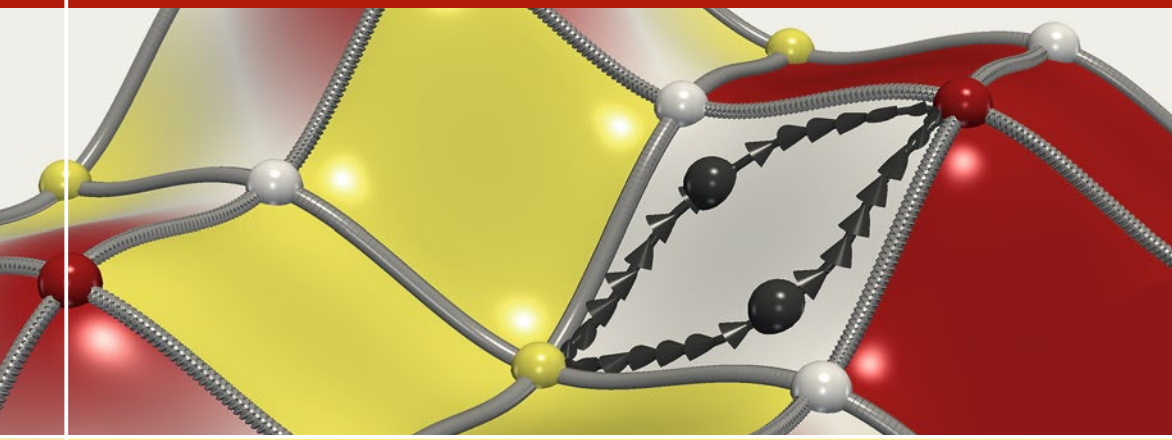
Mathematics and Visualization



Julien Tierny

Topological Data Analysis for Scientific Visualization

Springer

Mathematics and Visualization

More information about this series at http://www.springer.com/series/4562

Julien Tierny

Topological Data Analysis for Scientific Visualization

Springer

Julien Tierny
CNRS, Sorbonne Université, LIP6
Department of Scientific Computing
Paris, France

ISSN 1612-3786 ISSN 2197-666X (electronic)
Mathematics and Visualization
ISBN 978-3-319-71506-3 ISBN 978-3-319-71507-0 (eBook)
https://doi.org/10.1007/978-3-319-71507-0

Library of Congress Control Number: 2017960812

Mathematics Subject Classification (2010): 68U01

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Any problem which is non-linear in character, which involves more than one coordinate system or more than one variable, or where structure is initially defined in the large, is likely to require considerations of topology and group theory for its solution.

In the solution of such problems classical analysis will frequently appear as an instrument in the small, integrated over the whole problem with the aid of topology or group theory.

Marston Morse [87]

This book is dedicated to all those who supported me along the years.

Preface

This book is adapted from my habilitation thesis manuscript, which reviewed my research work since my Ph.D. thesis defense (2008), as a postdoctoral researcher at the University of Utah (2008–2010) and a permanent CNRS researcher at Telecom ParisTech (2010–2014) and at Sorbonne Universités UPMC (2014-present).

This book presents results obtained in collaboration with several research groups (University of Utah, Lawrence Livermore National Laboratory, Lawrence Berkeley National Laboratory, Universidade de Sao Paulo, New York University, Sorbonne Universités, Clemson University, University of Leeds) as well as students whom I informally or formally advised.

This research has been partially funded by several grants, including a Fulbright fellowship (US Department of State), a Lavoisier fellowship (French Ministry for Foreign Affairs), a Digiteo grant (national funding, "Uncertain Topo-Vis" project 2012-063D, Principal Investigator), an ANR grant (national funding, "CrABEx" project ANR-13-CORD-0013, local investigator), a CIFRE partnership with Renault, a CIFRE partnership with Kitware, a CIFRE partnership with Total, and a BPI grant (national funding, "AVIDO" project, local investigator).

During this period, I taught regularly at the University of Utah (2008–2010), Telecom ParisTech (2011–present), Sorbonne Universités (2011–present), and since 2013 at ENSTA ParisTech and University of Versailles, where I am the head instructor for the scientific visualization course.

This book describes most of the results published over this period (Chap. 3: [118, 125], Chap. 4: [63, 125, 128], Chap. 5: [16, 17, 55, 101], Chap. 6: [21, 56–58, 74, 117, 130]). I refer the interested reader to the following publications [12, 49, 50, 52, 80, 83, 95, 96, 107, 108, 111–113, 119–124, 126, 127, 132, 133] for additional results not described in this document.

The reading of this book only requires a basic background in computer science and algorithms; most of the mathematical notions are introduced in a dedicated chapter (Chap. 2).

Paris, France
October 2017

Julien Tierny

Acknowledgements

First, I would like to express my gratitude to the editors of the Springer's series *Mathematics and Visualization*, who gave me the opportunity to publish this book. I am sincerely grateful to Isabelle Bloch, Jean-Daniel Fekete, Pascal Frey, Hans Hagen, Chris Johnson, Bruno Lévy, Philippe Ricoux, and Will Schroeder, who did me the honor of accepting to be part of my habilitation committee. I am especially thankful to the reviewers of the manuscript for accepting this significant task.

Second, I would like to thank all my collaborators over the last 8 years. The results presented in this book would not have been possible without them. Since my Ph.D. defense, I had the opportunity to work with more than 50 coauthors, and I like to think that I learned much from each one of them. More specifically, I would like to thank some of my main collaborators (in alphabetical order), hoping the persons I forgot to mention will forgive me: Timo Bremer, Hamish Carr, Joel Daniels, Julie Delon, Tiago Etiene, Attila Gyulassy, Pavol Klacansky, Josh Levine, Gustavo Nonato, Valerio Pascucci, Joseph Salmon, Giorgio Scorzelli, Claudio Silva, Brian Summa, Jean-Marc Thiery. Along the years, some of these persons became recurrent collaborators with whom I particularly enjoyed working and interacting. Some of them even became close friends (even best men!) and I am sincerely grateful for that. Special thanks go to Valerio Pascucci, who gave me a chance back in 2008 when he hired me as a postdoc, although we had never met before. I have no doubt that my career path would have been very different if we had not worked together. Working with Valerio and his group has been a real pleasure and a source of professional and personal development. I am both glad and proud to be able to say that our collaboration lasted well beyond my stay at the University of Utah and that it still continues today. Along the last 8 years, Valerio has been a careful and inspiring mentor and I am sincerely grateful for that.

Next, I would like to thank all of the colleagues I had the chance to interact with at the University of Utah, Telecom ParisTech, and Sorbonne Universités UPMC, in particular my students, who are a daily source of motivation.

Finally, I would like to thank my friends, my family, my wife, and my daughter for their constant love and support.

Contents

Notations

$\mathbb{X}$	Topological space
$\partial\mathbb{X}$	Boundary of a topological space
$\mathbb{M}$	Manifold
$\mathbb{R}^d$	Euclidean space of dimension d
$\sigma,\ \tau$	d-simplex, face of a d-simplex
$v,\ e,\ t,\ T$	Vertex, edge, triangle, and tetrahedron
$Lk(\sigma),\ St(\sigma)$	Link and star of a simplex
$Lk_d(\sigma),\ St_d(\sigma)$	d-simplices of the link and the star of a simplex
$\mathscr{K}$	Simplicial complex
$\mathscr{T}$	Triangulation
$\mathscr{M}$	Piecewise linear manifold
β_i	i-th Betti number
χ	Euler characteristic
α_i	ith barycentric coordinates of a point p relatively to a simplex σ
$f : \mathscr{T} \to \mathbb{R}$	Piecewise linear scalar field
∇f	Gradient of a PL scalar field f
$Lk^-(\sigma),\ Lk^+(\sigma)$	Lower and upper link of σ relatively to f
$o(v)$	Memory position offset of the vertex v
$\mathscr{L}^-(i),\ \mathscr{L}^+(i)$	Sub- and sur-level set of the isovalue i relatively to f
$\mathscr{D}(f)$	Persistence diagram of f
$\mathscr{C}(f)$	Persistence curve of f
$\mathscr{R}(f)$	Reeb graph of f
$l(\mathscr{R}(f))$	Number of loops of $\mathscr{R}(f)$
$\mathscr{T}(f)$	Contour tree of f
$\mathscr{J}(f),\ \mathscr{S}(f)$	Join and split trees of f
$\mathscr{MS}(f)$	Morse-Smale complex of f

Chapter 1
Introduction

In early 2013, a group of researchers led by French scientists published in Nature a paper entitled *"A vast, thin plane of corotating dwarf galaxies orbiting the Andromeda galaxy"* [69]. This paper reported new intriguing observations that showed that a majority of the dwarf galaxies which orbit the larger Andromeda galaxy was actually rotating in a very thin, common plane structure. These observations then contradicted the state-of-art models which assumed that dwarf galaxies' locations followed an isotropic random distribution. This discovery raised many fundamental open questions that can potentially reshape the entire understanding of the universe formation process, as it implies that a still-to-be-found phenomenon seems to control the geometry of cosmos gas flow.

Beyond its academic outreach, this work drew a lot of attention from the French media, as one of the co-authors of the paper was a French teenager (and probably one of the youngest co-authors of a Nature publication). This student was doing a summer internship in a French astrophysics laboratory where he was assigned the design of a simple software prototype for the visualization of dwarf galaxy measurements. This is only when they started to visualize these measurements in 3D that these researchers made the astonishing observation of a coplanar orbit distribution, an hypothesis that was later confirmed through numerical estimations. In this study, while the numerical verification of the co-planarity hypothesis can be considered as a trivial task, formulating the original idea of this hypothesis cannot. Here, simple visualization tools precisely enabled this initial discovery as they helped these researchers formulate such original insights about their data.

This anecdote effectively illustrates one of the key motivations of Scientific Visualization, which is a sub-field of Computer Science that aims at developing efficient algorithms for the graphical and interactive exploration of scientific data, for the purpose of hypothesis formulation, analysis and interpretation.

While galaxy orbits are made of moderately simple geometries, recent acquisition devices or high-performance computing simulations nowadays generate large-scale data-sets of extremely precise resolution, which can encompass features with

J. Tierny, *Topological Data Analysis for Scientific Visualization*,
Mathematics and Visualization, https://doi.org/10.1007/978-3-319-71507-0_1

highly complex geometry, challenging their visualization and analysis. Therefore, research in Scientific Visualization aims at addressing several general challenges which impact distinct stages of the scientific methodology:

1. **Abstraction:** The definition of efficient analysis algorithms able to abstract high-level features (that humans can visualize, measure and understand) from raw data;
2. **Interaction:** The definition of efficient algorithms for the interactive manipulation, simplification and exploration of these high-level features;
3. **Analysis:** The definition of efficient algorithms for the geometrical measurement of these features, to serve as base tools for interpretation tasks in specific application problems.

Regarding scalar valued data, Topological Data Analysis forms a family of techniques that gained an increasing popularity in the Scientific Visualization community over the last two decades, since it precisely enables the robust capture and multi-scale representation of geometrical objects that often directly translate into features of interest application wise.

This book gives an introduction to the core concepts of Topological Data Analysis for Scientific Visualization, by covering each of the sub-topics mentioned above (abstraction, interaction and analysis). In particular, after a formalization of the main notions of Topological Data Analysis, it provides detailed explanations of some of its reference algorithms as well as application examples in computational fluid dynamics, topography, mechanical engineering, histology, combustion and chemistry. It also discusses upcoming challenges and research perspectives for Topological Data Analysis as well as preliminary results addressing such challenges, regarding the analysis of multivariate and uncertain data.

The rest of the book is organized as follows:

- Chapter 2 describes the theoretical background of Topological Data Analysis and briefly reviews the state-of-the-art;
- Chapter 3 describes examples of reference algorithms for the computation and simplification of topological abstractions of scalar data;
- Chapter 4 describes examples of efficient algorithms for user interactions with topological data abstractions;
- Chapter 5 describes concrete application examples as well as tailored data analysis pipelines based on topological data abstractions;
- Chapter 6 describes the perspectives and upcoming challenges for Topological Data Analysis and includes preliminary results regarding the analysis of multivariate and uncertain data.
- Chapter 7 finally concludes the book.

Chapter 2
Background

2.1 Data Representation

In scientific visualization, scalar data is in general defined on an input geometrical object (hereafter named *"Domain"*). It is represented by a finite set of sample values, continuously extended in space to the entirety of the domain thanks to an interpolant. In the following, a generic domain representation is first formalized. Next, a representation of the scalar data attached to this object (hereafter termed *"Range"*) is then formalized.

2.1.1 Domain Representation

In the following, a generic domain representation is formalized. This notion is introduced constructively. The end of this sub-section further describes topological notions relative to this domain representation, that will be used in the remainder of the book.

Preliminary Notions

Definition 2.1 (Topology) A topology on a set $\mathbb{X}$ is a collection T of subsets of $\mathbb{X}$ having the following properties:

- The sets $\emptyset$ and $\mathbb{X}$ are in T;
- The union of any sub-collection of T is in T;
- The intersection of a finite sub-collection of T is in T.

J. Tierny, *Topological Data Analysis for Scientific Visualization*,
Mathematics and Visualization, https://doi.org/10.1007/978-3-319-71507-0_2

Definition 2.2 (Topological Space) A set $\mathbb{X}$ for which a topology T is defined is called a *topological space*.

For example, the space of real numbers $\mathbb{R}$ is a topological space.

Definition 2.3 (Open Set) A subset $\mathbb{A} \subset \mathbb{X}$ of the topological space $\mathbb{X}$ is an *open* set of $\mathbb{X}$ if it belongs to T.

Definition 2.4 (Closed Set) A subset $\mathbb{B} \subset \mathbb{X}$ of the topological space $\mathbb{X}$ is a *closed* set of $\mathbb{X}$ if its complement $\mathbb{X} - \mathbb{B}$ is open.

Intuitively, open sets are subsets of topological spaces which do not contain their boundaries. For example, considering the space of real numbers $\mathbb{R}$, $(-\infty, 0) \cup (1, +\infty)$ and $[0, 1]$ are complements and respectively open and closed sets.

Property 2.1 (Open Sets)

- The set $\emptyset$ is open;
- The union of any number of open sets is open;
- The intersection of a finite number of open sets is open.

These properties follow from the definition of topology.

Definition 2.5 (Covering) A collection of subsets of a topological space $\mathbb{X}$ is a *covering* of $\mathbb{X}$ if the union of all its elements is equal to $\mathbb{X}$.

Definition 2.6 (Compact Topological Space) A topological space $\mathbb{X}$ is *compact* if every open covering of it contains a finite sub-collection that is also a covering of $\mathbb{X}$.

Definition 2.7 (Function) A *function* $f : \mathbb{A} \to \mathbb{B}$ associates each element of the topological space $\mathbb{A}$ with a unique element of the topological space $\mathbb{B}$.

Definition 2.8 (Injection) A function $f : \mathbb{A} \to \mathbb{B}$ is an *injection* if for each pair $a_1, a_2 \in \mathbb{A}$ such that $a_1 \neq a_2, f(a_1) \neq f(a_2)$. f is said to be *one-to-one*.

Definition 2.9 (Bijection) A function $f : \mathbb{A} \to \mathbb{B}$ is a *bijection* if for each element $b \in \mathbb{B}$ there is exactly one element $a \in \mathbb{A}$ such that $f(a) = b$. f is said to be bijective. It is also said to be *one-to-one* (injective) and *onto* (surjective).

Definition 2.10 (Continuous Function) A function $f : \mathbb{A} \to \mathbb{B}$ is *continuous* if for each open subset $\mathbb{C} \in \mathbb{B}$, the set $f^{-1}(\mathbb{C})$ is an open subset of $\mathbb{A}$.

Definition 2.11 (Homeomorphic Spaces) Two topological spaces $\mathbb{A}$ and $\mathbb{B}$ are *homeomorphic* if and only if there exists a continuous bijection $f : \mathbb{A} \to \mathbb{B}$ with a continuous inverse $f^{-1} : \mathbb{B} \to \mathbb{A}$. f is a *homeomorphism*.

Definition 2.12 (Manifold)

A topological space $\mathbb{M}$ is a *d-manifold* if every element $m \in \mathbb{M}$ has an open neighborhood $\mathbb{N}$ homeomorphic to an open Euclidean d-ball.

An intuitive description of a d-manifold is that of a curved space, which has locally the structure of an Euclidean space of dimension d, but which has a

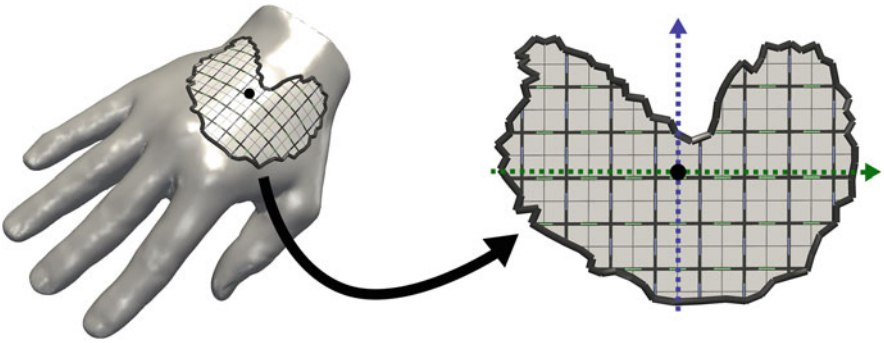


Fig. 2.1 Example of 2-manifold: any point of the surface (left, black dot) has an open neighborhood (textured chart) that is homeomorphic to an open Euclidean 2-ball (that can be *unfolded* to the plane, right)

more complicated global structure (Euclidean spaces are therefore special cases of manifolds). Figure 2.1 illustrates this with the example of a 2-manifold (surface).

Domain Formalization

In the following we formally introduce our domain representation as well as representations for connectivity information.

Definition 2.13 (Convex Set) A set $\mathbb{C}$ of an Euclidean space $\mathbb{R}^n$ of dimension n is *convex* if for any two points x and y of $\mathbb{C}$ and all $t \in [0, 1]$ the point $(1 - t)x + ty$ also belongs to $\mathbb{C}$.

Intuitively, a convex set is a set such that any two points of the set can be linked by a line segment that belongs to the set, as illustrated with 3-manifolds (volumes) in Fig. 2.2.

Definition 2.14 (Convex Hull) The *convex hull* of a set points $\mathscr{P}$ of an Euclidean space $\mathbb{R}^n$ is the unique minimal convex set containing all points of $\mathscr{P}$.

Definition 2.15 (Simplex) A *d-simplex* is the convex hull σ of $d + 1$ affinely independent points of an Euclidean space $\mathbb{R}^n$, with $0 \leq d \leq n$. d is the dimension of σ.

Definition 2.16 (Vertex) A *vertex* v is a 0-simplex of $\mathbb{R}^3$.

Definition 2.17 (Edge) An *edge* e is a 1-simplex of $\mathbb{R}^3$.

Definition 2.18 (Triangle) A *triangle* t is a 2-simplex of $\mathbb{R}^3$.

Definition 2.19 (Tetrahedron) A *tetrahedron* T is a 3-simplex of $\mathbb{R}^3$.

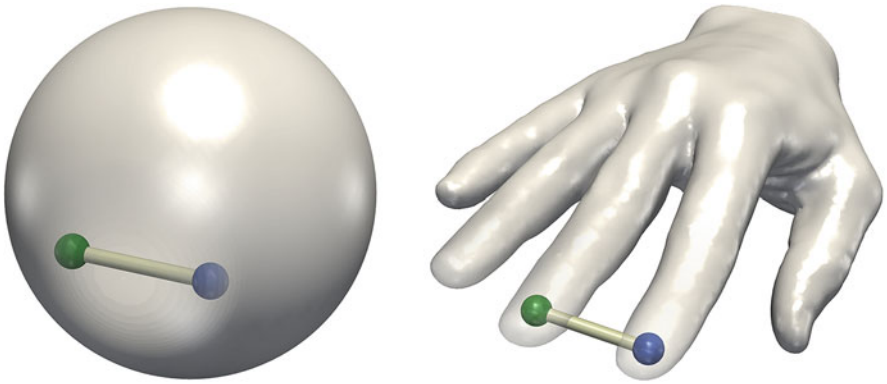


Fig. 2.2 Examples of convex (left) and non-convex (right) 3-manifolds (volumes). On the left, any two points (green and blue spheres) can be linked by a line segment that belongs to the volume (white cylinder). This is not the case for the right volume

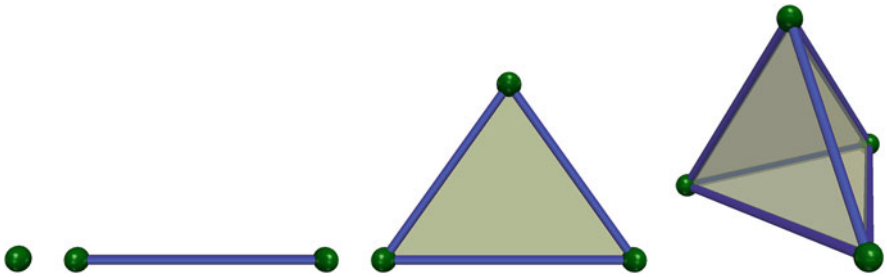


Fig. 2.3 Illustrations of 0 (green), 1 (blue), 2 (white) and 3-simplices (transparent), from left to right, along with their faces

Definition 2.20 (Face) A *face* τ of a d-simplex σ is the simplex defined by a non-empty subset of the $d+1$ points of σ, and is noted $\tau \leq \sigma$. We will note τ_i a face of dimension i.

In summary, a d-simplex is the smallest combinatorial construction that can represent a neighborhood of a d-dimensional Euclidean space. As illustrated in Fig. 2.3, it is composed of *faces*, that are themselves $(d-1)$, $(d-2)$, ..., and 0-simplices.

Definition 2.21 (Simplicial Complex) A *simplicial complex* $\mathcal{K}$ is a finite collection of non-empty simplices $\{\sigma_i\}$, such that every face τ of a simplex σ_i is also in $\mathcal{K}$, and any two simplices σ_i and σ_j intersect in a common face or not at all.

Definition 2.22 (Star) The *star* of a simplex σ of a simplicial complex $\mathcal{K}$ is the set of simplices of $\mathcal{K}$ that contain σ: $St(\sigma) = \{\tau \in \mathcal{K}, \sigma \leq \tau\}$. We will note $St_d(\sigma)$ the set of d-simplices of $St(\sigma)$.

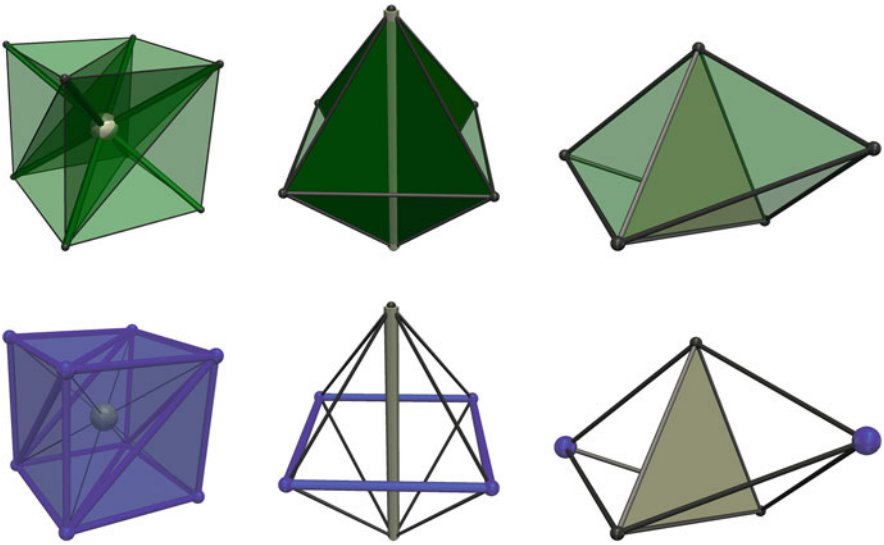


Fig. 2.4 Illustrations of stars (green, top) and links (blue, bottom) for 0, 1 and 2-simplices (white, from left to right) of a 3-dimensional simplicial complex

Definition 2.23 (Link)

The *link* of σ is the set of faces of the simplices of $St(\sigma)$ that are disjoint from σ: $Lk(\sigma) = \{\tau \leq \Sigma, \Sigma \in St(\sigma), \tau \cap \sigma = \emptyset\}$. We will note $Lk_d(\sigma)$ the set of d-simplices of $Lk(\sigma)$.

In other words, the star of a simplex σ is the set of simplices having σ as a face, as illustrated Fig. 2.4 (top). The notion of link is illustrated at the bottom of Fig. 2.4.

Definition 2.24 (Underlying Space) The *underlying space* of a simplicial complex $\mathscr{K}$ is the union of its simplices $|\mathscr{K}| = \cup_{\sigma \in \mathscr{K}} \sigma$.

Definition 2.25 (Triangulation)

The *triangulation* $\mathscr{T}$ of a topological space $\mathbb{X}$ is a simplicial complex $\mathscr{K}$ whose underlying space $|\mathscr{K}|$ is homeomorphic to $\mathbb{X}$.

The notion of triangulation has been preferred here to other competing representations for its practical genericity: any mesh representation (regular grid, unstructured grid, etc.) can be easily converted into a triangulation by subdividing each of its d-cells into valid d-simplices (having only $(d + 1)$ linearly independent points), as illustrated in Fig. 2.5 for the case of a regular grid. Also, note that for regular grids, the resulting triangulation can be implicitly encoded (i.e. adjacency relations can be retrieved on demand, without storage, thanks to the recurring subdivision pattern of the regular grid). Moreover, as detailed in the next subsection, triangulations can be accompanied with well-behaved interpolants, which facilitate reasoning and computation with scalar data.

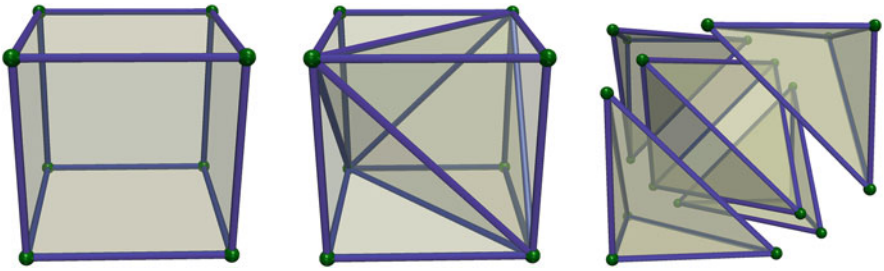


Fig. 2.5 A 3-dimensional regular grid (left) can be easily converted into a triangulation by subdividing each of its voxels independently into 5 tetrahedra (center, right: exploded view). This subdivision can be implicitly encoded

Fig. 2.6 Example of PL 3-manifold (left, right: clipped view)

As discussed further down this book, for reasoning and robustness purposes, the following, more restrictive, notion is often preferred over triangulations.

Definition 2.26 (Piecewise Linear Manifold)

The triangulation of a manifold $\mathbb{M}$ is called a *piecewise linear manifold* and is noted $\mathscr{M}$.

Therefore, a piecewise linear (PL) manifold is a combinatorial representation of a manifold that derives from the notion of triangulation, as illustrated in Fig. 2.6. It can be efficiently represented in memory by storing for each dimension d, the list of d-simplices as well as their stars and links. In the remainder of this book, we will consider PL-manifolds as our generic domain representations.

Topological Invariants

In the following, we present a few *topological invariants*: entities that do not change under continuous transformations of the domain (variations in point positions but no variation in connectivity). These notions are instrumental in Topological Data Analysis.

Definition 2.27 (Path) A homeomorphism $p : (a, b) \to \mathbb{C}$ from an open interval $(a, b) \subseteq \mathbb{R}$ to a subset $\mathbb{C}$ of a topological space $\mathbb{X}$ is called a *path* on $\mathbb{X}$ between $p(a)$ and $p(b)$.

Definition 2.28 (Connected Topological Space) A topological space $\mathbb{X}$ is *connected* if for any two points of $\mathbb{X}$ there exists a path between them on $\mathbb{X}$.

Definition 2.29 (Connected Components) The maximally connected subsets of a topological space $\mathbb{X}$ are called its *connected components*.

Definition 2.30 (Homotopy) A *homotopy* between two continuous functions f and g is a continuous function $H : \mathbb{X} \times [0, 1] \to \mathbb{Y}$ from the product of a topological space $\mathbb{X}$ with the closed unit interval to a topological space $\mathbb{Y}$ such that for each point $x \in \mathbb{X}$, $H(x, 0) = f(x)$ and $H(x, 1) = g(x)$. If there exists a homotopy between them, f and g are said to be *homotopic*.

While homeomorphism deals with the matching between neighborhoods, homotopies additionally require that a continuous transformation exists between them, by considering neighborhoods as images of functions (the notion of homotopy is then refined to that of isotopy). Here, the second parameter of an homotopy can be seen as time in this continuous transformation process. For instance, a circle and a knot are homeomorphic but are not homotopic since the knot needs to be cut and stitched back to be turned into a circle, which is not a continuous transformation.

Definition 2.31 (Simply Connected Topological Space)

A topological space $\mathbb{X}$ is *simply connected* if it is connected and if for any two points of $\mathbb{X}$, any two paths between them on $\mathbb{X}$ are homotopic.

As illustrated in Fig. 2.7, a domain is simply connected if for any two points, any pair of paths between them can be continuously transformed into one another (black paths in Fig. 2.7, right).

Definition 2.32 (Boundary) The *boundary* of a topological space $\mathbb{X}$, noted $\partial\mathbb{X}$, is the complement in $\mathbb{X}$ of the subspace of $\mathbb{X}$, called the *interior* of $\mathbb{X}$, composed of all the elements $x \in \mathbb{X}$ such that x has an open neighborhood $\mathbb{N}$.

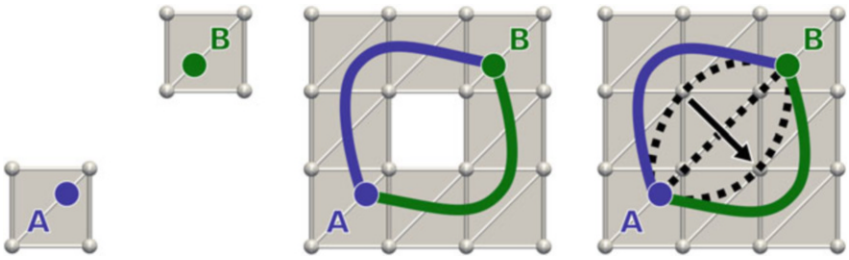


Fig. 2.7 Examples of disconnected, connected and simply connected domains (from left to right)

Definition 2.33 (Boundary Component) A *boundary component* of a topological space $\mathbb{X}$ is a connected component of its boundary $\partial\mathbb{X}$.

Definition 2.34 (p-Chain) A *p-chain* of a triangulation $\mathscr{T}$ of a topological space $\mathbb{X}$ is a formal sum (with modulo two coefficients) of p-simplices of $\mathscr{T}$.

Definition 2.35 (p-Cycle) A *p-cycle* of a triangulation $\mathscr{T}$ of a topological space $\mathbb{X}$ is a p-chain with empty boundary.

Definition 2.36 (Group of p-Cycles) The *group of p-cycles* of a triangulation $\mathscr{T}$ of a topological space $\mathbb{X}$ is the group of all p-cycles of $\mathscr{T}$, noted $Z_p(\mathscr{T})$, which forms a sub-group of all p-chains of $\mathscr{T}$.

Definition 2.37 (p-Boundary) A *p-boundary* of a triangulation $\mathscr{T}$ of a topological space $\mathbb{X}$ is the boundary of a $(p+1)$-chain.

Property 2.2 (p-Boundary) A p-boundary is a p-cycle.

Definition 2.38 (Group of p-Boundaries) The *group of p-boundaries* of a triangulation $\mathscr{T}$ of a topological space $\mathbb{X}$ is the group of all p-boundaries of $\mathscr{T}$, noted $B_p(\mathscr{T})$, which forms a sub-group of all p-cycles of $\mathscr{T}$.

Definition 2.39 (Homology Group) The pth *homology group* of a triangulation $\mathscr{T}$ of a topological space $\mathbb{X}$ is its p*th* cycle group modulo its pth boundary group: $H_p(\mathscr{T}) = Z_p(\mathscr{T})/B_p(\mathscr{T})$.

Intuitively, two p-cycles are said to be equivalent, or homologous, if they can be continuously transformed into each other (through formal sums with modulo two coefficients) without being collapsible to a point. Then, one can further group p-cycles into *classes* of equivalent p-cycles. Each class can be represented by a unique representative p-cycle that is called *generator* (and that is homologous to any other p-cycle of the class), as illustrated in Fig. 2.8 with a green 1-cycle (center) and a green 2-cycle (right). Enumerating the number of generators of a homology group enables to introduce intuitive topological invariants called Betti numbers.

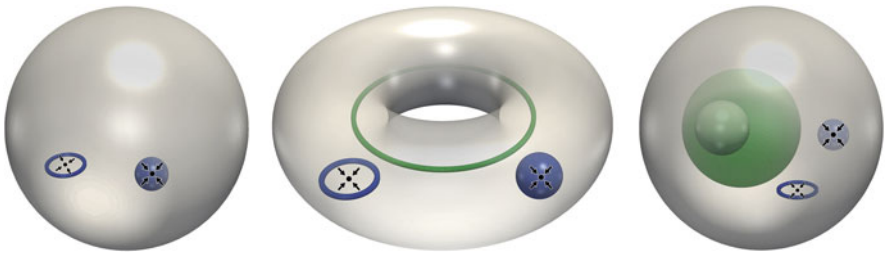


Fig. 2.8 Examples of PL 3-manifolds with varying Betti numbers. From left to right: a 3-ball, a solid torus, a 3-ball with a void. From left to right, $(\beta_0, \beta_1, \beta_2)$ is equal to $(1, 0, 0)$, $(1, 1, 0)$, and $(1, 0, 1)$). Generators are displayed in green, while examples of non-generator p-cycles are displayed in blue

Definition 2.40 (Betti Number)

The pth *Betti number* of a triangulation $\mathscr{T}$ of a topological space $\mathbb{X}$ is the rank of its p^{th} homology group: $\beta_p(\mathscr{T}) = rank(H_p(\mathscr{T}))$.

In low dimensions, Betti numbers have a very concrete interpretation. For instance, for PL 3-manifolds, β_0 corresponds to the number of connected components, β_1 to the number of handles and β_2 to the number of voids, as illustrated in Fig. 2.8 (β_3 is equal to 0 for PL 3-manifolds with boundary, i.e. that can be embedded in $\mathbb{R}^3$).

Definition 2.41 (Euler Characteristic)

The *Euler characteristic* of a triangulation $\mathscr{T}$ of a topological space $\mathbb{X}$ of dimension d, noted $\chi(\mathscr{T})$, is the alternating sum of its Betti numbers: $\chi(\mathscr{T}) = \sum_{i=0}^{i=d}(-1)^i\beta_i(\mathscr{T})$.

Property 2.3 (Euler Characteristic) The *Euler characteristic* of a triangulation $\mathscr{T}$ of a topological space $\mathbb{X}$ of dimension d is also equal to the alternating sum of the number of its i-simplices: $\chi(\mathscr{T}) = \sum_{i=0}^{i=d}(-1)^i|\sigma_i|$.

2.1.2 Range Representation

In the following, we formalize a range representation based on the previously introduced domain representation. Additionally, we will introduce a few related geometrical constructions that will be instrumental to Topological Data Analysis.

Piecewise Linear Scalar Fields

Definition 2.42 (Barycentric Coordinates)

Let p be a point of $\mathbb{R}^n$ and σ a d-simplex. Let $\alpha_0, \alpha_1, \ldots, \alpha_d$ be a set of real coefficients such that $p = \sum_{i=0}^{i=d}\alpha_i\tau_0^i$ (where τ_0^i is the ith zero dimensional face of σ) and such that $\sum_{i=0}^{i=d}\alpha_i = 1$. Such coefficients are called the *barycentric coordinates* of p relatively to σ.

Property 2.4 (Barycentric Coordinates) The barycentric coordinates of p relative to σ are unique.

Property 2.5 (Barycentric Coordinates) If and only if there exists an i for which $\alpha_i \notin [0, 1]$, then p does not belong to σ, otherwise it does.

Fig. 2.9 Example of PL scalar field f defined on a PL 3-manifold $\mathscr{M}$. From left to right: restriction $\hat{f}$ of f on the 0-simplices of $\mathscr{M}$, f (the color coding denotes the linear interpolation within each simplex), clipped view of f

Definition 2.43 (Piecewise Linear Scalar Field)

Let $\hat{f}$ be a function that maps the 0-simplices of a triangulation $\mathscr{T}$ to $\mathbb{R}$. Let $f : \mathscr{T} \to \mathbb{R}$ be the function linearly interpolated from $\hat{f}$ such that for any point p of a d-simplex σ of $\mathscr{T}$, we have: $f(p) = \sum_{i=0}^{i=d} \alpha_i \hat{f}(\tau_0^i)$ (where τ_0^i is the ith zero dimensional face of σ). f is called a *piecewise linear (PL) scalar field*.

Piecewise linear scalar fields will be our default representation for scalar data. Typically, the input data will then be given in the form of a triangulation with scalar values attached to its vertices ($\hat{f}$). The linear interpolation provided by the barycentric coordinates can be efficiently computed on demand (on the CPU or the GPU, as illustrated in Fig. 2.9) and has several nice properties that makes it well suited for combinatorial reasonings.

Property 2.6 (Gradient of a Piecewise Linear Scalar Field) The *gradient* ∇f of a PL scalar field $f : \mathscr{T} \to \mathbb{R}$ is a curl free vector field that is piecewise constant (constant within each d-simplex of $\mathscr{T}$).
This property has several implications that will be discussed in the following subsections.

Definition 2.44 (Lower Link) The *lower link* $Lk^-(\sigma)$ (respectively the *upper link* $Lk^+(\sigma)$) of a d-simplex σ relatively to a PL scalar field f is the subset of the link $Lk(\sigma)$ such that each of its zero dimensional faces has a strictly lower (respectively higher) f value than those of σ.

Given the above definition, it is often useful to disambiguate configurations of equality in f values between vertices (thus equality configurations in $\hat{f}$). Therefore, $\hat{f}$ is often slightly perturbed with a mechanism inspired by simulation of simplicity [43] to turn $\hat{f}$ into an injective function. This can be achieved in the following way, by adding to $\hat{f}$ a second function $\hat{g}$ that is injective. Let $o(v)$ denote the position integer offset of the vertex v in memory. $o(v)$ is injective. Then, to turn $\hat{f}$ into an injective function, one needs to add to it $\epsilon o(v)$ where ϵ is an arbitrarily small real value. As the original simulation of simplicity, this mechanism can be implemented numerically (by choosing the smallest possible value for ϵ depending on the machine precision) or preferably symbolically by re-implementing the necessary

predicates. For instance, to decide if a vertex v_0 is lower than a vertex v_1, one needs to test $\hat{f}(v_0) < \hat{f}(v_1)$ and, in case of equality, test $o(v_0) < o(v_1)$ to disambiguate. In the following, we will therefore consider that $\hat{f}$ is always injective in virtue of this mechanism. Therefore, no d-simplex of $\mathscr{T}$ collapses to a point of $\mathbb{R}$ through f for any non-zero d.

Related Geometrical Constructions

Based on our representation for scalar data on geometrical domains, we will now introduce a few geometrical constructions that will be instrumental in Topological Data Analysis.

Definition 2.45 (Sub-level Set) The *sub-level set* $\mathscr{L}^-(i)$ (respectively the *sur-level set* $\mathscr{L}^+(i)$) of an isovalue $i \in \mathbb{R}$ relatively to a PL scalar field $f : \mathscr{M} \to \mathbb{R}$ is the set of points: $\{p \in \mathscr{M} \mid f(p) \leq i\}$ (respectively $\{p \in \mathscr{M} \mid f(p) \geq i\}$).

Definition 2.46 (Level Set)

The *level-set* $f^{-1}(i)$ of an isovalue $i \in \mathbb{R}$ relatively to a PL scalar field $f : \mathscr{M} \to \mathbb{R}$ is the pre-image of i onto $\mathscr{M}$ through f: $f^{-1}(i) = \{p \in \mathscr{M} \mid f(p) = i\}$.

Property 2.7 (Level Set) The level set $f^{-1}(i)$ of a regular isovalue $i \in \mathbb{R}$ relatively to a PL scalar field $f : \mathscr{M} \to \mathbb{R}$ defined on a PL d-manifold $\mathscr{M}$ is a $(d-1)$-manifold.

Property 2.8 (Level Set) Let $f : \mathscr{T} \to \mathbb{R}$ be a PL scalar field and σ be a d-simplex of $\mathscr{T}$. For any isovalue $i \in f(\sigma)$, the restriction of $f^{-1}(i)$ within σ belongs to an Euclidean subspace of $\mathbb{R}^d$ of dimension $(d-1)$.

This latter property directly follows from Property 2.6 on the gradient of PL scalar fields, which is piecewise constant (a level set is everywhere orthogonal to the gradient). It follows that the level sets of PL scalar fields defined on PL manifolds can be encoded as PL manifolds, as illustrated with the white PL 2-manifold in Fig. 2.10 (right).

Property 2.9 (Level Set) Let $f : \mathscr{T} \to \mathbb{R}$ be a PL scalar field and σ be a d-simplex of $\mathscr{T}$. For any two isovalues $i \neq j$ belonging to $f(\sigma)$, the restrictions of $f^{-1}(i)$ and of $f^{-1}(j)$ within σ are parallel.

This property also follows from Property 2.6 on the gradient of PL scalar fields and is illustrated in Fig. 2.10 (right, dark gray isosurfaces), which shows an *isosurface* restricted to a 3-simplex (i.e. a level set of a PL scalar field defined on a PL 3-manifold). Such strong properties (planarity and parallelism) enable to derive robust and easy-to-implement algorithms for level set extraction (called *"Marching Tetrahedra"* for PL 3-manifolds, and *"Marching Triangles"* for PL 2-manifolds).

Definition 2.47 (Contour) Let $f^{-1}(i)$ be the level set of an isovalue i relatively to a PL scalar field $f : \mathscr{T} \to \mathbb{R}$. Each connected component of $f^{-1}(i)$ is called a contour.

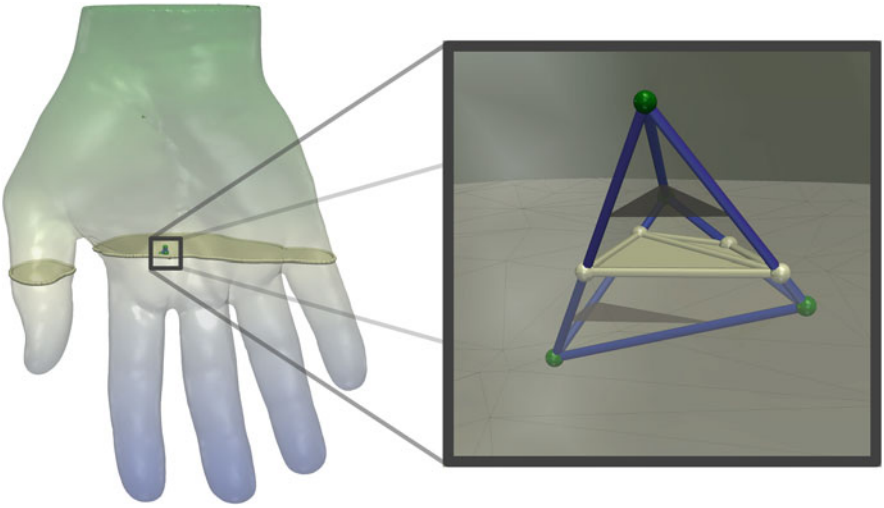


Fig. 2.10 Example of level set (isosurface, left) of a PL scalar field defined on a PL 3-manifold. Right: restriction of the isosurface to a 3-simplex

Definition 2.48 (Integral Line)

Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a PL scalar field defined on a PL manifold $\mathcal{M}$. An *integral line* is a path $p : \mathbb{R} \rightarrow \mathcal{C} \subset \mathcal{M}$ such that $\frac{\partial}{\partial t}p(t) = \nabla f(p(t))$. $lim_{t \rightarrow -\infty} p(t)$ and $lim_{t \rightarrow \infty} p(t)$ are called the *origin* and the *destination* of the integral line respectively.

In other words, an integral line is a path which is everywhere tangential to the gradient. In virtue of Property 2.6 on the gradient of PL scalar fields, it follows that integral lines can be encoded as PL 1-manifolds, as illustrated with the white PL 1-manifold in Fig. 2.11 (right).

2.2 Topological Abstractions

Level sets (and especially contours) and integral lines are fundamental geometrical objects in Scientific Visualization for the segmentation of regions of interests (burning flames in combustion, interaction pockets in chemistry, etc.) or the extraction of filament structures (galaxy backbones in cosmology, covalent interactions in chemistry, etc.).

Intuitively, the key idea behind Topological Data Analysis is to segment the data into regions where these geometrical objects are *homogeneous* from a topological perspective, and to summarize these homogeneity relationships into a *topological abstraction*. Such a segmentation strategy enables to access these features more

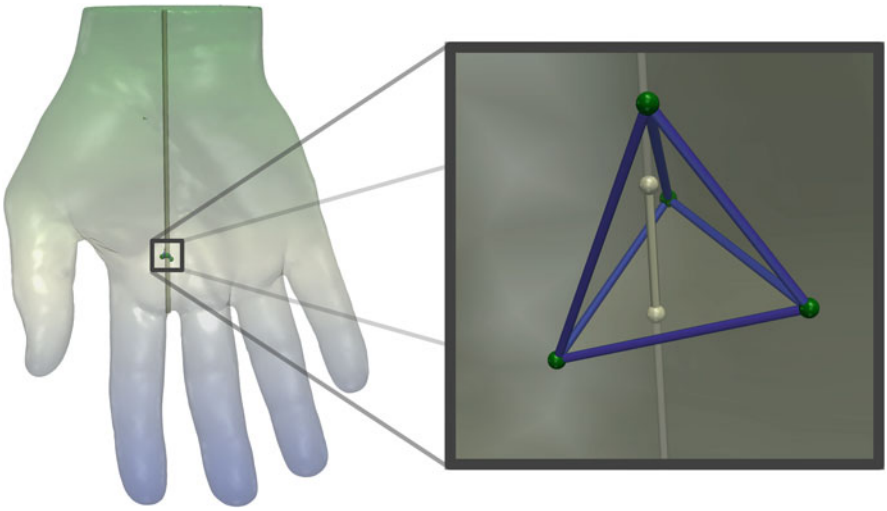


Fig. 2.11 Examples of integral line (left) of a PL scalar field defined on a PL 3-manifold. Right: restriction of the integral line to a 3-simplex

efficiently and to classify them according to application dependent metrics for further processing.

In the following, we introduce such topological abstractions for feature extraction, segmentation and classification purposes.

2.2.1 Critical Points

In the smooth setting, critical points are points of a manifold where the gradient of a smooth scalar field vanishes. Unfortunately, this notion does not directly translate into the PL setting since the gradient of a PL scalar field is piecewise constant. This requires the usage of an alternate definition, which interestingly involves topological and combinatorial reasonings.

Morse theory [84] relates the study of the topology of manifolds to the study of a specific group of smooth scalar fields defined on them (called *Morse* functions). One of the key results of Morse theory is the following property: the Betti numbers of the sub-level sets of a Morse function only change in the vicinity of a critical point. In other words, the topology of the sub-level sets only evolves when crossing a critical point. This observation is at the basis of a formalization of critical points in the PL setting.

Definition 2.49 (Critical Point)

Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a PL scalar field defined on a PL manifold $\mathcal{M}$. A vertex v of $\mathcal{M}$ is a *regular point* if and only if both $Lk^-(v)$ and $Lk^+(v)$ are simply connected. Otherwise, v is a *critical point* of f and $f(v)$ is called a *critical isovalue* (as opposed to *regular isovalues*).

Definition 2.50 (Critical Contour) Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a PL scalar field defined on a PL manifold $\mathcal{M}$. A contour of f which contains one of its critical points is called a *critical contour*.

In virtue of these definitions and Property 2.6 on their gradient, PL scalar fields have many nice properties regarding their critical points.

Property 2.10 (Critical Points) Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a PL scalar field defined on a compact PL manifold $\mathcal{M}$. The set of critical points of f, noted C_f, contains only isolated critical points and its cardinality $|C_f|$ is finite.

These properties follow from the fact that the gradient of a PL scalar field is piecewise constant: $\hat{f}$ is assumed to be injective, thus any d-simplex with $d \neq 0$ is mapped to a non null gradient vector. Therefore, critical points are isolated and can only occur on vertices. This makes their number finite for compact PL manifolds. This property is essential for a combinatorial reasoning on critical points. Also, note that the above definitions are independent of the dimension of $\mathcal{M}$.

Definition 2.51 (Extremum) Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a PL scalar field defined on a PL manifold $\mathcal{M}$. A critical point v is a *minimum* (respectively a *maximum*) of f if and only if $Lk^-(v)$ (respectively $Lk^+(v)$) is empty.

Definition 2.52 (Saddle) Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a PL scalar field defined on a PL manifold $\mathcal{M}$. A critical point v is a *saddle* if and only if it is neither a minimum nor a maximum of f.

