

# Visual Analysis of High-Dimensional Point Clouds using Topological Abstraction

Von der Fakultät für Mathematik und Informatik  
der Universität Leipzig  
angenommene

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

DOCTOR RERUM NATURALIUM

(Dr. rer. nat.)

im Fachgebiet

INFORMATIK

Vorgelegt

von Diplom-Informatiker Patrick Oesterling

geboren am 24. November 1982 in Halle (Saale)

Die Annahme der Dissertation wurde empfohlen von:

1. Professor Dr. Geric Scheuermann, Universität Leipzig
2. Professor Dr. Thomas Wischgoll, Wright State University

Die Verleihung des akademischen Grades erfolgt mit Bestehen  
der Verteidigung am 14. April 2016 mit dem Gesamtpredikat *magna cum laude*





# Selbstständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbstständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialien oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, den 14. April 2016

.....

(Ort, Datum)

.....

(Unterschrift)



# Danksagung

Mein größter Dank gilt Gerik Scheuermann, der mir als Doktorvater ein überaus spannendes und ertragreiches Thema für die Forschungsarbeit an dieser Dissertation anvertraut hat. Ich danke Gerik für die vielen fruchtbaren Diskussionen, die lehrreichen Lektionen und Anekdoten und die kollegiale und persönliche Atmosphäre am Lehrstuhl und auf Dienstreisen.

Ich danke Karin und allen ehemaligen Kollegen, dass sie das Lehrstuhlleben im Alltag und bei Lehrstuhlabenden entspannt gehalten haben und dass jeder für jeden da war und ein offenes Ohr für wissenschaftliche Diskussionen und private Gespräche hatte. Ein besonderer Dank geht an die Entwickler von OpenWalnut für ihre hervorragende Arbeit, welche es mir ermöglicht hat, meine Forschungsergebnisse effektiv zu implementieren. Außerdem danke ich Alexander, Christian, Dominic, Gerik, Heike und Mario, dass sie mir 2009 die Entscheidung zu promovieren leicht gemacht haben.

Gunther danke ich für unsere jahrelange Zusammenarbeit und die tolle Möglichkeit, in Berkeley wertvolle Erfahrungen gesammelt zu haben. Ich danke Herrn Professor Heyer für die hervorragende Zusammenarbeit am gemeinsamen DFG-Projekt und ich danke Gunther und Christian dafür, dass sie sich intensiv über Jahre am Projekt beteiligt und sehr zum Erfolg unserer gemeinsamen Publikationen und dem Gelingen dieser Dissertation beigetragen haben.

Besonderer Dank gilt—in zeitlich sortierter Reihenfolge—meinen fünf persönlichen Mentoren Markus Jäger, Herrn Dr. Joachimi, Sebastian Eichelbaum, Gerik Scheuermann und Christian Heine, die mich zu verschiedenen Zeiten in meinem Werdegang in die richtige Richtung gelenkt haben und von denen ich menschlich, methodisch und fachlich unglaublich viel lernen durfte.

Danken möchte ich vor allem auch meinen Eltern, dass sie mich stets gefördert und unterstützt haben und mir immer mit Rat und wenn nötig auch Trost zur Seite standen. All meinen Freunden und insbesondere meiner Kathrin danke ich für die tatkräftige moralische Unterstützung, das Motivieren in schwierigen Zeiten und das Ertragen unzähliger Monologe meinerseits. Danke!