Figure 2.12 illustrates the notion of critical points on a toy example. The evolution in the topology of the level sets can be observed in the right insets, which illustrate vertex stars (the smallest combinatorial neighborhood of a vertex in a triangulation). For a minimum (respectively a maximum) $\beta_0(f^{-1}(i))$ increases

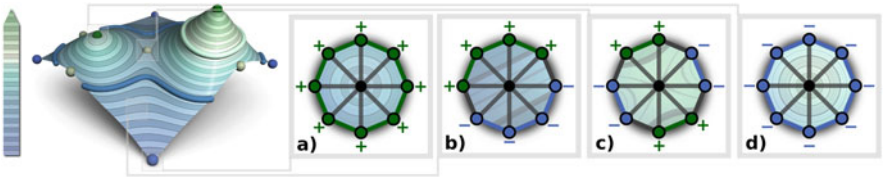


Fig. 2.12 Scalar field on a terrain (left). A level set is shown in blue, a contour is shown in white. Vertices can be classified according to the connectivity of their lower (blue) and upper (green) links. From left to right: a minimum (*a*, blue spheres on the left), a regular point (*b*), a simple saddle (*c*, white spheres on the left) and a maximum (*d*, green spheres on the left). ©2012 IEEE. Reprinted, with permission, from [118]

(respectively decreases) by one in the vicinity of the extremum. For a regular point, this number does not evolve when crossing a regular point. When crossing a saddle, the number of connected components of the restriction of $f^{-1}(i)$ to the star of the vertex first decreases by one exactly at the saddle and then increases by one right above it.

Critical points are usually classified according to their index. For PL scalar fields defined on PL 2-manifolds, minima have index 0, saddles index 1 and maxima index 2. As the dimension of the domain increases, the number of types of critical points also increases. For PL scalar fields defined on PL 3-manifolds, minima have index 0, 1-saddles (saddles that locally merge level sets) have index 1, 2-saddles (saddles that locally split level sets) have index 2 and maxima index 3. In the following, we will note C_f^i the set of critical points of f of index i.

Definition 2.53 (Saddle Multiplicity) Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a PL scalar field defined on a PL manifold $\mathcal{M}$ and let v be a saddle of f. Let k be the maximum value between $\beta_0(Lk^-(v))$ and $\beta_0(Lk^+(v))$. The *multiplicity* of a saddle is equal to $(k-1)$. A saddle of multiplicity 1 is called a *simple saddle*. It is called a *multi-saddle* otherwise, or $(k-1)$-fold saddle, or alternatively a *degenerate critical point*.

Definition 2.54 (PL Morse Scalar Field)

Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a PL scalar field defined on a PL manifold $\mathcal{M}$. f is a *PL Morse scalar field* if and only if *(i)* all its critical points have distinct f values and *(ii)* f has no degenerate critical point.

In practice, any PL scalar field can be easily perturbed into a PL Morse scalar field. The first condition can be easily satisfied by forcing $\hat{f}$ to be injective as described in the previous subsection. The second condition can be satisfied by a process called multi-saddle unfolding [42], that locally re-triangulates the star of a $(k-1)$-fold saddle into $(k-1)$ simple saddles.

Interestingly, PL Morse scalar fields inherit from most of the properties of their smooth counter-parts. In particular, the Morse-Euler relation, as first shown by Banchoff [7], still holds for PL Morse scalar fields.

Property 2.11 (Morse-Euler Relation)

Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a PL Morse scalar field defined on a closed PL manifold $\mathcal{M}$ of dimension d. Then the *Morse-Euler relation* holds: $\chi(\mathcal{M}) = \sum_{i=0}^{i=d}(-1)^i|C_f^i|$.

This property nicely summarizes the relation between the critical points of a PL Morse scalar field and the topology of its domain. Moreover, it also illustrates the global consistency of the local critical point classification definition (Definition 2.49).

In practice, critical points often directly translate into points of interest application wise. For instance, in 2D vector fields obtained in computational fluid dynamics, extrema of the magnitude of the curl indicate the locations of vortices, a high-level notion that has important implications in the efficiency evaluation of a flow (Fig. 2.13, bottom).

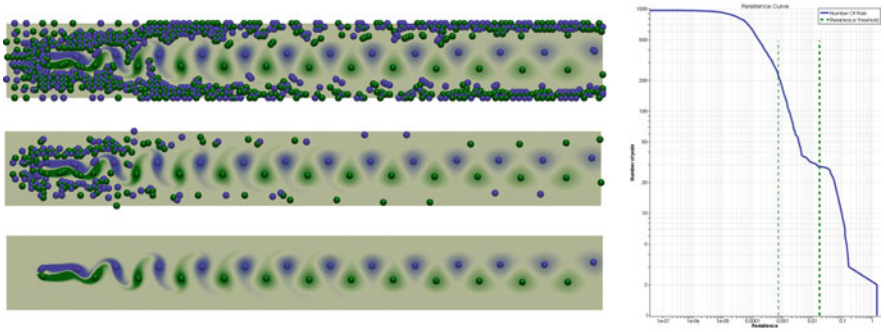


Fig. 2.13 Minima (blue) and maxima (green) of the orthogonal curl component of a flow simulation of the von Kármán street (flow turbulence behind an obstacle, here at the left of the domain). Right: persistence curve of the field. Selecting extrema involved in pairs more persistent than an increasing threshold (vertical lines, right) yields a hierarchy of critical point sets (left). Here, the light green vertical line (right) corresponds to the middle level (left) while the dark green line (right) corresponds to the bottom level (left). In practice, a flat plateau in the persistence curve (right) often indicates a separation between noise and features

2.2.2 Notions of Persistent Homology

As described in the previous subsection, critical points of PL scalar fields can be extracted with a robust, localized yet globally consistent, combinatorial and inexpensive classification (Definition 2.49). However, in practice, this classification strategy will identify, among others, critical points corresponding to slight function undulations coming from the noise in the data generation process (acquisition noise, numerical noise in simulations), as illustrated in Fig. 2.13 (top). Therefore, to make critical point extraction reliable and useful in practice, one needs to derive a mechanism to further classify critical points into noise or signal, given some application dependent metric. This is the purpose of Persistent Homology.

Definition 2.55 (Filtration) Let $f : \mathscr{K} \to \mathbb{R}$ be an injective scalar field defined on a simplicial complex $\mathscr{K}$ such that $f(\tau) < f(\sigma)$ for each face τ of each simplex σ. Let n be the number of simplices of $\mathscr{K}$ and let $\mathscr{L}^-(i)$ be the sub-level set of f by the ith value in the sorted set of simplex values. The nested sequence of subcomplexes $\mathscr{L}^-(0) \subset \mathscr{L}^-(1) \subset \cdots \subset \mathscr{L}^-(n-1) = \mathscr{K}$ is called the *filtration* of f.

The general notion of filtration is preferred here to the more specific notion of lower star filtration (specifically adapted to PL scalar fields) as this general introduction will be useful in the next subsections.

Definition 2.56 (Homomorphism) A *homomorphism* is a map between groups that commutes with the group operation.

For instance, the group operation for the group of p-chains is the formal sum of p-simplices (see Definition 2.34).

Fig. 2.14 Sub-complexes induced by the filtration of a PL scalar field defined on a PL 3-manifold (dark blue: $\mathscr{L}^-(i)$, light blue: $\mathscr{L}^-(j)$). From left to right: $\beta_0(\mathscr{L}^-(i)) = 3$, $\beta_0(\mathscr{L}^-(j)) = 4$, $\beta_0(\mathscr{L}^-(i,j)) = 2$

The filtration of a scalar field f induces a sequence of homomorphisms between the homology groups of the subcomplexes of $\mathscr{K}$:

$$H_p(\mathscr{L}^-(0)) \to H_p(\mathscr{L}^-(1)) \to \cdots \to H_p(\mathscr{L}^-(n-1)) = H_p(\mathscr{K}) \qquad (2.1)$$

Definition 2.57 (Persistent Homology Group) The pth *persistent homology groups* are the images of the homomorphisms induced by inclusion, noted $H_p^{i,j}$, for $0 \leq i \leq j \leq n-1$. The corresponding pth *persistent Betti numbers* are the ranks of these groups, $\beta_p^{i,j} = rank(H_p^{i,j})$.

Figure 2.14 provides a visual interpretation of the notion of 0th persistent Betti number, which characterizes connected components. Given two nested subcomplexes $\mathscr{L}^-(i)$ and $\mathscr{L}^-(j)$, with $i < j$, the subcomplex induced by inclusion with regard to the 0th homology group is noted $\mathscr{L}^-(i,j)$: it is defined by the connected components of $\mathscr{L}^-(j)$ which have non empty intersections with these of $\mathscr{L}^-(i)$ (which *includes* them). The 0th homology group of $\mathscr{L}^-(i,j)$ is composed of the classes of the 0th homology group that already existed at the ith isovalue and which still exist at the jth isovalue.

In this example, only two of the four connected components of $\mathscr{L}^-(j)$ include connected components of $\mathscr{L}^-(i)$. Therefore, among the three connected components of $\mathscr{L}^-(i)$, only two of them are *persistent* in the interval $[i,j]$.

Therefore, persistent homology provides a mechanism to characterize the importance of topological features (here connected components) with regard to a specific measure (here a PL scalar field), at multiple scales (here the interval $[i,j]$). This interpretation can be generalized to other topological features (for PL 3-manifolds cycles and voids) by extending it to other Betti numbers.

The previous example illustrated the case where a class of the 0th homology group (representing a connected component) disappeared in between the ith and jth isovalues: $\beta_0(\mathscr{L}^-(i)) = 3$, $\beta_0(\mathscr{L}^-(i,j)) = 2$. One can further track the

precise isovalue where classes appear, disappear or merge with others. Such events correspond to a change in the Betti numbers of the sub-level set of the scalar field. As discussed in the previous subsection, these changes occur at critical points. Therefore, the notions of *birth* and *death* of classes of persistent homology groups can be associated with pairs of critical points of the input scalar field, and the absolute value of their f value difference is called the *persistence* of the pair.

In particular, the merge of two classes represent a *death* event. In such a case, the least persistent class (the youngest of the two) is chosen as the dying class. This choice is often called the *Elder's rule* [42]. Once this is established, one can pair without ambiguity all the critical points of a scalar field. Note that this observation could already be foreseen with the Morse-Euler relation. For instance, the removal of an index 2 critical point (a maximum) of a PL Morse scalar field defined a PL 2-manifold implies the removal of a paired index 1 critical point (a saddle) in order to keep the Euler characteristic constant (Property 2.11).

Therefore, it is possible to enumerate all the classes of all pth homology group by enumerating the critical point pairs identified with the above strategy. This list of critical point pairs can be concisely encoded with a topological abstraction called the *Persistence Diagram* (Fig. 2.15), noted $\mathscr{D}(f)$. This diagram is a one-

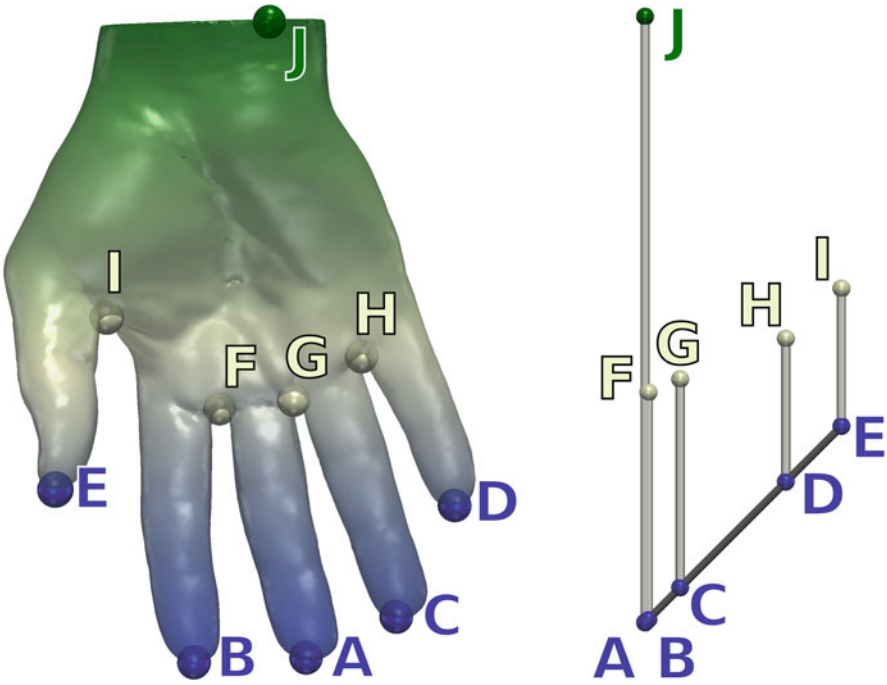


Fig. 2.15 Critical points of a PL scalar field f defined on a PL 3-manifold (left) and its persistence diagram $\mathscr{D}(f)$ (right). In the diagram, each pair of critical points is represented by a vertical bar (white) and its persistence is given by the height of the bar

dimensional simplicial complex that embeds each pair in $\mathbb{R}^2$ by using its *birth* value as a first component and its *birth* and *death* values as second components. The persistence diagram comes with several interesting properties. In particular, the stability theorem [27] states that given two PL scalar fields f and g defined on a common domain, the bottleneck distance between their persistence diagrams is bounded by the difference between the two functions with regard to the infinity norm: $d_B(\mathscr{D}(f), \mathscr{D}(g)) \leq ||f - g||_\infty$. Intuitively, this means that given a slight perturbation of a scalar field, its persistence diagram will only slightly vary. This stability result further motivates the usage of the persistence diagram as a stable topological abstraction of a scalar field (used for instance in function comparison).

In practice however, an alternate representation is often preferred to isolate critical point pairs corresponding to important features from these corresponding to noise. The *Persistence Curve*, noted $\mathscr{C}(f)$, is a diagram that plots the number of critical point pairs with persistence higher than a threshold ϵ, as a function of this threshold ϵ. When displayed in logarithmic scale, such curves often exhibit a flat plateau separating features with very low persistence from these of higher persistence (see Fig. 2.13). In practice, such a plateau is instrumental to manually identify a relevant persistence threshold for the user-driven selection of the most important critical points of the field, as illustrated in Fig. 2.13 (bottom). Also, note that extracting the critical point pairs more persistent than a threshold ϵ for increasing values of ϵ yields a hierarchy of sets of critical points, that enables to interactively explore them at multiple scales of importance, as showcased throughout Fig. 2.13.

2.2.3 Reeb Graph

The persistence curve and the persistence diagram provide concise representations of the critical point pairs of a PL scalar field, along with their persistence. However, they do not provide any information related to the adjacency relations of these pairs on the domain. This is the purpose of more advanced topological abstractions, such as the Reeb graph [98].

Definition 2.58 (Reeb Graph)

Let $f : \mathscr{M} \to \mathbb{R}$ be a PL Morse scalar field defined on a compact PL manifold $\mathscr{M}$.
Let $f^{-1}(f(p))_p$ be the contour of f containing the point $p \in \mathscr{M}$.
The Reeb graph $\mathscr{R}(f)$ is a one-dimensional simplicial complex defined as the quotient space on $\mathscr{M} \times \mathbb{R}$ by the equivalence relation $(p_1, f(p_1)) \sim (p_2, f(p_2))$, which holds if:

$$\begin{cases} f(p_1) = f(p_2) \\ p_2 \in \left(f^{-1}(f(p_1))\right)_{p_1} \end{cases}$$

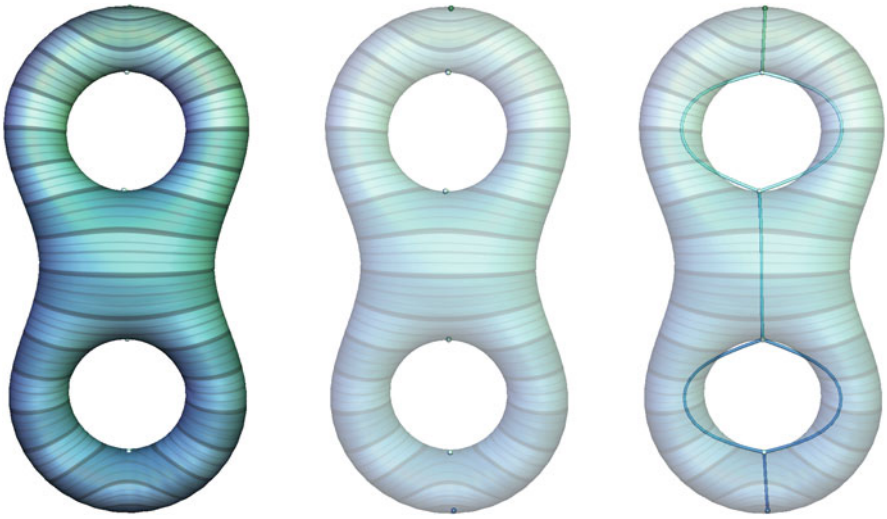


Fig. 2.16 PL Morse scalar field defined on a PL 2-manifold (left and center) and its Reeb graph (right)

The Reeb graph can also be defined alternatively as the *contour retract* of $\mathscr{M}$ under f (a continuous map that retracts each contour to a single point, such that its image is a subset of its domain and its restriction to its image is the identity). Note that f can be decomposed into $f = \psi \circ \phi$, where $\phi : \mathscr{M} \to \mathscr{R}(f)$ is the contour retraction and $\psi : \mathscr{R}(f) \to \mathbb{R}$ is a continuous function that maps points of $\mathscr{R}(f)$ to their f value in $\mathbb{R}$.

Intuitively, as suggested by the previous definition, the Reeb graph continuously contracts each connected component of level sets to a point, yielding a one-dimensional simplicial complex that can be optionally embedded in $\mathbb{R}^3$, as illustrated in Fig. 2.16.

Since the Betti numbers of the level sets change at critical points (in particular β_0 may change), the Reeb graph has a tight connection with the critical points of the function. In particular, branching occurs when $\beta_0(f^{-1}(i))$ changes as i evolves, as further detailed below:

Property 2.12 (Images Through ϕ [98]) Let $\mathscr{R}(f)$ be the Reeb graph of a PL Morse scalar field $f = \psi \circ \phi$ defined on a PL d-manifold. Let the valence of a 0-simplex $v \in \mathscr{R}(f)$ be the number of 1-simplices in its star $St(v)$.

- All regular points of f map through ϕ to a point in the interior of a 1-simplex of $\mathscr{R}(f)$.
- All critical points of index 0 or d (all extrema of f) map through ϕ to 0-simplices of $\mathscr{R}(f)$ of valence 1.

- If $d = 2$, all critical points of index 1 (all saddles of f) map through ϕ to 0-simplices of $\mathscr{R}(f)$ of valence 2, 3 or 4.
- If $d \geq 3$, all critical points of index 1 or $(d-1)$ (subsets of saddles of f) map through ϕ to 0-simplices of $\mathscr{R}(f)$ of valence 2 or 3.
- If $d > 3$, all critical points of index different from 0, 1, $(d-1)$ or d map through ϕ to 0-simplices of $\mathscr{R}(f)$ of valence 2.

The original description of the Reeb graph [98] (including the above properties) implies that all critical points of f map to 0-simplices of $\mathscr{R}(f)$. In more contemporary descriptions, two 1-simplices sharing a valence-2 0-simplex as a face are considered to form only one 1-simplex. Therefore, in the contemporary vision of the Reeb graph, extrema map to valence-1 vertices while only the saddles where $\beta_0(f^{-1}(i))$ evolves map to vertices of higher valence. In particular, saddles where $\beta_0(f^{-1}(i))$ decreases (respectively increases) are called *join* (respectively *split*) saddles. For PL 3-manifolds, join (respectively split) saddles have index 1 (respectively 2). In this vision, not all 1 and 2-saddles map to vertices of the Reeb graph.

Since the Reeb graph has a tight connection with the critical points of its scalar field, some of the properties of PL Morse scalar fields translate in the Reeb graph setting.

Definition 2.59 (Loops in a Reeb Graph) Let $\mathscr{R}(f)$ be the Reeb graph of a PL Morse scalar field f defined on a PL d-manifold $\mathscr{M}$. Each independent cycle of $\mathscr{R}(f)$ is called a *loop*. The number of loops of a Reeb graph is noted $l(\mathscr{R}(f))$.

The two saddles of each loop with the highest and lowest ψ values are usually called *loop saddles*.

Property 2.13 (Loops in a Reeb Graph) Let $\mathscr{R}(f)$ be the Reeb graph of a PL Morse scalar field f defined on a compact PL d-manifold $\mathscr{M}$. $l(\mathscr{R}(f))$ is bounded by $\beta_1(\mathscr{M})$:

$$l(\mathscr{R}(f)) \leq \beta_1(\mathscr{M}).$$

As discussed by Cole-McLaughlin et al. [28], this property follows from the fact that the construction of the Reeb graph can lead to the removal of 1-cycles of $\mathscr{M}$, but not to the creation of new ones. In the case of PL 2-manifolds, tighter bounds have be shown [28].

Property 2.14 (Loops in a Reeb Graph on PL 2-Manifolds) Let $\mathscr{R}(f)$ be the Reeb graph of a PL Morse scalar field f defined on a PL 2-manifold $\mathscr{M}$. Let $b(\mathscr{M})$ be the number of boundary components of $\mathscr{M}$ and $g(\mathscr{M})$ its genus. The number of loops of $\mathscr{R}(f)$ can be described as follows:

- If $\mathscr{M}$ is orientable (admits a non-null and continuous normal vector field):
  - if $b(\mathscr{M}) = 0$, then $l(\mathscr{R}(f)) = g(\mathscr{M})$;
  - otherwise $g(\mathscr{M}) \leq l(\mathscr{R}(f)) \leq 2g(\mathscr{M}) + b(\mathscr{M}) - 1$

- otherwise,
  - if $b(\mathscr{M}) = 0$, then $0 \leq l(\mathscr{R}(f)) \leq g(\mathscr{M})/2$
  - otherwise $0 \leq l(\mathscr{R}(f)) \leq g(\mathscr{M}) + b(\mathscr{M}) - 1$

Figure 2.16 illustrates this property for a closed and orientable 2-manifold of genus 2 (here $\chi(\mathscr{M}) = 2-2g(\mathscr{M})-b(\mathscr{M})$) and a Reeb graph having consequently two loops.

It follows from Property 2.13 that the Reeb graph of a PL Morse scalar field defined on a simply-connected PL d-manifold $\mathscr{M}$ is loop free ($\beta_1(\mathscr{M}) = 0$). In this specific case, the Reeb graph is usually called a *Contour tree* and is noted $\mathscr{T}(f)$. Variants of the Reeb graph, called the *Join* (respectively *Split*) *trees* are defined similarly by contracting connected components of sub (respectively sur) level sets to points (instead of level sets) and are noted $\mathscr{J}(f)$ (respectively $\mathscr{S}(f)$). Note that the join (respectively split) tree of a PL Morse scalar field admitting only one maximum (respectively minimum) is equal to its contour tree.

Since the Reeb graph is a one-dimensional simplicial complex, a filtration of $\psi : \mathscr{R}(f) \rightarrow \mathbb{R}$ can be considered and therefore persistent homology concepts (previous subsection) readily apply to the Reeb graph without specialization. In particular, one can directly read the $(0, 1)$ critical point pairs of f (minima and join saddles) from the join tree by removing its 1-simplices attached to a minimum, one by one in order of their persistence, as illustrated in Fig. 2.17 (top). A similar strategy applied to the split tree enumerates all $(d-1, d)$ critical point pairs ($(d-1)$-saddles and maxima). The time-efficient enumeration of these types of critical point pairs is one

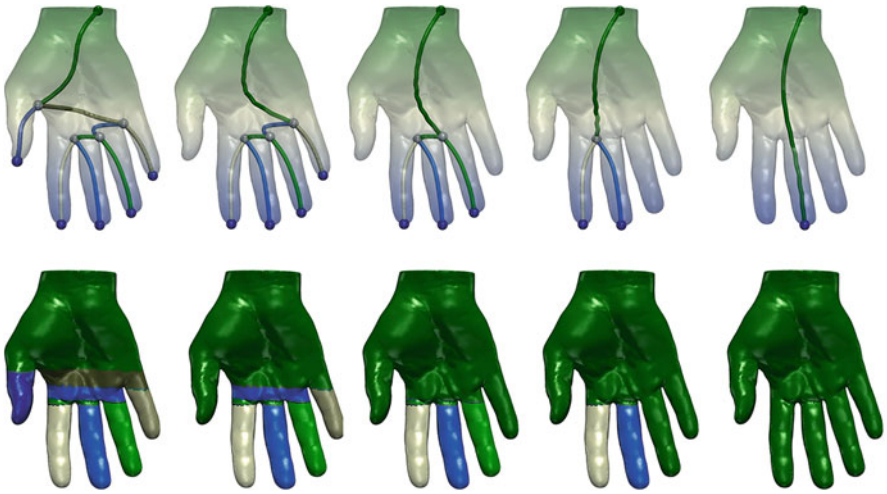


Fig. 2.17 Hierarchy of Reeb graphs obtained by repeated persistence-driven removal of their 1-simplices (top) and hierarchy of data segmentations (bottom) obtained by considering the pre-image by ϕ of each 1-simplex of the Reeb graphs (matching colors)

of the primary applications of the Reeb graph in practice. Note that simplifying in such a way the Reeb graph, similarly to the critical points in the previous subsection, yields a hierarchy of Reeb graphs (for each of which a corresponding, simplified, PL Morse scalar field is guaranteed to exist), that enables to interactively explore it at multiple scales of importance, as showcased throughout Fig. 2.17 (top).

Finally, note that the pre-image by ϕ of $\mathscr{R}(f)$ induces a complete partition of $\mathscr{M}$. In particular, the pre-image of a 1-simplex $\sigma \in \mathscr{R}(f)$ is guaranteed by construction to be connected (ϕ^{-1} is implemented in practice by marking during the construction of $\mathscr{R}(f)$ each vertex with the identifier of the 1-simplex where it maps to). This latter property is instrumental in various tasks in scientific visualization, including the efficient indexing of contours (for fast level set extraction) or the automatic and interactive data segmentation into regions of interest (especially when feature boundaries coincide with level sets). This latter capability of the Reeb graph can be nicely combined with persistent homology concepts, yielding hierarchies of data segmentations, as illustrated in Fig. 2.17 (bottom).

2.2.4 Morse-Smale Complex

As discussed in the previous subsection, in the modern interpretation of the Reeb graph, not all critical points are captured as 0-simplices. Therefore, the Reeb graph only describes the adjacency relations of a sub-set of critical point pairs. To capture such exhaustive adjacency relations, one needs to consider another topological abstractions, called the Morse-Smale complex, that is constructed by considering equivalence classes on integral lines instead of contours (see [59] for further details).

Property 2.15 (Integral Lines) Let $f : \mathscr{M} \to \mathbb{R}$ be a PL Morse scalar field defined on a closed PL d-manifold $\mathscr{M}$. Then, the following properties hold:

- Two integral lines are either disjoint or the same;
- Integral lines cover all of $\mathscr{M}$;
- The origin and the destination of an integral line are critical points of f.

The latter property is particularly interesting. It means that an integral line can be characterized by its extremities, which are guaranteed to be critical points of f. Then, one can introduce an equivalence relation that holds if two integral lines share the same extremities. This is the key idea behind the Morse-Smale complex.

Definition 2.60 (Ascending Manifold) Let $f : \mathscr{M} \to \mathbb{R}$ be a PL Morse scalar field defined on a PL d-manifold $\mathscr{M}$. The *ascending* (respectively *descending*) *manifold* of a critical point p of f is the set of points belonging to integral lines whose origin (respectively destination) is p.

Property 2.16 (Ascending Manifolds) Let $f : \mathscr{M} \to \mathbb{R}$ be a PL Morse scalar field defined on a PL d-manifold $\mathscr{M}$. Let p be an index-i critical point of f. The ascending

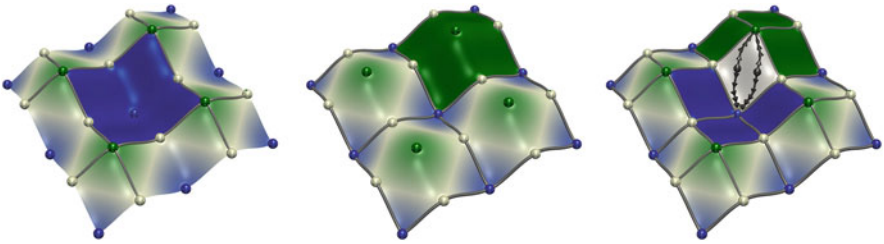


Fig. 2.18 Ascending (left) and descending (center) manifolds and Morse-Smale complex (right) of a PL Morse scalar field f defined on a PL 2-manifold

(respectively descending) manifold of p is an open set of $\mathscr{M}$ of dimension $(d - i)$ (respectively i).

Figure 2.18 illustrates these properties with a PL Morse scalar field defined on a PL 2-manifold $\mathscr{M}$. In particular, the ascending manifold of a minimum is a subset of $\mathscr{M}$ of dimension 2 (shown in dark blue, left). Similarly, the descending manifold of a maximum is also a subset of $\mathscr{M}$ of dimension 2 (show in green, center). For PL 2-manifolds, in both cases, ascending and descending manifolds of saddles have dimension 1 (grey integral lines in both images).

Definition 2.61 (Morse Complex) Let $f : \mathscr{M} \rightarrow \mathbb{R}$ be a PL Morse scalar field defined on a PL d-manifold $\mathscr{M}$. The complex formed by all descending manifolds of f is called the *Morse complex*.

Given this definition, the complex of all ascending manifolds shown on the left of Fig. 2.18 is the Morse complex of $-f$, while the complex of all descending manifolds shown in the center is that of f.

Definition 2.62 (Morse-Smale Function) Let $f : \mathscr{M} \rightarrow \mathbb{R}$ be a PL Morse scalar field defined on a PL d-manifold $\mathscr{M}$. f is a *Morse-Smale* function if the ascending and descending manifolds only intersect transversally.

Intuitively, the transversal intersection condition implies that ascending and descending manifolds are not parallel at their intersection (this condition is enforced in practice through local remeshing). This implies that when these intersect exactly at one point, such a point is critical. This also implies that given an integral line, the index of its origin is smaller than that of its destination.

Definition 2.63 (Morse-Smale Complex) Let $f : \mathscr{M} \rightarrow \mathbb{R}$ be a Morse-Smale scalar field defined on a PL d-manifold $\mathscr{M}$. The complex formed by the intersection of the Morse complex of f and that of $-f$ is called the *Morse-Smale* complex and noted $\mathscr{MS}(f)$.

Figure 2.18 (right) illustrates such an intersection. As shown with the white region, all integral lines (black curves) of a given cell of the complex (irrespectively of its dimension) share the same origin and destination. Note that one can derive a simplicial decomposition of the Morse-Smale complex by subdividing each d-dimensional cell into valid d-simplices. By construction, all the critical points of f

will therefore map to distinct 0-simplices of such a decomposition, since the Morse-Smale complex captures all ascending and descending manifolds (and therefore all critical points). Then, as for the Reeb graph (previous subsection), persistent homology concepts also readily apply to the simplicial decomposition of the Morse-Smale complex and simplifying it for increasing values of persistence also yields a hierarchy that enables to interactively explore the Morse-Smale complex at multiple scales of importance.

Finally, by construction, the Morse-Smale complex provides a partition of the domain that is instrumental in scientific visualization, especially when features or their boundaries coincide with the gradient. Alike the Reeb graph, this segmentation capability can be nicely combined with persistent homology concepts, yielding hierarchies of data segmentations, as detailed in the following subsection.

2.3 Algorithms and Applications

In this section, we briefly review the state-of-the art algorithms for computing the topological abstractions described above, and some of their applications are also briefly discussed (see [10, 66] for recent surveys). Reference open source implementations of these algorithms can be found in the Topology ToolKit library [130].

2.3.1 Persistent Homology

Critical points of PL scalar fields are usually extracted with a simple, robust, localized yet globally consistent, and easily parallelizable algorithm that directly implements the definitions presented in Sect. 2.2.1, and which derive from a seminal paper by Banchoff [7].

Critical point pair extraction as well as their persistence evaluation (Sect. 2.2.2) are usually implemented through sparse matrix reduction [44] with an algorithm with $O(n^3)$ worst case time complexity (where n is the number of simplices). Note however, that for the purpose of feature selection, only extrema-saddle pairs seem to have a practical interest and these can be computed more efficiently with the Reeb graph as described previously.

Persistence diagrams (which encode all critical point pairs along with their persistence) have been widely used for the purpose of function comparison, especially for high-dimensional domains where more advanced topological abstractions are more difficult to compute and simplify (see for instance [23, 53] and [99]).

Cohen-Steiner et al. [27] showed that the bottleneck distance between the persistence diagrams of two PL scalar fields f and g computed on a common domain was bounded by the distance between the two functions with regard to the infinity norm ($||f - g||_\infty$). This result raises the reciprocal question: *given a persistence*

diagram $\mathscr{D}(f)$ where all pairs less persistent than a threshold ϵ have been removed (noted $\mathscr{D}(g)$), can we compute a function g sufficiently close from f that admits $\mathscr{D}(g)$ as persistence diagram? This question has major practical implications since the time complexity of the algorithms for the construction or processing of topological abstractions is often dictated by the number of critical points in the input scalar field. Often in practice, it is possible to easily discriminate critical points that are not relevant application-wise. Therefore, there exists an applicative interest for an efficient pre-processing of an input scalar field, that would minimally perturb it to remove a given set of critical points. This question has first been addressed in the case of PL scalar fields defined on PL 2-manifolds by Edelsbrunner et al. [47], who showed that such a function g existed and that its difference to the input was bounded by ϵ: $||f-g||_\infty \leq \epsilon$. These authors also provided an algorithm to compute it. However this algorithm is complicated and difficult to implement. Moreover, as persistence pairs are processed in order of their highest extremity, the same vertices are swept several times when cancelable persistence pairs are nested. Attali et al. [3] and Bauer et al. [8] presented independently a similar approach to this problem for filtrations and Discrete Morse functions. However, converting the output of these algorithms to PL scalar fields (which is the standard scalar field representation for many applications) requires an important subdivision of the domain. Also, these approaches only deal with closed surfaces. We introduced in 2012 a general algorithm [118] for the topological simplification of PL scalar fields on closed or open PL 2-manifolds, capable of removing arbitrary critical point pairs (not necessarily the least persistent), which enables the usage of application-dependent metrics for feature selection. Thanks to its speed, ease of implementation, robustness and generality, we consider this algorithm as the reference for the problem of topological simplification of scalar data on surfaces. A survey on the concepts of Persistent homology and their applications can be found in [41].

2.3.2 Reeb Graph

Reeb graphs have been introduced in Computer Science independently by Boyell and Ruston [13] (for simply connected surfaces) and by Shinagawa et al. [102].

Efficient algorithms for their computation have first been investigated in the simpler cases of simply connected domains (for which the Reeb graph is called the contour tree). Time-efficient algorithms have first been introduced for 2D domains [131], then 3D domains [110] and last for domains of arbitrary dimension, with an algorithm [19] with optimal time complexity: $O(|\sigma_0|log(|\sigma_0|) + (\sum_{i=0}^{i=1}|\sigma_i|)\alpha(\sum_{i=0}^{i=1}|\sigma_i|))$, where $|\sigma_i|$ is the number of i-simplices in the domain and $\alpha()$ is an extremely slowly growing function (i.e. the inverse of the Ackermann function). In particular, the latter algorithm is considered by the community as the reference for the problem of contour tree computation thanks to its optimal time complexity, its ease of implementation, its robustness, and its practical performances. Sequential [35] and multi-threaded [57, 58] implementations are also

available. This algorithm first computes the join and split trees by tracking the merge events of a union-find data-structure, by processing the vertices in ascending (respectively descending) order. Last, the two trees are combined to form the contour tree in a linear pass.

Regarding more general domains, an algorithm [28] has been introduced for PL scalar fields defined on PL 2-manifolds, with optimal time complexity: $O(|\sigma_1|log(|\sigma_1|)$. A more recent, non-optimal algorithm [92] would be recommend instead however, due to its ease of implementation and acceptable performances in practice. This algorithm explicitly constructs the partition induced by the pre-image of ϕ to retrieve the 1-simplices of $\mathscr{R}(f)$, by computing the critical contours of all saddles of f as boundaries. This makes the algorithm output-sensitive but leads to a worst-case complexity of $O(|\sigma_0| \times |\sigma_2|)$.

Regarding higher dimensional (non simply-connected) domains, several attempts have been proposed, especially with the more practical target of PL 3-manifolds in mind. However, many properties of the 2-dimensional domains exploited by the above algorithms do not hold for 3-dimensional domains, making the problem more challenging. Pascucci et al. [91] introduced the first algorithm for the computation of the Reeb graph on domains of arbitrary dimension. Its streaming nature however, while appealing in specific applications, makes it implementation difficult. An open-source implementation, which I wrote in 2009 based on Giorgio Scorzelli's original implementation, is available in the official release of the open-source library the Visualization ToolKit [116]. Doraiswamy and Natarajan [38] extended the quadratic complexity algorithm by Patane et al. [92] from PL 2-manifolds to PL 3-manifolds. We introduced the first practical algorithm [125] (further described in Chap. 3) for the efficient computation of the Reeb graph of PL scalar fields defined on PL 3-manifolds in 2009. Thanks to its speed, it enabled in practice to transfer all of the contour tree based interactive applications to more general non-simply connected domains. We considered this algorithm as the reference for the problem of Reeb graph computation on PL 3-manifolds, until a dimension-independent, optimal time complexity ($O((\sum_{i=0}^{i=2}|\sigma_i|)log(\sum_{i=0}^{i=2}|\sigma_i|))$) algorithm was introduced 3 years later [89].

The contour tree and the Reeb graph have been massively applied in scientific visualization, in particular because of the property that the pre-image by ϕ of a 1-simplex $\sigma \in \mathscr{R}(f)$ is guaranteed by construction to be connected. This enables for instance to extract an optimal number of vertex seeds for optimal time level set extraction: each vertex seed initiates the construction of a contour by breadth-first search traversal of the domain (limiting the traversal to the exact set of simplices projecting on the queried isovalue). This seed extraction process requires to store each 1-simplex of $\mathscr{R}(f)$ in a balanced interval tree [30]. At query time, given an isovalue i, all 1-simplices of $\mathscr{R}(f)$ projecting on i can be efficiently retrieved in $O(|\sigma_1|log(|\sigma_1|))$ steps, where $|\sigma_1|$ is here the number of 1-simplices in $\mathscr{R}(f)$. Further, given a 1-simplex $\sigma \in \mathscr{R}(f)$ that projects on i, a seed vertex can be efficiently extracted if the vertices of the domain projecting to σ through ϕ (let $|\sigma_0'|$ be their number) are stored in a balanced search tree in $O(|\sigma_0'|log(|\sigma_0'|))$ steps. This mechanism can be nicely combined with persistent homology concepts to

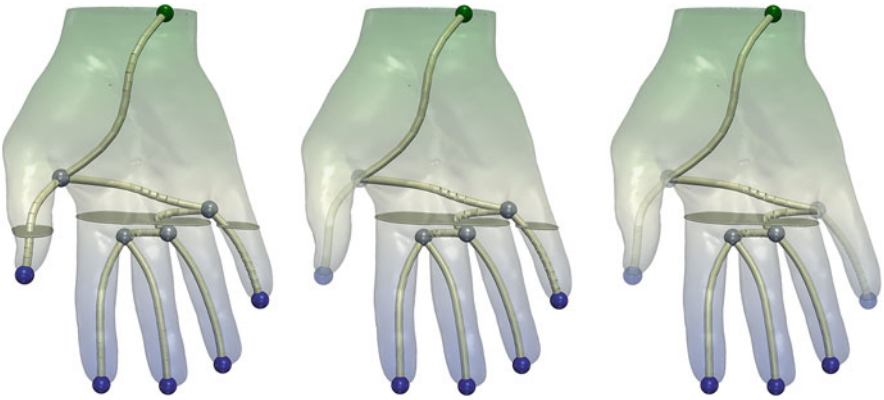


Fig. 2.19 Topological simplification of isosurfaces (white surface, middle). The 1-simplices of $\mathscr{R}(f)$ that are less persistent than an increasing threshold (transparent, from left to right) are not considered for contour seed extraction, yielding a progressive removal of the least prominent contours from the isosurface

interactively simplify isocontours, as illustrated in Fig. 2.19, by only considering the 1-simplices of $\mathscr{R}(f)$ that are more persistent than a threshold ϵ. Variants of this strategy have been presented in the case of the contour tree by van Kreveld et al. [131] and Carr et al. [20]. In particular, the latter approach introduced, as an alternative to persistence, several geometrical measures enabling to filter the 1-simplices of $\mathscr{T}(f)$ according to more application-relevant metrics, which reveals to be of major importance in practice.

The partitioning capabilities of the Reeb graph have also been instrumental in scientific visualization for data segmentation tasks, especially in cases where the boundaries of regions of interest coincide with level sets. In that context, the Reeb graph enables (with the fast isosurface extraction algorithm presented above) to rapidly extract and distinguish each of these boundaries, at multiple scales of importance when combined with persistent homology mechanisms. Chapter 5 discusses two applications of these capabilities to combustion [17] and chemistry [55]. These segmentation capabilities also serve as the basis of more advanced techniques, for the tracking of features over time [105], for the design of transfer functions in volume rendering [135] or the similarity estimation between data features [114].

Apart from scientific visualization, the Reeb graph has also been used as a core data-structure in computer graphics, as discussed in [11, 115, 132].

2.3.3 Morse-Smale Complex

The computation of Morse-Smale complexes was first investigated in the case of PL 2-manifolds. An initial algorithm was introduced by Edelsbrunner et al. [45].

This algorithm constructs for each saddle of f the integral lines originating and terminating at the saddle, yielding the set of ascending and descending 1-manifolds. Ascending and descending 2-manifolds are then retrieved through breadth-first search, by growing 2-dimensional regions until ascending or descending 1-manifolds are attained. Further, the authors apply persistent homology concepts to remove the least persistent critical point pairs and describe a mechanism to update the Morse-Smale complex accordingly (by removing ascending and descending 1-manifolds attached to removed critical points, merging the adjacent 2-manifolds and re-routing their 1-manifold boundaries). Since the number of saddles of f can be proportional to the number of vertices in the domain and since the integral lines can intersect a number of triangles which is proportional to that of the domain, the construction algorithm has a worst case time complexity of $O(|\sigma_0| \times |\sigma_2|)$. This algorithm was latter improved and applied for the first time to scientific visualization by Bremer et al. [14].

The problem of computing the Morse-Smale complex in higher dimensions is far more challenging. First, as the dimension increases, new types of critical points appear (of increasing index), which translates into the apparition of ascending and descending manifolds of higher and higher dimensions. The construction of each of these types of manifolds implies a quadratic term in the runtime complexity. Second, degenerate cases become more challenging to resolve. This is in particular the case of the resolution of the degenerate critical points and the enforcement of the transversal intersection condition. A complicated algorithm has been proposed for PL scalar fields defined on PL 3-manifolds [46] but seems highly challenging to implement and no implementation has been reported so far to the best of our knowledge.

Recently, efficient algorithms have been introduced by walking around the degeneracies of the PL setting and considering a competing formalism, Discrete Morse Theory [51], which comes with many nice combinatorial properties. In this setting for example, critical points occur on simplices of arbitrary dimension and the index of a critical point coincides with the dimension of its simplex. Moreover, degenerate critical points cannot occur by construction. This formalism considers as *Discrete Morse Functions* scalar fields that map each simplex of the domain to a single scalar value, in such a way that for each simplex σ, there exists at most one simplex for which σ is a face with lower function value, and that there exists at most one simplex that is a face of σ with a higher function value. Then a simplex is critical if no such face and no such coface exist. This formalism additionally introduces the notion of *V-path*, used as an analog to PL integral lines, which is a sequence of pairs of simplices of alternating dimensions (d and $(d + 1)$) with descending function values. Then, the notion of *discrete gradient* can be introduced as a pairing of simplices that induces V-paths that are all monotonic and loop-free (Fig. 2.20). Given a discrete gradient, critical points correspond to simplices that belong to no simplex pair. As mentioned before, degenerate critical points cannot occur by construction in this setting. Moreover, ascending and descending manifolds are guaranteed to intersect transversally [59]. Therefore, by construction, this formalism avoids all the degeneracies found in the PL setting which make Morse-Smale

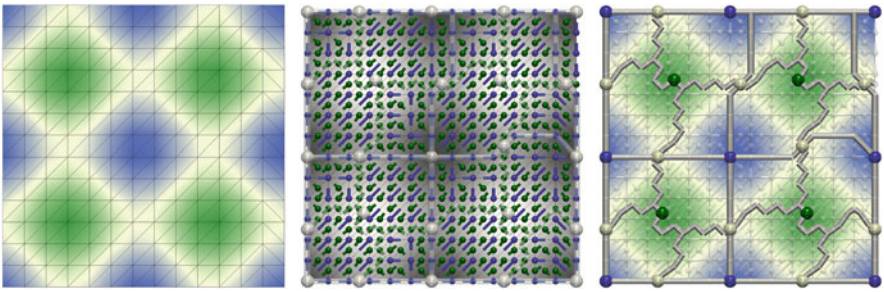


Fig. 2.20 Simple synthetic function (left) and its discrete gradient field (center). Blue glyphs illustrate the pairing of 0-cells (vertices) with 1-cells (edges) while green glyphs show the pairing of 1-cells (edges) with 2-cells (faces). The i-cells that are left unpaired by the discrete gradient field correspond to the critical points of the discrete Morse function, shown with blue ($d = 0$), white ($d = 1$) and green ($d = 2$) spheres (right). *V-paths* (discrete analogs of integral lines) between these critical points represent the separatrices of the Morse-Smale complex (grey cylinders, right). ©2017 IEEE. Reprinted, with permission, from [130]

complex computation challenging. Thus, several efficient algorithms have been proposed for the computation of a discrete gradient from a PL scalar field as well as the construction and simplification of the Morse-Smale complex [61, 100, 136], including a shared-memory parallel algorithm [104] whose implementation has been released in open-source [103]. Note that this discrete formalism has been described here in the context of PL manifolds for consistency, but it readily applies to arbitrary cellular complexes. This is another factor that motivates its popularity, as regular grids no longer need to undergo simplicial subdivisions in this framework.

The combinatorial consistency of the Discrete Morse Theory setting comes however with a price in terms of geometrical accuracy: in particular, locally, V-paths have to follow the simplices of the domain and therefore do not match the integral lines induced by any interpolant. Gyulassy et al. [62] addressed this issue by introducing a novel, probabilistic, discrete gradient construction algorithm whose V-paths are shown to converge to integral lines as the domain sampling increases. We recently improved this approach to avoid the need for domain re-sampling by introducing a discrete gradient construction algorithm that conforms to input constraints [63], as further discussed in Chap. 4. When used with integral lines computed through numerical approximation, this gradient construction algorithm yields Morse-Smale complexes whose manifolds better align with the gradient induced by the domain interpolant.

Morse-Smale complexes have been a popular topological abstraction for data analysis and visualization, especially for data segmentation tasks in applications where features of interest (or their boundaries) coincide with the gradient. Then such features can be efficiently captured at multiple scales of importance (thanks to persistent homology mechanisms) by considering the cells of the Morse-Smale complex. This overall strategy has been applied with tailored analysis algorithms to various applications, including the analysis of the Rayleigh-Taylor instability

[78], vortical structures [72], porous media [60], cosmology [106], combustion [64], computational fluid dynamics [24, 50, 80] or chemistry [55] as further discussed in Chap. 5. A recent survey [33] provides further details regarding the construction, simplification and application of the Morse-Smale complex.

Chapter 3
Abstraction

The previous chapter introduced the theoretical background required for the reading of this book. In particular, it introduced several topological abstractions that play a fundamental role in scientific visualization for feature extraction and data segmentation. In this chapter, we further detail examples of reference algorithms for the robust and efficient computation of topological abstractions. First, we focus on the case of PL scalar fields defined on PL 2-manifolds (with a pre-processing algorithm that speedups subsequent topological abstraction computations), then on the case of PL scalar fields defined on PL 3-manifolds (with a fast Reeb graph computation algorithm).

3.1 Efficient Topological Simplification of Scalar Fields

As described in the previous chapter, the computation of the Morse-Smale complex has a quadratic time complexity. This is also the case for popular Reeb graphs algorithms, such as the approach by Patane et al. [92].

However, in practice, it is often easy to discriminate critical points that are non relevant application-wise. Therefore, if one could perturb the input such that such the non-relevant critical points no longer exist, the subsequent computation of topological abstraction would be greatly accelerated. Moreover, while existing schemes for the simplification of topological abstractions produce multi-resolution representations of the abstractions (previous chapter), they do not perform an actual simplification of the underlying scalar field. This can often be useful for further analysis. Finally, in contexts such as scalar field design, it is desirable to obtain a simplified version of the input field directly without having to compute a computationally expensive topological abstraction.

J. Tierny, *Topological Data Analysis for Scientific Visualization*,
Mathematics and Visualization, https://doi.org/10.1007/978-3-319-71507-0_3

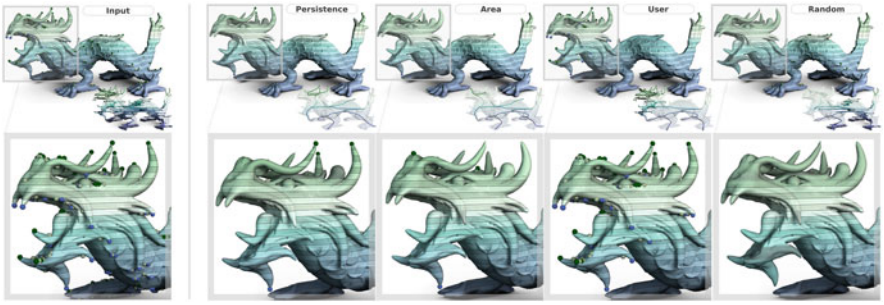


Fig. 3.1 Given an input scalar field f (left), our combinatorial algorithm generates a simplified function g that provably admits only critical points from a constrained subset of the singularities of f. Our approach is completely oblivious to the employed feature selection strategy, while guaranteeing a small distance $||f - g||_\infty$ for data-fitting purpose. Thus it supports application-dependent simplification scenarios such as the removal of singularities based on local geometrical measures, interactive user selection or even random selection. The topology of the resulting field is summarized with the inset Reeb graphs for illustration purpose. ©2012 IEEE. Reprinted, with permission, from [118]

In this section, we present a combinatorial algorithm for the general simplification of scalar fields on surfaces [118]. This algorithm is simple, fast in practice, and more general than previous techniques. Given a scalar field f, our algorithm generates a simplified function g that provably admits only critical points from a constrained subset of the singularities of f, while guaranteeing a small distance $||f - g||_\infty$ for data-fitting purpose. In contrast to previous combinatorial approaches, our algorithm is oblivious to the strategy used for selecting features of interest and allows critical points to be removed arbitrarily (Fig. 3.1). In the special case where topological persistence is used as a feature identification criteria, our algorithm generates a standard ϵ-simplification [47]. The algorithm is simple to implement, handles surfaces with or without boundary, and is robust to the presence of multi-saddles (the input is not restricted to true Morse functions). Extensive experiments show the generality of our algorithm as well as its high performance. In particular, the iterative nature of the approach could require a large number of passes but in practice we have not found examples requiring more than five iterations (normally only two are needed). For this reason the experimental results show an $O(n\ log(n))$ practical performance. To demonstrate the use of our approach, we present applications in terrain simplification as well as the acceleration of topological abstraction computation.

This section further details the following aspects:

1. *Approach:* An approach for the topological simplification of scalar fields that does not rely on persistent homology. It yields a simpler, more intuitive, and general setting. We enumerate the critical points that are *non-removable* because of the topology of the domain. We consequently derive a strategy that supports

the suppression of arbitrary *removable* critical points of the field. This enables the development of a more general simplification framework than previous approaches, for which ϵ-simplification is a special case.

2. *Algorithm:* An iterative, combinatorial simplification algorithm which is very simple to implement (only a few dozens of lines of C++ code). Given a set of user constraints on the extrema of the output function, our algorithm automatically identifies and removes the optimal set of saddles with regard to $||f-g||_\infty$, hence guaranteeing a small distance between the input and the output. In contrast to previous approaches, our technique works directly on PL scalar field representations, is robust to multi-saddles, and handles surfaces with or without boundary. The algorithm uses no computationally expensive topological abstraction, such as a Morse-Smale complex or even a contour tree; hence it is very fast in practice. Our extensive experiments on approximated worst-case scenarios show that this iterative algorithm rarely takes more than two iterations to converge.

3.1.1 Preliminaries

In the following, we consider a PL Morse scalar field $f : \mathcal{M} \rightarrow \mathbb{R}$ defined on an orientable PL 2-manifold $\mathcal{M}$.

General Simplification of Scalar Fields on Surfaces

Definition 3.1 *(General Topological Simplification)* Given a PL scalar field $f : \mathcal{M} \rightarrow \mathbb{R}$ with its set of critical points C_f, we call a *general simplification* of f a PL scalar field $g : \mathcal{M} \rightarrow \mathbb{R}$ such that the critical points of g form a sub-set of C_f: $C_g \subseteq C_f$ (with identical indices and locations).

It is often additionally desired that $||f - g||_\infty$ is minimized for data-fitting purpose. In other words, a general simplification consists in constructing a close variant of the input field f from which a set of critical points has been removed. We describe the possible removals:

Closed Surfaces

The Morse-Euler relation (Property 2.11) defines a dependency between the number of critical points of f and $\chi(\mathcal{M})$, where C_f^i is the set of critical points of f of index i:

$$\chi(\mathcal{M}) = \sum_{i \in \{0,1,2\}} (-1)^i |C_f^i| \; = \; \#_{min(f)} - \#_{saddles(f)} + \#_{max(f)} \tag{3.1}$$

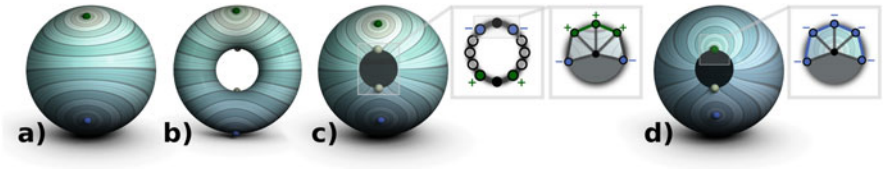


Fig. 3.2 Non removable critical points: (**a**) A global minimum and a global maximum have to be maintained for the field not to be constant. (**b**) 2 $g(\mathscr{M})$ saddles cannot be removed. Each boundary component has 2 non-removable global *stratified* extrema, which turn into non-removable saddles (**c**) or (possibly) *exchangeable* extrema (**d**). ©2012 IEEE. Reprinted, with permission, from [118]

It follows that removing only one extremum, such that the total number of critical points strictly decreases, implies the removal of one saddle (to maintain $\chi(\mathscr{M})$ invariant) and reciprocally. In other words, the removal of the saddles of f are dependent on the removal of its extrema. Certain saddles of f cannot be removed:

$$\chi(\mathscr{M}) = 2 - 2g(\mathscr{M}) = \#_{min(f)} - \#_{saddles(f)} + \#_{max(f)} \tag{3.2}$$

It follows that f counts exactly $2g(\mathscr{M})$ non-removable saddles, located along the $g(\mathscr{M})$ handles of the surface (Fig. 3.2b).

Surfaces with Boundary

The above properties are valid for surfaces with boundary. In addition, for each boundary component $\mathscr{B} \subseteq \partial\mathscr{M}$, certain saddles cannot be removed. Let $f_{\mathscr{B}}$ be the restriction of f to $\mathscr{B}$, and $C_{f_{\mathscr{B}}}$ its critical points that we call *stratified* critical points. By construction, $\mathscr{B}$ is a closed PL 1-manifold. Then:

$$\chi(\mathscr{B}) = \sum_{i\in\{0,1\}} (-1)^i |C^i_{f_{\mathscr{B}}}| \; = \; \#_{min(f_{\mathscr{B}})} - \#_{max(f_{\mathscr{B}})} = 0 \tag{3.3}$$

It follows that $\mathscr{B}$ has an even number of stratified critical points. These cannot be regular points of f on $\mathscr{M}$. For instance, if a maximum of $f_{\mathscr{B}}$ is only surrounded on the interior of $\mathscr{M}$ by vertices with higher f values (Fig. 3.2c, right), its lower link is by construction not simply connected; therefore it turns into a saddle of f. If it is only surrounded by vertices with lower f values (Fig. 3.2d, right), then it turns into a maximum of f (otherwise it is a multi-saddle but f is assumed so far to have only simple saddles). The symmetric property holds for the minima of $f_{\mathscr{B}}$. Since f is required to admit distinct values for each vertex, $f_{\mathscr{B}}$ admits a pair of global stratified extrema (Fig. 3.2c, middle). Consequently, each boundary component of $\mathscr{M}$ necessarily has a pair of critical points of f. If these two points are extrema, they can be removed only if they are not the only extrema of f, leaving a necessary saddle in place in exchange. Otherwise, these necessary critical points are non-

removable saddles of f (Fig. 3.2c, left). In conclusion, the removal of the saddles of f is completely dependent on the removal of its extrema, and for particular cases (summarized in Fig. 3.2) certain critical points are non-removable.

Surface Scalar Field Constrained Topology

As discussed in the previous sub-section, the removal of the saddles of f is dependent on the removal of its extrema. Then, given the sets of input constraints C_g^0 and C_g^2 (the extrema of g), the target of general simplification is to constrain the topology of the level lines of f such that the output field g is close to f and admits as saddles a valid subset of C_f^1 (such that the Morse-Euler relation still holds, Eq. (3.1)). To constrain the topology of the level lines $f^{-1}(i)$ of f, our strategy is to constrain the topology of both the sub- and sur-level sets of f ($\mathscr{L}^-(i)$ and $\mathscr{L}^+(i)$ respectively), which is more practical to achieve. We show in the following that, for surfaces, controlling the connectivity only of $\mathscr{L}^-(i)$ and $\mathscr{L}^+(i)$ is sufficient to enforce the removal (or the preservation) of the removable critical points of f.

Let $f^- : \mathscr{M} \to \mathbb{R}$ be a PL Morse scalar field with only one maximum and several minima (Fig. 3.3). Since f^- has only one maximum, $\beta_0(\mathscr{L}^+(i)) = 1$ for all the i values under the maximum (each vertex of $\mathscr{L}^+(i)$ has a non-empty upper link and thus admits a connected path to the maximum).

Minima

A minimum at isovalue i has an empty lower link; then there exists no connected path on $\mathscr{L}^-(i)$ linking this minimum to other lower minima. Thus, as i changes continuously in $\mathbb{R}$, when passing through a minimum of f^-, a new connected component of $\mathscr{L}^-(i)$ has to be created and $\beta_0(\mathscr{L}^-(i))$ increases (Fig. 3.3b).

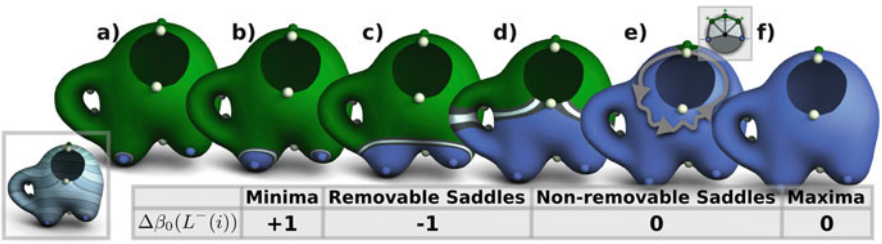


Fig. 3.3 Given a Morse function f^- admitting one maximum and several minima (left inset), the number of connected components of the sub-level set (in blue) increases when passing local minima (**a**, **b**), decreases when passing a removable saddle (**c**) and does not change when passing the non-removable saddles (**d**, **e**) and the maximum (**f**). ©2012 IEEE. Reprinted, with permission, from [118]

Regular Vertices

The lower link of a regular vertex at isovalue i is made of one connected component. Then, *(a)* it cannot merge distinct components of $\mathscr{L}^-(i)$ and *(b)* there exists connected paths on $\mathscr{L}^-(i)$ linking it to lower minima. Thus, passing through a regular point does neither *(a)* decrease nor *(b)* increase $\beta_0(\mathscr{L}^-(i))$.

Interior Saddles

By construction, the lower link of a simple saddle on the interior of $\mathscr{M}$ is made of two connected components (Fig. 2.12c). These components can either be linked to *(a)* the same or to *(b)* distinct connected components of $\mathscr{L}^-(i)$. In the latter case *(b)*, $\beta_0(\mathscr{L}^-(i))$ decreases when passing the saddle. In the former case *(a)*, neither $\beta_0(\mathscr{L}^+(i))$ nor $\beta_0(\mathscr{L}^-(i))$ changes ($\beta_0(\mathscr{L}^+(i)) = 1$ for all i below the maximum). However, since f^- is Morse, $f^{-1}(i)$ changes its topology at the saddle. In the interior of surfaces, the only possible topological change of $f^{-1}(i)$ is a change of $\beta_0(f^{-1}(i))$. Saddles which change $\beta_0(f^{-1}(i))$ while preserving $\beta_0(\mathscr{L}^+(i))$ and $\beta_0(\mathscr{L}^-(i))$ have been shown to correspond to the saddles opening or closing the loops of the Reeb graph of the function [125] and for surfaces, these loops correspond to the handles of the surface [28]. Thus, the only interior saddles of f^- for which $\beta_0(\mathscr{L}^-(i))$ does not change are the $2g(\mathscr{M})$ non-removable saddles (Figs. 3.2b and 3.3d).

Boundary Saddles

Simple boundary saddles can be classified into two categories. In the first case (*join* saddles), the lower link is made of two components (each lying on the same boundary component $\mathscr{B} \subseteq \partial\mathscr{M}$) and the upper link of one (Fig. 3.3e). The second case (*split* saddles) is symmetric: the lower link is made of one component and the upper link is made of two components on the boundary. Since $\beta_0(\mathscr{L}^+(i)) = 1$ (f^- has only one maximum) and since their lower link is made of only one component, neither $\beta_0(\mathscr{L}^+(i))$ nor $\beta_0(\mathscr{L}^-(i))$ changes when passing split saddles. For a join saddle, noted s_j, the two components of the lower link can either be linked to *(a)* the same or to *(b)* distinct connected components of $\mathscr{L}^-(i)$. In the latter case *(b)*, $\beta_0(\mathscr{L}^-(i))$ decreases when passing s_j. Otherwise *(a)*, neither $\beta_0(\mathscr{L}^-(i))$ nor $\beta_0(\mathscr{L}^+(i))$ changes and there exists a connected path on $\mathscr{L}^-(i)$ connecting the two components of the lower link of s_j. This path encloses the boundary component $\mathscr{B}$ on which s_j lies (Fig. 3.3e). This implies that $\{\mathscr{B}-s_j\} \subset \mathscr{L}^-(i)$ since $\mathscr{L}^+(i)$ is made of only one component (the presence of a vertex on $\mathscr{B}$ with a value higher than i would then imply that $\beta_0(\mathscr{L}^+(i)) > 1$). Thus, s_j is the *stratified* global maximum of $f^-_{\mathscr{B}}$. In other words, for each boundary component $\mathscr{B} \subseteq \partial\mathscr{M}$, the only join saddle of f^- for which $\beta_0(\mathscr{L}^-(i))$ does not change is a non-removable saddle (Fig. 3.2).

Maximum

The lower link of a maximum is made of only one connected component. Then, a maximum cannot merge or create a new component of $\mathscr{L}^-(i)$. Thus $\beta_0(\mathscr{L}^-(i))$ does not change when passing through a maximum (Fig. 3.3f).

The symmetric properties hold for a Morse function $f^+ : \mathscr{M} \to \mathbb{R}$ admitting only one minimum and several maxima. Since the input fields are assumed to be Morse functions, when passing through a given critical point, only one topological event can occur at a time on the level set [84], which enables the viewing of each critical point as an instance of the cases reviewed above. Then, in conclusion, the only critical points of the field for which both $\beta_0(\mathscr{L}^-(i))$ and $\beta_0(\mathscr{L}^+(i))$ do not evolve are the non-removable saddles due to the topology of $\mathscr{M}$; for the removable critical points of f, either $\beta_0(\mathscr{L}^-(i))$ or $\beta_0(\mathscr{L}^+(i))$ changes. Thus, it is possible to constrain the topology of the output field g by only controlling the connectivity of $\mathscr{L}^-(i)$ and $\mathscr{L}^+(i)$. The next subsection presents an algorithm exploiting this property.

3.1.2 Algorithm

Our algorithm naturally follows from the properties discussed in the previous subsection. Given the constraints C_g^0 and C_g^2, it iteratively reconstructs the corresponding sub- and sur-level sets, while removing the optimal set of saddles with regard to the L_∞ norm.

Algorithm Description

In the following, we start by describing the algorithm for sub-level set constrained reconstruction.

Sub-level Set Constrained Reconstruction

The pseudo-code for sub-level constrained reconstruction is given in Algorithm 1. To guarantee that the input field admits distinct values on each vertex, symbolic perturbation is used, as described in Sect. 2.1.2. In addition to its scalar value, each vertex v is associated with an integer *offset* (noted $o(v)$) initially set to the actual offset of the vertex in memory. When comparing two vertices (for critical point classification for instance), if these share the same scalar value, their order is disambiguated by their offset o. Algorithm 1 modifies both vertex scalar values and offsets.

input : Scalar field $f : \mathscr{M} \to \mathbb{R}$ (with n scalar (f) and offset (o) values);
input : Set of minima constraints to enforce C_g^0;
output: Scalar field $g : \mathscr{M} \to \mathbb{R}$ with enforced minima in C_g^0.

begin
  // T: set of vertices (self-balancing binary search tree).
  $T \leftarrow \emptyset$;
  // i: time (integer) when a vertex was last processed.
  $i \leftarrow 0$;

  // Initialize T with the minima constraints.
  foreach $m \in C_g^0$ **do** $T \leftarrow \{T + m\}$;

  repeat
    $v \leftarrow argmin_{x \in T} \; f(x)$;
    $T \leftarrow \{T - \{v\}\}$;
    mark v as **visited**;
    // Add unvisited neighbors.
    $T \leftarrow \{T \cup \{v_n \in Lk(v) \mid v_n \text{ is not } \textbf{visited}\}\}$;
    $A[i] \leftarrow v$;
    $i \leftarrow i + 1$;
  until $T = \emptyset$;

  // Scalar and offset value update, for all the vertices.
  // Make the ordering on g (scalar and offset values) consistent with the order of visit.
  for $j \leftarrow 0$ **to** n **do**
    if $j \neq 0 \;\; \&\& \;\; f(A[j]) < g(A[j-1])$ **then**
      $g(A[j]) \leftarrow g(A[j-1])$;
    else
      $g(A[j]) \leftarrow f(A[j])$;
    end
    $o(A[j]) \leftarrow j$;
  end
end

Algorithm 1: Sub-level set constrained reconstruction

The algorithm starts by pushing the minima constraints C_g^0 into a self-balancing binary search tree (noted T) ordered by scalar value and offset. Then, the sub-level sets are iteratively reconstructed, Algorithm 1, one vertex at a time, in a flooding fashion: the unvisited neighbors of the visited vertex are added to T and the vertices of T are uniquely visited in increasing order of scalar value (and offset) until the entire domain is processed. For instance, Fig. 3.4 shows the removal of the lowest minimum of f. Hence, all the minima of f have been added to C_g^0 except for the global minimum. The corresponding flooding is progressively shown in the middle of Fig. 3.4, where the visited and unvisited vertices appear in blue and green respectively. The resulting order of visit of each vertex is stored in the array A. The last step of the algorithm traverses A and updates the vertex scalar values and offsets such that the order defined by the output field is equivalent to the order of visit of the vertices (then the sub-level sets $\mathscr{L}^-(i)$ of the output field indeed correspond to the

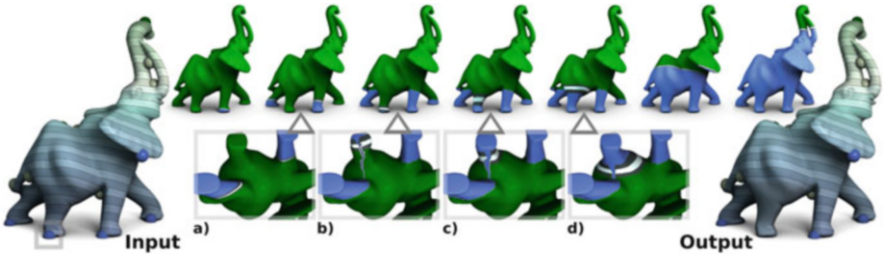


Fig. 3.4 Removing the lowest minimum (small box on the input) by sub-level set constrained reconstruction (in blue). The algorithm enforces the minima constraints (insets a to d) while implicitly removing saddles. ©2012 IEEE. Reprinted, with permission, from [118]

iteratively reconstructed sub-level sets). As shown in Fig. 3.4 (right), this strategy for function value update has the effect of flattening the output in the vicinity of the removed minima.

Since the sub-level sets are grown by adding the vertices of T with smallest function value first, if a connected component of $\mathscr{L}^-(i)$ were to hit a minimum constraint $m \in C_g^0$ before the latter was popped out of T, this would imply that m had a neighbor with lower initial function value, through which the component entered the link of m. This implies that m was not a minimum in the input, which is an invalid constraint. Then, for each $m \in C_g^0$, m is visited before the vertices of its link; hence the vertices of C_g^0 are all minima in the output. The other vertices which do not belong to C_g^0 can only be visited by the iterative growth of $\mathscr{L}^-(i)$, after that one of the vertices of their link has been visited. Hence, their lower link is not empty in the output and the vertices $m \in C_g^0$ are then the only minima of the output (Fig. 3.4).

When passing a saddle s of the input which used to join distinct components of sub-level sets, if only one component of $\mathscr{L}^-(i)$ hits the saddle, this implies that the minimum which created initially the other component does not belong to the constraints C_g^0. Then, the component of $\mathscr{L}^-(i)$ traverses s and continues to grow by visiting vertices with smallest values first, eventually sweeping the removed minimum (Fig. 3.4b–d). Otherwise, s is maintained as a saddle in the output.

Hence, the algorithm implicitly removes saddles given the extrema constraints and guarantees a valid topology of the output. Note that, since the algorithm visits the vertices of T with smallest value first, the saddle removed with one minimum m is the lowest saddle s which used to join the sub-level set component created by m in the input (i.e. the next saddle from m up the join tree): thus the algorithm minimizes $|f(m) - f(s)|$ when removing one saddle s along with one minimum m. Since the update of function value will lift m up to the level of s ($g(m) \leftarrow f(s)$, Algorithm 1), $||f - g||_\infty$ will be equal to $|f(m) - f(s)|$ (for instance, in Fig. 3.5, B is lifted up to the level of E). Thus the algorithm removes the optimal set of saddles with regard to $||f - g||_\infty$, hence guaranteeing a small distance between the input and the output (for the regions where no simplification is needed, the function is unaltered).

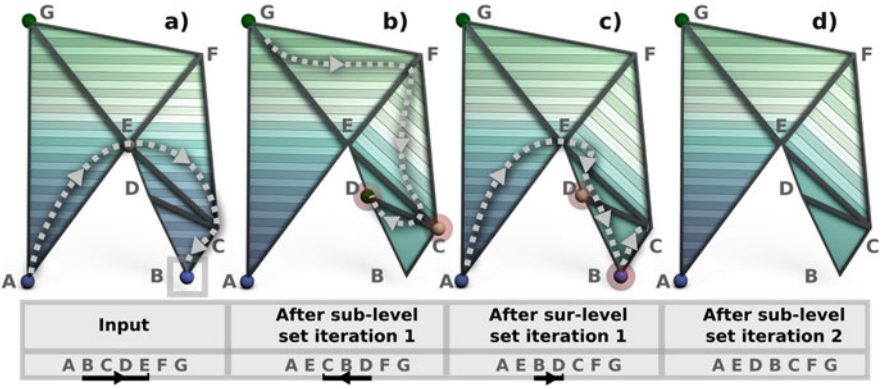


Fig. 3.5 Sub-level set constrained reconstruction can introduce residual maxima (red spheres): in (**a**), all the neighbors of D are visited before it, hence yielding a maximum (**b**). Symmetrically, in (**b**), all the neighbors of B are visited before it, yielding a minimum (**c**). Alternating sub- and sur-level set reconstruction reduces the (offset) function difference between the residual extrema and their corresponding saddle (cf. vertex ordering, bottom), and converges to the removal of all the residuals. ©2012 IEEE. Reprinted, with permission, from [118]

Sur-level Set Constrained Reconstruction

The constraints C_g^2 are enforced with the symmetric algorithm: the vertices of C_g^2 are initially pushed in T and the vertices of T are visited in decreasing order of function value (the update of the function values is also symmetric).

Overall Algorithm

As shown in Fig. 3.5, while Algorithm 1 guarantees that the constraints C_g^0 will be the only minima of the output, it does not guarantee that C_g^2 will be the only maxima. When the reconstructed sub-level set removes a saddle, the algorithm visits the vertices of lowest function value in priority, possibly leaving islands of non-visited vertices behind (Fig. 3.5a), yielding residual maxima in the output function (Fig. 3.5b). The symmetric remark goes for the sur-level set reconstruction regarding minima constraints. To remove these residuals, our overall algorithm successively alternates sub- and sur-level set reconstructions until C_g^0 and C_g^2 are the only extrema. We show in the following that this process converges.

Convergence

Algorithm 1 lifts up the minima to remove, since they are visited after their associated saddle (Fig. 3.5a). Then, when removing a minimum m (B, Fig. 3.5a), residual critical points can only occur higher than the minimum's associated saddle

(E, Fig. 3.5a), but lower than its next vertices in the global vertex ordering (F and G, Fig. 3.5a). Symmetrically, sur-level set reconstruction pushes down the maxima to remove. Then new residual critical points (B and D, Fig. 3.5c) occur lower than the residual saddle of the previous step (C, Fig. 3.5b), but still higher than the original (E, Fig. 3.5a). Alternating sub- and sur-level set reconstruction will keep on reducing the function range where the residual extremum and its corresponding saddle appear. Eventually these will be consecutive in the global vertex ordering (Fig. 3.5, bottom) leaving no more room for new residuals at the next iteration.

From Symbolic to Numerical Perturbation

After convergence, it may be useful to convert the symbolic perturbation (vertex offset $o(v)$) into numerical perturbation, to represent the output field g with a numerical value only for each vertex. The final array A (Algorithm 1) is traversed in increasing order and whenever a vertex is at the same value (or lower) than its predecessor ($g(A[i]) \leq g(A[i-1])$), its function value is increased by an arbitrarily small value ξ: $g(A[i]) \leftarrow g(A[i-1]) + \xi$. This numerical perturbation should be restricted only where it is needed (flat regions of g) to maintain $||f - g||_\infty$ to a small value. For instance, in Fig. 3.5, the vertices D, B and C should all have a final function value in the interval $(f(E), f(F))$. Hence, ξ should be smaller than $\frac{\delta_f}{n}$, where δ_f is the smallest (non-zero) function value absolute difference in the input and n is the number of vertices in $\mathscr{M}$.

Algorithm Properties

Relation to ϵ-Simplification

The implicit pairing performed by our algorithm is compatible with the pairing of critical points based on persistence: given one extremum removal, it pairs a minimum (respectively, a maximum), with its closest saddle up the join tree (respectively, down the split tree). Moreover, given one extremum removal, $||f - g||_\infty$ will be equal to the absolute difference in function value between the extremum and its paired saddle. For instance in Fig. 3.5, B is paired with E and $||f - g||_\infty$ is equal to $|f(B) - f(E)|$. Thus, if the input constraints are selected according to topological persistence (the persistence of the pairs associated with each critical point of C_g^0 and C_g^2 is higher than ϵ), then $||f - g||_\infty \leq \epsilon$.

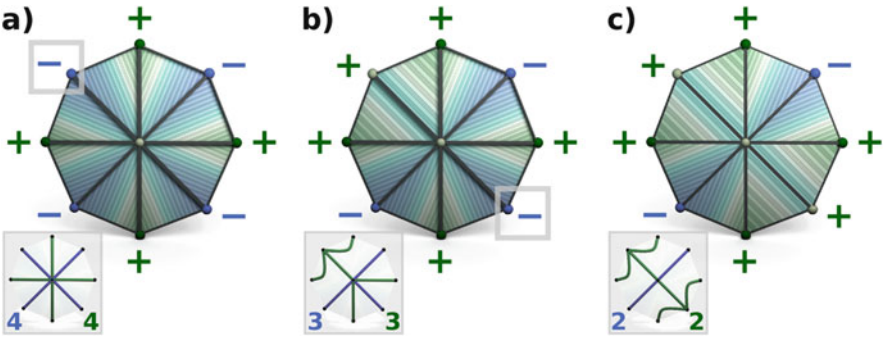


Fig. 3.6 Simplifying a saddle with high multiplicity (**a**): four components of sub- and sur-level sets merge in the saddle (inset Reeb graph: sub- and sur- level sets are marked in blue and green respectively). Removing one extremum (box in (**a**)) decreases the number of components of the lower and upper links by 1 (**b**). Removing other extrema (box in (**b**)) eventually decreases this number to 2, yielding a simple saddle (**c**). ©2012 IEEE. Reprinted, with permission, from [118]

Non-Morse Inputs

Multi-saddles may occur in the input, preventing f from being a Morse function. For these, the lower and upper links can be made of more than two components. Our algorithm handles these degenerate cases with no modification: removing one extremum associated with a multi-saddle will simply decrease the saddle's multiplicity in the output by one (Fig. 3.6).

3.1.3 Results and Discussion

In this section, we present practical results of our algorithm obtained with a C++ implementation on a computer with an i7 CPU (2.93 GHz).

Time Requirement

The algorithm uses no computationally expensive topological abstraction, such as a Morse-Smale complex or even a contour tree. Therefore each iteration is extremely fast in practice. Given a surface with n vertices, inserting and removing a vertex from the self-balancing binary search tree T takes $O(log(n))$ time. Each vertex is uniquely visited. Thus, the complexity of an iteration is $O(n\ log(n))$, irrespectively of the number of critical points to remove. In theory, the number of iterations required for the algorithm to converge could possibly be non-negligible. Given one extremum removal, after each reconstruction, the distance in the global vertex ordering which separates a new residual extremum from its corresponding saddle

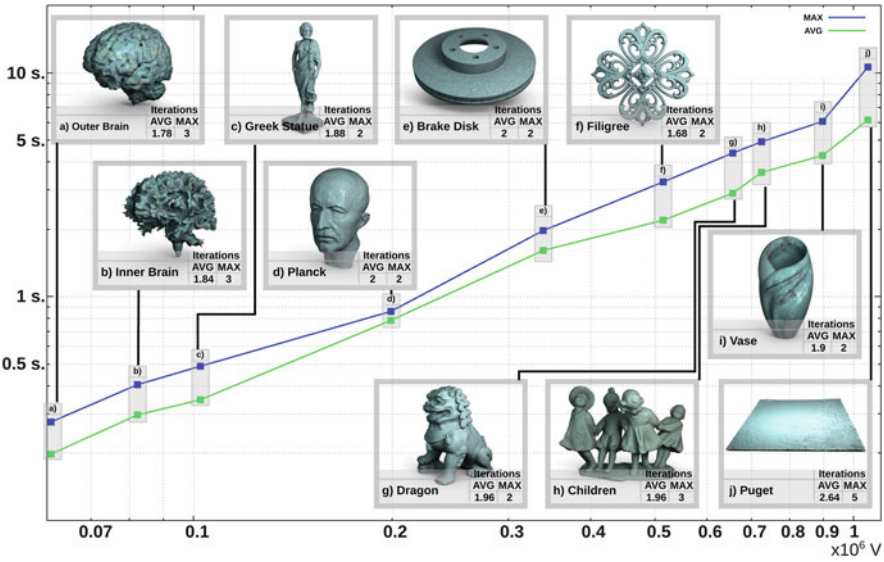


Fig. 3.7 Running times (log scale) for the simplification of random functions, with a random constraint selection (C_g^0, C_g^2), 50 runs per data-set. ©2012 IEEE. Reprinted, with permission, from [118]

decreases at least by one (this distance is illustrated by a black arrow in Fig. 3.5, bottom). As this distance can initially be close to the number of vertices in the mesh, n reconstructions could be required, yielding an $O(n^2\ log(n))$ worst case complexity, irrespectively of the number of removed extrema.

Traditionally, random scalar fields are considered as relevant approximations of worst-case scenarios. Moreover, considering multiple instances increases the chances to get a proper worst-case approximation. We show in Fig. 3.7 the average and maximum running times (in log scale) for the algorithm to achieve convergence on a set of meshes (including examples with high genus, up to 116), for which 50 instances of random fields have been considered and for which the constraints C_g^0 and C_g^2 are random subsets of the fields' extrema. In particular, critical points from C_f^0 are uniquely added to C_g^0 in random order until $|C_g^0|$ is equal to a random fraction of $|C_f^0|$ (the constraint set C_g^2 is constructed similarly). For most data-sets, the average number of required iterations is smaller than or equal to 2 and the maximum number of iterations is never greater than 5. This shows that, from a practical point of view, the number of required iterations is negligible with regard to n, hence yielding $O(n\ log(n))$ practical running time. In our experiments, the algorithm took at most 10.7 s to compute on a mesh with 1 million vertices (6.3 million simplices total).

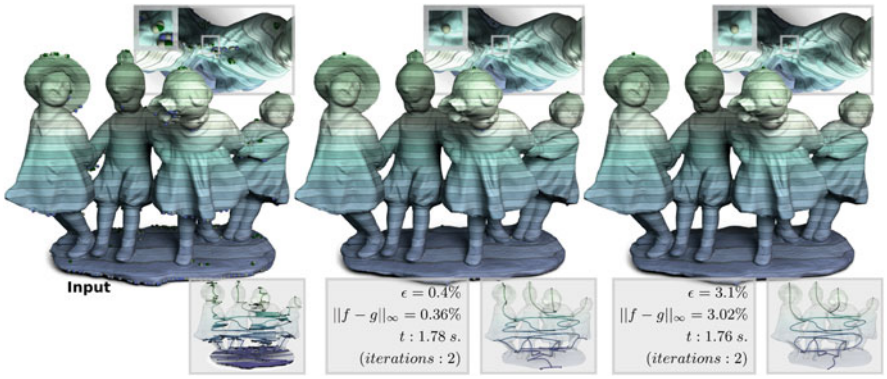


Fig. 3.8 In the special case where the critical points in C_g^0 and C_g^2 are all more persistent than ϵ, our algorithm produces an ϵ-simplification ($||f - g||_\infty \leq \epsilon$). Both the (vertex) position and the function value of the remaining removable critical points are preserved after simplification (top insets), even in the presence of multi-saddles (6 in the input). The topology of the resulting field is summarized with the inset Reeb graph for illustration purpose (input surface: 725k vertices). ©2012 IEEE. Reprinted, with permission, from [118]

Discussion

A unique aspect of our algorithm is its ability, given the constraints C_g^0 and C_g^2, to automatically identify and remove the optimal set of saddles with regard to the L_∞ norm. Moreover, this is accomplished without the need to carry a union-find data-structure unlike previous techniques. Although this data-structure has nearly linear time complexity in theory, practically it could cause slowdowns by a non-negligible constant factor, given the algorithm's low resource requirement. Also, after simplification, the (vertex) position of the remaining critical points is preserved. In the special case where the critical points of C_g^0 and C_g^2 are selected based on topological persistence, the algorithm produces a standard ϵ-simplification, as shown in Fig. 3.8. In contrast to previous approaches, our algorithm directly works on a PL representation of the field, which is more acceptable application-wise. Importantly, it is also more general as critical points can be removed arbitrarily (at the exception of the non-removable critical points summarized in Fig. 3.2), irrespectively of the employed feature-selection strategy.

Limitations

Given our formulation of general simplification, for specific constraint configurations, the value of the remaining critical points may change after simplification. For instance, if the lowest minimum in C_g^0 is initially higher than the highest maximum in C_g^2, the algorithm will change their values to satisfy the topological constraints. Also, when removing only one extremum from the boundary, it will be replaced by

a boundary saddle if it is associated by the algorithm (with regard to the L_∞ norm) with an interior saddle (Fig. 3.6). This is due to the fact that the number of critical points must be even on each boundary component, which may be counter-intuitive from a user's perspective. Moreover, whereas our algorithm removes the optimal set of saddles with regard to $||f - g||_\infty$ given some extrema constraints, our strategy for function value update (which is purely based on *flooding*) does not guarantee a minimization of $||f-g||_\infty$, although the resulting L_∞ norm is close to the theoretical minimum. In the context of persistence driven simplification, Bauer et al. [8] showed that optimality could be reached by a combination of *carving* (i.e. pulling the saddles halfway towards their associated extremum) and *flooding* (i.e. pushing the extrema halfway towards their associated saddle) at the expense of no longer guaranteeing a function value lock on the maintained critical points. By simplifying all the pairs less persistent than ϵ, their approach yields $||f - g||_\infty \leq \frac{\epsilon}{2}$. In contrast, like in [47], our approach yields $||f - g||_\infty \leq \epsilon$ but locks the maintained critical points in terms of function value. In the context of general simplification, the optimal balance between carving and flooding is more difficult to evaluate, as it is no longer a local decision which depends only on the pair of removed critical points (excessive carving could break the enforcement of near-by extrema located in the vicinity of the removed saddles). We addressed this issue and provided an optimal algorithm, derived from this approach but slower in practice and more difficult to implement, in a follow-up work [129]. Like any combinatorial approach, our algorithm provides strong guarantees on the topology of the output at the expense of its geometrical smoothness. Unlike previous combinatorial algorithms using *carving* [3, 8, 47], our algorithm uses *flooding* only. Hence, it does not suffer from the usual artifacts of carving (visible thin paths linking sets of removed critical points), but from those of flooding (flat regions in the output, Fig. 3.4, right). In our experiments, we found that these artifacts were usually little noticeable, unless the features removed by the user span a large region of the domain (Fig. 3.4, right). If smoothness is desired, our approach can be combined with a numerical technique to provide smooth outputs that still benefit from the topological guarantees of our algorithm [118].

Application

In practice, non-relevant critical points can often be easily discriminated, by discarding either certain range values or domain areas. We illustrate this process with terrain simplification, where we show that ad-hoc application driven constraints can be easily incorporated in our simplification scheme to ease further analysis. This is particularly useful if this latter analysis involves the computation of topological abstractions (for segmentation tasks for instance). Then, we show that a pre-simplification of the data based on application driven constraints can greatly accelerate the computation of topological abstractions. Other application examples of this technique are demonstrated further down this book.

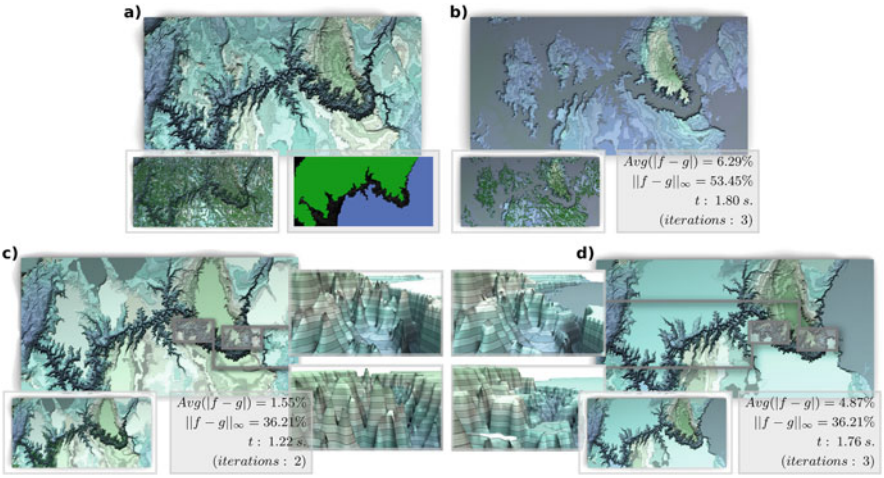


Fig. 3.9 Simplifying the Grand Canyon ((**a**), 500k vertices, 65,526 critical points, bottom) with a location driven feature selection (black: canyon, green: north rim, blue: south rim). (**b**) Maintaining only one minimum and removing all the critical points from the canyon (16,756 critical points remain) emphasizes the topological features of the rims and simulates a massive flooding of the canyon. (**c**) Removing all the critical points from the rims (2296 remaining) emphasizes the topological features inside the canyon. An ϵ-simplification with compatible L_∞ norm (**d**) completely discards the features irrespectively of their location (zoom insets) while yielding a worse average data fitting ($Avg(|f-g|)$). ©2012 IEEE. Reprinted, with permission, from [118]

Topological simplification of terrains can be particularly useful for topographic analysis or water flow simulations as discussed by Bremer et al. [15]. However, the original topological persistence measure can be unsatisfactory for selecting features in such a context, as the characterization of features of interest is application dependent and at times can also be data-set dependent. Figure 3.9 exemplifies this observation on the Grand Canyon elevation data-set, which initially counts 65,526 critical points. The Grand Canyon can be decomposed in three major regions from a geographic point of view: the canyon itself, its north rim and its south rim. These regions (in black, green and blue respectively in Fig. 3.9a) have been initially extracted by segmenting the image along high elevation gradient and interactively completed by the user. Based on this initial decomposition, we present two simplification scenarios. First, in Fig. 3.9b, the topology of only the rims has been emphasized: only the lowest minimum above 53% of the total elevation difference has been maintained, the critical points inside the canyon have been selected for removal and all the maxima on the rims above 53% of elevation have been maintained. In less than 2 s, our algorithm constructs the corresponding *flooded* Grand Canyon, while enforcing the preservation of the selected topological features on the rims. A contrary scenario would consist in emphasizing the peaks inside the canyon. For instance, the result of such a simplification strategy can drive a mesh simplification procedure for an interactive fly-through within the canyon. In that

Table 3.1 Computation times and speedups (bold) for the Reeb graph computation of the simplified data, with the algorithm by Patane et al. [92]

Data-set	$\lvert C_g \rvert$	Simplification time (s)	Reeb graph time (s)	Speedup
Original—Fig. 3.9a	65,526	–	125.723	–
Rims—Fig. 3.9b	16,756	1.783	12.976	**9.69**
Canyon—Fig. 3.9c	2296	1.123	4.432	**28.37**
Persistence—Fig. 3.9d	12	1.702	0.313	**402.28**

case (Fig. 3.9c), only the global minimum has been maintained and all the maxima outside of the canyon have been selected for removal. Note that in comparison, a standard ϵ-simplification with compatible L_∞ norm (Fig. 3.9d) is unsatisfactory as it completely discards the topological features irrespectively of their location (zooms in Fig. 3.9), while yielding a worse average data-fitting ($Avg(|f - g|)$).

Table 3.1 shows the computation times for the Reeb graph construction of the simplified data, with the algorithm by Patane et al. [92] (as described in the previous chapter, the construction of this topological abstraction is instrumental for data segmentation for instance). This table shows that the simplification pre-processing step usually takes much less time than the Reeb graph computation itself, while considerably improving its own performances with up to two orders of magnitude speedups (depending on the number of critical points in g).

Concluding Remarks

This section presented a combinatorial approach for the general simplification of piecewise linear scalar fields on surfaces. By abstracting the approach from the concepts of persistent homology, we believe to have presented a simpler, more intuitive and more general description of scalar field topological simplification. Also, we enumerated all the configurations for which critical points were *non-removable* given the topology of the domain, for surfaces with or with out boundary. From this, we derived a strategy that allows for the arbitrary suppression of the *removable* critical points of the field. We presented a simple iterative algorithm for general topological simplification which, given some constraints on the extrema of the output field, implicitly identifies and removes the optimal set of saddles with regard to the L_∞ norm, hence guaranteeing a small distance $||f - g||_\infty$. Although it is iterative, extensive experiments on approximated worst-case scenarios showed that the algorithm rarely takes more than two iterations to converge. The algorithm uses no computationally expensive topological abstraction, such as a Morse-Smale complex or even a contour tree; hence it is very fast in practice. We demonstrated that it could be used as a pre-processing step to accelerate topological abstraction computation algorithms. In contrast to previous combinatorial approaches, our approach works directly on a PL representation of the field, is robust against multi-saddles, and handles surfaces with or without boundary. Moreover, our approach solves a more general problem, for which ϵ-simplifications have been shown to be a special case.

Thanks to its generality, robustness, good time complexity, ease of implementation and practical performances, we consider this algorithm as the reference for the topological simplification of scalar data on surfaces.

A natural direction for future work is the extension of this approach to volumetric data-sets. As the algorithm works on the 0 and 1-simplices of the domain only, it can be used in principle directly for domains of arbitrary dimension. However, as the dimension of the domain increases, new types of saddle points appear and more subtle topological transitions occur on the level sets (genus changes through 1-2-saddle pairs). Hence, enforcing the connectivity of the sub- and sur-level sets is insufficient for the removal of 1-2-saddle pairs; the genus of the iso-surfaces also needs to be efficiently controlled. However, this problem is NP-hard as recently shown by Attali et al. [4]. This indicates that the design of a practical algorithm with strong topological guarantees is challenging.

3.2 Efficient Reeb Graph Computation for Volumetric Meshes

As discussed in the previous chapter, the Reeb graph is a fundamental data-structure in scientific visualization for contour indexing with applications in fast contour handling, data segmentation and feature extraction. However, as detailed in Sect. 2.3, algorithms with strong quadratic behavior in practice were only available for volumetric meshes and were therefore not practically applicable for use in interactive applications.

In this section, we present a practical algorithm to compute Reeb graphs on volumetric meshes in $\mathbb{R}^3$ (in particular tetrahedral meshes) that runs in practice with comparable efficiency to a contour-tree algorithm, enabling the practical generalization of contour tree based visualization techniques to meshes of arbitrary topology. Our approach is based on the key concept of *loop surgery*, inspired from surgery theory [134]. In particular, we transform the input domain by a sequence of symbolic cuts such that the Reeb graph of the input scalar field defined on the transformed domain is guaranteed to be loop free, and hence computable with efficient contour tree algorithms. Then, some *inverse symbolic cuts* are performed in the topology domain to convert the computed contour tree into the Reeb graph of the original scalar field. We show that these *symbolic cuts* can be computed in an efficient manner, with reasonable computation overhead with respect to contour tree computation. Extensive experiments show that our approach improves state-of-the-art techniques for computing Reeb graphs by several orders of magnitude in terms of running time, with reasonable memory footprint.

This section further describes the following aspects:

1. A procedure called *loop surgery* to reduce the problem of computing a Reeb graph to that of a contour tree. We believe this is an important result, since the join-split algorithm [19] for computing contour trees is well known to have not only optimal theoretical complexity, but also simple and practical implementation.
2. A practical algorithm for computing Reeb graphs with complexity $O(n \log n + N\alpha(N) + g(\mathscr{M}_\partial) \times N_S)$. For practical examples, g, which is equal to the number of handles of the domain, is a small constant, and systematic experiments show a speedup over previous algorithms by several orders of magnitude on average.
3. A proof showing necessary and sufficient conditions for a loop free Reeb graph to be computed correctly by the join-split contour tree algorithm.

3.2.1 Preliminaries

In the following, we consider a PL Morse scalar field $f : \mathscr{M} \to \mathbb{R}$ defined on a PL 3-manifold $\mathscr{M}$ embedded in $\mathbb{R}^3$.

Loops in Reeb Graphs on PL 3-Manifolds in $\mathbb{R}^3$

The key idea of *loop surgery* is to define a sequence of operations that transform $\mathscr{M}$ to $\mathscr{M}'$ with $f' : \mathscr{M}' \to \mathbb{R}$ valued by f such that $\mathscr{R}(f')$ becomes loop-free, and then efficiently computable with contour tree algorithms. Therefore, we carefully characterize the loops (independent cycles) of $\mathscr{R}(f)$ prior to introducing loop surgery.

Since $\mathscr{M}$ is compact and embeddable in $\mathbb{R}^3$, $\partial\mathscr{M}$ is necessarily non-empty, orientable and closed [34] but possibly disconnected.

Let f_∂ be the restriction of f to $\partial\mathscr{M}$. We will first assume that both f and f_∂ are PL Morse functions (degenerate cases will be discussed later). Let $\mathscr{R}(f_\partial)$ be the Reeb graph of f_∂. Then, we have the following relation [28]:

$$l(\mathscr{R}(f_\partial)) = g(\partial\mathscr{M}) \tag{3.4}$$

where $g(\partial\mathscr{M})$ is the sum of the genuses of the boundary components of $\mathscr{M}$. The key property that will allow us to implement loop surgery in an efficient manner is the fact than the topology of $\mathscr{M}$ is closely related to that of $\partial\mathscr{M}$. In particular, the number of handles of $\mathscr{M}$ is the first Betti number, $\beta_1(\mathscr{M})$, which is given by the following relation [34]:

$$\beta_1(\mathscr{M}) = g(\partial\mathscr{M}) \tag{3.5}$$

In simpler words, this relation expresses the fact that each handle of the volume $\mathscr{M}$ corresponds to a tunnel of its boundary surface.

As discussed in [28] in any dimension the construction of the Reeb graph can lead to the removal of 1-cycles, but not the creation new ones. Therefore, the number of loops of $\mathscr{R}(f)$ cannot be greater than the first Betti number of $\mathscr{M}$:

$$l(\mathscr{R}(f)) \leq \beta_1(\mathscr{M}) \tag{3.6}$$

In conclusion, the number of loops of $\mathscr{R}(f)$ cannot be greater than the number of loops of $\mathscr{R}(f_\partial)$:

$$l(\mathscr{R}(f)) \leq \beta_1(\mathscr{M}) = g(\partial\mathscr{M}) = l(\mathscr{R}(f_\partial)) \tag{3.7}$$

A direct extension of the Eq. (3.7) is the following property:

Property 3.1 (Existence of Loops) The existence of loops in $\mathscr{R}(f)$ implies the existence of corresponding tunnels in both $\mathscr{M}$ and $\partial\mathscr{M}$, and thus of corresponding loops in $\mathscr{R}(f_\partial)$. The inverse is not necessarily true.

A consequence of this property is that one can deduce information about the loops of $\mathscr{R}(f)$ by just studying $\mathscr{R}(f_\partial)$. In particular, a loop of $\mathscr{R}(f_\partial)$ will yield a loop in $\mathscr{R}(f)$ if its preimage through ϕ_∂ at a given isovalue contains contours of f_∂ that form the boundary components of distinct contours of f. It follows that each loop $\mathscr{L}$ of $\mathscr{R}(f)$ maps through ψ to an interval $\psi(\mathscr{L}) \subseteq \psi_\partial(\mathscr{L}_\partial)$ being a subset of the image by ψ_∂ of a distinct loop $\mathscr{L}_\partial$ of $\mathscr{R}(f_\partial)$. It follows that the loop saddles of each loop $\mathscr{L} \in \mathscr{R}(f)$ lie on a monotone path on $\mathscr{J}(f)$ (or $\mathscr{S}(f)$), being a subset of the monotone path linking the loop saddles of the corresponding loop $\mathscr{L}_\partial \in \mathscr{R}(f_\partial)$. This latter observation will be instrumental in our algorithm.

Loop Surgery

When f_∂ is PL Morse, there exists a unique pair of loop saddles for each loop, consisting of an "opening" split saddle and a "closing" join saddle. The existence of such pairs is guaranteed by extended persistence [2]. We uniquely associate each loop of $\mathscr{R}(f_\partial)$ with the closing saddle of this pair. Moreover, by Property 3.1, it is possible to associate each loop in $\mathscr{R}(f)$ with the loop saddles of the corresponding loop in $\mathscr{R}(f_\partial)$. Notice that some loop saddles of $\mathscr{R}(f_\partial)$ may not be associated with any loop of $\mathscr{R}(f)$. We will describe in the algorithm section how to identify candidates for the loop saddles of f given those of f_∂.

Loop surgery consists of transforming the domain $\mathscr{M}$ such that $\mathscr{R}(f)$ becomes loop-free. In other words, loop surgery breaks the loops of $\mathscr{R}(f)$ and reflects that transformation on $\mathscr{M}$. For each loop 1-saddle s_i of f with value $f(s_i)$, we define its *cutting surface* S_i as a contour of f (a connected component of isosurface inside the volume) at value $f(s_i) - \epsilon$ such that $f(s_i) - \epsilon$ is a regular value of f, there is no critical value in $[f(s_i) - \epsilon, f(s_i))$ and S_i intersects one of the connected components

Fig. 3.10 Overview of Reeb graph computation based on loop surgery: The input defines a scalar function f (color gradient) on $\mathscr{M}$ (**a**). The Reeb graph of the function restricted to the boundary $\mathscr{R}(f_\partial)$ (**b**) is used to identify *loop saddles* (blue squares). *Cutting surfaces* (**c**) of each loop saddle are used to transform the domain (**d**) to $\mathscr{M}'$. The Reeb graph $\mathscr{R}(f')$ is loop free (**e**). *Inverse cuts* are applied to circled critical point pairs to obtain the Reeb graph $\mathscr{R}(f)$ of the input function. ©2012 IEEE. Reprinted, with permission, from [125]

of the lower star $St^-(s_i)$ in the volume. The symbolic cuts transforming $\mathscr{M}$ into $\mathscr{M}'$ consists of cutting $\mathscr{M}$ along each defined cutting surface, as illustrated in Fig. 3.10c, d.

On $\mathscr{R}(f)$, cutting along a cutting surface at a regular value is equivalent to cutting a 1-simplex of $\mathscr{R}(f)$ and creating a new pair of critical nodes (a minimum and a maximum, as illustrated in Fig. 3.10e). Then, the Reeb graph of the function $f' : \mathscr{M}' \to \mathbb{R}$, f' being the function valued by f after symbolic cuts, is guaranteed to be loop free: all the possible loops have indeed been broken, as shown in Fig. 3.10e.

Once the loop-free Reeb graph $\mathscr{R}(f')$ is computed, *inverse cuts* can be applied in a straightforward manner by removing pairs of minimum and maximum nodes generated by the same cutting surface, and gluing together the corresponding 1-simplices.

Loop Free Reeb Graph Computation

The algorithm presented by Carr et al. [19] computes the contour tree by tracking the joining of sub- and sur-level set components. Traditionally, simply-connectedness of $\mathscr{M}$ has been used as a condition to ensure correctness of this algorithm. However, a Reeb graph can be loop free even when the domain is not simply connected, as shown in Fig. 3.11b. This is especially important, since our loop surgery procedure does not guarantee that the domain is divided into simply connected regions. Therefore, we prove necessary and sufficient conditions for this contour tree algorithm to work. The following property shows when a saddle creates a loop in the Reeb graph.

Property 3.2 (Loop Saddles) Let $f : \mathscr{M} \to \mathbb{R}$ be a PL Morse scalar field and σ be a valence-3 0-simplex of $\mathscr{R}(f)$ being the image by ϕ of a join saddle s (see Fig. 3.12). If the contours joined by the saddle are on the boundary of the same sub-level set component, then there exists a loop in $\mathscr{R}(f)$ for which σ is the highest loop saddle.

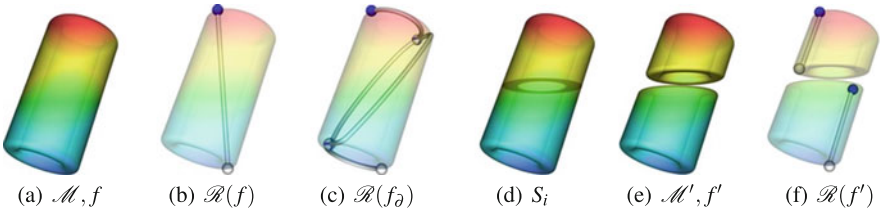


Fig. 3.11 Example where the domain (**a**) is not simply connected and $\mathscr{R}(f)$ is loop free (**b**). The Reeb graph of the boundary, $\mathscr{R}(f_\partial)$, has a loop (**c**), therefore loop surgery is performed (**d**), (**e**). Even after loop surgery, each component of $\mathscr{M}'$ is not necessarily simply connected, but the Reeb graph of each component is loop free (**f**). ©2009 IEEE. Reprinted, with permission, from [125]

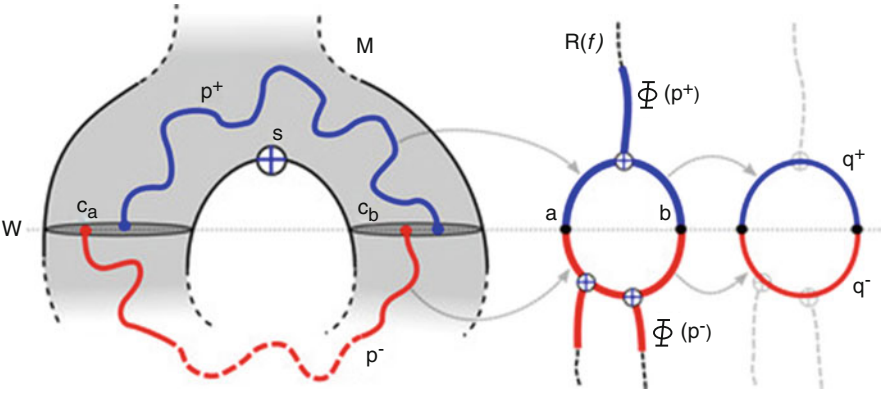


Fig. 3.12 Reference figure for Property 3.2. Paths forming a loop in $\mathscr{M}$ map to connected components forming a loop in $\mathscr{R}(f)$. ©2009 IEEE. Reprinted, with permission, from [125]

Proof Refer to Fig. 3.12 in the following. Given σ, s, f, and $\mathscr{R}(f)$, let ϵ be a small number such that there is no critical value in the range $[f(s)-\epsilon, f(s))$. The existence of such an epsilon is guaranteed because the critical values of a Morse function are distinct. Let $w = f(s) - \epsilon$, and a and b be the points on the two downward 1-simplices of $\mathscr{R}(f)$ from σ such that the value of their corresponding contours c_a and c_b is w, i.e., $f(c_a) = f(c_b) = w$. By construction, s is a join saddle, therefore there exists a path p^+ in $\mathscr{M}$ connecting a point in c_a with a point in c_b such that all its interior is in the sur-level set $\mathscr{L}^+(w)$. ϕ is continuous, therefore a connected component p^+ in $\mathscr{M}$ is mapped to a connected component $\phi(p^+)$ in $\mathscr{R}(f)$, such that a and b are connected in $\mathscr{R}(f)$ by a path $q^+ \subseteq \phi(p^+)$ whose interior is strictly above w. Since by hypothesis c_a and c_b are on the boundary of the same sub-level set component, there exists also a path p^- in $\mathscr{M}$ connecting a point in c_a with a point in c_b such that all its interior is in the sub-level set $\mathscr{L}^-(w)$. Therefore a and b are also connected in $\mathscr{R}(f)$ by a path $q^- \in \phi(p^-)$ whose interior is strictly below w. The two paths $q^+ + q^-$ form a loop.

By applying Property 3.2, we prove necessary and sufficient conditions for computing correct loop free Reeb graphs using tracking of sub- and sur-level sets.

Property 3.3 (Loop Free Reeb Graphs) Let $f : \mathcal{M} \to \mathbb{R}$ be a PL Morse scalar field. Its Reeb graph $\mathcal{R}(f)$ is loop free if and only if every valence-3 0-simplex of $\mathcal{R}(f)$ is the image through ϕ of a saddle of f where distinct components of sub- or sur-level sets join.

Proof First we prove that if a Reeb graph $\mathcal{R}(f)$ is loop free, every valence-3 0-simplex σ is the image through ϕ of a saddle s of f that joins distinct components of sub- or sur-level sets. Assume that there exists a valence-3 0-simplex that joins contours that are on the boundary of the same sub- or sur-level set component. Then by Property 3.2, there must be a loop in $\mathcal{R}(f)$. This contradicts the hypothesis statement that $\mathcal{R}(f)$ is loop free.

Next, we prove that if every valence-3 0-simplex of $\mathcal{R}(f)$ being the image through ϕ of a saddle joins distinct sub- or sur-level set components, then $\mathcal{R}(f)$ is loop free. Assume for contradiction that $\mathcal{R}(f)$ has a loop, s is the highest join saddle in the loop, and σ its image through ϕ. Pick ϵ, w, a, b, c_a, and c_b as before. The paths q^+ and q^- exist in $\mathcal{R}(f)$ connecting a and b in a loop. The inverse map ϕ^{-1} is a continuous mapping of points of the Reeb graph to contours of $\mathcal{M}$, therefore $\phi^{-1}(q^-)$ is a connected component in $\mathcal{M}$, connecting c_a to c_b in $L^-(w)$. Therefore, c_a and c_b are on the boundary of the same connected component of $\mathcal{L}^-(w)$, which contradicts the hypothesis. The same argument holds for split saddles.

Property 3.3 shows that we can use the contour tree algorithm presented by Carr et al. [19] to compute loop free Reeb graphs, by tracking the connectivity evolution of sub- and sur-level sets. Performing loop surgery guarantees that $\mathcal{R}(f')$ is loop free, and then computable with the contour tree algorithm.

3.2.2 Algorithm

Our strategy for Reeb graph computation is summarized in Algorithm 2. First, the domain is symbolically cut to ensure that the Reeb graph of f' is loop free. If a diagnostic shows that the domain has no handle, then no loop surgery needs to be done. Otherwise, loop saddles are extracted and the domain is symbolically cut along cutting surfaces. We keep track of the extra pairs of 0-simplices of $\mathcal{R}(f')$ created by each cut. Since the Reeb graph is guaranteed to be loop free, we compute it using a modified version of the join-split tree algorithm [19]. Finally, the loop free Reeb graph is transformed into the correct Reeb graph of the input function by inverse cuts.

```
input : Scalar field f : M → ℝ;
begin
    M' ← M, GL = ∅;
    if genus_diagnostic(∂M) > 0 then
        S = find_surgery_loop_saddles(M, ∂M, f);
        foreach s ∈ S do
            M' ← cut(M', cutting_surfaces(s));
            GL = GL ∪ simplex_pairs(s);
        end
    end
    R(f) ← contour_tree(M', f');
    foreach p ∈ GL do
        R(f) ← glue(R(f), p);
    end
end
```

Algorithm 2: Reeb graph algorithm overview

Loop Surgery

The purpose of *loop surgery* is to symbolically cut the domain such that $\mathscr{R}(f')$ is guaranteed to be loop free.

Genus Diagnostic

We first check if any loop surgery is needed by checking the presence of tunnels on the boundary $\partial\mathscr{M}$. This implements the *genus_diagnostic* procedure of Algorithm 2. In particular, we use the Euler formula on $\partial\mathscr{M}$:

$$\chi(\partial\mathscr{M}) = 2 - 2g(\partial\mathscr{M}) = n_v^\partial - n_e^\partial + n_f^\partial \tag{3.8}$$

where n_v^∂, n_e^∂ and n_f^∂ stand for the numbers of vertices, edges and triangles of $\partial\mathscr{M}$. The genus $g(\partial\mathscr{M})$ then gives the number of tunnels in $\partial\mathscr{M}$ and hence the number of cuts needed to ensure that $\mathscr{R}(f')$ is loop free. Loop surgery is needed only if this number $g(\partial\mathscr{M})$ is non-zero.

Loop Saddles

If the domain is cut at every loop saddle then $\mathscr{R}(f')$ is loop free. In this section, we implement *find_surgery_loop_saddles* of Algorithm 2. One technique for finding the loop saddles of f_∂ is computing $\mathscr{R}(f_\partial)$ (using an existing technique such

as [91]), and then identifying each loop of $\mathscr{R}(f_\partial)$ using extended persistence [2]. Alternatively, we found in practice that the benefits of using a simpler technique for finding overall a small superset of surgery loop saddles of f outweigh the cost of performing additional cuts.

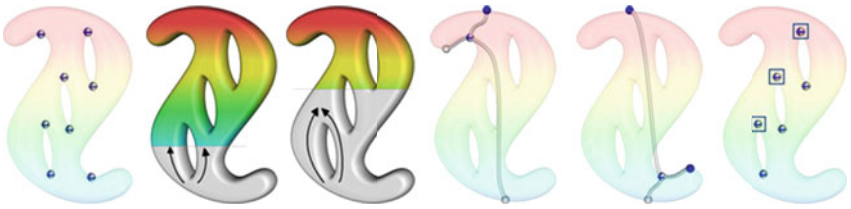
f_∂ Loop saddles of f_∂ are first selected in a three-step process: *(i)* all saddles of f_∂ are identified; *(ii)* we apply Property 3.3 and remove from these the ones that join distinct sub- or sur-level set components; and *(iii)*, we further extract a list of candidate surgery loop saddles of f. The rules we employ resolve degenerate saddles implicitly. These steps are explained in detail below.

(i) Saddles of f_∂ occur at vertices of $\partial\mathscr{M}$. A vertex $x \in \partial\mathscr{M}$ is a saddle if and only if the number of connected components of its lower link is greater than one. This number can be computed by a simple link traversal technique [28]. The set of all the saddles on the boundary is denoted S_∂.

(ii) We remove saddles S_∂ that do not open or close a loop. To use the result of Property 3.2 in determining whether or not the saddle can be part of a loop, we identify the sub- and sur-level set associated with each connected component of the lower and upper link of a saddle of S_∂. The classification of sub- and sur-level set components is computed by constructing the join tree $\mathscr{J}(f_\partial)$ and the split tree $\mathscr{S}(f_\partial)$. The join sweep that builds the join tree processes vertices of $\partial\mathscr{M}$ in order of increasing function value, maintaining sub-level sets via a Union-Find data structure. In the computed join tree of f_∂, the number of downward 1-simplices of each node is equal to the number of distinct sub-level set components of f_∂ that are joined at the corresponding vertex. Similarly, a split sweep builds the split tree by processing vertices of $\partial\mathscr{M}$ from highest to lowest. The resulting split tree provides the information about the evolution of sur-level set components.

When the number of downward 1-simplices in the join tree equals the number of connected components of the lower link and the number of upward 1-simplices in the split tree equals the number of connected components of the upper link, the saddle does not open or close a loop in $\mathscr{R}(f_\partial)$. These saddles are removed from S_∂.

(iii) The set S_∂ now exactly contains the loop saddles of f_∂. As described in the previous subsection, each loop $\mathscr{L} \in \mathscr{R}(f)$ maps to a monotone path in $\mathscr{J}(f)$ (or in $\mathscr{S}(f)$) being a subset of the monotone path linking the loop saddles of the corresponding loop $\mathscr{L}_\partial \in \mathscr{R}(f_\partial)$. We identify candidates for *surgery loop saddles* of f by walking upward from the lowest loop saddle of $\mathscr{L}_\partial$ in $\mathscr{J}(f)$ to its highest loop saddle (blue squares in Fig. 3.13e) and extracting the list of 1-saddles of f that map to the interior of 1-simplices of $\mathscr{J}(f)$ (boundary loop saddles are paired on $\mathscr{J}(f)$ based on their function values, yielding possible over estimations when boundary loops overlap through ψ).



(a) $\partial\mathscr{M}$, f_∂ (b) Sublevel sets of f_∂ (c) Join tree of f_∂ (d) Split tree of f_∂ (e) Loop saddles

Fig. 3.13 Identifying loop saddles: f_∂ and its saddle points (**a**). Loops are "absorbed" by sub-level sets of f_∂ (**b**). Join tree (**c**) and split tree (**d**) of f_∂ identify non-loop saddles. Loop saddles are returned (**e**). ©2009 IEEE. Reprinted, with permission, from [125]

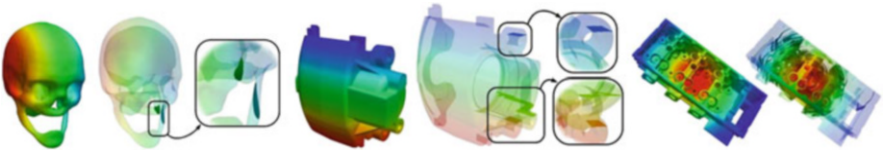


Fig. 3.14 Examples of *cutting surfaces* of 3D scalar fields defined on non-simply connected domains: skull (2 handles), brake caliper (3 handles), cylinder head (82 handles). ©2009 IEEE. Reprinted, with permission, from [125]

Cutting Surfaces

We symbolically cut $\mathscr{M}$ through a sequence of symbolic cuts (Figure 3.14), implementing the procedure *cutting_surfaces* of Algorithm 2. According to Property 3.3, to ensure that $\mathscr{R}(f)$ be loop free, every 1-saddle must join distinct sub-level set components. A surgery loop saddle s of f does not have this property, therefore we perform symbolic cuts on $\mathscr{M}$ such that each connected component of the lower link of s have a unique sub-level set component.

A symbolic cut is a contour traversal that updates pointers in the tetrahedra that are crossed. A cutting surface S_i is a simple data structure with a unique identifier that is the record of the symbolic cut. Let s_i be a surgery loop saddle with value $f(s_i)$ and let n be the number of connected components of $Lk^-(s_i)$. Cutting surface traversals are started for $(n-1)$ connected components $Lk^-(s_i)$, in particular at a tetrahedron adjacent to each of these. Each symbolic cut produces a new sub-level set component, which is recorded in the cutting surface data structure.

To keep track of the symbolic cuts in the rest of the algorithm, each tetrahedron crossed by any cutting surface stores a pointer for each of its vertices to the highest cutting surface passing below and the lowest cutting surface passing above the vertex. Additionally, each vertex is marked with a *top* flag (Fig. 3.15, blue circle) if it lies above a cutting surface crossing the tetrahedron, and also a *bottom* flag (Fig. 3.15, red disc) if it lies below a cutting surface crossing the tetrahedron. Finally, each saddle that generates symbolic cuts stores pointers to the corresponding cutting surfaces.

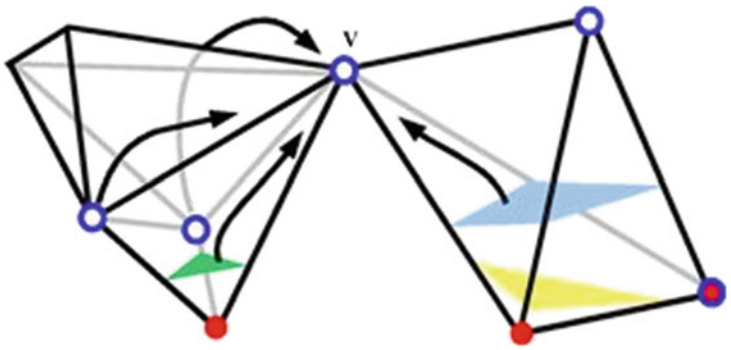


Fig. 3.15 Simulating symbolic minima in the presence of cutting surfaces (green, blue and yellow) when visiting a vertex v during the join tree construction. Vertices with a *top* (respectively *bottom*) flag are marked with a blue circle (respectively a red disc)

Loop Free Reeb Graph Computation

By Property 3.3, a loop free Reeb graph can be computed using a contour tree algorithm. Since we cut $\mathscr{M}$ only symbolically, we use a modified version of the join-split algorithm [19] that behaves "as if" $\mathscr{M}$ were actually transformed into $\mathscr{M}'$. Building the loop free Reeb graph using the modified join-split algorithm implements the procedure *contour_tree* of Algorithm 2. This modified contour tree algorithm simulates the cuts and uses a Union-Find data structure (using path compression and union by rank) implemented to return the highest element of a set.

Vertices are processed in order of increasing function value, and each vertex v is added to the join tree and a new set is created in the Union-Find. If the star of v contains tetrahedra pointing to cutting surfaces, we simulate $\mathscr{M}$ being cut by simulating the existence of a new sub-level set and a new minimum by adding each cutting surface S_i to the join tree and the Union-Find. The tetrahedra in the star of v that have a vertex in the lower link of v are iterated upon. For each such tetrahedron T, if there is a cut S_i crossing T, we pick the highest S_i that is below v, as illustrated in Fig. 3.15 (black arrows emanating from cutting surfaces). This is a constant time operation due to having stored pointers in each tetrahedron in the cutting surface computation. We perform a find and then merge the sets, also adding an arc to the join tree. This simulates having cut $\mathscr{M}$ by S_i, since it disconnects part of the lower link, and instead connects v to an "artificial" minimum, which is the node returned by the Union-Find. Next, the vertices in the lower link of v that were not disconnected by any cutting surface are processed (black arrows emanating from vertices in Fig. 3.15). If no tetrahedron in the lower star of v is crossed by a cutting surface (i.e., v did not get marked as *top*) then the lower link of v can be processed with no changes to the algorithm in [19]. The join tree of f' is returned. The split tree is computed symmetrically, and merging the two trees occurs exactly as in the join-split algorithm. This computes the loop free Reeb graph $\mathscr{R}(f')$.

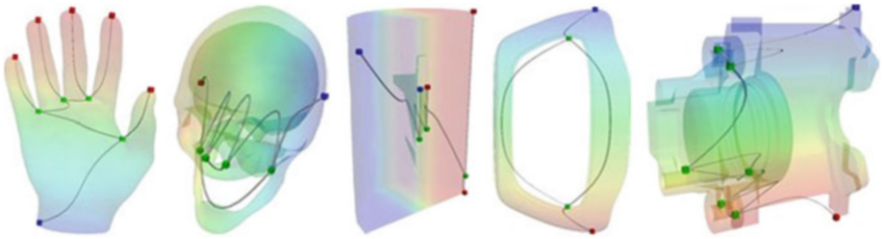


Fig. 3.16 Examples of Reeb graphs of scalar fields defined on the experiment data-sets. Persistence-based simplification and arc smooth embedding can optionally be computed in a post-process. ©2009 IEEE. Reprinted, with permission, from [125]

Inverse Cuts

Transforming the loop-free Reeb graph $\mathscr{R}(f')$ into $\mathscr{R}(f)$ requires gluing minimum-maximum pairs of each cutting surface. This implements the procedure *glue* of Algorithm 2. For each cutting surface, pointers in the data structure identify the minimum it generated in the join tree and the maximum it generated in the split tree. The two 0-simplices are found in $\mathscr{R}(f')$, and are glued together by concatenating the up-arc of the minimum, and the down-arc of the maximum. This inverts the changes made to $\mathscr{R}(f')$ by loop surgery (Fig. 3.16).

3.2.3 Results and Discussion

We implemented Reeb graph computation based on loop surgery in standard C under GNU/Linux. All the experiments presented below were run on a standard desktop computer with a 64-bit 2.83 GHz CPU and 8 GB of memory. Data sets are courtesy of the AIM@SHAPE shape repository and collaborating mechanical design experts.

In our experiments, we compare running times with recent Reeb graph computation techniques for tetrahedral meshes [38, 91]. We used the original implementations of these approaches, kindly provided by their respective authors. Furthermore, we compared the output of our approach to that presented in [91] using the exact same simulation of simplicity [43], and found the two algorithms output identical Reeb graphs for all the available datasets.

Time Complexity

Let n and N be respectively the number of vertices and simplices of $\mathscr{M}$. Moreover, let n_∂ and N_∂ be respectively the number of vertices and simplices of $\mathscr{M}_\partial$. Finally, let N_S be the number of simplices of $\mathscr{M}$ crossed by cutting surfaces. We present a complexity analysis for each step in our algorithm:

Loop saddle extraction: Saddle identification by link traversal requires $O(n_\partial)$ steps. Join tree and split tree computation both require $O(n_\partial log(n_\partial) + N_\partial \alpha(N_\partial))$ where $\alpha()$ is an exponentially decreasing function (inverse of the Ackermann function). Surgery loop saddle candidate extraction takes $O(N)$ steps.

Symbolic cuts computation: $O(g(\mathscr{M}_\partial) \times N_S)$ steps: each of the $g(\mathscr{M}_\partial)$ handles of $\mathscr{M}$ generates cutting surfaces and yields a traversal of at most N_S tetrahedra.

Loop-free Reeb graph computation: the contour tree algorithm variant requires $O(nlog(n) + N\alpha(N))$ steps [19].

Inverse cuts: $O(g(\mathscr{M}_\partial))$ steps: there is an explicit list of cutting surfaces, and gluing the min-max pairs of each takes constant time.

Overall bound: $O(nlog(n) + N\alpha(N) + g(\mathscr{M}_\partial) \times N_S)$.

The worst case scenario is reached when both $g(\mathscr{M}_\partial)$ and N_S (loop surgery process) are linear with the size of the mesh, in which case our algorithm will exhibit a strong quadratic behavior. However, with real-life data, $g(\mathscr{M}_\partial)$ is a small constant, resulting in virtually linear scalability in practice.

Performance Comparison

We compare the running times of our approach with those of the two fastest previous techniques, presented in [91] and [38]. The scalar valued data represents a variety of physical phenomena: air turbulence, pressure, liquid oxygen diffusion, rock density, etc. Table 3.2 reports the running times of the three methods on these data.

Our approach achieves significant improvement in terms of running time for each data-set, including those with the highest number of handles, resulting in an average speedup factor of 6500.

The approaches presented in [91] and [38] rapidly stress and exhibit a quadratic behavior on real-life data. Despite the theoretical quadratic complexity of our loop surgery, the worst-case scenario seems difficult to reach with real-life data.

Asymptotic Stress Tests

As described previously, the loop surgery procedure has a quadratic worst-case complexity. In practice, for the experiments reported in Table 3.2, we observed this step took 43% of the overall computation in average and that its time requirement was increasing both with the size of the mesh and the number of handles. The pressure field on the trunk data-set is a special case: it is an extremely noisy field that illustrates that we are performing more symbolic cuts than necessary. Overall, we observe that the loop surgery overhead is proportional to the cost of computing the contour tree with real-life data.

Next, we provide a stress test to show the performance of the algorithm in a worst-case scenario. We generate meshes by tetrahedralizing a rectilinear grid on from which rows and columns have been removed, allowing us to increase the

Table 3.2 Comparing Reeb graph computation run times

			LS	SA [91]		OS [38]	
Data	Tets	#H	Time	Time	Speedup	Time	Speedup
Langley fighter	70,125	0	0.346 s	43.70 s	**126.3**	650.1 s	**1879**
Cylinder head (low res.)	116,274	82	0.660 s	10.20 s	**15.45**	257.188 s	**389.68**
Hood (low res.)	120,501	31	0.340 s	45.77 s	**134.62**	52.602 s	**157.71**
Trunk (low res.)	143,366	1	1.866 s	27.97 s	**14.99**	406.665 s	**217.93**
Liquid oxygen post	616,050	1	0.686 s	435.2 s	**634.4**	15.54 s	**22.65**
Brake caliper (med res.)	1,155,317	3	2.242 s	–	–	180,638 s	**80,570**
Buckminster Fullerene	1,250,235	0	2.508 s	9887 s	**3942**	781.0 s	**311.4**
Plasma	1,310,720	0	2.202 s	11,983 s	**5442**	–	–
S. Fernando earthquake	2,067,739	0	4.068 s	15,949 s	**3921**	–	–
Brake disk (high res.)	3,554,828	54	7.798 s	–	–	–	–

Bold figures indicate speedups
#H stands for the number of handles in the mesh, *LS* for the loop surgery approach, *SA* for the streaming approach [91] and *OS* for the output sensitive approach [38]. The loop surgery approach processes in average 428 thousand tets per second and provides an observed average speedup of 6510. The "–" symbol means that either the process interrupted on a memory exception or that the process had not completed after several days of computation

number of tets and the number of handles independently. The function value of each vertex is its y coordinate. Figure 3.17 illustrates these meshes. Figure 3.18 shows the running time and the memory footprint as a function of the number of handles and the size of the mesh. This experiment shows that for a constant number of handles, our algorithm scales linearly with the size of the mesh. When the number of handles increases, the linear ratio also increases. The memory footprint follows the same linear behavior as the time execution, due to non-optimized data structures for storing cutting surfaces.

Limitations

A limitation of the algorithm is that it requires the mesh to be manifold with boundary to ensure that the boundary of the mesh is a 2-manifold without boundary. Another is that a given boundary loop can span several 1-saddles that map to the interior of 1-simplices of $\mathscr{J}(f)$. Each of these will be extracted as a surgery loop saddle, yielding potentially several cutting surfaces per handle.

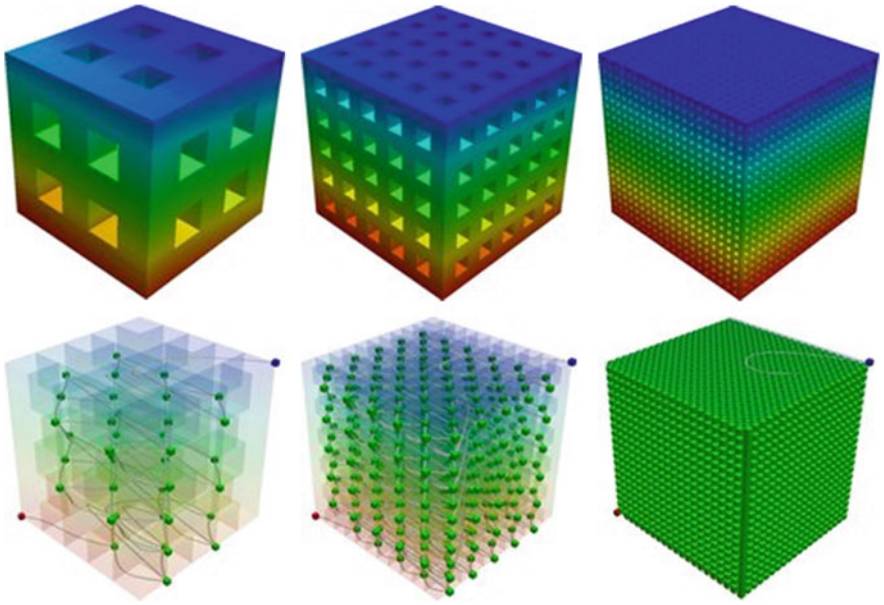


Fig. 3.17 Handle stress experiment: examples of generated meshes with increasing number of handles (top) and the Reeb graph of their height function. ©2009 IEEE. Reprinted, with permission, from [125]

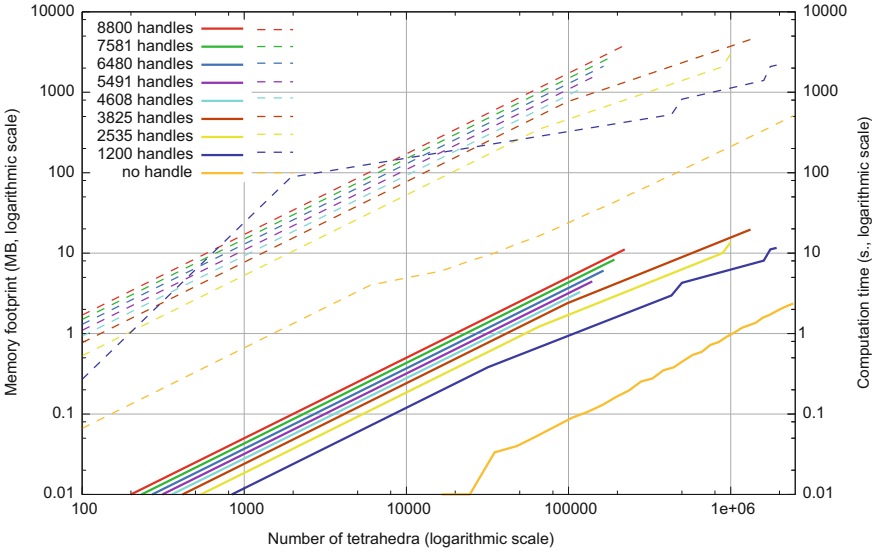


Fig. 3.18 Running time (solid lines) the memory footprint (dashed lines) when increasing the number of tets for a mesh with constant number of tunnels. ©2009 IEEE. Reprinted, with permission, from [125]

Concluding Remarks

In this section, we presented a practical algorithm for fast Reeb graph computation on tetrahedral meshes in $\mathbb{R}^3$. By providing theoretical results on the topology of such Reeb graphs, we showed their computation could be reduced to a contour tree computation through a technique called loop surgery. Experiments demonstrated in practice the scalability of the algorithm. Moreover, we showed that our approach improves, in terms of running time, the fastest previous techniques for real-life data by several orders of magnitude.

Reducing the computational requirements of Reeb graphs to that of contour trees enables the generalization of the contour-tree based interactive techniques to volumetric meshes of arbitrary topology (as described in the next chapter) and thus opens several avenues for future visualization research. An extension of our approach to volumetric meshes not embeddable in $\mathbb{R}^3$ and of higher dimensions would address a larger class of problems. However, as it is no longer true that such meshes necessarily admit boundary, the loop surgery concept would need to be extended and generalized. Doraiswamy and Natarajan [39] later extended this approach to manifolds of arbitrary dimension, by inserting cutting surfaces at each 1-saddle lying in the interior of a 1-simplex of the join tree (and not necessarily those spanned by boundary loops), yielding a more general algorithm at the expense of computing more cutting surfaces than our approach. Due to its fast performances in practice, we considered our loop surgery approach as the reference algorithm for the problem of Reeb graph computation on tetrahedral meshes in $\mathbb{R}^3$ until an optimal time complexity algorithm was introduced 3 years later [89].

Chapter 4
Interaction

As described in Chap. 2, topological abstractions are fundamental data structures in scientific visualization, to accelerate the computation of geometrical constructions (such as level-sets) or to drive segmentation algorithms. In this chapter, we describe examples of efficient algorithms for the interactive manipulation of topological abstractions. After the data abstraction step (discussed in the previous chapter), interactive manipulation capabilities are often needed in practice to explore the features of interest, prior to their quantitative analysis (discussed in the next chapter). In this chapter, we describe two types of interactive manipulation approaches, for visual exploration and data segmentation purposes.

4.1 Topological Simplification of Isosurfaces

Scientific data is often affected by noise. When extracting isosurfaces for visual exploration purposes, noise can lead to the appearance of many connected components of isosurfaces, some of which being not relevant application wise but still occluding the components of interest of the isosurface. This occlusion problem can prevent users from visualizing and understanding the geometry of the main features of the data.

Carr et al. [20] described an approach for the topological simplification of isosurfaces on simply connected domains that addresses this issue. As described in Chap. 2, the contour tree can be used as an indexing data-structure for the fast extraction of level sets. This capability can be nicely combined with persistent homology concepts to only extract the contours corresponding to the most persistent features of the data (Fig. 2.19). Carr et al. [20] described other geometrical measures to prioritize the cancellations of the 1-simplices of the contour tree, such as the volume of their pre-image through ϕ or the integral of f over these pre-images (called hypervolume). This work demonstrated the importance of such measures

J. Tierny, *Topological Data Analysis for Scientific Visualization*,
Mathematics and Visualization, https://doi.org/10.1007/978-3-319-71507-0_4

for segmentation tasks, as they induce more stable and intuitive segmentation parameters.

In this section, thanks to our fast Reeb graph computation algorithm (previous chapter), we generalize this interactive approach to non simply-connected PL 3-manifolds. In particular, we focus on the analysis of pressure fields in mechanical design (where the majority of meshes have handles), a case study where contour-tree based techniques could not previously apply. One experiment in the process of mechanical design involves the analysis of the resistance of mechanical pieces made of different materials to pressure stress. In such experiments, mechanical experts first consider simulation previews computed on low resolution meshes. This step illustrates the approximate behavior of the material. Its understanding is crucial to define correct parameters for the actual simulation at high resolution. However, as shown in Figs. 4.1 and 4.2, this preview can be noisy, making its interpretation difficult.

We overcome this problem with a fast topologically clean isosurface extraction system. First, the Reeb graph of the low-resolution pressure stress function is computed in a pre-process. Then, local geometric measures [20] (extended to tetrahedral meshes) are computed for each arc of the Reeb graph. Then, users may select thresholds for geometric measures to simplify the Reeb graph, as described in [91]. This filtering of the arcs in our examples took at most 0.03 s. Our approach maintains degree two nodes in the Reeb graph, representing regular vertices of the function. Consequently, the Reeb graph provides a seed vertex for each contour the user wants to display. We store the non-simplified arcs of the Reeb graph in a balanced interval tree. An isosurface extraction query consists of searching in this tree for a valid seed set. In our examples, this is performed in less than nanoseconds, starting standard isosurface traversal techniques at these seeds in interactive times.

Figures 4.1 and 4.2 illustrate this process on pressure stress functions computed on a brake disk and a cylinder head, where the user progressively increases a

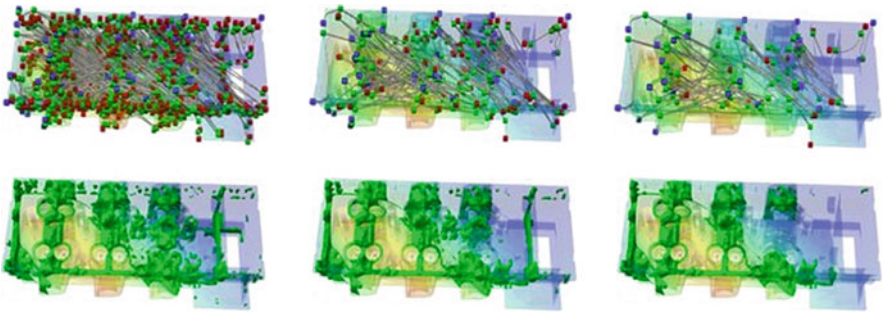


Fig. 4.1 Topologically clean isosurface extraction on a pressure stress simulation on a cylinder head. The Reeb graph is progressively simplified (from left to right) with increasing hyper-volume scale. As a result, small components (noise) of the considered isosurface are progressively removed (bottom) and the most important features are progressively highlighted (215, 58 and 9 connected components). ©2009 IEEE. Reprinted, with permission, from [125]

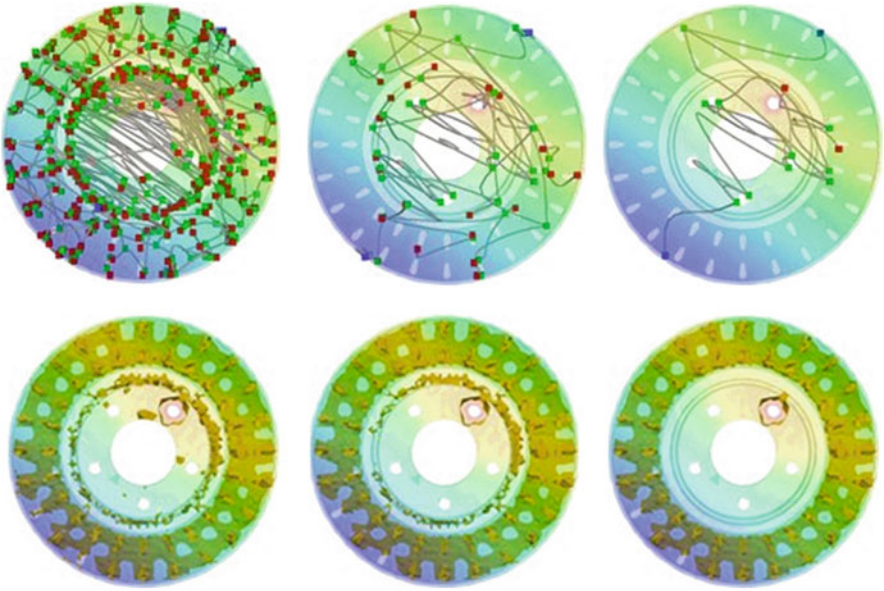


Fig. 4.2 Topologically clean isosurface extraction on a pressure stress simulation on a brake disc. The Reeb graph is progressively simplified (from left to right) with increasing hyper-volume scale. As a result, small components (noise) of the considered isosurface are progressively removed (bottom) and the most important features are progressively highlighted (92, 8 and 2 connected components). ©2009 IEEE. Reprinted, with permission, from [125]

simplification threshold with the hyper-volume measure [20]. The Reeb graphs are simplified, and as a result, the small connected components (noise) of the queried isosurface are progressively removed and the most important features are highlighted. This enables a direct visualization of the major trends of the simulation.

Based on this approach, we introduced next an isosurface based visualization widget for a large-scale simulation monitoring system. Large scale numerical simulations running on super-computers are often analyzed on-line with advanced monitoring systems [75], providing the end users with real time quantitative indicators describing the behavior of the simulation, allowing them to identify instantly possible run-time issues (mistakes in parameter settings for instance). However, the interpretation of these indicators require strong user expertise and sometimes problematic configurations are difficult to read from those. Therefore, there exists an application need for on-the-fly visualizations of numerical simulations for monitoring purposes. However, large scale simulation time steps are usually too large to be interactively transferred to a remote workstation for visualization. Moreover, due to their size, not all of the time-steps of such a simulation can be stored on the super-computer where they have been generated. This prevents remote rendering and interaction and requires in situ visualization generation.

We addressed this problem by designing a prototype [101] for the eSimMon monitoring system [75] developed by Oak Ridge National Laboratory, capable of

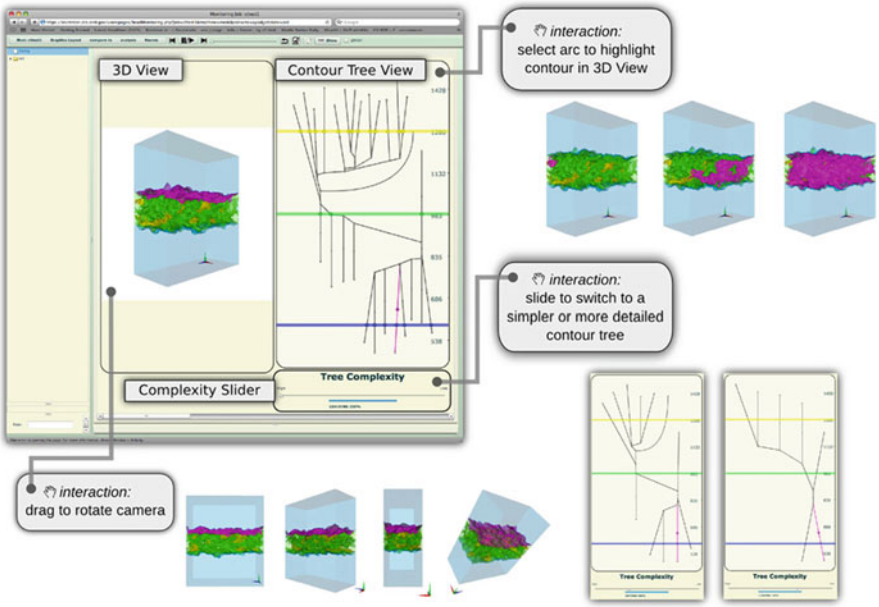


Fig. 4.3 Isosurface based visualization widget for the eSimMon [75] web-based simulation monitor: isosurface rendering (top left), Reeb graph planar display (top right), persistence slider (bottom right). Other view examples are provided on the side. ©2009 IEEE. Reprinted, with permission, from [101]

generating in situ isosurface renderings (Fig. 4.3). The visualization of an isosurface is dictated by a number of parameters such as view point and isovalue and possibly topological simplification threshold. Therefore we implemented an in situ algorithm that finely samples in batch mode this parameter space (26 view points, four level of possible topological simplification, plus the individual extraction of a contour for each 1-simplex of the Reeb graph, in purple in Fig. 4.3) and generates for each parameter combination an in situ offline 2D rendering at a resolution of 1024×1024. Even by finely sampling this parameter space, the data size of the output collection of 2D renderings for a given time-step is configurable and in practice guaranteed to be orders of magnitude smaller than that of the time-step itself. In our experiments, the space required for this collection of 2D renderings was 140 Mb. Next, a web-based client integrated into eSimMon enables the user to explore this collection remotely by emulating the interactive control of these parameters as showcased in Fig. 4.3. Note in particular that only the 2D renderings selected by the user are transferred to the web-based client. This technique was enabled by our fast isosurface extraction and simplification algorithms described previously. This prototype enabled simulation users to obtain qualitative visual insights from their running simulations. A similar strategy has been developed

independently by Kitware Inc. 5 years later in its visualization system ParaView Cinema [73].

In our experiments, only data-sets of moderate size were considered to enable the development of this proof-of-concept prototype. As discussed in Chap. 6, the in situ processing of real-life sized data-sets raises a number of important algorithmic challenges that require to revisit entirely Topological Data Analysis algorithms.

4.2 Interactive Editing of Topological Abstractions

As discussed in Chap. 2, topological abstractions are fundamental data-structures in scientific visualization for data segmentation purposes. In particular, the Reeb graph and its variants are well suited when the boundary of the regions of interest align with level-sets, while the Morse-Smale complex is well suited when features (or their boundaries) align with the gradient. In many applications, such segmentations often directly correspond to meaningful segmentation application-wise, with excellent classification scores. However, such approaches still result in general in the identification of false negatives as compared to a manual labeling by a domain expert. In the following, I describe two approaches for the interactive editing of topological abstractions, in order to integrate users' knowledge in the process.

4.2.1 Morse-Smale Complex Editing

As described in Chap. 2, when using the Discrete Morse Theory setting, the first step for the computation of the Morse-Smale complex involves the computation of a discrete gradient (Fig. 2.20). In this sub-section, we describe the first approach for the computation of a discrete gradient that *conforms* to some alignment constraints provided automatically or interactively by a user. Such a mechanism enables to correct in a pre-process potential false positives in the segmentation, while still benefiting from the multi-scale nature of persistent homology concepts applied to the Morse-Smale complex. To define the concept of a conforming Morse-Smale complex, we first introduce conforming discrete gradient vector fields. Given a regular cell complex K, let $\hat{f} : K^0 \rightarrow \mathbb{R}$ be the input scalar data evaluated on the vertices of K and let $I : K \rightarrow S$ be a surjective map from cells of K to an index set $S \subset \mathbb{Z}$.

Conforming Discrete Gradient

We define a discrete gradient vector field V as *conforming to a map* I if and only if for each discrete vector $\langle \alpha^i, \beta^{i+1} \rangle \in V, I(\alpha^i) = I(\beta^{i+1})$, meaning both the head and tail of a discrete vector have the same label in I. We remark that the space of discrete

gradient vector fields is the same as the space of conforming discrete gradient vector fields. This is easily seen, since (1) every conforming discrete gradient vector field is also a discrete gradient vector field, and (2) any discrete gradient vector field V is also a conforming discrete vector field under the map $I(\alpha) = 1$. Since the Morse-Smale complex $\mathscr{MS}(f)$ is uniquely determined by a discrete gradient vector field V, we say $\mathscr{MS}(f)$ *conforms* to a map I if and only if V also conforms I.

Algorithm

Computing discrete gradient vector fields so that they conform to any arbitrary map I involves only a slight variation of previous techniques. In particular, the added restriction that a gradient vector $\langle \alpha, \beta \rangle$ can only be created if $I(\alpha) = I(\beta)$ by definition creates a conforming discrete gradient. Note that any valid discrete gradient V can be converted to a conforming one by simply making each cell α and β critical when $\langle \alpha, \beta \rangle$ and $I(\alpha) \neq I(\beta)$. However, such a trivial modification does not represent how the flow behaves when restricted to the boundary of a labeled section. In the following algorithm, we create a discrete gradient vector field that conforms to I but nevertheless avoids creating critical cells where a discrete gradient vector $\langle \alpha, \beta \rangle$ can be created where $f(\beta) \leq f(\alpha)$ and $I(\alpha) = I(\beta)$, i.e., there exists *some* gradient flow restricted to the label $I(\alpha)$.

Our algorithm is inspired by Robins et al.: first create a discrete gradient vector for a vertex-edge pair and then perform simple homotopic expansions in the lower star [100]. The main difference is that we only consider for pairing those cells that also share the same label. Robins' algorithm has a strict ordering for performing homotopic expansions in the lower star, and straightforward restriction of pairing based on labels would create many extra critical cells. We instead take a very pragmatic approach to reducing the number of spurious critical cells produced by reordering the homotopy expansion to perform any possible pairing in the lower star that is a simple homotopy expansion (the inverse of homotopic collapses [26]) yet also conforms to the labeling.

In the following, we denote a cell that has been identified as critical by pairing it with itself, for example, $\langle \alpha, \alpha \rangle$. Furthermore, a cell is *assigned* if and only if it has been identified as critical or paired in a discrete gradient vector. The function $\#UCF(\gamma)$, (Number of Unassigned Conforming Faces) counts the number of faces $\beta \leq \gamma$ of a cell γ restricted to the lower star $St^{-}(\alpha)$ such that β has not been assigned and $I(\beta) = I(\gamma)$.

The algorithm processes each vertex independently, first creating a vertex-edge vector in the direction of steepest descent, restricted to the set of edges sharing the same segmentation label as the vertex. If no pairing for the vertex is possible, it is made critical.

Next, simple homotopy type expansions are performed in order of increasing dimension (where d stands for the dimension of the domain), again restricting possible candidates for pairing to those sharing the same segmentation label. For each dimension i, while there exists unassigned i-cells in the lower star of

input : Scalar field $\hat{f} : K \to \mathbb{R}$, map $I : K \to \mathbb{Z}$;
output: Conforming discrete gradient V;

begin
  $V \leftarrow \emptyset$;
  foreach $\alpha \in K^0$ **do**
    $S \leftarrow \{\beta^1 \in St^-(\alpha) \mid I(\alpha) = I(\beta^1),\ \alpha \leq \beta^1\}$;
    if $S = \emptyset$ **then**
      $V \leftarrow V \cup \langle \alpha, \alpha \rangle$;
    else
      $V \leftarrow V \cup \langle \alpha, \beta^1 \rangle$, $\beta^1 \in S$ is in direction of steepest descent;
    end
    foreach $i \in [1, \ldots, d]$ **do**
      while $\exists$ *unassigned* $\beta^i \in St^-(\alpha)$ **do**
        while $S^{i+1} \leftarrow \{\gamma^{i+1} \in St^-(\alpha) \mid \#UCF(\gamma) = 1\} \neq \emptyset$ **do**
          $V \leftarrow V \cup \langle \beta^i, \gamma^{i+1} \rangle$, β^i is the assigned conforming face of $\gamma^{i+1} \in S^{i+1}$;
        end
        if $\exists$ *unassigned* $\beta^i \in St^-(\alpha)$ **then**
          $V \leftarrow V \cup \langle \beta^i, \beta^i \rangle$, β^i is unassigned;
        end
      end
    end
  end
end

Algorithm 3: Conforming gradient construction algorithm

α, simple homotopy expansions of an unassigned i-cell with unassigned $i + 1$ cells are attempted, marking the i-cell critical when such an expansion is not possible. The test to check if there exist unassigned i-cells in the lower star of α can be implemented by placing the i-cells in $St^-(\alpha)$ in a list, whose size is typically bounded by a small constant. The output is guaranteed to produce a discrete gradient vector field, since all pairings are restricted to the lower star of a vertex and a homotopy expansion is only performed when all faces of the i-cell have previously been assigned, and the $i + 1$-cell has only one unassigned face. These two conditions along with the fact that every cell of the domain is either paired or marked critical ensure that all V-paths produced are monotonically decreasing and V is acyclic, hence a discrete gradient vector field (just as in [100]). Algorithm 3 can be applied to every vertex in the domain in an embarrassingly parallel manner.

This same kind of modification is also possible for discrete gradient construction algorithms other than Robbin's [61, 62]. Overall, just as in our *ConformingGradient*() algorithm, the modifications to these include checking the segmentation label when assigning the first vertex-edge gradient vector, and subsequently checking the label when assigning the higher dimensional vectors during simple homotopy expansion. Furthermore, slight reorderings may be necessary to remove spurious critical points. Once the discrete gradient vector field has

been computed, any one of the algorithms for traversal of the discrete gradient field [61, 100, 104] can be used to compute the conforming Morse-Smale complex.

Editable Morse-Smale Complexes

The *ConformingGradient*() algorithm for computing conforming Morse-Smale complexes can be used to enable editing of Morse-Smale complexes. Editing can be used by domain experts to correct errors when doing feature extraction. For instance, in bio-medical imaging, overlapping morphological structures, insufficient dye penetration, shadows, and light diffraction often contribute to poor feature representation during analysis. In this case, the image data contains insufficient or incorrect information for an accurate Morse-Smale complex based feature identification. However, a domain scientist may understand the deficiency and have the interpretation skills necessary to understand what the segmentation *should* be. Often, long pipelines of image processing filters are used to prepare image data for semi-automated segmentation, such as deconvolution, noise removal, color/contrast correction, thresholding, and smoothing. Furthermore, the Morse-Smale complex itself may be simplified and filtered to extract the desired features. Each stage of this pipeline may depend on several parameters, each one affecting the accuracy and precision of the resulting feature identification, when compared to the gold standard of a domain expert manually identifying features. While most features can be extracted with such an approach, there may still remain problematic cases. In such an instance, we allow the user to edit the segmentation iteratively as part of the analysis pipeline.

Once a Morse-Smale complex has been computed on a d-dimensional domain, its ascending and descending d-manifolds define an origin/destination map. Then, reconstructing the Morse-Smale complex using that map as a constraint with our conforming gradient algorithm results in the same Morse-Smale complex. However, it is also possible to *edit* the maps prior to the second computation, effectively editing the reconstructed Morse-Smale complex. The combinatorial nature of the algorithm guarantees that the result is a valid Morse-Smale complex that conforms to the new edits. In this following, I restrict the discussion to two-dimensional images.

Identity Map

Let V be a discrete gradient vector field computed with any algorithm on the mesh K and function $\hat{f}$. There are many maps I such that *ConformingGradient*() produces the exact same discrete gradient vector field V. Two trivial examples are $I(\alpha) = 1$ for any cell α, and the map taking each distinct gradient vector and assigning both cells a their own label. Such maps are called *identity* maps. Our approach for editing the Morse-Smale complex is to generate an identity-like map from the Morse-Smale

complex as described below, allow the user to modify that map, and then recompute the Morse-Smale complex with the conforming algorithm.

Termination Map

We construct the initial map based on the origin/destination of V-paths in V. In particular, the ascending and descending manifolds of $\mathscr{MS}(f)$ allow us to construct a *termination map*, $\omega : K^{0,d} \rightarrow S \subset \mathbb{Z}$, that stores a labeling of the destination of each vertex and d-cell. Let $M_n = \{\alpha_0, \alpha_1, \ldots, \alpha_k\}$ be the set of critical vertices, i.e., the minima. For each minimum α_j, we compute the set of vertices belonging to its ascending d-manifold, and assign those the label j. Symmetrically, we compute the descending manifolds of critical d-cells, i.e. the maxima, and assign all d-cells the same label. Given a Morse-Smale complex $\mathscr{MS}(f)$ constructed from a discrete gradient field V, the termination map constructed from its ascending/descending manifolds as described above is denoted ω_f. Note that ω_f can be seen a the vertex-based (respectively face-based) segmentation of the domain induced by the Morse complex of $-f$ (respectively by the Morse complex of f).

Boundary Map

We use ω_f to identify those cells in K that are on the boundaries of ascending/descending d-manifolds. We can extend ω_f to construct a new label map ∂_f over all simplices that encodes the appropriate boundary information. In this case, $\partial_f : K \rightarrow \{0, 1, 2, 3\}$ and simply records whether a cell is the boundary of ascending d-manifolds, descending d-manifolds, both, or neither (as illustrated in Fig. 4.4):

$$\partial_f(\alpha) = \begin{cases} 0 & \text{if } \alpha \text{ is not a boundary cell} \\ 1 & \text{if } \alpha \text{ is a boundary between ascending } d\text{ -manifolds} \\ 2 & \text{if } \alpha \text{ is a boundary between descending } d\text{ -manifolds} \\ 3 & \text{if } \alpha \text{ is a boundary between both} \end{cases} \tag{4.1}$$

User Edits

Given the initial termination map ω_f, we allow the user to modify it directly into ω'_f. After each edit, the boundary map ∂'_f is recomputed. Then, the Morse-Smale complex $\mathscr{MS}(f')$ is computed from the gradient field produced by *ConformingGradient*(). If persistence simplification/filtering was used to generate $\mathscr{MS}(f)$, that same level of simplification is performed on $\mathscr{MS}(f')$. Practically,

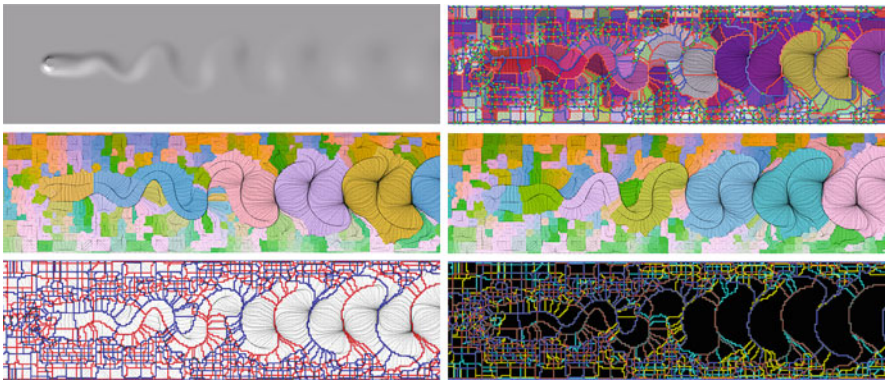


Fig. 4.4 Example of termination maps and boundary map for the vorticity field of the 2D von Kármán street (top left). The Morse-Smale complex of the data is first computed (top right) to compute the minima (middle, left) and maxima (middle right) termination maps. The separatrices of the Morse-Smale complex (bottom left) are used to construct the boundary map (bottom right). ©2014 IEEE. Reprinted, with permission, from [63]

the difference between $\mathscr{MS}(f)$ and $\mathscr{MS}(f')$ is restricted to the region of ∂'_f modified by the user. Note that user edits may force changes to the Morse-Smale complex in non-trivial ways, such as splitting regions and changing the connectivity. For edits such as moving the boundaries of regions, critical points are necessarily added when there is no monotonic path that can be found on the boundary.

An additional simplification rule must be observed to ensure that the edits are not immediately removed as low-persistence features: a critical point of $\mathscr{MS}(f')$ that is a boundary of ω'_f but not of ω_f can only be canceled with another similarly labeled critical point. Figure 4.5 illustrates this process for two simple examples. In the first (a–d), overlapping features cause a false negative to appear, where the finest resolution Morse-Smale complex image insufficiently segments the domain. The user modifies the termination map ω_f, and the complex is regenerated with the false negative corrected. In the next example (e–h), the geometric embedding of an arc of the Morse-Smale complex is fixed by modifying the spatial extents of a label in ω'_f.

A Histological Example

Applying Morse-Smale complexes for the automated segmentation of nuclei in phenotypic analysis of histological sections is an active area of research [109]. We apply a straightforward approach, identifying nuclei as simplified basins in a processed image. However, the basins identified often may not sufficiently segment the domain to separate all the nuclei. Our goal is to remove all false negatives using

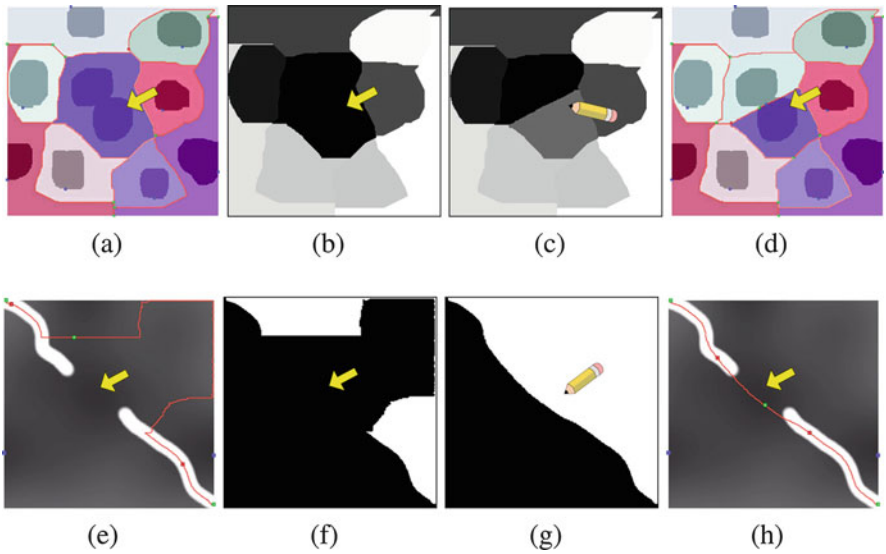


Fig. 4.5 The Morse-Smale complex is computed for a simple example showing dark blobs on a light background (**a**). Two overlapping features are identified as part of the same basin. This Morse-Smale complex generates a termination map (**b**), that can be edited by a user (**c**) to split this region. The edited map is used to generate a new MS complex (**d**) with the false negative corrected. Similarly, a light ridge-like structure (**e**) is disconnected, resulting in a ridge reconstruction with poor geometric embedding. The termination map (**f**) is again edited (**g**) to reconstruct the desired embedding (**h**) ©2014 IEEE. Reprinted, with permission, from [63]

a semi-automated approach. As Fig. 4.6 illustrates, we apply grayscale conversion, thresholding, and smoothing to the initial image data, and compute basins of the Morse-Smale complex simplified to some persistence threshold.

This pipeline results in some false negatives where two or more nuclei overlap, as compared to a labeling by a domain expert [137]. The computed Morse-Smale complex segmentation is used to generate a termination map ω_f, which is edited to match the input of a domain expert. The boundary ∂'_f is computed from ω'_f, and finally the complex $\mathscr{MS}(f')$ is recomputed and simplified, with the new features present. While the Morse-Smale complex that results contains no false negatives, it still contains false positives. However, as it is still a valid Morse-Smale complex, further filtering/simplification is possible to account for such errors. Having every nuclei in its own basin makes it later possible to identify which basins contain nuclei and which do not, for instance, by applying standard filtering/simplification pipelines.

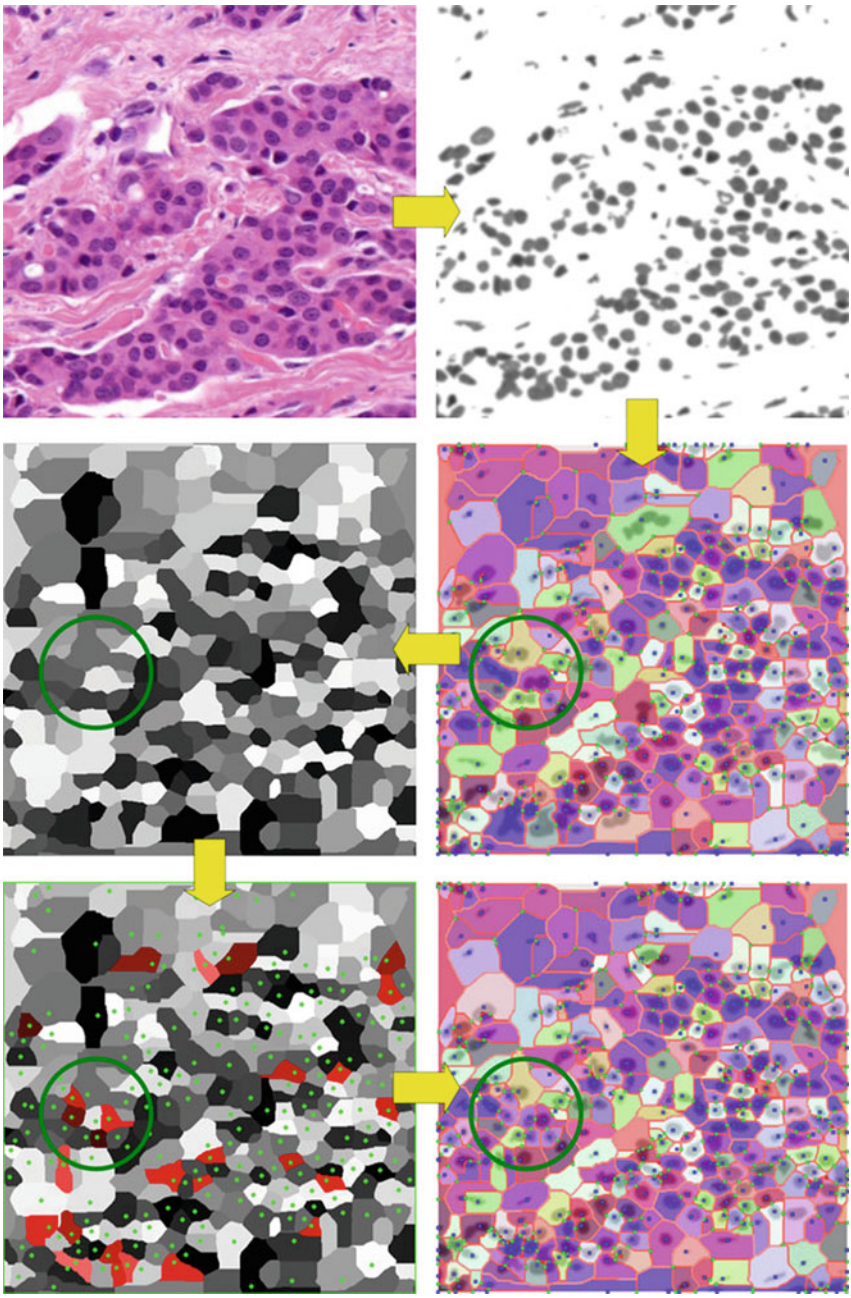


Fig. 4.6 A section of tissue is imaged in a histology study. We apply grayscale conversion, thresholding, and smoothing to generate an input for an initial MS complex computation. The termination map is edited to match a domain expert's labeling (red regions, bottom left), and a complex is regenerated that has no false negatives. The green circle highlights a region that is corrected through editing. ©2014 IEEE. Reprinted, with permission, from [63]

Concluding Remarks

This sub-section introduced an algorithm to construct Morse-Smale complexes that conform with a user-supplied segmentation. integration strategies, is investigation. An investigation of the breadth of interactions that must be supported and user interface requirements to make such an approach a viable component of the analyst's toolbox is future work. Additionally, an interesting direction for future work is to restrict the (re-)computation necessary for a local region of influence around the modified segmentation, to enable instantaneous updates of the Morse-Smale complex. Furthermore, editing may introduce non-local effects both in the feature generation and simplification process, and fully exploring these in a robust manner would be a valuable future contribution.

Finally, another application area for future work involves robust mesh generation. The Morse-Smale complex has started to emerge as vehicle for surface meshing [37, 68]. We view conforming Morse-Smale complexes as a first step towards utilizing Morse-Smale complexes for volumetric mesh generation. User edits with a rich segmentation interface could drive the repair of mesh artifacts interactively, similar to the approach presented in the following sub-section.

4.2.2 Reeb Graph Editing

In Computer Graphics, surface quadrangulation representations are often preferred over triangulations for tasks such as texture mapping or animation. Quadrangulations can be obtained by partitioning the surface in a set of quadrangular charts (which can be further refined as desired). However, end-users need to control the overall layout and topology of this partitioning (chart number and boundary alignment, position, valence and number of extraordinary vertices) as it can affect the output animation.

In this subsection, we describe an approach for the semi-automatic segmentation of surfaces into quadrangular charts, based on editing mechanisms of the Reeb graph. Given an initial segmentation automatically generated from the pre-image of $\mathscr{R}(f)$ through ϕ, we introduce atomic operations for the control of the number of charts, the geometrical control of their boundaries, as well as the robust control of the number and valence of chart corners. Each of these operations is accompanied with an intuitive user interface and our core algorithms perform edits at interactive rates, enabling users to constructively enhance the quadrangular segmentation of the surface for quad-meshing purposes (further described in the next chapter).

Harmonic Scalar Fields

In order to control the geometry of the boundaries of the quadrangular charts, we will consider as an input scalar field the solution to an optimization process,

integrating user constraints for the definition of the geometry of the level sets. Moreover, since it is often desirable in practice to construct quadrangular charts with smooth boundaries, we will add a regularization term in the optimization.

The solution to the Laplace equation under Dirichlet boundary conditions is a good candidate for such requirements, since it smoothly interpolates sparse constraints on the entire domain. Given a finite set of extrema constraints C along with corresponding target values, this equation is defined as follows:

$$f(c_i) = f_{c_i} \qquad \forall c_i \in C \tag{4.2}$$

$$\Delta f(v) = 0 \qquad \forall v \notin C \tag{4.3}$$

where Δ stands for a discretization of the Laplace-Beltrami operator on surfaces. The scalar fields being solutions of this equation are called *harmonic scalar fields*.

In practice, to compute such harmonic scalar fields, the Laplace-Beltrami operator is discretized using cotangent weights [97], which leads to a symmetric positive-definite sparse matrix $L = W - D$ whose elements w_{ij} of W are defined as:

$$w_{ij} = \begin{cases} -\frac{1}{2}(\cot \alpha_{ij} + \cot \beta_{ij}) & \text{if edge } [i,j] \in \mathscr{M} \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

where α_{ij} and β_{ij} are opposite angles to edge e_{ij} and D is a diagonal matrix with elements d_{ii} given by row sums of W.

We make use of the penalty method to impose constraints to the linear system derived from Eq. (4.3). Consider C, the set of indices of constrained vertices, then the harmonic scalar field is obtained by solving the linear system,

$$(L + P)f = Pb, \tag{4.5}$$

where P is a diagonal matrix with non-zero entries $p_{ii} = \alpha$ only if $i \in C$ and α is the penalty weight ($\alpha = 10^8$ [138]). Constrained values are set within the vector b,

$$b_i = \begin{cases} f_{c_i}, & \forall c_i \in C \\ 0, & \forall v \notin C \end{cases} \tag{4.6}$$

where f_{c_i} is the desired scalar value assigned to vertex c_i. The main advantage of using the penalty method to impose constraints is that supernodal schemes [31] can be used to update (and downdate) the Cholesky factorization, making it possible to include and remove constraints efficiently [138], at interactive rates. Figure 4.7 shows an example of harmonic scalar field along with its corresponding Reeb graph.

In practice, the set of extrema constraints can be either provided by the user, or selected as extrema of relevant functions computed automatically, such as the integral of the geodesic distance function [67].

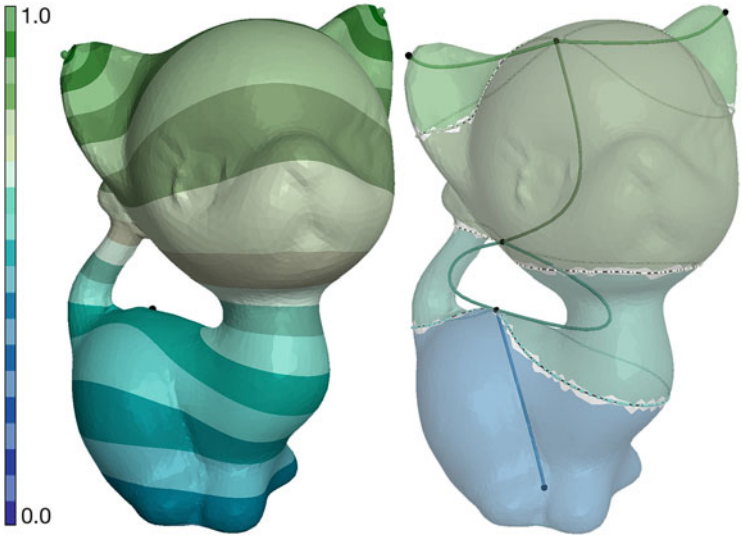


Fig. 4.7 Harmonic scalar field (left, extrema constraints are located on the ears, top, and at the bottom) and its Reeb graph (right) as well as its pre-image through ϕ (transparent chart colors, right). ©2012 IEEE. Reprinted, with permission, from [128]

Reeb-Graph Based Surface Segmentation and Parameterization

Given a closed PL 2-manifold $\mathscr{M}$ embedded in $\mathbb{R}^3$, a *Reeb chart* is the pre-image by ϕ of the interior of a 1-simplex of $\mathscr{R}(f)$. By construction, Reeb charts are continuous pilings of closed 1-dimensional contours. Since they are the pre-image of the *interior* of 1-simplices, Reeb charts do not include critical contours and are thus open sets with the topology of an open annulus (a connected genus zero surface, with two boundary components excluded). Note that a boundary component collapses to a point if a 1-simplex is linked to the image through ϕ of an extremum. Because Reeb charts are constructed from the regular contours of f, their definition does not require f to be strictly PL Morse (i.e. degenerate saddles are allowed).

Given the segmentation of the surface into Reeb charts, the *Reeb atlas* is defined as the union of the charts with respective local parameterizations. Because Reeb charts have a controlled topology, they are robustly, easily and efficiently parameterized with a generic strategy, which will be instrumental for quadrangulation purposes.

Each Reeb chart $\mathscr{M}_i$ of $\mathscr{M}$ is built by duplicating the triangles of $\mathscr{M}$ that fully map to the *interior* of a 1-simplex σ_i via ϕ. *Boundary triangles*, intersected by the critical contours whose image by ϕ is a face of σ_i, are also inserted into $\mathscr{M}_i$, illustrated as grey triangles in Fig. 4.7 (right). In order to obtain smooth boundary components for the charts, the *boundary triangles* are shrunk such that their vertices being outside of the chart get snapped along the boundary critical contour (by sliding them along an incoming edge crossing the contour).

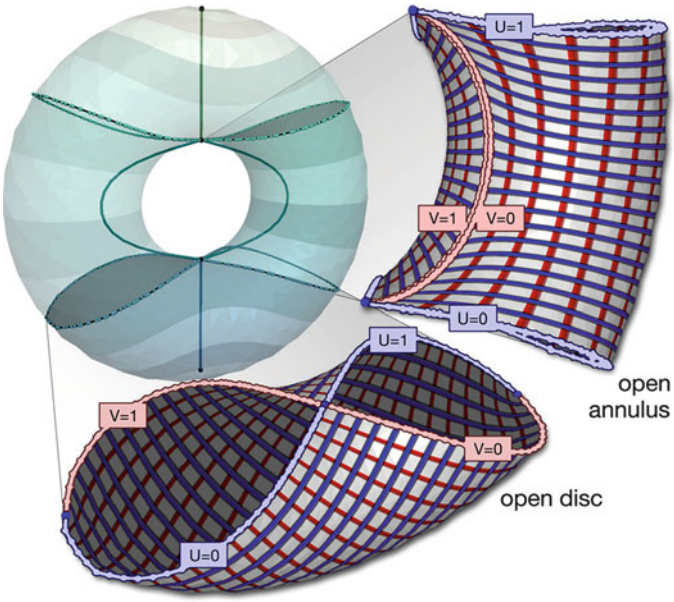


Fig. 4.8 Our parameterization strategy maps the boundaries of the Reeb charts to the unit square by defining UV Dirichlet boundary conditions within the Laplace system. Open annuli are cut into quadrangular charts by an integral line of f. ©2012 IEEE. Reprinted, with permission, from [128]

A parameterization maps the open annulus $\mathscr{M}_i$ to the unit square by solving two harmonic functions with Dirichlet boundary conditions (Fig. 4.8) using the solver presented previously. The field $U : \mathscr{M}_i \rightarrow [0, 1]$ is computed to align with the level lines of f by constraining the boundary vertices of $\mathscr{M}_i$, projected to the two critical contours, to either $U = 0$ or $U = 1$ (Fig. 4.8). The orthogonal field $V : \mathscr{M}_i \rightarrow [0, 1]$ is computed by tracing a cutting integral line along the mesh edges of $\mathscr{M}_i$ guided by the gradient of U, turning the annulus into a quadrangular chart with disc topology. The vertices of the cutting edges are duplicated and assigned values, $V = 0$ and $V = 1$, to map the boundary of $\mathscr{M}_i$ to the unit square.

Each Reeb chart $\mathscr{M}_i$ mapping through ϕ to a 1-simplex σ_i of $\mathscr{R}(f)$ having as face the image by ϕ of an extremum of f (Fig. 4.8) are parameterized differently. The boundary triangles adjacent to the extremum are included within $\mathscr{M}_i$ so that $\mathscr{M}_i$ has a single boundary component and is homeomorphic to a disc. The boundary vertices are segmented into four contiguous polylines and assigned values mapping the boundary to the unit square.

At this stage, the Reeb atlas represents a coarse quadrangular segmentation of the surface. Moreover, each Reeb chart is equipped with its own local parameterization to the unit square, enabling further quadrangular subdivisions. Note that saddles of f can be seen as extraordinary vertices in this segmentation (i.e. vertices whose valence is different from 4).

Table 4.1 Summary list of the Reeb atlas editing operations

Operation	Add chart	Del chart	Move bound.	Add bound.	Saddle align.	Frac. poles	Frac. saddles
Figure	Fig. 4.10 (top)	Fig. 4.10 (top)	Fig. 4.10 (mid.)	Fig. 4.10 (bot.)	Fig. 4.11	Fig. 4.12	Fig. 4.13
Interaction	Click	Click	Drag	Click	Clicks	Clicks	Clicks + Drag

Editing Operations

In the following, we describe a set of atomic operations (detailed in Table 4.1) on the Reeb graphs for the interactive control of the initial quadrangular segmentation provided by the Reeb atlas.

Chart Boundary

The Reeb chart boundaries are defined by critical contours of f. While the relocation of minima and maxima is well understood, consisting of removing the original constraint and replacing it with a new one at a different location, moving saddle contours requires a bit more machinery but it is however mandatory to edit the geometry of charts' boundaries.

For each saddle contour, additional constraints are added to the Laplace system at the vertices of *saddle triangles* (triangles intersected by the critical contour) to ensure that the scalar field level-sets respect the user designed geometry. Assume that the user modified the geometry of a saddle contour with a scalar value s_c so as to intersect the triangle $t = \{p_0, p_1, p_2\}$ on edges $e_0 = \overline{p_0 p_1}$ and $e_2 = \overline{p_2 p_0}$ (Fig. 4.9). The intersection points $p_{in} = p_0 + \alpha_0(p_1 - p_0)$ and $p_{out} = p_2 + \alpha_1(p_0 - p_2)$ and scalar values $s_{in} = s_0 + \alpha_0(s_1 - s_0)$ and $s_{out} = s_2 + \alpha_1(s_0 - s_2)$, where s_0, s_1, s_2 are associated with the vertices of t, assist in the definition of vertex constraints. The vertex p_i of t is projected onto the segment $\overline{p_{in} p_{out}}$ yielding the point p'_i with a scalar value s'_i. The scalar constraint assigned to p_i for t is $\tilde{s}_i = s_c + (s_i - s'_i)$. The final constraint of each vertex is averaged with values of adjacent saddle triangles. This novel constraint computation enables strict control of the contour of f, aligning to the user's designed polyline.

Note that initially, and also after each editing operation, *all* the saddle contours of f are constrained using the above scheme (even if the contours are not displaced). Then, the effects of the editing operations are localized to the charts of interest. For instance, when displacing a saddle contour at the boundary of a chart of interest, the other boundaries remain in position and conserve their f values. Then to guarantee that the saddle contour displacement does not alter the topology of f, our interface discards any interaction that generates a contour which is not a closed loop, or that makes the contour overlap another saddle contour or sweep an extremum. Finally, note that after each editing operation, the Reeb graph is recomputed globally. Then, the list of Reeb charts which need an update of their individual triangle list is tracked

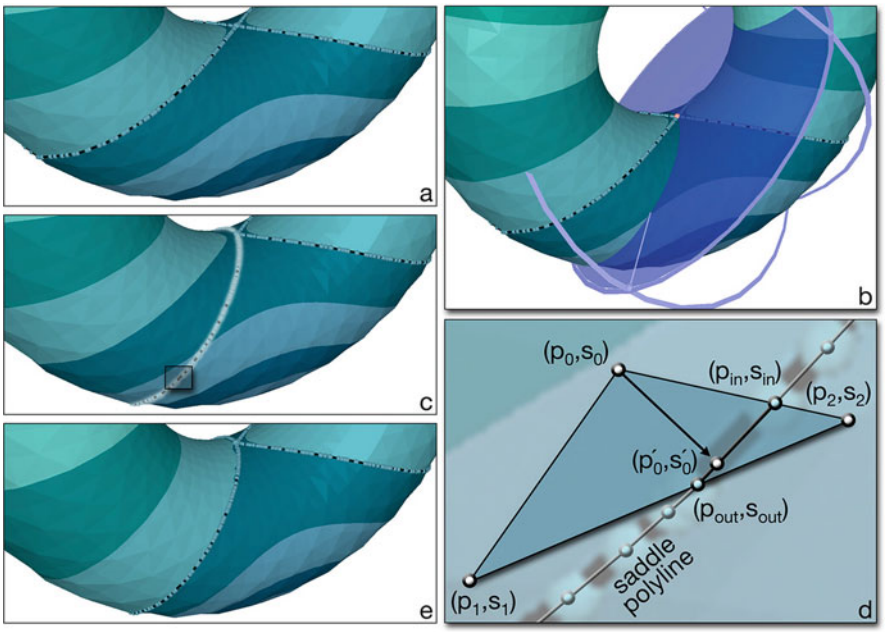


Fig. 4.9 Editing a chart boundary by saddle contour manipulation: the original level-set curve (**a**) is modified through the critical contour widget (**b**) where the mesh-plane intersection describes the new contour geometry (**c**). Saddle triangles are constrained (**d**) to ensure the scalar field respects the new critical contours (**e**) ©2012 IEEE. Reprinted, with permission, from [128]

based on the lists of regular vertices of the arcs of the Reeb graph (as described in Fig. 4.10, center).

Chart Number

The number of charts can also be edited by the user through a set of atomic operations for the insertion, deletion, subdivision and merging of Reeb charts.

As illustrated in Fig. 4.10 (top), disc-like Reeb charts can be inserted or deleted by inserting or deleting extrema constraints from the Dirichlet boundary conditions.

Reeb charts can also be subdivided. In particular, because the Reeb chart is defined as a collection of contours, splitting a chart into two can be achieved by flagging a particular contour (i.e. clicking on a vertex, Fig. 4.10, bottom) and by construction each child chart maintains the topological guarantees of the Reeb atlas segmentation. Reeb chart splitting facilitates alignment of the scalar field, the charts' parameterizations and consequently of the final quadrangulation to surface features.

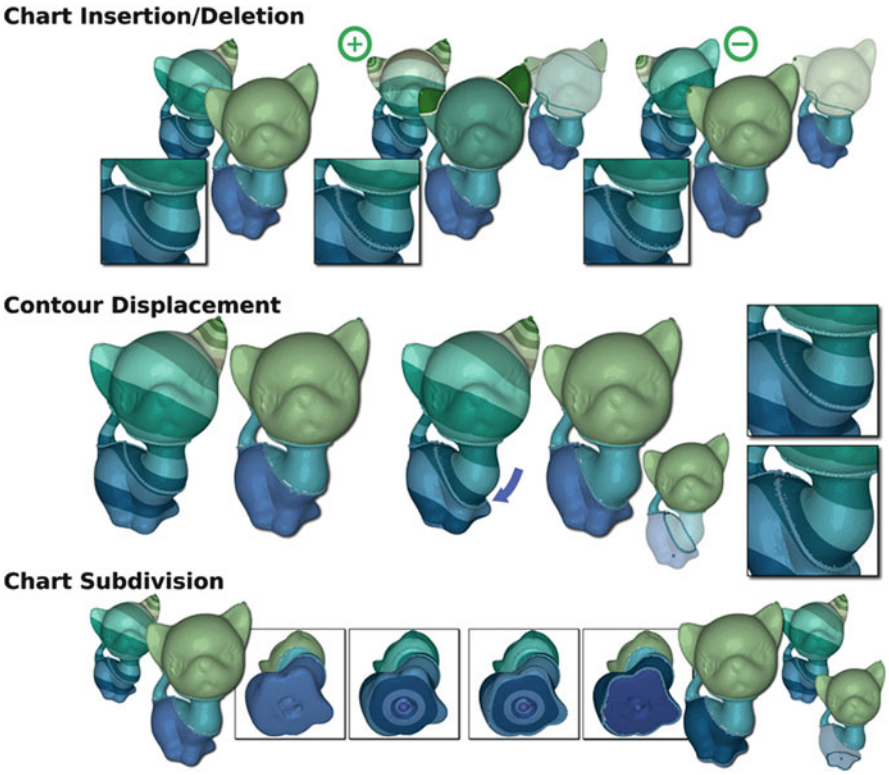


Fig. 4.10 Global impact of Reeb atlas editing operations. Reeb charts whose triangle list changes after the operation are transparent. Top: chart insertion and deletion (through insertion and deletion of extrema). Middle: contour displacement. Bottom: chart subdivision. ©2012 IEEE. Reprinted, with permission, from [128]

When the scalar field f admits a succession of nearby saddles (Fig. 4.11), it may be desirable to align the associated critical contours. In effect, this functionality coarsens the Reeb atlas by removing thin Reeb charts to align extraordinary vertices in the final quad mesh. The atlas coarsening maintains the total number of saddle vertices while decreasing the number of saddle contours (multiple critical points are enforced to share the same function value, forcing f not to be a PL Morse scalar field), yet the Reeb charts remain well defined. Aligning multiple saddles (Fig. 4.11) is achieved interactively by first deleting the critical contours to align. Next, the user clicks on pairs of saddles to be connected with automated mesh traversals (shortest paths) providing initial curve segments. Then, the user can further re-orient them with the critical contour widget (the aligned curves are then constrained as discussed previously).

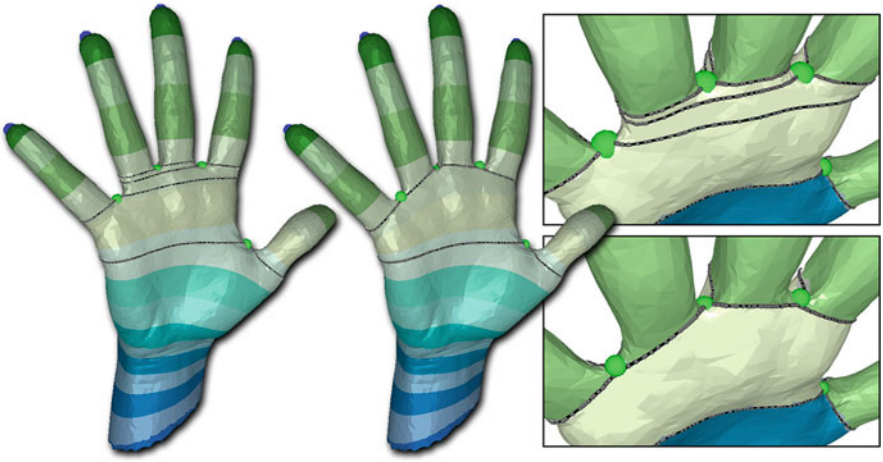


Fig. 4.11 *Thin Reeb charts* (left and top) result where multiple saddles have nearly equivalent scalar values. Our global editing operations support the geometric control of the contours, linking the saddle vertices and removing thin Reeb charts (right and bottom). ©2012 IEEE. Reprinted, with permission, from [128]

Chart Corners

The Reeb atlas provides a coarse quadrangular segmentation that can be refined by contouring the parameterization of each Reeb charts and locally solving potential t-junctions [128]. With this strategy, the corners of each Reeb chart will yield extraordinary vertices (i.e. with a valence different from 4). Here we describe how such vertices can be controlled with atomic edits with the notion of *fractional singularity*.

When a boundary component of a Reeb chart is an extremum vertex, parameterizing the chart with a cutting streamline (as an open annulus) generates a polar singularity that leads to triangular elements around a high valence extraordinary vertex (Fig. 4.12, left). To guarantee the generation of a quad-only output, we use the notion of *fractional singularity*. In particular, our default parameterization strategy for disc charts splits a polar singularity into quarter poles, where the resulting quad mesh contains 4 valence 3 vertices (Fig. 4.12, right).

An alternative proposed to the user is to split the polar singularity into 2 half-poles, constraining a sequence of mesh edges with constant min/max f values (0 or 1); then, the chart is parameterized with a cutting streamline (Fig. 4.12, middle). This configuration corresponds to the concept of *non-isolated critical points* in the smooth setting. We use *Simulation of Simplicity* (*SoS*) [43] in the PL setting to maintain a consistent combinatorial representation of f. The resulting quad mesh has 2 valence-2 extraordinary vertices at the endpoints of the extremum segment.

In the spirit of handling fractional polar singularities, we design fractional saddle singularities within the scalar field design. Saddle vertices correspond to

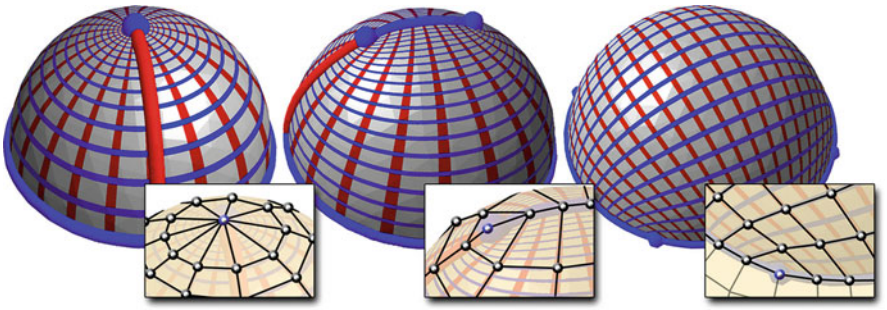


Fig. 4.12 Fractional poles: a polar vertex (left) split into 2 half- poles (val. 2, middle) and 4 quarter-poles (val. 3, right). ©2012 IEEE. Reprinted, with permission, from [128]

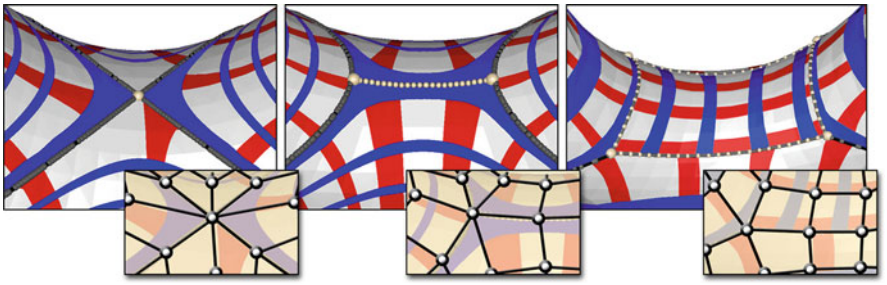


Fig. 4.13 Fractional saddles: a saddle vertex (val. 8, left) split into 2 half-saddles (val. 6, middle) and 4 quarter-saddles (val. 5, right). ©2012 IEEE. Reprinted, with permission, from [128]

extraordinary vertices within the final quad mesh (Fig. 4.13, left). We provide a set of atomic editing operations that enable the user to redistribute easily the high valency of saddles with the notions of *half-* and *quarter*-saddles. While there exists multiple possible combinations of adjacent Reeb chart parameterization configurations, we abbreviate this discussion to the example shown in Fig. 4.13. A non-degenerate saddle contour is a set of two closed curves admitting exactly one common point. Half-saddle splitting is supported by modifying the geometry of the saddle contour to be described, for example, with two closed curves linked by a *middle segment* that is aligned to the edges of the mesh. The half-saddles are defined at the intersection of the middle segment and the two closed curves (Fig. 4.13, middle).

To design half-saddle configurations (Fig. 4.13, middle), the user deletes the original saddle contours and vertex, then initiates the tracing of two contours from manually chosen vertices. The middle segment is automatically computed as the shortest path defined along mesh edges between the two points. User-defined half-saddle contours can be geometrically edited via the critical contour widget to align to surface features. The network of critical contours defining the half-saddle is assigned a single constraint isovalue. Note, the middle segment relates to the notion of non-isolated critical point in the smooth setting, handled in the PL setting with

SoS. Splitting a saddle reduces the valence of the related vertex by redistributing it among the multiple, created extraordinary vertices. The quarter-saddle configuration (Fig. 4.13, right) further supports this observation. Quarter saddles are designed by first deleting the original saddle contour and vertex. Then the user clicks on a reference vertex to extract its isocontour. Three other reference vertices are selected along this isocontour and pairs of reference vertices are connected through shortest path computations (Fig. 4.13, right). Finally, an extremum is inserted at the location of the original saddle to maintain a valid field topology. The user can use the critical contour widget to further align the contour (constrained as discussed previously).

Preliminary Results

Figure 4.14 illustrates a few preliminary quadrangulation results obtained after Reeb atlas editing and a direct contouring of the Reeb chart parameterization. In particular, the hand example (center, top) has been obtained after saddle alignment

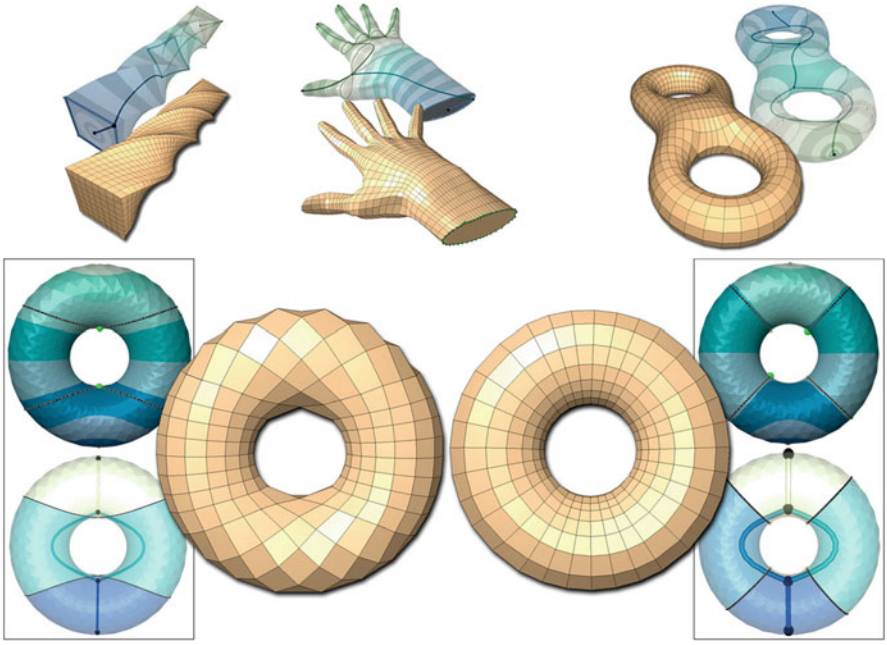


Fig. 4.14 Preliminary quadrangulation results obtained after Reeb atlas editing and a direct contouring of the Reeb chart parameterizations. ©2012 IEEE. Reprinted, with permission, from [128]

and chart subdivision at the finger tips. The torus example (bottom) illustrates the effects of fractional singularity editing (left: before, right: after). In particular, due to our default parameterization method for discs, splitting polar vertices into quarter-poles, the construction of half-saddles can lead to the removal of pairs of extraordinary vertices in the quad mesh. In particular, the singularities of *min/split-saddle* and *merge-saddle/max* Reeb charts are removed. On the torus examples, all singularities can be classified as these types, resulting in a completely regular quadrangulation (right). The bitorus example (top right) further exemplifies fractional saddle editing.

Concluding Remarks

In this subsection, we described an interactive framework for the complete editing of the geometry and topology of a surface segmentation derived from the Reeb graph, with a specific application to surface quadrangulation. Beyond this application, our Reeb atlas interactive framework also opens new possibilities to incorporate users knowledge into segmentation tasks in scientific visualization, as described in the previous subsection with the Morse-Smale complex but this time for segmentation tasks where features of interest align with level sets.

Chapter 5
Analysis

5.1 Exploration of Turbulent Combustion Simulations

In this section, we describe an analysis framework for large-scale time varying combustion simulations based on hierarchical split trees. This data abstraction enables a massive data reduction (two orders of magnitude) while still abstracting the relevant information for the representation of the features of interest. This reduction enabled the design of a user interface for the interactive exploration of the features of the simulation, yielding new visual insights later confirmed by quantitative analysis.

5.1.1 Applicative Problem

Low-swirl injectors [9, 25, 85, 88, 94] are emerging as an important new combustion technology. In particular, such devices can support a lean hydrogen-air flame that has the potential to dramatically reduce pollutant emissions in transportation systems and turbines designed for stationary power generation. However, hydrogen flames are highly susceptible to various fluid-dynamical and combustion instabilities, making them difficult to design and optimize. Due to these instabilities, the flame tends to arrange itself naturally in localized cells of intense burning that are separated by regions of complete flame extinction.

Figure 5.1a shows the detail of a low-swirl nozzle. The annular vanes inside the nozzle throat generate a swirling component in the fuel stream. Above the nozzle the resulting flow-divergence provides a quasi-steady aerodynamic mechanism to anchor a turbulent flame. Figure 5.1b illustrates such a flame for a lean premixed CH_4-air mixture (the illustration shows a methane flame since H_2 flames do not emit light in the visible spectrum). Figure 5.1c, d show typical experimental data from laboratory low-swirl, lean H_2-air flames. Such data is used to extract the

J. Tierny, *Topological Data Analysis for Scientific Visualization*,
Mathematics and Visualization, https://doi.org/10.1007/978-3-319-71507-0_5

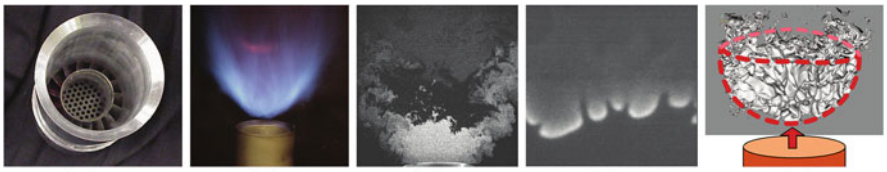


Fig. 5.1 Turbulent combustion applicative context (from left to right): (**a**) Photo of a typical laboratory low-swirl nozzle; (**b**) Photo of a lean premixed CH_4 low-swirl flame; (**c**) Experimental Mie scattering image of a lean premixed H_2 flame; (**d**) Planar laser-induced fluorescence data imaging the OH concentration in a lean premixed H_2 flame; (**e**) Rendering of the burning cells of the SwirlH2 simulation data. The cells form a bowl shaped structure with the arrow indicating the direction of the fuel stream. ©2011 IEEE. Reprinted, with permission, from [17]

mean location and geometrical structure of instantaneous flame profiles. The images indicate highly wrinkled flame surfaces that respond in a complex way to turbulent structures and cellular patterns in the inlet flow-field.

Existing approaches to analyze the dynamics of flames, including most standard experimental diagnostic techniques, assume that the flame is a connected interface that separates the cold fuel from hot combustion products. In cellular hydrogen-air flames, many of the basic definitions break down: there is no connected interface between the fuel and products, and in fact there is no concrete notion of a "progress variable" that can be used to normalize the progress of the combustion reactions through the flame. As a consequence, development of models for cellular flames requires a new paradigm of flame analysis.

The computational model used to generate the simulation results explored in this study incorporates a detailed description of the chemical kinetics and molecular transport, thus enabling a detailed investigation of the interaction between the turbulent flow field and the combustion chemistry. As in the physical device, the low-swirl burner simulation achieves a statistically stationary flame in a time-dependent turbulent flow field above the inlet nozzle. Results from the simulation are in the form of a sequence of snapshots in time of the state data. We considered two simulations (labeled SwirlH2 and SwirlH2Fast) having different flow profiles. The SwirlH2Fast case has a mean fueling rate of 2.5 times that of the SwirlH2 case. In the simulations, the time-dependent integrated inventory of fuel in the domain is used to monitor the developing flame. Once a quasi-steady configuration is obtained, snapshots were collected at intervals of approximately 2 and 1 ms for SwirlH2 and SwirlH2Fast, respectively and used for the analysis here. The data sets consist of 332 and 284 snapshots for the slow and fast version, respectively, at an effective resolution of 1024^3. The resulting snapshots are roughly 12–20 GB in size totaling a combined 8.4 TB of raw data.

The main features of interest are the intensely burning cells defined by a threshold on the local fuel consumption rate. All regions with a local fuel consumption rate above this threshold are tagged as "burning." Note, however, that no single "correct" threshold exists, requiring that we characterize the influence of this threshold value on the resulting diagnostics. However, previous approaches for the analysis of this

type of data relied on the identification of a single consumption rate threshold and no approach was available to interactively explore through time and characterize quantitatively the geometry of burning flames for interactively defined thresholds. The work described in this section addresses this issue.

5.1.2 Algorithm

The cellular regions of intense burning are identified by thresholding the local consumption rate and we work directly with the resulting three-dimensional, time-dependent regions. The process may be regarded as a generalized subsetting strategy, whereby subregions of a computational result may be sampled, explored and categorized in terms of a volume of space with an arbitrary application-specific definition for its boundary. Since we are interested in regions of high fuel consumption we have identified split trees which encode the topology of sur-level sets, see Chap. 2, as appropriate data structure.

Data Segmentation

As discussed in Sect. 2.2.3, the Reeb graph and its variants (the join, split and contour trees) induce data segmentations with boundaries aligning with level sets, by considering the pre-image through ϕ of each of their 1-simplices. In this application, since burning flames are defined as connected components of space whose combustion rate is higher than a queried threshold $t \in \mathbb{R}$, we follow and extend this segmentation strategy in the case of the split tree.

Given a threshold, t, for the fuel consumption rate $f : \mathscr{M} \to \mathbb{R}$, we determine the corresponding burning cells by conceptually cutting the split tree of f at t creating a forest of trees. Each individual tree represents one connected burning cell, see Fig. 5.2e. In practice, rather than cutting the split tree and traversing sub-trees, the

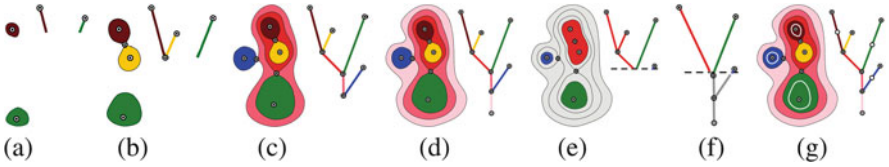


Fig. 5.2 (**a**)–(**d**) Constructing a split tree and corresponding segmentation by recording the merging of contours as the function value is swept top-to-bottom through the function range. (**e**) The segmentation for a particular threshold can be constructed by cutting the split tree at the threshold, ignoring all pieces below the threshold and treating each remaining (sub-)tree as a cell. (**f**) The segmentation of (**e**) constructed by simplifying all saddles above the threshold. (**g**) The split tree of (**d**) augmented by splitting all 1-simplices spanning more than a given range. ©2011 IEEE. Reprinted, with permission, from [17]

same information is stored more efficiently as a simplification sequence. A split tree is simplified by successively merging leaf branches with their sibling branch. We order these simplifications by decreasing function value of the merge saddles and store the resulting simplification sequence. In this framework, burning cells at threshold t are defined as sets of all leaf branches with function value greater than or equal to t of the tree simplified to t, see Fig. 5.2f.

Storing only the split tree, the scheme described above allows us to determine the number of burning cells for all possible thresholds. However, in practice we also need an accurate representation of cell geometry and of any number of additional attributes such as volume (see below). Using only the original segmentation this is difficult since we must exclude the lower portions of branches intersecting the threshold. As the distribution of branch attributes can, in general, not be predicted, excluding a portion of a branch would require us to access the original data at significant cost. Instead, we augment the split tree with additional valence two nodes by splitting all branches longer than some threshold, as seen in Fig. 5.2g. Furthermore, we also compute a number of additional attributes for each branch. For example, we compute the volume of each branch as well as a number of kth order moments such as means and variances for any variable of interest (not necessarily just f). This splitting is performed with little overhead during the initial computation and allows us to approximate cutting at any threshold with a pre-defined accuracy. It is important to note that this approximation only affects the geometry of the segments and their attributes but not their structure. We are guaranteed to not erroneously merge or split cells due to the approximation.

The augmented split trees form the fundamental data structure in our framework, storing the one-parameter family of possible segmentations along with an arbitrary number of attributes. For efficient access during the visualization we store the segmentation information, a list of vertices per cell, separately.

Feature Tracking

Given the data segmentations for all time steps, we track over time features defined by a given static threshold. We track features by spatial overlap which appears to be adequate for our purposes. However, since we have a complete description of features the framework can be easily extended to any number of more sophisticated techniques. To determine the potential overlap we load the split trees of two consecutive time-steps and adapt them to the given threshold. We then traverse the vertices of both segmentations in parallel determining their active segmentation index and if both vertices are above the threshold add a (tracking graph) arc between the corresponding features. The active segmentation index is computed as the segmentation index stored in the file adapted to threshold simplification.

Due to the large number of timesteps involved, creating a tracking graph cannot yet be performed interactively. Furthermore, creating a layout for a given graph, even using state of the art tools, remains too slow for interactive techniques. However, it is important to point out that the tracking graphs are created from the

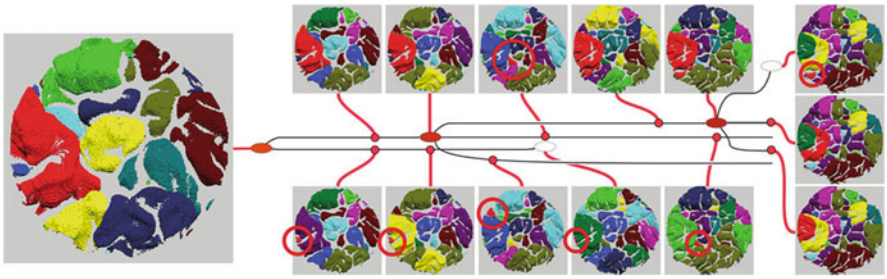


Fig. 5.3 Example of burning cells being tracked over time (in red). The graph shows a small portion of the tracking graph for the SwirlH2 data set spanning time steps 1880 through 1895. The embedded screen shots show the corresponding segmentation as different nodes are selected in the graph. Over the course of these time steps the large cell represented by the left most node slow sheds smaller cells until it finally breaks apart into three independent pieces. ©2011 IEEE. Reprinted, with permission, from [17]

pre-computed segmentations and not the original data so all processing involved can easily be handled by a standard desktop computer.

For regular grids the tracking is by design a streaming process. Each vertex is treated independently and since the vertices are in the same order for both time steps only two consecutive split trees must be kept in memory. For each time interval we dump the partial tracking graph to disk to be assembled at the end.

An example of features getting tracked through time is shown in Fig. 5.3. The figure shows a small portion of the tracking graph for the SwirlH2 data set for time steps 1880–1895. The embedded screen shots show the main segmentation display when the indicated node has been selected.

Tracking Graph Simplification

As illustrated in Fig. 5.3, the tracking graphs can become highly complex and difficult to understand. Furthermore, they contain artifacts of the thresholding such as tiny features existing for only one or very few time steps. To reduce the graph complexity and eliminate some of the artifacts we simplify the tracking graphs by removing all valence zero nodes as well as nodes with a volume smaller than a given threshold (in practice we use the number of vertices corresponding to each node as a measure of volume). Such simplification significantly streamlines the tracking graph by suppressing unnecessary details. In order to avoid disconnecting segments above the volume threshold and thus structurally changing the tracking graph we restrict the simplification to successively removing leafs below the volume threshold. To choose an adequate volume threshold we study how the number of nodes in the tracking graph changes as we increase the volume threshold, see Fig. 5.4. These plots indicate a separation between noise and features. To reduce the complexity of the graphs and the cost of the layout we chose values on the upper range of the

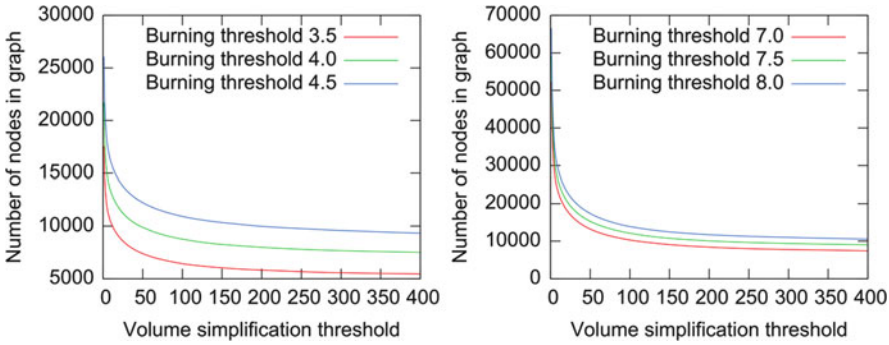


Fig. 5.4 Determining the tracking graph simplification threshold: The graphs show the number of nodes remaining in the tracking graphs vs. the simplification threshold for the restricted portions of the SwirH2 (left) and SwirlH2Fast (right) data set. ©2011 IEEE. Reprinted, with permission, from [17]

suggested noise level for simplification. All examples shown here use a threshold at around 100 vertices.

5.1.3 Results

The data reduction with our hierarchical split tree representation was performed in parallel on an SGI Altix 350, with 32 Itanium-2, 1.4 GHz processors using one processor per time step. The interactive exploration of the data and its quantitative analysis was carried out on a commodity desktop computer. For a single representative time step, the computation of the hierarchical split tree and corresponding segmentation took 1067 and 2684 s for the SwirlH2 and SwirlH2Fast data-sets respectively (excluding I/O). The hierarchical split trees pre-process the data with respect to one of the most important aspects of the data (the burning cells) and store all additional information in accordance with this segmentation, allowing for an interactive post-exploration. Even for the largest data sets, the resulting split trees consist of only around 6 Mb ASCII information per time step and their corresponding segmentation to 144 Mb, compared to several Gigabytes of raw data. In fact, the trees are small enough to be loaded interactively from disk. Using standard gzip compression, these reduce to roughly 70 Mb. Overall, for all time-steps, our data representation roughly required 13 and 20 Gb of gzipped files for the SwirlH2 and SwirlH2Fast case, respectively. Given the 3.9 and 4.5 TB of original data, this corresponds to a data reduction of more than two orders of magnitude, while still providing greater flexibility in the segmentation and selection than possible using standard techniques based on isosurfacing for instance.

Interactive Exploration Interface

The primary focus of this work was to provide the application scientists with the ability to comfortably explore their data in a manner meaningful in their particular problem space. For example, allowing the user to explore easily variables conditioned on the extracted features provides a simple way to understand whether various conditional statistics may provide new insights into the data.

Graph Display

Our user interface presents a fully linked system in which the user can explore the tracking graph, the corresponding segmentation, and the conditional statistics simultaneously with on-demand data loading, as illustrated in Fig. 5.5. One of our two main windows (V) is dedicated to the display of the tracking graph (layout with *dot*.) To reduce the visual clutter, only the non-valence two nodes of the tracking graph are shown while sequences of valence two nodes are indicated by unbroken arcs. For exploration we typically use the cell volume (represented by the number of vertices within the cell) to highlight larger cells. To display the graph, we load its geometry into OpenGL, which allows us to draw even the largest graphs fully interactively. The graph display not only provides a visualization of the graph but the user can also select nodes or arcs. When selecting an arc, the system automatically selects the closest valence two node along this arc. A selection triggers two actions. First, the system loads the split tree of the corresponding time step and, if desired, a number of neighboring time steps. Since split trees are comparatively small, the

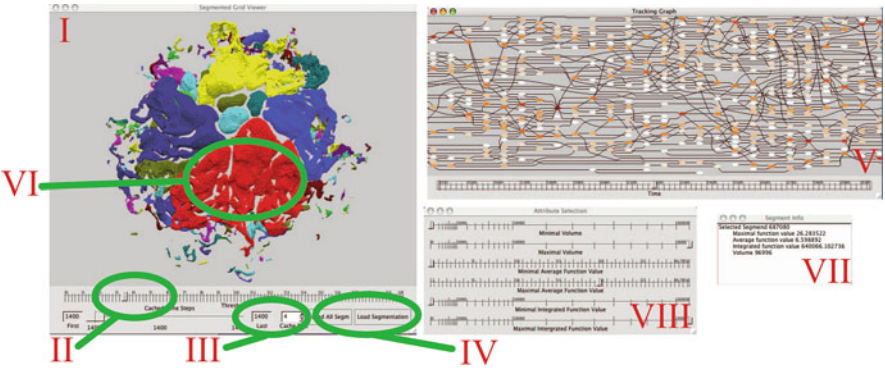


Fig. 5.5 Illustration of the different components of the user interface. (I) 3D display of the segmentation including a slider to select the fuel consumption threshold (II); (III) the interface to determine the number of in-memory time steps; (IV) the button to load the geometry; (V) interactive display of the tracking graph. Selecting node in either the 3D viewer or the graph display causes the corresponding cell to be highlighted (VI) and its attribute information to be displayed in the info window (VII). The last window (VIII) provides the ability to sub-select segments based on attribute values. ©2011 IEEE. Reprinted, with permission, from [17]

trees are loaded interactively from disk without any caching or other acceleration mechanism. Second, the segment id and all its corresponding attribute information are extracted from the split tree and displayed in the info window (VII).

Note that the current threshold of the split tree is driven by the segmentation display (slider II), while the time tracking graph uses a single fixed threshold. Thus, during selection, the tracking graph and the split tree can use different thresholds, in which case the system automatically adapts the selection: If the split tree threshold is smaller (bigger cells) the segment containing the picked one is selected; If the split tree threshold is larger (smaller cells) the information for the appropriate sub-segment is shown. Finally, if the node the user has selected corresponds to any segment currently shown, this segment will be highlighted (VI).

Segmentation Display

The other main window (I) displays the segmentation and allows the user to vary the threshold (II) and pick the number of in-memory time steps (III). Individual cells are displayed using one of eleven colors at random, reserving bright-red for highlighted cells. Similar to the graph display, the segmentation view supports selection of individual segments displaying their information in a separate window (VII). Finally, we provide an additional window (VIII) that makes it possible to sub-select segments based on the various attributes. Overall, the system supports to exploring the entire time series of a combustion simulation at arbitrary thresholds and using conditional selection criteria.

Quantitative Analysis

The flexible, one-parameter families of segmentations generated by our framework and their corresponding statistical information enable new in-depth analysis capabilities in studying turbulent flames. We report in this section quantitative analysis scenario derived from our framework as well as the preliminary insights our collaboration partners gained with these. The main significant observation is that the flames in the low-swirl configuration seem to burn in two different modes. Overall, the burning cells create a bowl shaped structure centered above the burner. Around the center of this bowl, cells appear to behave much like the idealized flames studied in [32]. On the outside, however, the flames burn more chaotically in smaller, irregularly shape regions. The behavior of these *fringe* cells is very unlike that of the idealized flames and it is not yet clear how to model them. Therefore, the initial analysis has focused on the center of the bowl.

Our interactive exploration user interface helped our collaboration partners identifying a better and refined fuel consumption rate than used in previous studies. In particular, the threshold $2.6\,\mathrm{kg}_{H_2}/\mathrm{m}^3\,\mathrm{s}$ used previously exhibited under segmentations yielding only one large region for the SwirlH2Fast data-set. Our interface, with its visual exploration and statistical measure capabilities, helped them refine

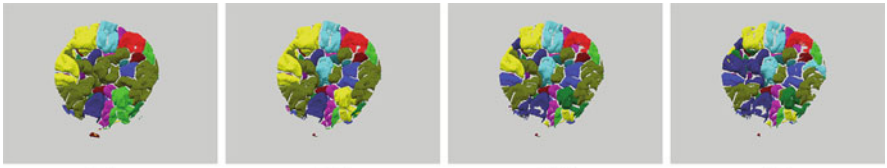


Fig. 5.6 Burning cells in the center of the SwirlH2 data at time step 1500 (randomly colored) using fuel consumption thresholds of 4.0 (**a**), 5.0 (**b**), 6.0 (**c**) and 7.0 (**d**) $\mathrm{kg}_{H_2}/\mathrm{m}^3\,\mathrm{s}$ respectively. ©2011 IEEE. Reprinted, with permission, from [17]

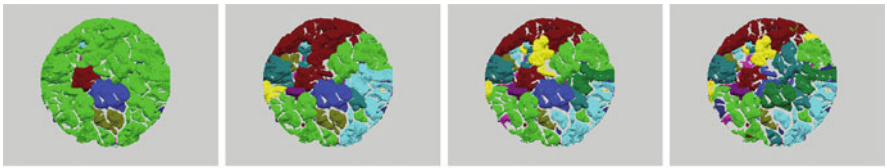


Fig. 5.7 Burning cells in the center of the SwirlH2Fast data at time step 3000 (randomly colored) using fuel consumption thresholds of 6.0 (**a**), 7.0 (**b**), 8.0 (**c**) and 9.0 (**d**) $\mathrm{kg}_{H_2}/\mathrm{m}^3\,\mathrm{s}$ respectively. ©2011 IEEE. Reprinted, with permission, from [17].

these thresholds to 5 and $8\,\mathrm{kg}_{H_2}/\mathrm{m}^3\,\mathrm{s}$ for the SwirlH2 and the SwirlH2Fast datasets respectively (Figs. 5.6 and 5.7).

To confirm these insights, we generated the time tracking graphs for various fuel consumption thresholds and reported the distribution of the size of the burning cells maintained in the graph in Fig. 5.8. As suggested by the visualization, the distributions show a markedly different behavior for lower fuel consumption thresholds. For small thresholds the distributions become exponential indicating a small number of larger cells rather than the logarithmic behavior seen in previous studies. However, as the threshold increases, the distributions continuously change to a logarithmic shape. Combining the visual observations of Figs. 5.6 and 5.7 with the statistical results shown in Fig. 5.8 might suggest that the swirling flames behave similar to the idealized flames but at much higher fuel consumption rates.

Overall, it appears that the low-swirling flames are in a substantially different regime than the idealized flames. Our interactive framework coupled with the data analysis made possible by using augmented hierarchical split trees has been instrumental in trying to better understand the underlying dynamics controlling the low-swirling flames. It has open several new research directions for our collaboration partners, as improved fuel consumption parameters have been identified in this study to refine their theoretical models.

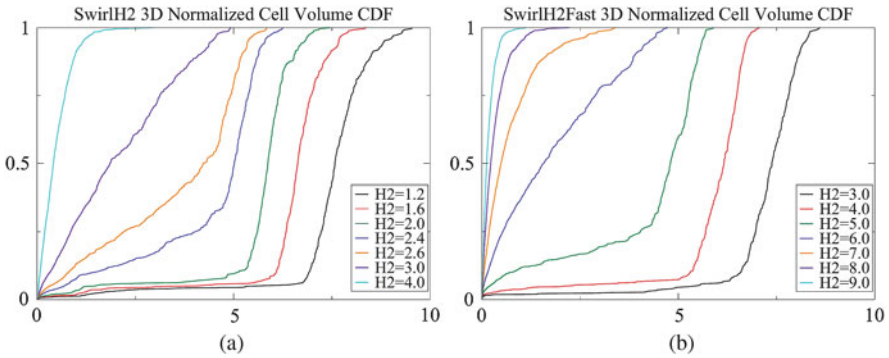


Fig. 5.8 Cumulative density functions of the distributions of cell sizes for various fuel consumption thresholds for the SwirlH2 (**a**) and SwirlH2Fast (**b**) data sets. Unlike the idealized flame [32], these distributions indicate few large cells for lower thresholds than the expected many small cells. ©2011 IEEE. Reprinted, with permission, from [17]

Concluding Remarks

While we can compute the tracking graphs for the full data including the cells on the fringes, the resulting graphs are difficult to handle. Dot currently does not scale gracefully to these large graphs and creating a layout can take hours or fail all together. Furthermore, assuming a layout is created the resulting graphs are difficult to interpret even after heavy simplification. Currently the graphs, unlike the segmentations, are computed for a static threshold. The data structures contain sufficient information to create graphs efficiently for variable thresholds. However, to view these graphs would require an interactive layout, which is beyond the current state of the art. Thus, new paradigms are needed to handle such graphs potentially involving more sophisticated simplification and hierarchical representations.

In conclusion, this work nicely exemplified the three main steps of data analysis and visualization introduced in Chap. 1: abstraction, interaction and analysis. By using hierarchical split trees, our approach enabled an important data-reduction (two orders of magnitude) while still abstracting the relevant information for the representation of the features of interest of the simulation. This massive data reduction made it possible to derive a user interface for the interactive exploration of the features of the simulation, yielding new visual insights later confirmed by quantitative analysis. While this approach has been applied to turbulent combustion simulations, it could be in principle used in any application where a generalized data subsetting (at interactively defined thresholds) and tracking is needed.

5.2 Quantitative Analysis of Molecular Interactions

In this section, we present an approach that combines the segmentation capabilities of the join tree and the Morse-Smale complex for the robust extraction of subtle features of interest (partly aligned with the gradient of a field and the level sets of another) in chemical simulations.

5.2.1 Applicative Problem

The chemical properties of a molecular system are mainly governed by the interactions between the composing atoms. In particular, an interaction type of special interest is covalent bonding, which describes the sharing of electrons in between atoms. *Covalent bonds* have been widely studied since the early twentieth century [79]. They give rise to the chemical structure of a molecular system, and are deeply investigated in molecular chemistry [93]. When dealing with complex molecular systems, a second type of interactions governs many chemical phenomena: *noncovalent interactions*. These are responsible for the bonding between several molecules and the folding of single molecules onto themselves. Examples of chemical processes driven by such interactions include the bonding between a protein and a drug, a catalyst and its substrate, or self-assembly materials. Thus, the understanding of these interactions is necessary for the interpretation of many biological processes and chemical design tasks (e.g., pharmaceutics, nanotechnology).

In contrast to covalent bonds, noncovalent interactions release much less energy and are characterized by low electron density values and only slight value variations. This challenges their extraction and analysis by solely studying the electron density. Recently, the signed electron density and the reduced gradient have drawn much attention in the chemistry community. These scalar quantities are derived from the electron density and enable a qualitative visualization of the interactions [22, 70]. However, the analysis of these quantities is mainly done manually [22, 29, 70, 77]. An automated extraction and characterization of encoded chemical interactions (in terms of the involved atoms) is still an issue, that we address in this section.

Interactions in Molecular Systems

Molecular interactions govern the structure of chemical systems by establishing attractive and repulsive balances in-between atoms. These interactions vary in strength and type. Here, we provide a brief characterization and highlight some of their properties. We refer the reader to [93] for further details. For the purpose of our discussion, one mainly differentiates between two classes of chemical interactions.

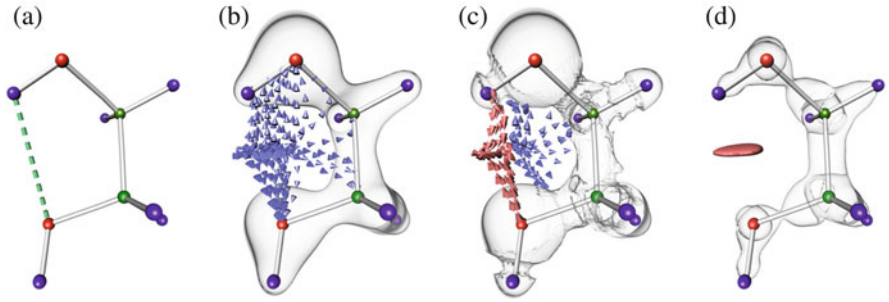


Fig. 5.9 Isosurfaces and gradient behavior (colored arrows) of the electron density ρ, its derived signed electron density $\tilde{\rho}$, and the reduced gradient s for the 1,2-ethanediol molecule. Oxygen, carbon, and hydrogen atoms are shown as red, green, and purple spheres, respectively. Covalent and noncovalent bonds are shown as white sticks and dashed green lines respectively. ©2014 IEEE. Reprinted, with permission, from [55]. (**a**) Molecular structure of the 1,2-ethanediol molecule. Noncovalent bonding (green) occurs between the hydrogen and oxygen. (**b**) The ρ-field does not capture the noncovalent bond. The gradient flow $\nabla\rho$ (blue arrows) uniformly covers all atoms. (**c**) The hydrogen-oxygen attraction is captured by $\nabla\tilde{\rho}$. The attraction (blue arrows) closes a molecular cycle generating a repulsion (red arrows). (**d**) The noncovalent bond is given in s by an isolated component (red surface) highlighting the interaction site of the oxygen and hydrogen atom.

A *covalent interaction* describes a chemical bond between atoms by sharing electrons. Based on electrostatic grounds, one can provide a simple picture of the physics behind it. Placing two atoms next to each-other, the two positively charged nuclei, i.e., the center of the atoms, both attract the outer negatively charged electrons. If the attraction is strong enough such that it overcomes the repulsion caused by the positively charged nuclei, a bond between the atoms is created. The bond represents an energetic equilibrium in which the electrons are equally attracted by the two nuclei. Thus, the two bonded atoms share these electrons. Multiple covalent interactions between atoms result in a system of bonded atoms called *molecule*—typically represented with balls (atoms) and sticks (covalent bonds) (Fig. 5.9a).

A *noncovalent interaction* does not involve the sharing of electrons. From an electrostatic point of view, they can be understood as weak electrostatic interactions between temporary and permanent partial charges. While this class of interaction constitutes the driving forces for intra-molecular folding and inter-molecular bonding, it spans a wide range of binding energies, which are typically from 1 to 15 kcal/mol and around one to two orders of magnitude smaller than covalent interactions. Hence, their extraction and characterization is much more involved. Noncovalent interactions can be caused by several physical phenomena and we detail here the most relevant ones:

- *Hydrogen bonds:* Among these electrostatic interactions, the hydrogen bonds deserve special attention due to their ever presence in biological systems. They link a hydrogen to an electronegative atom (e.g., oxygen, nitrogen, fluorine,

carbon) and occur within the same molecule or in-between molecules. (Dashed green line in Fig. 5.9a.)

- *Van der Waals forces:* These attractive forces have a purely quantum mechanical nature. In particular, the constant movement of electrons around the nucleus transforms an atom into a fluctuating multipole. These temporary charges can cause attraction between close oppositely charged atoms yielding a stable bonding of weak energy. Although the force of an individual van der Waals bond is relatively weak, the cumulative effect of multiple of them may strongly influence the global structure of large molecular systems—as shown in many chemical reactions and protein-ligand interactions.
- *Steric repulsion:* These repulsive forces are short range interactions which occur when two atoms approach one another. Intuitively, they are due to the fact that too many electrons occupy the same space (Pauli principle). This can be pictured as forces occurring in regions of space bounded by negatively charged elements, such as covalent bonds and negatively charged atoms forming molecular cycles [54]. The localization of these interactions is of major importance for chemical design tasks since they indicate regions of space that cannot receive additional electrons.

Input Data

The input of our analysis are two scalar fields derived from the electron density: the signed electron density and the reduced gradient.

Signed Electron Density

In quantum chemistry, electrons behave simultaneously as waves and particules, which only allows for a probabilistic evaluation of their positions. The relative probability that electrons can be found in a particular location of a space is described by the *electron density* $\rho : \mathscr{M} \to \mathbb{R}^+$. Density cusps are expected at the nuclei, the center of the atoms, whereas charges decrease exponentially away from them. Thus, the nuclei dictate the overall behavior of ρ. Weak atomic interactions are very often occluded and cannot be directly computed or visualized. For instance, while the ethanediol molecule admits a noncovalent bond (dashed green line, Fig. 5.9b), this bond is not captured by the electron density ρ [77]. Investigating the flow of $\nabla\rho$ in Fig. 5.9b reveals that the flow enters the molecular cycle from the outside and uniformly covers all atoms forming the cycle. The circular structure shown in Fig. 5.9a is not captured by the flow while it is crucial for the analysis of attractive and repulsive interactions. A differentiation of these interactions solely based on the density ρ is not possible, in general. To compensate therefore, a direct investigation of the Hessian $H\rho$ and its eigenvalues is needed [29]. Assuming the eigenvalues λ_i are given in increasing order, i.e., $\lambda_1 < \lambda_2 < \lambda_3$, we observe the following behavior. In the vicinity of the nuclei all eigenvalues are negative. Away from it,

λ_3 becomes positive and varies along the internuclear axis representing covalent bonds. λ_1 and λ_2 describe the density variation orthogonal to this internuclear axis. λ_1 represents the influence of the nuclei, and is always negative away from the nuclei. Contrarily, λ_2 can be either positive or negative depending on the type of interaction. While attractive interactions concentrate electron charge perpendicular to the bond ($\lambda_2 \leq 0$), repulsive interactions cause density depletion ($\lambda_2 \geq 0$). Using this localized information, the *signed electron density* $\tilde{\rho}$ is defined as $\tilde{\rho} : \mathscr{M} \to \mathbb{R}$ with $\tilde{\rho}(x) = sign(\lambda_2(x))\rho(x)$ [29]. In contrast to ρ which only assesses the interaction strength of atoms, the signed electron density $\tilde{\rho}$ additionally enables the differentiation of attracting and repulsive interactions. Figure 5.9c shows an isosurface of the signed electron density for the ethanediol molecule. In contrast to the electron density, the gradient $\nabla\tilde{\rho}$ captures nicely the attraction between the hydrogen and oxygen (red arrows), which forms a noncovalent bond creating a molecular cycle. This folded conformation also introduces repulsion in the molecule captured by $\nabla\tilde{\rho}$ (blue arrows).

Reduced Gradient

To further reveal weak noncovalent interactions, bonds, the *reduced gradient* $s : \mathscr{M} \to \mathbb{R}^+$ of ρ was introduced [70]

$$s = \frac{1}{2(3\pi^2)^{1/3}} \frac{|\nabla\rho|}{\rho^{4/3}} \tag{5.1}$$

The reduced density gradient s describes the deviation in atomic densities due to interactions [70]. Intuitively, covalent and noncovalent interactions both appear in s. In the presence of such interactions, s reports a strong change in its values in regions of space between interacting atoms. The denominator of s reduces herein the influence of the nuclei due to their high electron densities. In contrast to the electron density, s shows large values in regions far from the nuclei. The electron density exponentially decreases towards zero, and $\rho^{4/3}$ converges faster to zero than $|\nabla\rho|$. Points at which $\nabla\rho$ vanishes become zeros in s and are minima of s. In contrast to the infinite-behavior, the gradient $\nabla\rho$ dominates the denominator $\rho^{4/3}$ at those points.

The reduced gradient provides a qualitative visualization of chemical interactions by considering its isosurfaces chosen at appropriate values [22, 70]. Figure 5.9d shows an isosurface of s for the ethane-diol molecule. The interaction of the hydrogen and oxygen causes the creation of a component of isosurface (highlighted in red). The investigation of such components, which describe regions of space where interactions occur, is the starting point of our approach. In particular, the topological analysis of s reveals *interaction sites* which enables us to focus on locations of space which are relevant from a chemical perspective.

5.2.2 *Algorithm*

Feature Definition

Bader [6] proposed a theoretical model which relates molecular interactions to the critical points of the electron density [82]. In particular, nuclei can be considered as local maxima of the electron density ρ whereas bond interactions between atoms are represented by separatrices emanating from 2-saddles and ending in two distinct atoms. Figure 5.10a shows these features extracted from the Morse-Smale complex of ρ on the ethane-diol. However, as observed by Lane et al. [77], even without any topological simplification, ρ exhibits no 2-saddle in between the upper-left hydrogen and the lower oxygen atoms (cf. dashed line in Fig. 5.9a). Thus, no separatrix connecting these two atoms can be extracted and no chemical bond is identified between them. However, as discussed by Lane et al. [77], it is established from experimental evidence that this molecule exhibits a noncovalent hydrogen bond connecting these two atoms (Fig. 5.9a). Since $\tilde{\rho}$ and s better emphasize noncovalent interactions, we can overcome this limitation by transposing Bader's model from ρ to the analysis of $\tilde{\rho}$ and s. In particular, we focus on the following features.

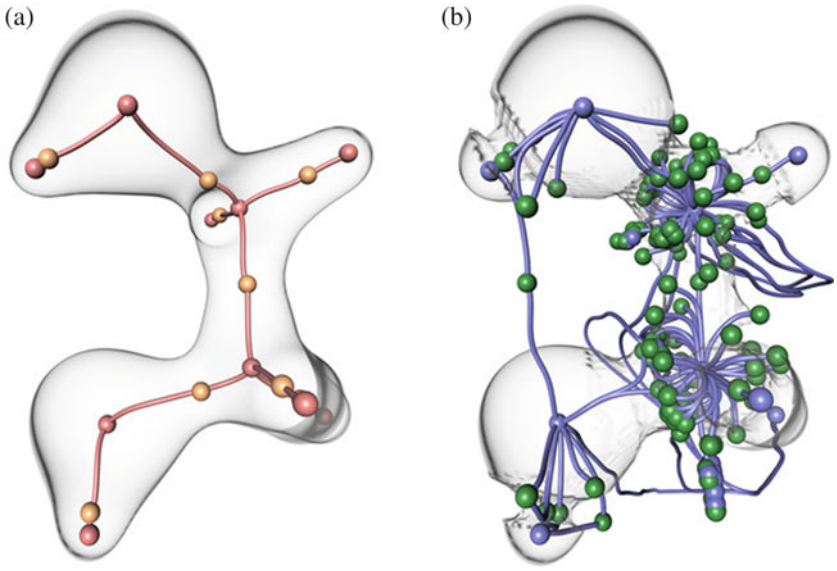


Fig. 5.10 Critical points (spheres) and separatrices (lines) in the ethanediol molecule for the considered scalar fields. Red, orange, green and blue spheres represent maxima, 2-saddles, 1-saddles and minima respectively. Red and blue lines represent separatrices connecting a 2-saddle to maxima and a 1-saddle to minima respectively. ©2014 IEEE. Reprinted, with permission, from [55]. (**a**) ρ. (**b**) $\tilde{\rho}$

Atoms

Near the atoms, the flow of $\nabla\rho$ exhibits an attractive behavior characterized by $\lambda_2 < 0$. Thus, $\tilde{\rho}$ is negative in the vicinity of the atoms. Moreover, since the electron population is maximal at the nuclei, $|\tilde{\rho}|$ is also maximal in these locations. Thus atoms are represented in $\tilde{\rho}$ by local minima. With regard to s, these atoms are also represented by minima of s since $\nabla\rho$ vanishes in the atoms. For typical quantum chemical calculations with Gaussian bases, atoms are characterized by $\tilde{\rho} < -0.35$ and $s = 0$.

Bond Paths

By transposing Bader's model to the analysis of $\tilde{\rho}$, a bonding interaction between two atoms is present if there exists a single line connecting the atoms and which is everywhere tangential to $\nabla\tilde{\rho}$. Since atoms are represented by minima of $\tilde{\rho}$, such a line is represented by two integral lines emanating from minima. Moreover, the existence of such a line between two minima implies the existence of a 1-saddle of $\tilde{\rho}$, which connects the two separatrices emanating from the minima. The latter saddle is called a *bond critical point*. Note that this saddle can be a merging or genus-change 1-saddle.

Bonding Graph

The set of atoms and bond paths in a molecular system form the *Bonding Graph*. Each node of this graph represents an atom and its edges represent all attractive interactions in the system.

Repulsive Bond Cycles

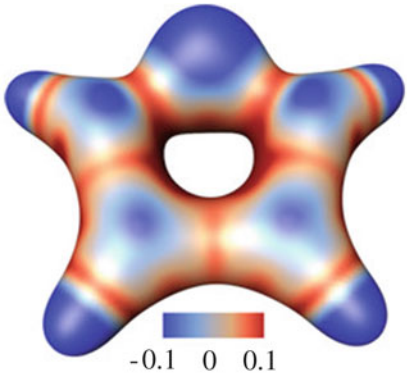
Bonds can form cycles in the bonding graph yielding steric repulsion. Thus, we define a cycle of minimal length in the bonding graph as a repulsive bond cycle.

Figure 5.10b shows the extraction of the minima and the ascending separatrices ending in 1-saddles, obtained from the Morse-Smale complex of $\tilde{\rho}$. In contrast to the topological analysis of ρ (Fig. 5.10a), this analysis reveals a bond path connecting the upper-left hydrogen and the lower oxygen atoms, hence, corroborating experimental observations [77]. Thus, this result further motivates the topological analysis of $\tilde{\rho}$ for noncovalent bond extraction. Note that s also captures this interaction site by a component of its isosurface (Fig. 5.9d).

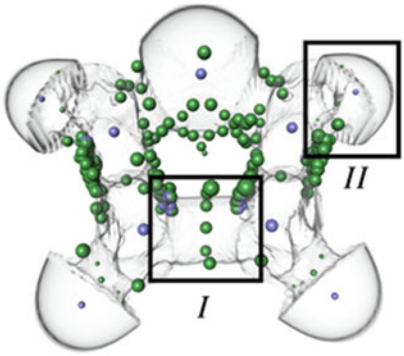
Technical Challenges

Although the Morse-Smale complex of $\tilde{\rho}$ nicely captures covalent and noncovalent bonds, this comes with a price to pay in terms of topological complexity (Fig. 5.10b). This complexity occludes the features of interest and challenges their direct extraction, as illustrated in Fig. 5.11.

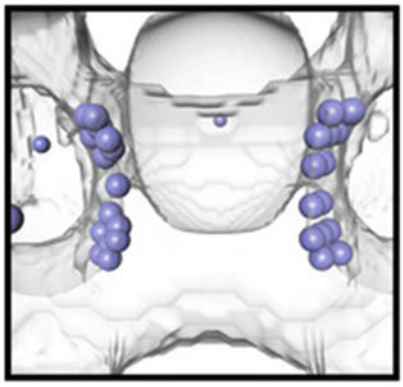
According to our feature definition, atoms can be identified by considering minima of $\tilde{\rho}$. Bond paths can be identified by extracting from the Morse-Smale complex of $\tilde{\rho}$ the two descending separatrices emanating from each 1-saddle. A bond path is considered *valid* if it connects two distinct atoms. Note that the robust extraction of the bond paths is of major importance for the extraction of the



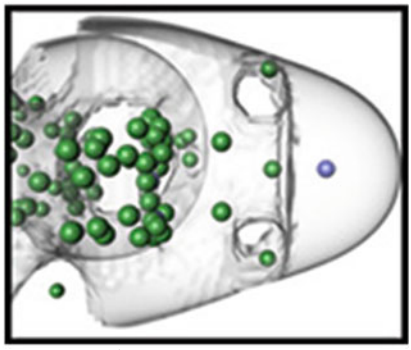
(a) Isosurface of ρ colored by the 2^{nd} eigenvalue λ_2 of $H\rho$



(b) Minima and 1-saddles of $\tilde{\rho}$ scale by persistence.



(c) Zoom-in *I* of (b).



(d) Zoom-in *II* of (b).

Fig. 5.11 Technical challenges associated with the extraction of molecular interactions from $\tilde{\rho}$ on the furan molecule: beyond low-persistence critical point pairs, $\tilde{\rho}$ also exhibits high-persistence minimum-saddle (**c**) and saddle-saddle pairs (**d**) which occlude the critical points of interests (**b**) representing atoms and bond critical points. ©2014 IEEE. Reprinted, with permission, from [55]

remaining features (covalent bonds, noncovalent bonds, Bonding Graph, repulsive bond cycles). However these features are also the most challenging to isolate, for the following reasons:

1. *Low persistence structures:* Critical points are present in the Morse-Smale complex due to the sampled representation of $\tilde{\rho}$. While spurious 1-saddles can yield false positive bond paths, spurious minima can additionally yield interrupted bond paths by preventing descending separatrices emanating from a valid bond critical point to reach the corresponding atoms. Thus, the Morse-Smale complex needs to undergo topological simplification to (a) remove low-persistence minima and to (b) connect bond critical points to atoms.
2. *Non-atom minima:* At the edge of attractive ($\tilde{\rho} < 0$) and repulsive ($\tilde{\rho} > 0$) regions (Fig. 5.11a), $\tilde{\rho}$ exhibits high-persistence minima, as illustrated in Fig. 5.11c. These minima also need to be discarded to avoid interrupted bond paths as well as false positives in atom identification.
3. *Non-bonding saddles:* As illustrated in Fig. 5.11d, $\tilde{\rho}$ also includes high-persistence (genus-change) saddle-saddle critical point pairs. However, this category of critical point pair may not be removable through Morse-Smale complex simplification [71]. Thus, the removal of these outliers challenges existing topological techniques and motivates the joint analysis of the reduced gradient s.

The above challenges motivate a tailored topological analysis pipeline described in the following section.

Interaction Sites

To address the third challenge of the previous paragraph, we describe a pre-processing stage whose purpose is to isolate regions of space containing 1-saddles of $\tilde{\rho}$ which are relevant bond critical point candidates from a chemical perspective. Interaction sites can be visualized by considering the regions of space bounded by the connected components of relevant isosurfaces of s [22, 70]. However, each type of interaction is visualized by a specific isovalue. The birth of these components happens at early isovalues for covalent bonds. Contrarily, for noncovalent bonds, the birth happens at arbitrarily larger isovalues. To extract all interaction sites in a single procedure, we analyze $\mathscr{J}(s)$, the join tree of s.

The reduced gradient s exhibits low values in the vicinity of the nuclei and larger values away from it. Moreover, as the isovalue increases, new connected components of isosurfaces appear in regions of spaces where interactions (covalent and noncovalent bonds) occur. The appearance of such components are characterized by minima of s. As the isovalue continues on increasing, the merge of these components occur at 1-saddles of s. These events are captured by $\mathscr{J}(s)$. In particular, each branch of $\mathscr{J}(s)$ which contains a leaf (a minimum) represents a connected region of space containing a single minimum of s. Therefore, we extract each interaction site by considering each branch of the join tree which contains a minimum.

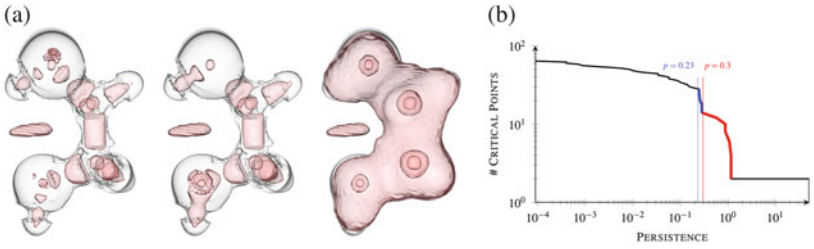


Fig. 5.12 Illustration of different simplification levels of the reduced gradient s based on the persistence p of critical point pairs. Reading the persistence curve from the right to the left side reveals a first slope (red) at the end of which interaction sites corresponding to covalent bonds have been simplified. The end of the next slope on the left ($p = 0.23$, blue) corresponds to a simplification level where both covalent and noncovalent interaction sites are maintained. ©2014 IEEE. Reprinted, with permission, from [55]. (**a**) Interaction sites obtained from the sub-level set components of the reduced gradient s for three simplification levels: no simplification (left, 32 sites), $p = 0.23$ (middle, 14 sites), $p = 0.3$ (right, 6 sites). (**b**) Persistence curve highlighted with two different simplification levels (blue and red vertical lines)

Figure 5.12(left) shows the interaction sites (bounded by red surfaces) extracted from the analysis of $\mathscr{J}(s)$. Due to the sampled representation of s and its numerical evaluation, spurious features are present—in particular in the vicinity of the atoms. To discard these, we apply a persistence based simplification of $\mathscr{J}(s)$. To select an appropriate simplification level, we manually inspect the persistence curve of the saddle-minimum pairs (Fig. 5.12) for the ethanediol molecule. Reading the curve from the right side (high persistence pairs) to the left (low persistence pairs) reveals a first slope (red in Fig. 5.12) finishing with a sharp kink at $p = 0.3$ (red vertical line). This level corresponds to the removal of the minima representing covalent bonds. Thus, at this level of simplification, the interaction sites extracted from $\mathscr{J}(s)$ (Fig. 5.12a, right) only reveal: the prominent atoms (oxygens and carbons), a larger component which represents the sub-level set containing all covalent interactions, and an isolated component representing a noncovalent interaction. The following slope to the left (blue) also finishes with a sharp kink at $p = 0.23$ (vertical blue line). This point corresponds to a simplification level where the components representing covalent bonds are maintained (Fig. 5.12a, middle). This observation was confirmed in all of our experiments. Thus, we select as an appropriate simplification level the end of the second slope (from the right side) on the persistence curve. As illustrated in Fig. 5.12a (middle), each interaction (covalent and noncovalent) is represented by a specific interaction site. Note that dominant atoms (oxygens and carbons) are also represented. However, the latter regions typically include no 1-saddle of $\tilde{\rho}$ which enables to discard them in a straightforward manner. Note that at a technical level, in contrast to the Morse-Smale complex of $\tilde{\rho}$, all saddle-minimum pairs captured by the Join Tree of s can be simplified, which enables to overcome the technical challenge introduced by the incomplete simplification of the Morse-Smale complex of $\tilde{\rho}$.

Bonding Graph

Covalent and noncovalent bonds form a subset of the More-Smale complex. We describe how the Morse-Smale complex of $\tilde{\rho}$ can be reduced to the Bonding Graph representing all atoms and their bonds only.

1. Removal of Low Persistence Structures

First, the Morse-Smale complex of $\tilde{\rho}$ needs to be simplified. To find a representative simplification level, we analyze the persistence p of all minima and saddles. Figure 5.13b shows such a curve for the ethanediol molecule in logarithmic scaling. The persistence curve exhibits a characteristic behavior. We can make use of this to guide the manual selection of an appropriate simplification level. Around 90% of the minima and saddles have a very low persistence ($p < 0.01$). After the first drop down in this curve, two different persistence-ranges are apparent. In the first range ($0.01 \leq p \leq 0.39$), the curve shows an almost constant slope (blue) indicating the transition from low to high persistent features. Figure 5.13a (middle) shows the minima and 1-saddles at the simplification level $p = 0.01$. Most of the spurious structures are removed at this level, and the critical points representing atoms and bonds are revealed. However, there exists also minima and 1-saddles which do not represent molecular features. Increasing the persistence threshold now iteratively removes pairs of minima and 1-saddles. At the beginning of the second range ($p = 0.39$, red), only the minima representing the heavy carbon and oxygen atoms are still present. The minima representing hydrogens are already removed due to the small mass of hydrogens. However, 1-saddles which do not represent bonds are still visible in the interface between attractive and repulsive regions. We made the above observation in all of our experiments. To guarantee that all atoms and their

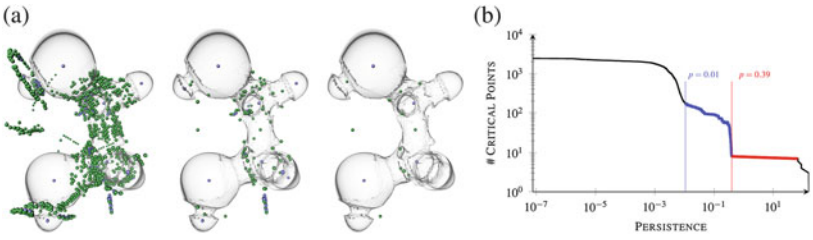


Fig. 5.13 Illustration of different simplification levels of the signed electron density $\tilde{\rho}$ based on the persistence p of critical points. In the initial level minima and 1-saddles representing atoms and bonds are occluded by low-persistence critical points. Simplifying the Morse-Smale complex removes spurious critical point revealing the features of interest. However, not all minima and 1-saddles represent atoms and bonds even in the simplified Morse-Smale complex. ©2014 IEEE. Reprinted, with permission, from [55]. (**a**) Minima (blue) and 1-saddles (green) for three simplification levels: no simplification (left), $p = 0.01$ (middle), $p = 0.39$ (right). (**b**) Persistence curve highlighted with two different simplification levels (blue and red line)

even weak bonds are well represented, we select a simplification level after the first drop down in the persistence curve. However, the point of first drop down varies depending on the overall structure and energy of the molecular system.

2. Removal of Non-atom Minima

As detailed previously, there are minima which do not represent atoms. These are characterized by their location at the interface between the attractive and repulsive regions. Thus, their signed density value needs to be larger than those of the atoms. This allows a differentiation of the minima, and we select those with $\tilde{\rho} > -0.35$. Since those minima can interrupt bond paths, we need to remove them from the Morse-Smale complex yielding a new connectivity between the remaining minima and 1-saddles. We follow here the general strategy for the simplification of the Morse-Smale complex, but restricted to the selected set of minima. For each minimum, we determine the lowest saddle in its neighborhood; and put all minima-saddle pairs with their weight, i.e., the height difference between minimum and saddle, in a priority queue. This queue is processed in ascending order starting with the pair with the lowest weight. This pair is removed from the Morse-Smale complex, which requires a subsequent update of the connectivity information of the critical points in the Morse-Smale complex. Due to the new connectivity, the weights of the minimum-saddle pairs also need to be updated yielding a new lowest minimum-saddle pair. We iteratively remove the selected minima until all of them are processed. This yields a Morse-Smale complex in which all minima represent atoms only.

3. Removal of Non-bonding Saddles

Next, only the 1-saddles of $\tilde{\rho}$ which are relevant from a chemical perspective should be considered for bond path extraction. Thus, we restrict the remainder of the analysis to the 1-saddles located in each of the interaction sites extracted previously (Fig. 5.12).

4. Bonding Graph Extraction

Each 1-saddle is connected to either one or two minima. Since bonding only occurs between atoms, we neglect all 1-saddles which are twice connected to the same minimum. From the remaining saddles, we collect the 1-saddles S^k and their connected minima M_1^k and M_2^k giving rise to a set of triplets (M_1^k, M_2^k, S^k). It may happen that they exist two saddles S^{k_1} and S^{k_2} which end in the same pair of minima, i.e., $M_1^{k_1} = M_1^{k_2}$ and $M_2^{k_1} = M_2^{k_2}$. However, only one bond can exist between the same pair of atoms. Considering an evolution of the isovalue in $\tilde{\rho}$, bonding critical points are given by the earliest contact of the sub-level set components emanating

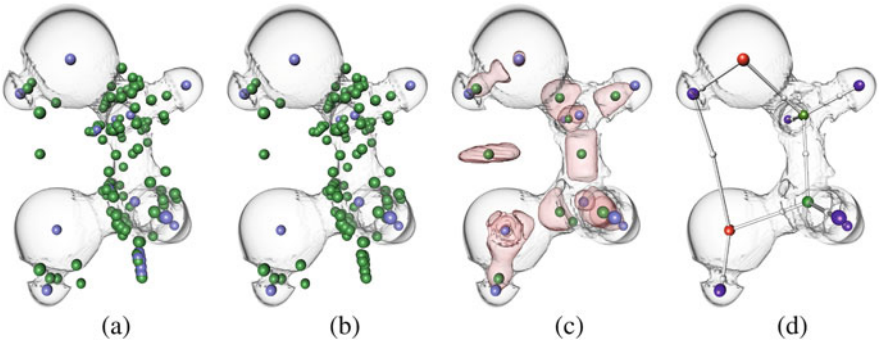


Fig. 5.14 Illustration of the different steps of our analysis algorithm: (**a**) low-persistence pairs are cancelled; (**b**) high-persistence minimum-saddle pairs are removed; (**c**) high-persistence saddle-saddle pairs are filtered out based on the interaction site extraction; (**d**) final bonding graph. ©2014 IEEE. Reprinted, with permission, from [55]

from the two atoms. Hence, for each interaction site, we choose the 1-saddle S which minimizes its $\tilde{\rho}$-value as the bonding saddle representing the bond between M_1 and M_2. Applying this procedure to all saddles S^k yields the Bonding Graph $G = (N, E)$ in which all minima represent atoms and each pair of minima is only connected by a single 1-saddle representing their bonding. The nodes N of the undirected graph G are given by the minima and bonding 1-saddles, its edges E by the separatrices connecting the saddles to the minima.

Figure 5.14 shows the different steps of our pipeline on the ethanediol molecule and illustrates its ability to isolate the features of interest, despite the presence of high-persistence and spurious critical points in $\tilde{\rho}$.

Repulsive Bond-Cycles

A repulsive bond cycle is given as a cycle of minimal length in the Bonding Graph. Thus, we need to compute the shortest path between two bonded atoms by omitting their bond, as illustrated in Fig. 5.15. To do so, we weight the edges E of the graph G by the Euclidean distance of the incident nodes. Note that all the weights are positive. Let us consider a node representing a 1-saddle $S \in N$ and its adjacent minima $M_1, M_2 \in N$. The two edges connecting these three nodes are denoted by $\mathbf{E_1}, \mathbf{E_2} \in E$. We apply Dijkstra's algorithm to compute the shortest path $P \subset E \setminus \{\mathbf{E_1}, \mathbf{E_2}\}$ between M_1 and M_2 in G omitting the edges $\mathbf{E_1}$ and $\mathbf{E_2}$. If it exists, a molecular repulsive cycle was found. We mark all minima covered by the shortest path and collect all bonding saddles connecting the minima. We assign to each of these saddles an identifier indicating that all of them belong to the same repulsive interaction. $\mathbf{E_1}$ Note that several saddles can describe the same cycle, i.e., all minima covered by two cycles coincide. In this case, we only keep one representative of this cycle.

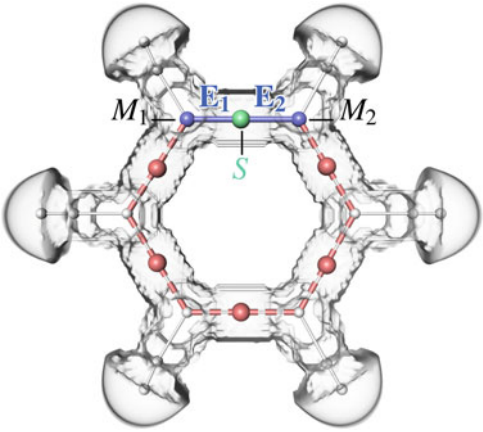


Fig. 5.15 Illustration of the repulsive bond-cycle extraction. ©2014 IEEE. Reprinted, with permission, from [55]

5.2.3 Results

This section presents experimental results of our analysis algorithm obtained with a C++ implementation on a computer with an i7 CPU (2.8 GHz) and 16 Gb of RAM. We investigate a variety of molecular systems obtained through quantum chemistry simulation with the Gaussian program and represented by regular grid data. In all of experiments, our analysis took in general a few seconds to compute and at most 8 min for the DNA helix data-set ($170 \times 178 \times 183$).

Validation

Figure 5.16 shows the bonding graph computed by our approach on a variety of simple molecular systems. Our analysis indeed reveals the covalent bonds of these systems. Moreover, it also reveals the noncovalent interactions responsible for the bonding of several dimers. In particular, our analysis reveals that two hydrogen bonds (noncovalent bonds linking a hydrogen to a heavier atom) are involved in the bonding of the Pyridoxine and Aminopyridine (leftmost, top), while a van der Waals attractive force bonds the bottom two hydrogen atoms together. Each of these bonding graphs reveal a covalent and noncovalent bonding structure.

Quantitative Analysis and Visual Exploration

Since our technique allows for a robust extraction of covalent and noncovalent interactions, it also enables to enumerate, classify and investigate such features from a quantitative point of view.

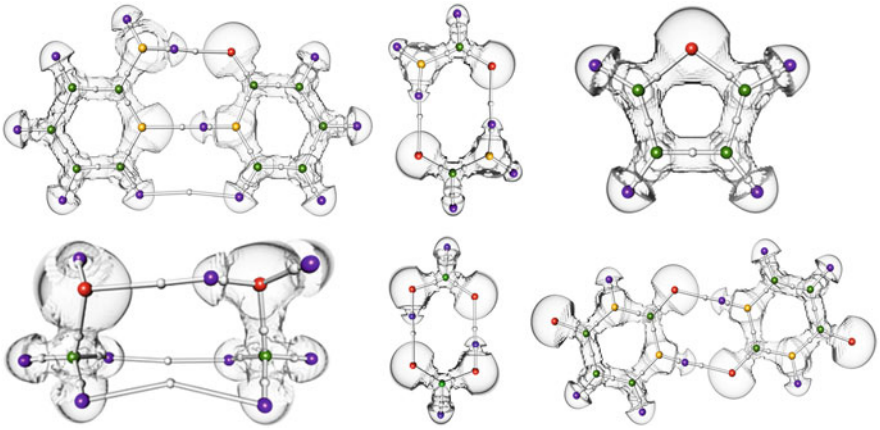


Fig. 5.16 Bonding Graphs of simple molecular systems. Oxygen, nitrogen, carbon, and hydrogen are shown as red, yellow, green, and purple spheres. Bonding saddles are depicted as white spheres. The Bonding Graph captures covalent and noncovalent bondings (white lines). ©2014 IEEE. Reprinted, with permission, from [55]

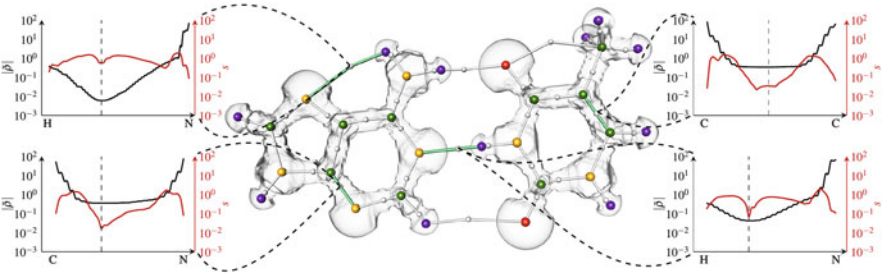


Fig. 5.17 Quantitative analysis of attractive interactions in the A-T DNA base pair. Thanks to the robust extraction of attractive bonds, each type of interaction can be investigated in the light of several quantum chemistry measures. By restricting the analysis of these measures to the separatrix representing each bond, our analysis enables to project these complex 3D information down to easily readable 1D plots, revealing distinct characteristic behaviors for covalent bonds (bottom-left, upper-right), hydrogen bonds (bottom-right) or Van der Waals interactions (upper-left). The bond critical points are highlighted as dashed lines in the 1D plots. ©2014 IEEE. Reprinted, with permission, from [55]

Bond Investigation and Classification

In addition to the enumeration of the bonding interactions, our analysis enables new investigation capabilities on a per interaction basis. In particular, one can further analyze various quantum chemistry measures for each extracted bond since our analysis provides a concrete geometrical representation for each interaction. Figure 5.17 illustrates such an analysis where several quantities ($|\tilde{\rho}|$ and s) were evaluated along the bond paths representing two covalent bonds (lower left and

upper right) and noncovalent bonds (upper left and lower right). For each interaction, such a procedure enables to project these complex 3D information down to easily readable 1D plots revealing the evolution of these measures along the bond path. In particular, these plots reveal a characteristic behavior for covalent bonds. A high-valued flat plateau of electron population ($|\tilde{\rho}|$) is located in the middle of the bond. Moreover, s indicates a dominant minimum in the vicinity of the bond critical point. In contrast, noncovalent bonds reveal a more subtle electronic structure. In the vicinity of the bonding critical point, this analysis reveals that a minimum of electron population ($|\tilde{\rho}|$) is achieved at distinct values for hydrogen bonds (10^{-1}, bottom right) and van der Waals (10^{-2}, upper left) interactions; which is one to two orders of magnitude lower than covalent bonds. As proposed by Bader [6], attractive bonds can be classified according to the $|\tilde{\rho}|$-value of their bonding critical point. We make use of this property to classify attractive bonds in the remainder.

Steric Repulsion

Figure 5.18 illustrates the extraction of repulsive bond cycles on the Pyridoxine-Aminopyridine. Steric repulsion is induced by a closed chain of atoms. As a first approach, we integrate $\log(|\tilde{\rho}|)$ along each cycle to assess its strength. This analysis reveals strong steric repulsions induced by the cycles formed by covalent bonds only (left and right orange cycles). In contrast, very weak steric repulsion is induced by the bottom cycle. This difference is caused by the presence of the electrons involved in covalent interactions and their absence in weak-energy van der Waals

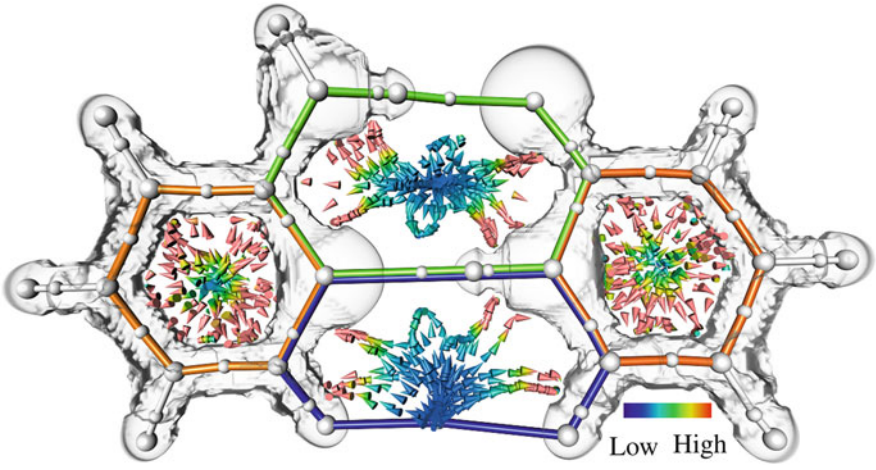


Fig. 5.18 Repulsive bond cycles of the Pyridoxine-Aminopyridine. Integrating $|\tilde{\rho}|$ along them enables to assess the strength of the steric repulsion, as confirmed visually by the $|\tilde{\rho}|$-color-coding of $\nabla\tilde{\rho}$ (arrows). ©2014 IEEE. Reprinted, with permission, from [55]

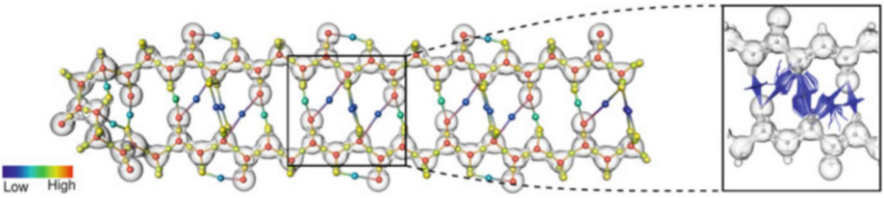


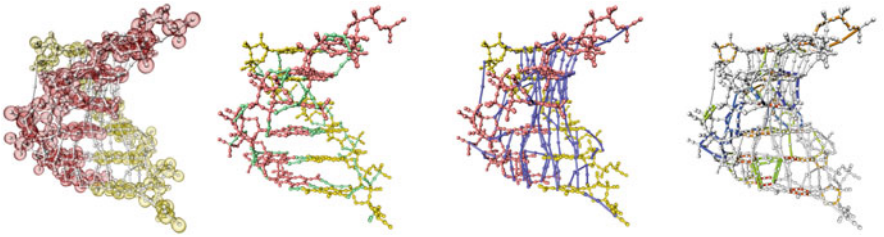
Fig. 5.19 Visual and quantitative exploration of covalent and noncovalent bonds in the β-sheet polypeptide. The amplitude of the signed electron density ($|\tilde{\rho}|$, color-coded from blue to red) enables to distinguish covalent bonds (yellow) from hydrogen bonds (cyan) and van der Waals interactions (dark blue). While the numerical integration of $\nabla\tilde{\rho}$ (right inset) enables to visually distinguish the latter two types of interactions, our combinatorial pipeline robustly extracts these features to support further quantitative analysis. In particular, our algorithm reveals the repeating pattern (black frame) of noncovalent interactions responsible for the folding of this molecule, which decomposes it in unitary building blocks corresponding to the elementary amino-acids composing the molecule. ©2014 IEEE. Reprinted, with permission, from [55]

bond (bottom hydrogens). This insight is also confirmed by the $|\tilde{\rho}|$-color-coding of the numerical streamline integration in $\nabla\tilde{\rho}$. *s*

Complex Molecular Systems

Figure 5.19 shows the extraction of attractive covalent and noncovalent bonds on a folded β-sheet polypeptide. Our analysis enables to classify them according to the $|\tilde{\rho}|$-value of their bond critical points. This classification highlights the noncovalent bonds responsible for the folding of this molecule. While hydrogen bonds (cyan) are involved in this folding, our analysis additionally reveals that van der Waals attractions (dark blue) also play a structural role to enforce the stability of this folded conformation. This distinction is confirmed by the continuous flow of $\nabla\tilde{\rho}$ (right inset), which exhibits the dispersed behavior of van der Waals attractions. Moreover, as highlighted with the black frame, our analysis reveals a repeating pattern of noncovalent bonds, which corresponds to the decomposition of the molecule in its elementary amino-acids.

Figure 5.20 illustrates an exploration of the molecular interactions within a portion of a nucleic acid double helix found in DNA. Due to the complexity of this data-set, our algorithm extracts a large number of interactions between the two parts of the helix, highlighted in yellow and red (Fig. 5.20a). To reveal the structural role of these interactions at a global level, we perform a quantitative analysis of each of the extracted bonds. In particular, plotting the number of bonds as a function of $|\tilde{\rho}|$ (Fig. 5.21 left) indicates three distinct ascending modes. The first two modes (blue and green) are separated by a small-scale kink (see inset) while the last two modes (green and black) are separated by a large-scale kink. As suggested by our initial quantitative analysis on the A-T DNA base (Fig. 5.17), each portion of the curve (blue, green and black) correspond to a specific type of interactions: van der



(a) Bonding Graph (b) Covalent and Hydrogen Bonds (c) Covalent and van der Waals Bonds (d) Cycles

Fig. 5.20 Quantitative exploration of the nucleic acid double helix data-set. Our analysis enables to enumerate and classify noncovalent interactions (Fig. 5.21), yielding query-based visualizations (**b**, **c**) revealing the role of each type of interaction in the helicoidal structure of DNA. A multi-scale exploration of the repulsive bond cycles (**d**) reveals high repulsions in the center of the helix and weaker repulsion on its outer boundary. ©2014 IEEE. Reprinted, with permission, from [55]

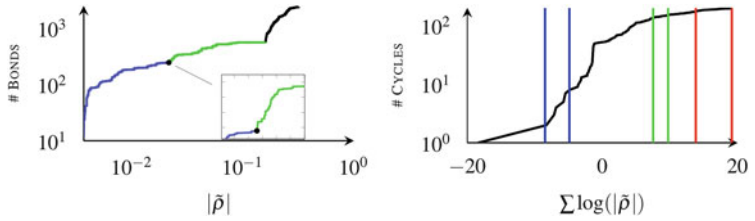


Fig. 5.21 Supporting plots for Fig. 5.20. Shown are the number of bonds and cycles present in the DNA example as a function of $|\tilde{\rho}|$. ©2014 IEEE. Reprinted, with permission, from [55]

Waals, hydrogen bonds and covalent bonds, respectively. Thus, this curve enables to select thresholds for a query-based exploration of the noncovalent bonds. In Fig. 5.20b, van der Waals attractions were removed while in Fig. 5.20c hydrogen-bonds were filtered out. Covalent bonds appear in red and yellow in these figures. These visualizations provide two complementary global insights on the helicoidal structure of this molecular system. In particular, Fig. 5.20b reveals that hydrogen bonds are mostly located in the planes orthogonal to the helix axis while van der Waals attractions follow the axis-direction. This confirms that the torsional stiffness of the double helix, which influences the circularisation of DNA, is mostly governed by hydrogen bonding, whereas the axial stiffness, which characterizes the wrapping properties of DNA, is mostly governed by weaker van der Waals attractions. A similar query-based visualization can be carried out regarding the steric repulsions. Since our analysis enables to quantify the strength of repulsive cycles (cf. previous paragraph), the evolution of their number can be plotted in function of their strength (Fig. 5.21 right). From this curve, the user can select relevant intervals (blue, green, red) and have a direct feedback in the visualization of the corresponding steric repulsions (Fig. 5.20d). This enables a multi-scale exploration of such features, which indicates a strong repulsion in the center of the base pairs (red cycles on

the horizontal steps of the helix) and weaker repulsion on the outer boundary of the helix (blue cycles).

Concluding Remarks

In this work, we showed that subtle features could be reliably extracted in molecular systems by combining the segmentation capabilities of the join tree and the Morse-Smale complex of two scalar fields. While such a tailored approach enables an interactive exploration of the structure of a molecular system as well as its quantitative characterization, it also indicates that the joint topological analysis of pairs of scalar fields may yield more generic analysis algorithms. However, this requires a generalization of topological analysis to bivariate data, as further discussed in Chap. 6.

Chapter 6
Perspectives

As illustrated throughout this book, Topological Data Analysis has demonstrated over the last two decades its effectiveness, robustness and practical interest for many data abstraction, exploration and analysis tasks. Moreover, as mentioned at the end of Chap. 2, these techniques gained a sufficient level of maturity such that several of their key algorithms are now available through open-source implementations [35, 103, 116, 130]. This level of maturity is also demonstrated by the numerous collaborations with domain experts for the resolution of problems that go well beyond the sole scope of Computer Science, including molecular chemistry, combustion, cosmology and fluid dynamics for instance. Thus, given this maturity, one can legitimately wonder:

Is there still any research to be done in this area?

To answer this question, one needs to observe the ongoing trends in Scientific Computing (see [66] for a complementary point of view on future trends). Three-dimensional numerical simulation established itself as a necessary tool for knowledge discovery in many fields of science. It enables to evaluate, improve and analyze theoretical models, especially when experimental validation is made difficult for financial, technical or legal reasons. In industry, simulation is ubiquitous in the modeling process of a product.

Traditionally, such simulations are run on High-Performance Computing (HPC) resources while their output (typically a scalar field representing a simulated quantity at a given time-step) is transferred to a remote work station for post-processing purposes: visualization and analysis.

This overall methodology turns out to be incompatible with the characteristics of the upcoming generation of super-computers (expected around 2018) with predicted computing performances at the ExaScale (10^{18} FLoating-point Operations Per Second, FLOPS), since:

J. Tierny, *Topological Data Analysis for Scientific Visualization*,
Mathematics and Visualization, https://doi.org/10.1007/978-3-319-71507-0_6

1. it will come with unprecedented technical challenges that cannot be addressed by simply extending existing frameworks [65] and which will impose new constraints on data analysis algorithms;
2. it will also enable radically novel simulation usages [36] which will result in novel types of data to analyze.

As described below, put together, these two challenges require to deeply re-visit the core algorithms of data analysis. This implies a complete *reboot* of the research effort in Topological Data Analysis.

6.1 Emerging Constraints

Current estimations regarding the next generation of super-computers [65] expect an increasing imbalance between predicted data throughputs (10^{12} bytes/s [36]) and persistent storage (10^{10} bytes/s) or global network bandwidths (most HPC users are located off-site). In other words, data will be produced at significantly higher rates than it can be stored to disk or transferred through the network. In this scenario, traditional off-line post-processing is no longer an option given this increasing bottleneck.

Therefore, to simply avoid this bottleneck, it becomes necessary to move data analysis algorithms as close as possible to their input source (the numerical simulation algorithms). In particular, to minimize the usage of persistent storage, it is necessary that analysis algorithms run on the same hardware as the simulation and that they run during and in synergy with the simulation process. This strategy, often called *In-situ data processing*, imposes three major constraints, described in the following.

6.1.1 Hardware Constraints

Supercomputers are built around hardware architectures that differ from commodity computers in several ways. To fully exploit these resources, Topological Data Analysis algorithms need to be adapted to fit the hardware specifications. In particular, a key challenge is to extend them to parallel computing models, including:

1. Shared-memory parallelism with uniform memory access (multi-core environments typically found in any recent computing devices, from handled devices to workstations and supercomputers);
2. Shared-memory parallelism with non-uniform memory access (multi-processor environment typically found in high-end workstations and supercomputers);
3. Distributed memory parallelism (multi-node environments typically found in supercomputers).

Each of these types of parallelism comes with increasing difficulty regarding memory transfer management.

Adapting Topological Data Analysis algorithms to these three types of parallelisms is a major algorithmic challenge since these approaches are intrinsically sequential, use only few floating point arithmetic operations (they are mostly based on graph traversal sub-routines) and always rely at some point on a global access to the data (in particular to enable their global consistency).

Moreover, another major concern to be addressed for the upcoming generation of supercomputers is energy consumption [65]. In particular, one well-accepted way to reduce the energy consumption related to the execution of a program is to reduce its memory usage and transfer. In terms of algorithms, this is also a major algorithmic challenge for Topological Data Analysis, as most of its algorithms are memory bound. Therefore, to enable Topological Data Analysis for the upcoming generation of supercomputers, most of its approaches need to be revisited to develop memory-efficient parallel algorithms.

Preliminary Results

To initiate this research effort, we first focus on the case of the Contour Tree and the Reeb graph. While several approaches attempted to parallelize the Contour Tree construction algorithm [1, 76, 81, 86, 90], these techniques only considered the case of a simpler variant (the join or split tree) or relied on a computation based on monotone paths (which drastically restricts the usage of the output data structures in applications, preventing for instance data segmentation features).

Gueunet et al. [57] present an algorithm for the fast, shared memory multi-threaded computation of contour trees on tetrahedral meshes. In contrast to previous multi-threaded algorithms, this technique computes the *augmented* contour tree. Such an augmentation is required to enable the full extent of contour tree based applications, including for instance data segmentation. This approach relies on a range-driven domain partitioning (Fig. 6.1) to balance the input data-set in between the different threads. The authors show how to exploit such a partitioning to rapidly compute *contour forests*. They also show how such forests can be efficiently turned into the output contour tree. In particular, this partitioning strategy makes the stitching of the intermediate results computed from the different threads into one global structure much easier and more efficient than previous approaches. Performance numbers compare this approach to a reference sequential implementation (libTourtre [35]) for the computation of augmented contour trees. These experiments (Table 6.1) demonstrate the practical run-time efficiency of the algorithm. We consider this approach as the reference algorithm for the problem of the efficient multi-threaded computation of the augmented contour tree and we released a VTK-based open source implementation of it in the hope of a rapid uptake of this method.

A natural perspective of this work is to extend this kind of data-division approach to account for non-simply connected domains, hence generalizing this technique from Contour Trees to Reeb Graphs. Further, it appears to be necessary to continue

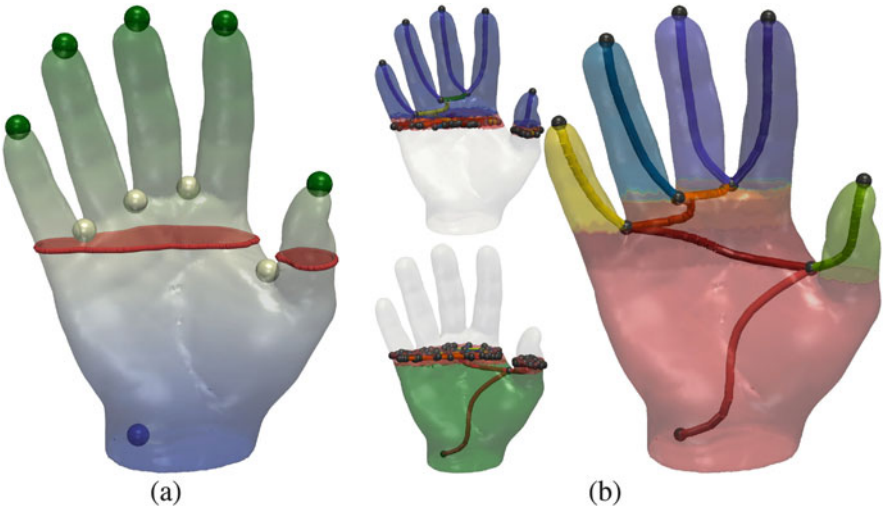


Fig. 6.1 Overview of the multi-threaded algorithm by Gueunet et al. [57] for augmented contour tree computation, on the height function f of a volume $\mathscr{M}$ with two threads. (**a**) Input scalar field f (color gradient) with its critical points (blue: min, white: saddle, green: max). The domain is split into two partitions $\mathscr{P}_i$ and $\mathscr{P}_j$ of roughly equal size corresponding to the pre-images of contiguous intervals $\mathscr{I}_i$ and $\mathscr{I}_j$ of $f(\mathscr{M})$. The *interface level-set* between such two partitions is shown in red. (**b**) The augmented contour trees $\mathscr{T}(f)_i$ (top) and $\mathscr{T}(f)_j$ (bottom) are constructed in parallel for each partition. These local trees can be easily and efficiently stitched together to form the output augmented contour tree (right) by collapsing the equivalence class corresponding to the interface level-set (show in red, left). ©2016 IEEE. Reprinted, with permission, from [57]

Table 6.1 Overall running time comparison (in seconds) between the sequential libTourtre implementation (sTourtre), a naive parallel implementation of libTourtre (pTourtre) and the multi-threaded approach by Gueunet et al. [57] (measured with a C++ implementation on an Intel Xeon CPU 2.4 GHz, eight cores, with eight threads)

			Speedup wrt.		Speedup wrt.	
Data-set	sTourtre	pTourtre	sTourtre	Ours	sTourtre	pTourtre
Elevation	20.63	10.07	2.04	5.42	3.81	2.64
EthaneDiol	23.47	13.96	1.68	7.81	3.00	1.79
Combustion	21.26	12.39	1.72	7.31	2.91	1.70
Boat	23.26	12.52	1.85	7.44	3.13	1.68
Jet	20.60	11.50	1.79	7.21	2.86	1.60
Enzo	32.51	18.07	1.80	12.40	2.62	1.46
Foot	13.52	8.40	1.60	9.72	1.39	0.86
Plasma	0.08	0.08	1.00	0.09	0.89	0.89
Bucky	0.07	0.06	1.16	0.08	0.88	0.75
SF earthquake	0.12	0.10	1.20	0.13	0.92	0.77

this research effort by revisiting other Topological Analysis algorithms in the perspective of shared-memory and distributed parallelism.

6.1.2 Software Constraints

In-situ data post-processing can be achieved either in (1) synchronous or (2) asynchronous manner. In the first case (1), the data production process (i.e. the simulation code) is temporarily put on pause to make the computing resources available to the in-situ post-processing program. This approach requires intrusive scheduling policies to be implemented in the simulation code (to observe periodic pauses) and also imposes to store in memory all of the data that will be required by the post-processing program. A more flexible and less memory-demanding strategy is the asynchronous data post-processing (2), where data is post-processed *on-the-fly* as it is produced. Unlike the synchronous mode, this strategy does not require to store in memory large portions of the generated data and is transparent in terms of scheduling from the simulation code's perspective. While this strategy facilitates the deployment of in-situ post-processing into existing simulation codes, it shifts the scheduling constraints to the post-processing program.

In particular, in the asynchronous framework, to optimize the scheduling between data generation and post-processing, one should be able to impose resource budgets (both in terms of memory and computation time) on the data-analysis algorithms. For instance, these should be able to process data as long as it is maintained in memory and produce a usable output when it is deleted.

These software management constraints constitute a major algorithmic shift for Topological Data Analysis, since it requires to develop *best-effort* algorithms whereas only exact algorithms have been designed by the community. In other words, in this setting, one would like to develop Topological Data Analysis algorithms capable of approximating their output and refining it progressively as long as the computation time budget has not expired. Note that this *coarse-to-fine* strategy is the complete opposite of current algorithms (which are *fine-to-coarse*), where the exact solution is first computed and then progressively simplified with persistent homology mechanisms (see Chap. 2). Designing *best-effort* Topological Data Analysis algorithms is one of the key problems that need to be addressed in the upcoming years.

Research Directions

A promising research direction to develop best-effort Topological Data Analysis algorithms is to consider hierarchical data representations, as illustrated in Fig. 6.2 for the Contour Tree in the case of a hierarchy of 2D regular grids. Then, while existing algorithms can be used for the coarsest level of the data hierarchy, new

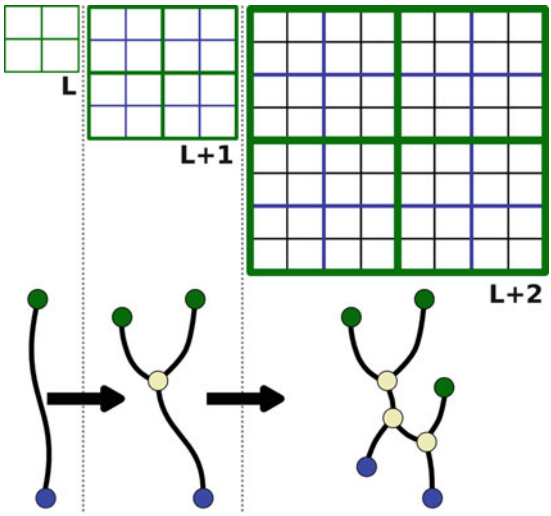


Fig. 6.2 Progressive Contour Tree Computation. A first computation is performed at the coarsest level (L) with existing algorithms, while new algorithms need to be designed to efficiently perform only the required updates (dark arrows) to obtain the Contour Tree of the next hierarchy levels ($L + 1, L + 2$)

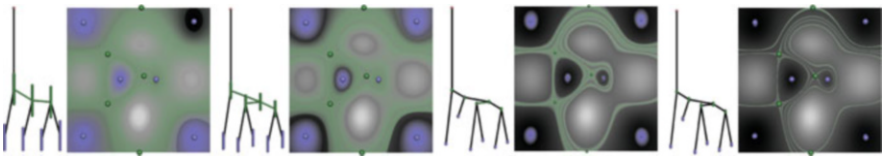


Fig. 6.3 Progressive Join Tree construction illustrated as an uncertain process. The minima and saddles of the finest resolution of the data (gradient of gray in the background) are shown with blue and green spheres respectively. For each data-resolution (from left to right: coarsest to finest) our uncertain topological analysis has been run. The predicted regions for the appearance of minima and saddles are shown in blue and green respectively

algorithms need to be designed to efficiently update the output data-structure at the next hierarchy levels by only performing the required updates.

Note that, if one considers a min-max data hierarchy (storing for each vertex of a resolution the minimum and maximum values of the vertices it represents at the next resolution), this progressive data computation can be viewed as an uncertain process, where the current resolution describes an error-affected representation of the data whose error is reduced along the data hierarchy.

As further described in the next section, an approach was recently introduced for the generalization of the Join Tree (a simpler variant of the Contour Tree) for uncertain scalar fields [56]. Figure 6.3 illustrates the computation of this structure for four different hierarchy levels (from left to right: coarsest to finest). In particular, blue and green regions denote regions of possible appearance of minima and saddles

at the finest hierarchy level. As showcased in this illustration, the Join Tree not only gets more accurate along the hierarchy from a topological point of view (number of critical points), but also from a geometrical point of view with more and more refined predictions of the locations of the critical points at the finest level of the hierarchy. Therefore, to accompany *best-effort* algorithms with prediction on the accuracy of the output, a promising direction consists in extending uncertain data processing algorithm [56] with update mechanisms to efficiently progress from a hierarchy level to another.

6.1.3 Exploration Constraints

The discussion about In-Situ data processing focused so far on the integration of Topological Data Analysis algorithms in an HPC context, mostly for data abstraction and analysis purposes, assuming these algorithms to run in batch-mode, alongside the simulation. Therefore, this discussion left the question of interactive exploration in an HPC context open.

An accepted strategy to offer interactive exploration capabilities in an HPC context is in-situ data reduction for a posteriori interactive exploration on a remote computer (or *post-mortem exploration*). Then, the challenge consists in efficiently reducing the generated data such that:

- the reduced data has a size that becomes manageable for persistent storage or network transfer (hence enabling a posteriori exploration on a remote computer);
- the reduced data still exhibits precisely the features of interest in the data.

A first obvious direction to address in-situ data reduction for post-mortem exploration is data lossy compression.

Another, orthogonal, approach for data reduction consists in anticipating the possible user interactions and pre-computing in-situ the result of each interaction. In particular, given a list of parameters that a user can control in an interactive task, the idea here is to sample this parameter space and pre-compute in-situ the visual outcome of each parameter combination. For instance, this idea is the core strategy of ParaView Cinema [73]. Given a visualization pipeline for a data-set and its list of parameters (camera position, view angle, color map, etc.), this system samples this parameter space and for each parameter combination generates a corresponding 2D rendering. Then the output collection of 2D renderings (which is orders of magnitude smaller than the actual data) can be interactively explored on a remote computer, with interactions that emulate changes in camera position, view angle, color map, etc.

While the example of ParaView cinema addresses the actual post-mortem visualization of the data, it does not address post-mortem interactive feature extraction and exploration. Thus, in the upcoming years, this strategy should be extended to Topological Data Analysis algorithms for interactive feature extraction and exploration tasks. This research problem requires to re-visit each interactive

technique based on topological abstractions and derive the appropriate output data encoding given the parameters of the interaction.

Preliminary Results

As described in Chap. 5, this research direction was already preliminarily explored for the interactive post-mortem exploration of features in the context of combustion simulations [16, 17]. In this work, we showed that a concise encoding of the possible segmentations of the data by the split tree enabled a post-mortem interactive exploration and tracking of the flames present in the simulation. A promise research direction consists in extending this strategy in multiple ways:

- Extending this approach and the segmentation encoding to only store on disk the finest segmentation and enable post-mortem progressive simplification;
- Enriching this strategy with optional lossy reduction by pre-simplifying the segmentations in-situ up to a user tolerance;
- Enriching this approach with out-of-core capabilities on the client side, to enable post-mortem exploration even with low-memory client workstations;
- Extending this approach to other, more complex, topological abstractions (Contour Tree, Reeb Graph, Morse-Smale complex);
- Specializing this general strategy to specific application scenarios;
- Extending this approach to unstructured meshes.

6.2 Emerging Data Types

In addition to the emergence of new algorithmic constraints (previous section), the current increase in the performance of supercomputers allows new simulation usages, yielding the emergence of new data types.

As described below, these new data types cannot be handled by current analysis algorithms, which also requires to revisit Topological Data Analysis algorithms to take them into account.

6.2.1 Multivariate Data

Given the recent HPC performances, it becomes now possible to model complex macroscopic processes by jointly simulating multiple physical phenomena (for instance: joint thermodynamic, computational fluid dynamics and material resistance simulations). In this context, a given simulation generates, for a given PL manifold $\mathscr{M}$, a *family* of n scalar functions that represent drastically (and physically)

different quantities (temperature, flow vorticity, material stress, etc.):

$$f : \mathscr{M} \rightarrow \mathbb{R}^n \tag{6.1}$$

This variability leads concretely to scalar fields having drastically different range values and dynamics. The joint analysis of several scalar fields defined on a common domain is therefore a major challenge that needs to be addressed to identify and quantify possible geometrical correlations between quantities.

However, traditionally, Topological Data Analysis only deals with the analysis of one scalar function defined on a single geometry. Therefore, an important research challenge consists in extending the concepts of Topological Data Analysis to *multivariate* scalar functions. This effort should be accompanied with the design of algorithms that are efficient in practice for the construction and simplification of these generalized topological abstractions, and their exploitation in specific application problems.

Preliminary Results

In the context of a collaboration with the University of Leeds, we recently introduced an algorithm with efficient practical performances for the problem of Reeb space computation [48]. This construction generalizes the notion of Reeb graph to multivariate scalar functions (Fig. 6.4) by tracking connected components of *fibers* (multivariate analogs of level-sets). In particular, we presented an efficient algorithm [117] for the computation of the Reeb space of an input bivariate piecewise linear scalar function $f : \mathscr{M} \rightarrow \mathbb{R}^2$ defined on a PL 3-manifold $\mathscr{M}$. By extending and generalizing algorithmic concepts from the univariate case to the bivariate one, we

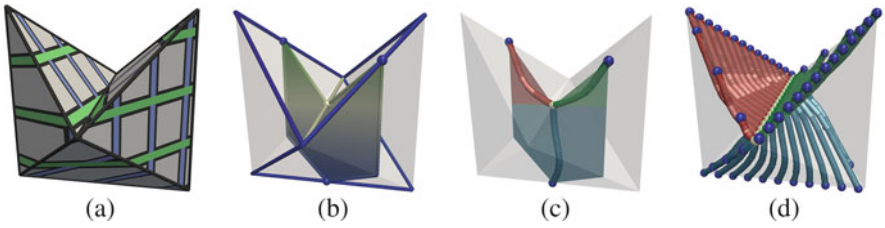


Fig. 6.4 Relation between the Reeb space [117] and the Reeb graph on a simple bivariate example (the X and Y 3D coordinates serve as input bivariate function). (**a**) Input bivariate function (left: green and blue perpendicular lines denote u and v level-sets). (**b**) Given an isovalue q_u, the restriction to $u^{-1}(q_u)$ of $v : \mathscr{M} \rightarrow \mathbb{R}$ is shown by the colored surface (from blue to green). The critical points [7] of this restriction (blue spheres: extrema, white sphere: saddle) are located on the Jacobi set [40] of f (extremum and saddle edges are shown in blue and white respectively). (**c**) Reeb graph of the restriction. (**d**) The Reeb space can be interpreted as a continuous stacking of such Reeb graphs, as q_u continuously evolves. ©2016 IEEE. Reprinted, with permission, from [117]

Table 6.2 Running time comparison with the Joint Contour Net [18] (one thread, two quantizations) and the chamber approach [48] (parallel)

Data-set	n_T	JCN-1 [18]		JCN-100 [18]		Chambers-P [48]	
Water dimer	302,715	20.3	**0.1**	290.4	**1.6**	1388.4	**50.7**
Gargoyle	1,098,602	226.5	**0.3**	2011.6	**2.6**	8654.6	**120.7**
Mechanical	1,482,764	490.0	**1.4**	5649.2	**16.0**	8116.6	**249.0**
Bitorus	2,555,904	521.5	**7.3**	4903.7	**68.3**	23,361.2	**3385.7**
Vortex street	5,060,475	587.7	**0.6**	15,660.0	**16.8**	103,862.0	**1128.9**

Running times are provided in seconds (obtained with a C++ implementation on a Xeon 2.6 GHz CPU, with 12 threads). The bold numbers indicate the speedups obtained by our algorithm

reported the first practical, output-sensitive algorithm for the exact computation of such a Reeb space. The algorithm starts by identifying the Jacobi set [40] of f (Fig. 6.4), the bivariate analogs of critical points in the univariate case. Next, the Reeb space is computed by segmenting the input mesh along the new notion of *Jacobi Fiber Surfaces*, the bivariate analog of critical contours in the univariate case. We additionally presented a simplification heuristic that enables the progressive coarsening of the Reeb space. Our algorithm is simple to implement and most of its computations can be trivially parallelized. Performance numbers demonstrate orders of magnitude speedups over previous approaches [18, 48] (Table 6.2), enabling for the first time the tractable computation of bivariate Reeb spaces in practice. Moreover, unlike range-based quantization approaches (such as the Joint Contour Net), our algorithm is parameter-free. We demonstrate the utility of this approach by using the Reeb space as a semi-automatic segmentation tool for bivariate data. In particular, we introduce *continuous scatterplot peeling* (Fig. 6.5), a technique which enables the reduction of the cluttering in the continuous scatterplot, by interactively selecting the features of the Reeb space to project. We consider this approach as the reference algorithm for the problem of the efficient computation of bivariate Reeb spaces and we released a VTK-based open source implementation of it in the hope of a rapid uptake of this method.

As Reeb graphs and Contour Trees offered new analysis and interaction capabilities for scalar fields (see Chap. 2), it seems reasonable to believe that Reeb spaces (and in particular our efficient algorithm) will enable a wide range of analysis methods for bivariate data, hence initiating a new line of research dedicated to the extensions of the application of the Reeb graph in Scientific Visualization to the case of bivariate data. In the following, we further motivate the applicative interest of such analysis capabilities.

In the process of defining our fast algorithm for Reeb space computation, we needed to introduce a novel construction called *Fiber Surfaces*, which are preimages of PL 1-manifolds through bivariate functions. Surprisingly we discovered that this novel construction was somehow implicitly known by the volume rendering community for multi-dimensional transfer function definition. However, this community could only visualize a volume rendering of these fiber surfaces and no algorithm was documented to extract them geometrically. We therefore presented

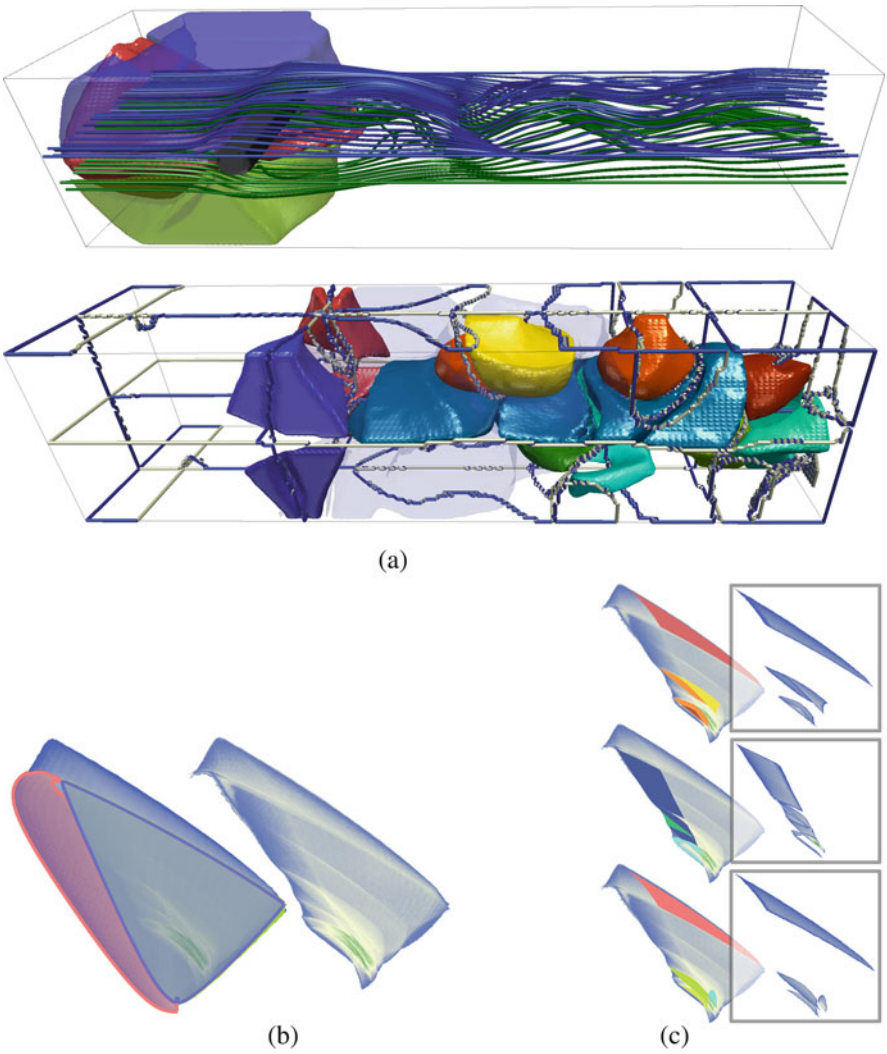


Fig. 6.5 The Reeb space $\mathscr{R}(f)$ of a bivariate function f segments the domain $\mathscr{M}$ into regions where *fibers*, multivariate analogs of level-sets, are made of a single connected component. This allows the automatic separation of volumetric regions that project to the same areas in the continuous scatterplot (CSP). In this flow example (streamlines are shown in green and blue in (**a**), top), the Reeb space of the velocity and curl magnitudes is used to segment the data into its main features (**a**). The largest regions of $\mathscr{R}(f)$ are located before the obstacle (in blue, red and green (**a**), top). While these features are not important for the understanding of the structure of the turbulent flow, their projections cover most of the CSP (red, blue and green polygons in (**b**), bottom left). In particular, due to the symmetry in the data, the blue and green regions nearly coincide in the CSP. Removing these regions from the projection results in a less cluttered CSP, better revealing the projections of the turbulent features of the flow ((**b**), bottom center). The user can further inspect these features with localized CSPs (**c**). These visualizations are enabled by our new Reeb space computation algorithm, which computes this segmentation in a minute and a half, while previous techniques either take days to compute or hours to approximate the result. ©2016 IEEE. Reprinted, with permission, from [117]

a simple approach to compute fiber surfaces and showed the applicative interests of such constructions for data segmentation purposes in various application fields [21]. For instance, Figs. 6.6 and 6.7 demonstrate the superiority of fiber surfaces over isosurfaces for the user-driven segmentation of simulated or acquired data.

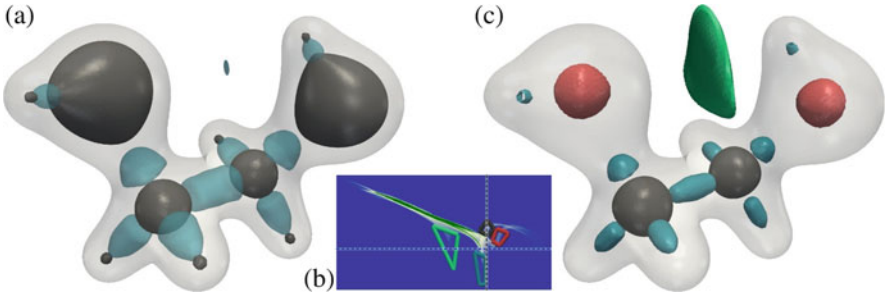


Fig. 6.6 Isosurfaces (**a**) and fiber surfaces (**c**) of a bivariate field representing chemical interactions within an ethane-diol molecule ((**b**) continuous scatter plot, X: electron density, Y: reduced gradient). While isosurfaces of the electron density capture regions of influence of atoms ((**a**) grey), they do not distinguish atom types. Similarly, isosurfaces of the reduced gradient capture regions of chemical interactions ((**a**) blue) but do not distinguish covalent from non-covalent interactions. In contrast, polygons isolating the main features of the continuous scatter plot (**b**) yield fiber surfaces (**c**) distinguishing atom types (red and grey) as well as interaction types (blue and green). ©2016 IEEE. Reprinted, with permission, from [74]

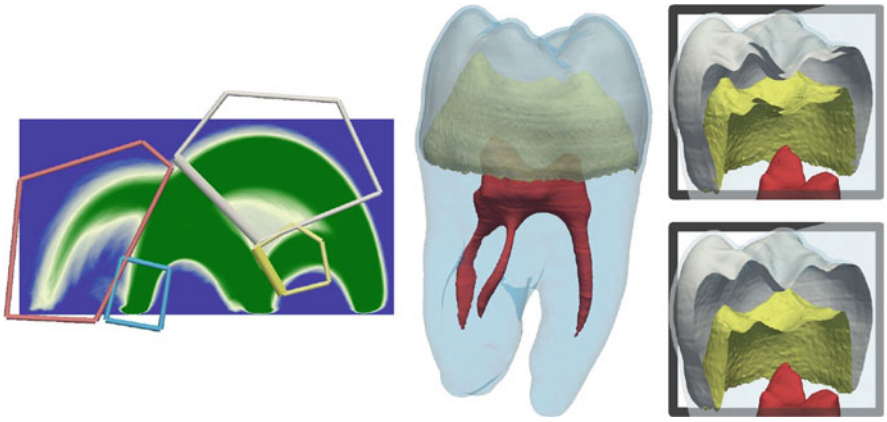


Fig. 6.7 Fiber surface extraction on an acquired data set (CT-scan). The considered bivariate data is given by the acquired value (X axis of the continuous scatter plot, left) and the magnitude of its gradient (Y axis, left). By manually contouring the main features of the continuous scatter plot (left), the user can easily extract with the corresponding fiber surfaces the boundary of the regions of interest of the volume (middle: pulp in red, dentin in blue, enamel in white, boundary between the dentin and the enamel in yellow). Thanks to our new algorithm, fiber surfaces can also be computed for non-closed polygons (thicker edges on the left, fiber surfaces in the bottom zoom-in view on the right). ©2016 IEEE. Reprinted, with permission, from [74]

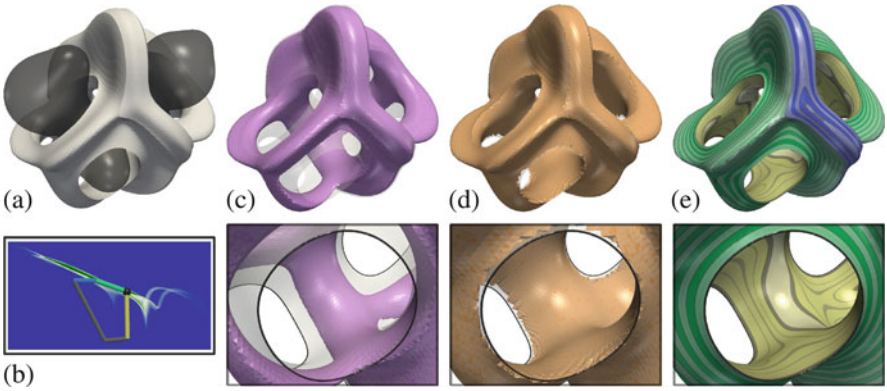


Fig. 6.8 Fiber surface extraction on a bivariate field (electron density and reduced gradient) representing chemical interactions within an ethane-diol molecule (dark surface in (**a**)). Fiber surfaces are defined as pre-images of polygons drawn in range space (i.e. the continuous scatter plot (**b**)). The first algorithm for their computation [21] relies on a distance field computation on a rasterization of the range. While increasing the raster resolution results in more accurate fiber surfaces ((**c**) 162, (**d**) 10,242), even for large resolutions, the distance field intrinsically fails at capturing sharp features of the fiber surface (here polygon bends in the range, black sphere (**b**)), as showcased in the zoom-views (bottom) where the corresponding fibers are displayed with black curves. Our new approach [74] introduces the first algorithm for the exact computation of fiber surfaces on tetrahedral meshes. It accurately captures sharp features (**e**) and enhances fiber surfaces with polygon-edge segmentation (colors in (**b**) and (**e**)) and individual fibers (**e**, bottom) to better convey the relation between fiber surfaces and range features. ©2016 IEEE. Reprinted, with permission, from [74]

However this first algorithm was slow in practice for large data-sets and was only approximate, as illustrated in Fig. 6.8. Therefore, we introduced a second algorithm for the exact computation of fiber surfaces as well as several acceleration mechanisms (generalized from isosurface extraction acceleration) which enabled an interactive exploration of the space of fiber surfaces [74]. We consider this latter algorithm as the reference for the problem of efficient and exact fiber surface computation in bivariate scalar fields and we released a VTK-based open source implementation of it in the hope of a rapid uptake of this method.

As illustrated in Figs. 6.6, 6.7 and 6.8, fiber surface extraction currently requires manual intervention based on the perceived features in the continuous scatter plot [5] of the data. However, as illustrated in the case of the two oxygen atoms in Fig. 6.6 (two red surfaces, right), several distinct features may exist in 3D for the same location in the continuous scatter plot (single red curve, middle). To disambiguate these configurations and to further help the user explore the continuous scatter plot, a promising direction consists in investigating the usage of the Reeb space for the automatic feature segmentation and simplification in bivariate data. Indeed, the Reeb space is guaranteed by definition to segment the 3D space into regions where fibers (and fiber surfaces) are made of a single connected component. Such a contribution would have an impact on any visualization task dealing with bivariate data, such as

feature extraction or transfer function design. However, to make such a topological abstraction useful in practice, as it was the case for scalar data, persistent homology concepts need to be generalized to bivariate data to allow for efficient simplification algorithms. Thus, in the upcoming years, based on the preliminary results described above, a major research challenge will consist in generalizing Topological Data Analysis to bivariate data by extending persistent homology concepts as well as applications of the Reeb graph in visualization to the bivariate case.

6.2.2 Uncertain Data

A physical model is often dictated by a number of parameters (initial conditions, boundary conditions, etc.). Given the recent HPC advances, the fine sampling of this parameter space becomes feasible (yielding one simulation output per combination of parameters). This process, called parameter study, is central to the understanding of the uncertainty that accompanies any physical process. For instance, it enables to identify parameter ranges for an efficient and safe functioning of a complex system. This type of simulation also generates, given a common domain, a *family* of n scalar fields, that model an uncertain process:

$$f : \mathscr{M} \rightarrow \mathbb{R}^n \tag{6.2}$$

This collection of n scalar fields is often called an *ensemble data-set* and each scalar field a *member* of this ensemble. Alternatively, each scalar field can be seen as an *observation* of an *uncertain scalar field*, which maps each point of the domain to a random variable. In this latter context, the data is typically represented for each vertex of the domain by a probability density function (estimated for instance by a histogram). Analyzing this family of scalar fields (ensemble members or observations) as a whole to identify, extract and understand the conditions of appearance of features is a major upcoming challenge in visualization and analysis.

For uncertain data, the number n of considered scalar fields is typically much higher than in the case of multivariate data (previous section). Moreover, from a theoretical point of view, the topological analysis of multivariate data seems to have an applicative interest only when the dimension of the range is lower than the dimension of the domain (typically three). For instance, as of $n = 3$, the Reeb space of a generic multivariate scalar field defined on a PL 3-manifold $\mathscr{M}$ is $\mathscr{M}$ itself. Thus, for n values beyond the dimension of the domain, another direction needs to be considered. Therefore, the topological analysis of multivariate fields seems of little importance for the processing of uncertain data. Instead, in this topic, a major research challenges consists in generalizing the constructions of Topological Data Analysis to uncertain scalar fields (that map each point of the domain to a random variable).

Preliminary Results

We recently introduced the first non-local, combinatorial characterization of critical points and their global relation in 2D uncertain scalar fields [56]. The characterization is based on the analysis of the support of the probability density functions of the input random variables. Given two scalar fields representing reliable estimations of the bounds of this support (noted $f^- : \mathcal{M} \rightarrow \mathbb{R}$ and $f^+ : \mathcal{M} \rightarrow \mathbb{R}$), based on the observation that their sub-level sets were point-wise nested, we described sufficient conditions (Fig. 6.9) for the appearance of *mandatory minima*. This latter construction generalizes the notion of local minimum from PL scalar fields to uncertain scalar fields:

Definition 6.1 (Mandatory Minimum) A *mandatory minimum* M is a minimal connected component $C \subset \mathcal{M}$ with a minimal interval $I \subset \mathbb{R}$ such that any realization field g (a scalar field that maps each vertex to a realization of its random variable) admits at least one minimum $m \in C$ with $g(m) \in I$. C is called the *critical component* of M and I its critical interval.

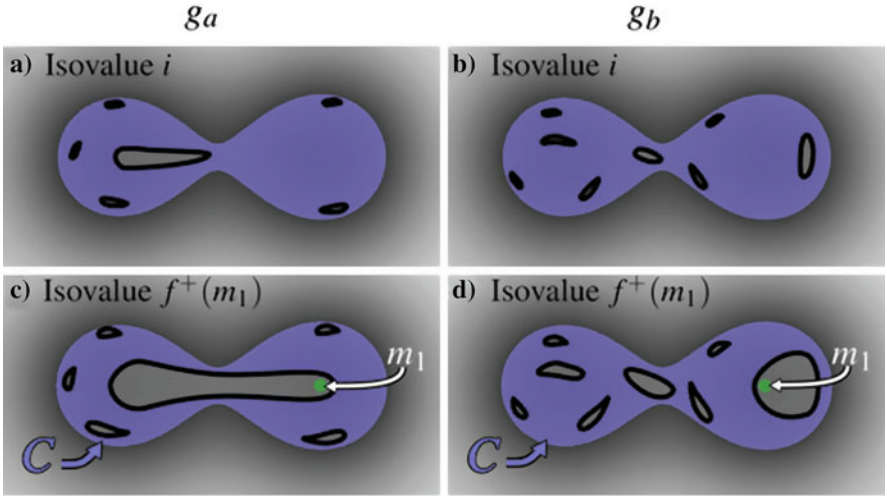


Fig. 6.9 Sub-levels of the lower (f^-, blue) and upper (f^+, green) bounds of an uncertain 2D scalar field for increasing isovalues (top to bottom). The sub-level sets of two realization fields of the uncertain data are shown in grey (left: g_a, right: g_b). All points in the blue region have a non-zero probability to get a value lower than the current isovalue, while all the points in the green region have a probability of 1 to be lower. Therefore, the appearance of a unique local minimum m of f^+ (in green) within a connected component C of the sub-level set of f^- is a sufficient condition for the appearance of a *mandatory minimum*, whose critical component is C. Such a configuration indeed implies the existence of at least one connected component of sub-level set of any realization field g, included in C and including m (bottom row). Such a component implies the existence of at least one local minimum of g in C. ©2014 Wiley. Reprinted, with permission, from [56]

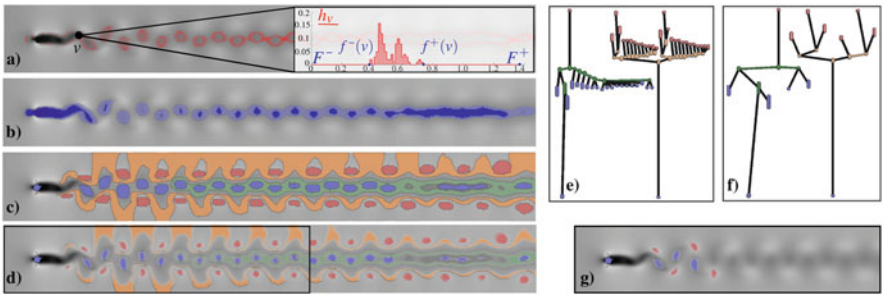


Fig. 6.10 Mandatory critical points of the velocity magnitude of the uncertain Kármán vortex street. (**a**) Each vertex v is assigned with a histogram h_v estimating its probability density function. The shades of red show the point-wise probability for the isovalue 0.85. (**b**) The support of h_v is visualized by the lower (f^-, light blue) and upper (f^+, dark blue) bound fields. (**c**) Depicts the mandatory critical points (blue: minimum, green: join saddle, yellow: split saddle, red: maximum), (**d**) illustrates the spatial uncertainty within the components. (**e**) Shows the mandatory join/split tree, and (**f**) and (**g**) the simplified visualization. ©2014 Wiley. Reprinted, with permission, from [56]

Note that this construction nicely generalizes the notions of critical points and critical values to critical components and critical intervals. The notions of *mandatory maxima*, *mandatory join saddles* and *mandatory split saddles* are defined similarly.

Thanks to the specification of the appearance conditions of mandatory critical points, we described a combinatorial algorithm for their extraction. This strategy hence identifies spatial regions and function ranges where critical points have to occur in any realization of the input uncertain data, as illustrated in Fig. 6.10. In other words, these regions form the common topological denominator to all realizations (or observations) of an uncertain scalar field: they describe the common topological features that appear in all of these scalar fields. From an application point of view, this approach enables to predict the location and the minimum number of vortices for instance in uncertain flow simulations, as illustrated in Fig. 6.10.

Our algorithm provides a global pairing scheme for mandatory critical points which is used to construct mandatory join and split trees. These trees enable a visual exploration of the common topological structure of all possible realizations of the uncertain data. To allow multi-scale visualization, we introduce a simplification scheme for mandatory critical point pairs revealing the most dominant features. Our technique is purely combinatorial and handles parametric distribution models and ensemble data. It does not depend on any computational parameter and does not suffer from numerical inaccuracy or global inconsistency. The algorithm exploits ideas of the established join/split tree computation. It is therefore simple to implement, and its time complexity is output-sensitive. Experiments demonstrated the accuracy and usefulness of our method on synthetic and real-world uncertain data-sets. Thanks to its robustness and multi-scale nature, we consider this approach

as the reference algorithm for the problem of mandatory critical point extraction in 2D uncertain scalar fields.

Despite this strong result, this first attempt at extending Topological Data Analysis to uncertain data raised even more questions than answers. In particular, despite their strong applicative interest, the topological features that are common to all realizations of an uncertain process (i.e. *that have a probability of appearance of 1*) only constitute a sub-set of the features users are interested in. From a theoretical point of view, a natural question that arise is:

What about the critical points with a probability of appearance lower than 1?

This question has strong applicative implications. Often, the phenomena under investigation can reveal distinct regimes and it is important to understand the probability of appearance of these regimes as well as the conditions (sets of parameters) for their appearance. Figure 6.11 shows such an example where the uncertain data clearly exhibits three distinct regimes that are not identified and characterized by our approach. Thus, in the upcoming years, an important challenge is the topological classification of ensemble data (or uncertain data) for regime extraction and characterization. As described above, such a research direction is important because of its applicative context (parameter studies) where such analysis capabilities are necessary to abstract, interact with and analyze uncertain data. From a technical perspective, this problem is highly challenging for several reasons. First, it requires to bridge the gap between algorithmic techniques coming from different communities (Topological Data Analysis and Machine Learning). Second, given the size of the ensemble data-sets to consider, it is likely that such analyses can only be performed in-situ. Last, it requires to address sub-problems which

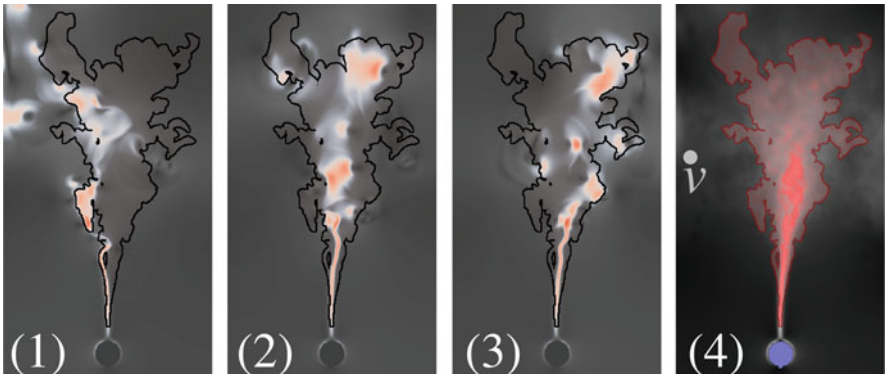


Fig. 6.11 Uncertain scalar field describing the velocity field caused by a heat source (bottom). Our algorithm for mandatory critical point computation extracts a dominant maximum (in red, rightmost) which describes the cone through which the flow travels (and attain velocity maxima) for all observations. However, as illustrated on the left with three observation fields (1–3), the flow describes three distinct regimes (leftward, center and rightward trajectories) which are not identified and characterized by our approach. ©2014 Wiley. Reprinted, with permission, from [56]

have not (or only partially) been addressed by the community. In particular, to develop relevant models for the topological classification of ensemble data-sets, the following research problems need to be addressed:

1. Designing stable and discriminative distance metrics between Topological Abstractions as well as efficient algorithms for their computation;
2. Designing efficient and relevant algorithms for the clustering of Topological Abstractions given the above distance metrics;
3. Analyzing the parameter space of parameter studies in the light of the topological clustering of their observations and specializing this overall strategy to several application scenarios.

Chapter 7
Conclusion

This book presented an overview of Topological Data Analysis for Scientific Visualization. After a concise tutorial and survey on the core concepts of Topological Data Analysis (Chap. 2), it presented examples of reference algorithms in the field, in particular in each of the following topics:

1. **Abstraction:** In this topic, we presented examples of reference algorithms for the computation of Topological Abstractions of raw scalar data. In particular, we described a general algorithm for the topological simplification of scalar data on surfaces [118] and showed that, when used as a pre-processing step, it could drastically improve the time performance of topological abstraction computation algorithms. Thanks to its generality, robustness, ease of implementation and practical performances, we consider this algorithm as the reference for the problem of topological simplification of scalar data on surfaces. Next, we presented an efficient algorithm for the computation of the Reeb graph of scalar data defined on PL 3-manifolds in $\mathbb{R}^3$ [125]. This approach described the first practical algorithm for volumetric meshes, with virtually linear scalability in practice and up to 3 orders of magnitude speedups with regard to previous work. Such an algorithm enabled for the first time the generalization of contour-tree based techniques to general, non simply-connected volumes. We considered this algorithm as the reference for the problem of Reeb graph computation on volumes until an optimal time complexity algorithm was introduced 3 years later [89].
2. **Interaction:** In this topic, we presented examples of efficient algorithms for the interactive editing of Topological Abstractions for data exploration and user-driven segmentation tasks. In particular, we described interactive algorithms based on the Reeb graph for the topological simplification of isosurfaces, both on-line [125] and in an in-situ context [101]. Moreover, we presented two algorithms for the interactive editing of Topological Abstractions (the Morse-Smale complex [63] and the Reeb graph [128]) to enable user-driven topological data segmentation. In particular, these algorithms enable to incorporate user

J. Tierny, *Topological Data Analysis for Scientific Visualization*,
Mathematics and Visualization, https://doi.org/10.1007/978-3-319-71507-0_7

knowledge in segmentation tasks where features of interest are aligned with the gradient of the data (Morse-Smale complex) or with its level sets (Reeb graph).

3. **Analysis:** In this topic, we presented examples of specializations of Topological Data Analysis techniques to specific application problems in combustion [16, 17] and chemistry [55]. In particular we showed how standard Topological Data Analysis could be adapted to extract and analyze features of interest in these fields. These two applications demonstrated that thanks to their robustness and relevance, Topological Data Analysis techniques could address diversified scientific issues well beyond the sole scope of Computer Science.

All of these results were obtained in collaboration with several research groups (University of Utah, Lawrence Livermore National Laboratory, Lawrence Berkeley National Laboratory, Universidade de Sao Paulo, New York University, Sorbonne Universites, Clemson University, University of Leeds) as well as students whom I informally or formally advised.

In the process of introducing the key concepts, algorithms and applications of Topological Data Analysis (through the tutorial Chap. 2 and Chaps. 3–5), we emphasized the robustness, time-performance and efficiency of such algorithms for several data abstraction, exploration and analysis tasks. We also underlined the maturity attained by these techniques, as demonstrated by the emergence of open-source implementations of these algorithms [130], to which I partially contributed. This level of maturity is also demonstrated by the numerous collaborations with domain experts for the resolution of problems that go well beyond the sole scope of Computer Science, including combustion, molecular chemistry, cosmology, and fluid dynamics for instance.

Despite this level of maturity, we described in Chap. 6 important research challenges in Topological Data Analysis for Scientific Visualization to be addressed in the future. In particular, we discussed how these challenges were greatly related to the upcoming generation of supercomputers, which will (1) impose new constraints on analysis algorithms and which will (2) enable the generation of new data types to analyze. Put together, these two challenges require to deeply revisit Topological Data Analysis algorithms and to reboot the research effort made by the community in this area over the last two decades.

In particular, due to the input-output bandwidth bottleneck, in-situ data processing will become inevitable. This requires Topological Data Analysis techniques to adapt to new hardware constraints (massively parallel and distributed architectures), new software constraints (with best effort computations, in order to ease the scheduling between data generation and post-processing) and exploration constraints (requiring to define in-situ data reduction algorithms for post-mortem interactive exploration). For each of these constraints, we described the corresponding algorithmic challenges to address.

Furthermore, the computing performances of the next generation of supercomputers will allow for a systematic usage of parameter studies or multi-physics simulations. Both of these usages will make emerging data types much more prominent such as (1) multivariate data and (2) uncertain data. Generalizing

Topological Data Analysis to these two data types requires not only a major algorithmic effort but also an important theoretical investigation about the extension of Morse theory concepts. As described through Chap. 6, preliminary results in this topic [21, 56, 74, 117] raised even more questions than answers, but still enabled to identify promising research directions to address these challenges in the future.

References

1. A. Acharya, V. Natarajan, A parallel and memory efficient algorithm for constructing the contour tree. Proc. Pacific Vis. **2015**, 271–278 (2015)
2. P.K. Agarwal, H. Edelsbrunner, J. Harer, Y. Wang, Extreme elevation on a 2-manifold, in *Proceeding of ACM Symposium on Computational Geometry* (2004)
3. D. Attali, M. Glisse, S. Hornus, F. Lazarus, D. Morozov, Persistence-sensitive simplification of functions on surfaces in linear time, in *TopoInVis Workshop* (2009)
4. D. Attali, U. Bauer, O. Devillers, M. Glisse, A. Lieutier, Homological reconstruction and simplification in R3, in *Proceeding of ACM Symposium on Computational Geometry* (2013)
5. S. Bachthaler, D. Weiskopf, Continuous scatterplots. IEEE Trans. Vis. Comput. Graph. **14**, 1428–1435 (2008)
6. R.F.W. Bader, *Atoms in Molecules: A Quantum Theory* (Oxford University Press, Oxford, 1994)
7. T.F. Banchoff, Critical points and curvature for embedded polyhedral surfaces. Am. Math. Mon. **77**, 475–485 (1970)
8. U. Bauer, C. Lange, M. Wardetzky, Optimal topological simplification of discrete functions on surfaces. Discret. Comput. Geom. **47**, 347–377 (2012)
9. B. Bedat, R.K. Cheng, Experimental study of premixed flames in intense isotropic turbulence. Combust. Flame **100**, 485–494 (1995)
10. S. Biasotti, L. De Floriani, B. Falcidieno, P. Frosini, D. Giorgi, C. Landi, L. Papaleo, M. Spagnuolo, Describing shapes by geometrical-topological properties of real functions. ACM Comput. Surv. **40**, 1–87 (2008)
11. S. Biasotti, D. Giorgio, M. Spagnuolo, B. Falcidieno, Reeb graphs for shape analysis and applications. Theor. Comput. Sci. **392**, 5–22 (2008)
12. R.A. Boto, J.C. Garcia, J. Tierny, J.-P. Piquemal, Interpretation of the reduced density gradient. Mol. Phys. **114**, 1406–1414 (2016)
13. R.L. Boyell, H. Ruston, Hybrid techniques for real-time radar simulation, in *Proceeding of the IEEE Fall Joint Computer Conference* (1963)
14. P.-T. Bremer, H. Edelsbrunner, B. Hamann, V. Pascucci, A multi-resolution data structure for 2-dimensional Morse functions, in *IEEE Visualization* (2003), pp. 139–146
15. P.-T. Bremer, H. Edelsbrunner, B. Hamann, V. Pascucci, A topological hierarchy for functions on triangulated surfaces. IEEE Trans. Vis. Comput. Graph. **10**, 385–396 (2004)
16. P.T. Bremer, G. Weber, J. Tierny, V. Pascucci, M. Day, J. Bell, A topological framework for the interactive exploration of large scale turbulent combustion, in *Proceeding of IEEE eScience* (2009)

J. Tierny, *Topological Data Analysis for Scientific Visualization*,
Mathematics and Visualization, https://doi.org/10.1007/978-3-319-71507-0

17. P.T. Bremer, G. Weber, J. Tierny, V. Pascucci, M. Day, J. Bell, Interactive exploration and analysis of large scale simulations using topology-based data segmentation. IEEE Trans. Vis. Comput. Graph. **17**, 1307–1324 (2011)
18. H. Carr, D. Duke, Joint contour nets. IEEE Trans. Vis. Comput. Graph. **20**, 1100–1113 (2014)
19. H. Carr, J. Snoeyink, U. Axen, Computing contour trees in all dimensions, in *Proceeding of Symposium on Discrete Algorithms* (2000), pp. 918–926
20. H. Carr, J. Snoeyink, M. van de Panne, Simplifying flexible isosurfaces using local geometric measures, in *Proceeding of IEEE Visualization* (2004), pp. 497–504
21. H. Carr, Z. Geng, J. Tierny, A. Chattopadhyay, A. Knoll, Fiber surfaces: generalizing isosurfaces to bivariate data. Comput. Graph. Forum **34**, 241–250 (2015)
22. R. Chaudret, B. de Courcy, J. Contreras-Garcia, E. Gloaguen, A. Zehnacker-Rentien, M. Mons, J.-P. Piquemal, Unraveling non-covalent interactions within flexible biomolecules: from electron density topology to gas phase spectroscopy. Phys. Chem. Chem. Phys. **16**, 9876–9891 (2014)
23. F. Chazal, D. Cohen-Steiner, L.J. Guibas, F. Memoli, S.Y. Oudot, Gromov–Hausdorff stable signatures for shapes using persistence. Comput. Graph. Forum **28**, 1393–1403 (2009)
24. F. Chen, H. Obermaier, H. Hagen, B. Hamann, J. Tierny, V. Pascucci, Topology analysis of time-dependent multi-fluid data using the Reeb graph. Comput. Aided Geom. Des. **30**, 557–566 (2013)
25. R.K. Cheng, Velocity and scalar characteristics of premixed turbulent flames stabilized by weak swirl. Combust. Flame **101**(1–2), 1–14 (1995)
26. M.M. Cohen, *A Course in Simple-Homotopy Theory* (Springer, Berlin, 1973)
27. D. Cohen-Steiner, H. Edelsbrunner, J. Harer, Stability of persistence diagrams, in *Proceedings of ACM Symposium on Computational Geometry* (2005)
28. K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, V. Pascucci, Loops in Reeb graphs of 2-manifolds, in *Proceedings of ACM Symposium on Computational Geometry* (2003), pp. 344–350
29. J. Contreras-Garcia, E. Johnson, S. Keinan, R. Chaudret, J.-P. Piquemal, D. Beratan, W. Yang, NCIPLOT: a program for plotting noncovalent interaction regions. J. Chem. Theory Comput. **7**(3), 625–632 (2011)
30. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms* (MIT Press, Cambridge, 2009)
31. T.A. Davis, W.W. Hager, Dynamic supernodes in sparse Cholesky update/downdate and triangular solves. ACM Trans. Math. Softw. **35**(4), 1–23 (2009)
32. M. Day, J. Bell, P.-T. Bremer, V. Pascucci, V. Beckner, M. Lijewski, Turbulence effects on cellular burning structures in lean premixed hydrogen flames. Combust. Flame **156**, 1035–1045 (2009)
33. L. De Floriani, U. Fugacci, F. Iuricich, P. Magillo, Morse complexes for shape segmentation and homological analysis: discrete models and algorithms. Comput. Graph. Forum **34**, 761–785 (2015)
34. T. Dey, S. Guha, Computing homology groups of simplicial complexes in $\mathbb{R}^3$. J. ACM **45**, 266–287 (1998)
35. S. Dillard, A contour tree library (2007), http://graphics.cs.ucdavis.edu/~sdillard/libtourtre/doc/html/
36. DOE and ASCAC, Synergistic challenges in data-intensive science and exascale computing. Technical report, DoE and Advanced Scientific Computing Advisory Committee, Data Subcommittee (2013)
37. S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, J.C. Hart, Spectral surface quadrangulation. ACM Trans. Graph. **25**, 1057–1066 (2006)
38. H. Doraiswamy, V. Natarajan, Efficient output sensitive construction of Reeb graphs, in *International Symposium on Algorithms and Computation* (2008)
39. H. Doraiswamy, V. Natarajan, Computing Reeb graphs as a union of contour trees. IEEE Trans. Vis. Comput. Graph. **19**, 249–262 (2013)

40. H. Edelsbrunner, J. Harer, *Jacobi Sets of Multiple Morse Functions*, Cambridge Books Online (2004)
41. H. Edelsbrunner, J. Harer, Persistent homology—a survey, in *Surveys on Discrete and Computational Geometry* (American Mathematical Society, Providence, 2008)
42. H. Edelsbrunner, J. Harer, *Computational Topology: An Introduction* (American Mathematical Society, Providence, 2009)
43. H. Edelsbrunner, E. Mucke, Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. ACM Trans. Graph. **9**, 66–104 (1990)
44. H. Edelsbrunner, D. Letscher, A. Zomorodian, Topological persistence and simplification. Discret. Comput. Geom. **28**, 511–533 (2002)
45. H. Edelsbrunner, J. Harer, A. Zomorodian, Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. Discret. Comput. Geom. **30**, 87–107 (2003)
46. H. Edelsbrunner, J. Harer, V. Natarajan, V. Pascucci, Morse-Smale complexes for piecewise linear 3-manifolds, in *Proceedings of ACM Symposium on Computational Geometry* (2003)
47. H. Edelsbrunner, D. Morozov, V. Pascucci, Persistence-sensitive simplification of functions on 2-manifolds, in *Proceedings of ACM Symposium on Computational Geometry* (2006), pp. 127–134
48. H. Edelsbrunner, J. Harer, A. Patel, Reeb spaces of piecewise linear mappings, in *Proceedings of ACM Symposium on Computational Geometry* (2008)
49. T. Etiene, L.G. Nonato, C. Scheidegger, J. Tierny, T. Peters, V. Pascucci, M. Kirby, C. Silva, Topology verification for isosurface extraction. IEEE Trans. Vis. Comput. Graph. **18**, 952–965 (2012)
50. G. Favelier, C. Gueunet, J. Tierny, Visualizing ensembles of viscous fingers, in *IEEE Scientific Visualization Contest* (2016)
51. R. Forman, A user's guide to discrete morse theory, in *Proceedings of the International Conference on Formal Power Series and Algebraic Combinatorics* (2001)
52. M. Gargouri, J. Tierny, E. Jolivet, P. Petit, E. Angelini, Accurate and robust shape descriptors for the identification of rib cage structures in ct-images with random forests, in *IEEE Symposium on Biomedical Imaging* (2013)
53. R. Ghrist, Barcodes: the persistent topology of data. Am. Math. Soc. **45**, 61–75 (2007)
54. N. Gillet, R. Chaudret, J. Contreras-Garcia, W. Yang, B. Silvi, J.P. Piquemal, Coupling quantum interpretative techniques: another look at chemical mechanisms in organic reactions. J. Chem. Theory Comput. **8**, 3993–3997 (2012)
55. D. Guenther, R. Alvarez-Boto, J. Contreras-Garcia, J.-P. Piquemal, J. Tierny, Characterizing molecular interactions in chemical systems. IEEE Trans. Vis. Comput. Graph. **20**, 2476–2485 (2014)
56. D. Guenther, J. Salmon, J. Tierny, Mandatory critical points of 2D uncertain scalar fields. Comput. Graph. Forum **33**, 31–40 (2014)
57. C. Gueunet, P. Fortin, J. Jomier, J. Tierny, Contour forests: fast multi-threaded augmented contour trees, in *Proceedings of IEEE Symposium on Large Data Analysis and Visualization (LDAV)* (2016)
58. C. Gueunet, P. Fortin, J. Jomier, J. Tierny, Task-based augmented merge trees with fibonacci heaps, in *Proceedings of IEEE Symposium on Large Data Analysis and Visualization (LDAV)* (2017)
59. A. Gyulassy, Combinatorial construction of Morse-Smale complexes for data analysis and visualization. PhD thesis, University of California at Davis, 2008
60. A. Gyulassy, V. Natarajan, M. Duchaineau, V. Pascucci, E. Bringa, A. Higginbotham, B. Hamann, Topologically clean distance fields. IEEE Trans. Vis. Comput. Graph. **13**, 1432–1439 (2007)
61. A. Gyulassy, P.T. Bremer, B. Hamann, V. Pascucci, A practical approach to Morse-Smale complex computation: scalability and generality. IEEE Trans. Vis. Comput. Graph. **14**, 1619–1626 (2008)
62. A. Gyulassy, P.T. Bremer, V. Pascucci, Computing Morse-Smale complexes with accurate geometry. IEEE Trans. Vis. Comput. Graph. **18**, 2014–2022 (2012)

63. A. Gyulassy, D. Guenther, J.A. Levine, J. Tierny, V. Pascucci, Conforming Morse-Smale complexes. IEEE Trans. Vis. Comput. Graph. **20**, 2595–2603 (2014)
64. A. Gyulassy, P.T. Bremer, R. Grout, H. Kolla, J. Chen, V. Pascucci, Stability of dissipation elements: a case study in combustion. Comput. Graph. Forum **33**, 51–60 (2014)
65. W. Harrold, A journey to exascale computing, in *Proceedings of SuperComputing* (2012)
66. C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. De Floriani, G. Scheuermann, H. Hagen, C. Garth, A survey of topology-based methods in visualization. Comput. Graph. Forum **35**, 643–667 (2016)
67. M. Hilaga, Y. Shinagawa, T. Kohmura, T. Kunii, Topology matching for fully automatic similarity estimation of 3D shapes, in *Proceedings of ACM SIGGRAPH* (2001)
68. J. Huang, M. Zhang, J. Ma, X. Liu, L. Kobbelt, H. Bao, Spectral quadrangulation with orientation and alignment control. ACM Trans. Graph. **27**, 147 (2008)
69. R. Ibata, G. Lewis, A. Conn, M. Irwin, A. McConnachie, S. Chapman, M. Collins, M. Fardal, A. Ferguson, N. Ibata, D. Mackey, N. Martin, J. Navarro, M. Rich, D. Valls-Gabaud, L. Widrow, A vast, thin plane of corotating dwarf galaxies orbiting the andromeda galaxy. Nature **493**, 62–65 (2013)
70. E.R. Johnson, S. Keinan, P. Mori-Sanchez, A.J. Contreras-Garcia, J. Cohen, W. Yang, Revealing noncovalent interactions. J. Am. Chem. Soc. **132**, 6498–6506 (2010)
71. M. Joswig, M. Pfetsch, Computing optimal Morse matchings. SIAM J. Discret. Math. **20**(1), 11–25 (2006)
72. J. Kasten, J. Reininghaus, I. Hotz, H.C. Hege, Two-dimensional time-dependent vortex regions based on the acceleration magnitude. IEEE Trans. Vis. Comput. Graph. **17**, 2080–2087 (2011)
73. Kitware, Paraview cinema (2014), http://www.kitware.com/blog/home/post/734
74. P. Klacansky, J. Tierny, H. Carr, Z. Geng, Fast and exact fiber surfaces for tetrahedral meshes. IEEE Trans. Vis. Comput. Graph. **23**(7), 1782–1795 (2017)
75. S. Klasky, M. Vouk, M. Parashar, A. Khan, N. Podhorszki, R. Barreto, D. Silver, S. Parker, Collaborative visualization spaces for petascale simulations, in *Proceedings of International Symposium on Collaborative Technologies and Systems* (2008)
76. A. Landge, V. Pascucci, A. Gyulassy, J. Bennett, H. Kolla, J. Chen, T. Bremer, In-situ feature extraction of large scale combustion simulations using segmented merge trees, in *SuperComputing* (2014)
77. J. Lane, J. Contreras-Garcia, J.-P. Piquemal, B. Miller, H. Kjaergaard, Are bond critical points really critical for hydrogen bonding? J. Chem. Theory Comput. **9**(8), 3263–3266 (2013)
78. D.E. Laney, P.T. Bremer, A. Mascarenhas, P. Miller, V. Pascucci, Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. IEEE Trans. Vis. Comput. Graph. **12**, 1053–1060 (2006)
79. G.N. Lewis, The atom and the molecule. Am. Chem. Soc. **38**, 762–785 (1916)
80. J. Lukasczyk, G. Aldrich, M. Steptoe, G. Favelier, C. Gueunet, J. Tierny, R. Maciejewski, B. Hamann, H. Leitte, Viscous fingering: a topological visual analytic approach, in *Conference on Physical Modeling for Virtual Manufacturing Systems and Processes* (2017)
81. S. Maadasamy, H. Doraiswamy, V. Natarajan, A hybrid parallel algorithm for computing and tracking level set topology, in *International Conference on High Performance Computing* (2012)
82. C.F. Matta, R.J. Boyd, *The Quantum Theory of Atoms in Molecules: From Solid State to DNA and Drug Design* (Wiley, Weinheim, 2007)
83. J.C. Michelin, J. Tierny, F. Tupin, C. Mallet, N. Paparoditis, Quality evaluation of 3d city building models with automatic error diagnosis, in *Proceedings of ISPRS Conference on SSG* (2013)
84. J. Milnor, *Morse Theory* (Princeton University Press, Princeton, 1963)
85. M. Mohi, C. Yung-Cheng, Stability characteristics and flame structure of low swirl burner. Exp. Thermal Fluid Sci. **32**(7), 1390–1395 (2008)
86. D. Morozov, G. Weber, Distributed merge trees, in *ACM Symposium on Principles and Practice of Parallel Programming* (2013)

87. M. Morse, *The Calculus of Variations in the Large*. Colloquium Publications—American Mathematical Society, vol. 18 (American Mathematical Society, Providence, 1934)
88. K. Nogenmyr, P. Peterson, X. Bai, A. Nauert, J. Olofsson, C. Brackman, H. Seyfried, Z.-S. Zetterberg, J. Li, M. Richter, A. Dreizler, M. Linne, M. Alden, Large eddy simulation and experiments of stratified lean premixed methane/air turbulent flames. Proc. Combust. Inst. **31**, 1467–1475 (2007)
89. S. Parsa, A deterministic O(m log m) time algorithm for the Reeb graph, in *Proceedings of ACM Symposium on Computational Geometry* (2012)
90. V. Pascucci, K. Cole-McLaughlin, Parallel computation of the topology of level sets. Algorithmica **38**, 249–268 (2003)
91. V. Pascucci, G. Scorzelli, P.T. Bremer, A. Mascarenhas, Robust on-line computation of Reeb graphs: simplicity and speed. ACM Trans. Graph. **26**, 58 (2007)
92. G. Patane, M. Spagnuolo, B. Falcidieno, Reeb graph computation based on a minimal contouring, in *Proceedings of IEEE Shape Modeling International* (2008)
93. L. Pauling, *The Nature of the Chemical Bond and the Structure of Molecules and Crystals: An Introduction to Modern Structural Chemistry* (Cornell University Press, Ithaca, 1960)
94. P. Peterson, J. Olofsson, C. Brackman, H. Seyfried, J. Zetterberg, M. Richter, M. Alden, M. Linne, R. Cheng, A. Nauert, D. Geyer, A. Dreizler, Simultaneous PIV/OH PLIF, Rayleigh thermometry/OH PLIF and stereo PIV measurements in a low-swirl flame. Appl. Opt. **46**, 3928–3936 (2007)
95. S. Philip, B. Summa, J. Tierny, P.T. Bremer, V. Pascucci, Scalable seams for gigapixel panoramas, in *Eurographics Symposium on Parallel Graphics and Visualization* (2013)
96. S. Philip, B. Summa, J. Tierny, P.T. Bremer, V. Pascucci, Distributed seams for gigapixel panoramas. IEEE Trans. Vis. Comput. Graph. **21**, 350–362 (2015)
97. U. Pinkall, K. Polthier, Computing discrete minimal surfaces and their conjugates. Exp. Math. **2**, 15–36 (1993)
98. G. Reeb, Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique. C. R. Acad. Sci. **222**, 847–849 (1946)
99. B. Rieck, H. Leitte, Persistent homology for the evaluation of dimensionality reduction schemes. Comput. Graph. Forum **34**, 431–440 (2015)
100. V. Robins, P. Wood, A. Sheppard, Theory and algorithms for constructing discrete morse complexes from grayscale digital images. IEEE Trans. Pattern Anal. Mach. Intell. **33**, 1646–1658 (2011)
101. E. Santos, J. Tierny, A. Khan, B. Grimm, L.D. Lins, J. Freire, V. Pascucci, C. Silva, S. Klasky, R. Barreto, N. Podhorszki, Enabling advanced visualization tools in a web-based simulation monitoring system, in *Proceedings of IEEE eScience* (2009)
102. Y. Shinagawa, T.L. Kunii, Y.L. Kergosien, Surface coding based on morse theory. IEEE Comput. Graph. Appl. **11**, 66–78 (1991)
103. N. Shivashankar, Parallel implementation of 3d Morse-Smale complex computation (2012), http://vgl.serc.iisc.ernet.in/mscomplex/
104. N. Shivashankar, V. Natarajan, Parallel computation of 3d Morse-Smale complexes. Comput. Graph. Forum **31**, 965–974 (2012)
105. B.S. Sohn, C.L. Bajaj, Time varying contour topology. IEEE Trans. Vis. Comput. Graph. **12**, 14–25 (2006)
106. T. Sousbie, The persistent cosmic web and its filamentary structure: theory and implementations. Royal Astron. Soc. **414**, 350–383 (2011)
107. B. Summa, J. Tierny, V. Pascucci, Panorama weaving: fast and flexible seam processing. ACM Trans. Graph. **31**, 83 (2012)
108. B. Summa, J. Tierny, V. Pascucci, Visualizing the uncertainty of graph-based 2d segmentation with min-path stability. Comput. Graph. Forum **36**, 133–143 (2017)
109. F. Sun, K. Cheng, Q. Du, Modeling and segmentation of nuclei based on Morse-Smale complex. Technical Report, Penn State University (2013)
110. S. Tarasov, M. Vyali, Construction of contour trees in 3d in O(n log n) steps, in *Proceedings of ACM Symposium on Computational Geometry* (1998)

111. J.M. Thiery, B. Buchholz, J. Tierny, T. Boubekeur, Analytic curve skeletons for 3d surface modeling and processing. Comput. Graph. Forum **31**, 2223–2232 (2012)
112. J.M. Thiery, J. Tierny, T. Boubekeur, CageR: cage-based reverse engineering of animated 3d shapes. Comput. Graph. Forum **31**, 2303–2316 (2012)
113. J.M. Thiery, J. Tierny, T. Boubekeur, Jacobians and Hessians of mean value coordinates for closed triangular meshes. Vis. Comput. **2013**, 1–15 (2013)
114. D.M. Thomas, V. Natarajan, Multiscale symmetry detection in scalar fields by clustering contours. IEEE Trans. Vis. Comput. Graph. **20**, 2427–2436 (2014)
115. J. Tierny, Reeb graph based 3D shape modeling and applications. PhD thesis, Lille1 University, 2008
116. J. Tierny, vtkReebGraph class collection (2009), http://www.vtk.org/doc/nightly/html/classvtkReebGraph.html
117. J. Tierny, H. Carr, Jacobi fiber surfaces for bivariate Reeb space computation. IEEE Trans. Vis. Comput. Graph. **23**(1), 960–969 (2017)
118. J. Tierny, V. Pascucci, Generalized topological simplification of scalar fields on surfaces. IEEE Trans. Vis. Comput. Graph. **18**, 2005–2013 (2012)
119. J. Tierny, J.-P. Vandeborre, M. Daoudi, 3D mesh skeleton extraction using topological and geometrical analyses, in *Proceedings of Pacific Graphics* (2006)
120. J. Tierny, J.-P. Vandeborre, M. Daoudi, Invariant high level Reeb graphs of 3D polygonal meshes, in *Proceedings of IEEE 3DPVT* (2006)
121. J. Tierny, J.-P. Vandeborre, M. Daoudi, Reeb chart unfolding based 3D shape signatures, in *Proceedings of Eurographics* (2007)
122. J. Tierny, J.-P. Vandeborre, M. Daoudi, Topology driven 3D mesh hierarchical segmentation, in *Proceedings of IEEE Shape Modeling International* (2007)
123. J. Tierny, J.-P. Vandeborre, M. Daoudi, Enhancing 3D mesh topological skeletons with discrete contour constrictions. Vis. Comput. **24**, 155–172 (2008)
124. J. Tierny, J.-P. Vandeborre, M. Daoudi, Fast and precise kinematic skeleton extraction of 3D dynamic meshes, in *Proceedings of IEEE ICPR* (2008)
125. J. Tierny, A. Gyulassy, E. Simon, V. Pascucci, Loop surgery for volumetric meshes: Reeb graphs reduced to contour trees. IEEE Trans. Vis. Comput. Graph. **15**, 1177–1184 (2009)
126. J. Tierny, J.-P. Vandeborre, M. Daoudi, Partial 3D shape retrieval by Reeb pattern unfolding. Comput. Graph. Forum **28**, 41–55 (2009)
127. J. Tierny, J. Daniels, L.G. Nonato, V. Pascucci, C. Silva, Inspired quadrangulation. Comput. Aided Des. **43**, 1516–1526 (2011)
128. J. Tierny, J. Daniels, L.G. Nonato, V. Pascucci, C. Silva, Interactive quadrangulation with Reeb atlases and connectivity textures. IEEE Trans. Vis. Comput. Graph. **18**, 1650–1663 (2012)
129. J. Tierny, D. Guenther, V. Pascucci, Optimal general simplification of scalar fields on surfaces, in *Topological and Statistical Methods for Complex Data* (Springer, Berlin, 2014)
130. J. Tierny, G. Favelier, J. Levine, C. Gueunet, M. Michaux, The topology ToolKit. IEEE Trans. Vis. Comput. Graph. **24**(1), 832–842 (2018), https://topology-tool-kit.github.io/
131. M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pasucci, D.R. Schikore, Contour trees and small seed sets for isosurface traversal, in *Proceedings of ACM Symposium on Computational Geometry* (1997)
132. A. Vintescu, F. Dupont, G. Lavoue, P. Memari, J. Tierny, Conformal factor persistence for fast hierarchical cone extraction, in *Eurographics* (2017)
133. A. Vintescu, F. Dupont, G. Lavoue, P. Memari, J. Tierny, Least squares affine transitions for global parameterization, in *WSCG* (2017)
134. C. Wall, *Surgery on Compact Manifolds* (American Mathematical Society, Oxford, 1970)
135. G. Weber, S.E. Dillard, H. Carr, V. Pascucci, B. Hamann, Topology-controlled volume rendering. IEEE Trans. Vis. Comput. Graph. **13**, 330–341 (2007)

136. K. Weiss, F. Iuricich, R. Fellegara, L. De Floriani, A primal/dual representation for discrete Morse complexes on tetrahedral meshes. Comput. Graph. Forum **32**, 361–370 (2013)
137. S. Wienert, D. Heim, K. Saeger, A. Stenzinger, M. Beil, P. Hufnagl, M. Dietel, C. Denkert, F. Klauschen, Detection and segmentation of cell nuclei in virtual microscopy images: a minimum-model approach. Sci. Rep. **2**, 503 (2012)
138. K. Xu, H. Zhang, D. Cohen-Or, Y. Xiong, Dynamic harmonic fields for surface processing. Comput. Graph. **33**, 391–398 (2009)

Index

J. Tierny, *Topological Data Analysis for Scientific Visualization*,
Mathematics and Visualization, https://doi.org/10.1007/978-3-319-71507-0