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Remarks . . . . .	4
<b>2</b>	<b>Thematic Classification, Difficulties and Solution Approach</b>	<b>7</b>
2.1	Cluster Analysis: A Brief Introduction . . . . .	8
2.1.1	Problems with Clustering High-Dimensional Data . . . . .	13
2.2	Visualization of High-Dimensional Data . . . . .	14
2.2.1	Projective Visualization Techniques . . . . .	15
2.2.2	Axis-based Visualization Techniques . . . . .	18
2.2.3	Problems and Difficulties . . . . .	19
2.3	Novel Topology-Based Solution Approach . . . . .	22
<b>I</b>	<b>Visual Analysis of Time-Invariant Clusterings</b>	<b>25</b>
<b>3</b>	<b>Topological Representation</b>	<b>29</b>
3.1	Related Work . . . . .	29
3.1.1	Neighborhood Description . . . . .	29
3.1.2	Scalar Field Topology . . . . .	31
3.2	Basic Algorithm . . . . .	34
3.2.1	Approximation of the Density Function . . . . .	35
3.2.2	Topology of the Density Function . . . . .	37
3.2.3	Merge Tree Simplification . . . . .	39
3.2.4	Topology-Based Region Properties . . . . .	40
3.3	Optimizations . . . . .	41
3.3.1	Approximation of the Delaunay Triangulation . . . . .	42
3.3.2	Faster Neighborhood Graph Construction . . . . .	45
3.3.3	Sampling and Reinsertion . . . . .	46
3.4	Parameters and Runtime . . . . .	49
3.4.1	Runtime Complexity . . . . .	52

3.5	Examples and Results . . . . .	53
3.5.1	Artificial 2-D Data Set . . . . .	53
3.5.2	High-Dimensional Data Sets . . . . .	60
3.6	Conclusion and Discussion . . . . .	63
<b>4</b>	<b>Topological Visualization</b>	<b>67</b>
4.1	Related Work . . . . .	68
4.2	Extended 3-D Topological Landscape . . . . .	72
4.2.1	Modified Metric-based Distortion . . . . .	72
4.2.2	Data Point Representation . . . . .	74
4.2.3	Labeling . . . . .	76
4.3	2-D Topological Atoll . . . . .	78
4.4	2-D Topological Landscape Profile . . . . .	79
4.4.1	Properties of the Landscape Profile . . . . .	81
4.4.2	Construction and Implementation . . . . .	84
4.5	Examples and Results . . . . .	86
4.5.1	Global Overview . . . . .	86
4.5.2	Unclassified Data . . . . .	97
4.5.3	Application 1: Visualization of Document Collections . . . . .	99
4.5.4	Application 2: Classification of Unclassified Data . . . . .	104
4.6	Conclusion and Discussion . . . . .	106
<b>5</b>	<b>Interactive Visual Analysis</b>	<b>109</b>
5.1	Related Work . . . . .	110
5.2	Global Overview and Parameter Widgets . . . . .	110
5.2.1	Filter Radius of the Density Function . . . . .	111
5.2.2	Simplification of Structural Noise . . . . .	115
5.3	Feature Selection and Local Analysis . . . . .	118
5.3.1	Features in the Landscape . . . . .	119
5.3.2	Selection and Linked Views . . . . .	120
5.4	Visual Analysis Framework . . . . .	123
5.4.1	Prototype Implementation . . . . .	124
5.4.2	Typical Workflow of the Analysis Process . . . . .	126
5.5	Examples and Results . . . . .	129
5.5.1	Data Analysis . . . . .	129
5.5.2	Unclassified Data . . . . .	137
5.6	Conclusion and Discussion . . . . .	138

<b>II</b>	<b>Visual Analysis of Time-Varying Clusterings</b>	<b>141</b>
<b>6</b>	<b>Topological Representation</b>	<b>145</b>
6.1	Related Work . . . . .	146
6.2	Algorithm Overview . . . . .	148
6.3	Merge Tree Transformation . . . . .	149
6.3.1	Algorithm . . . . .	149
6.3.2	Implementation . . . . .	152
6.3.3	Runtime Complexity . . . . .	153
6.4	Feature Tracking . . . . .	155
6.4.1	Tracking the Complete Function . . . . .	156
6.4.2	Tracking Superlevel Sets of the Function . . . . .	158
6.5	Simplification . . . . .	160
6.6	Conclusion and Discussion . . . . .	162
<b>7</b>	<b>Topological Visualization</b>	<b>165</b>
7.1	Related Work . . . . .	165
7.2	Visualization Design . . . . .	166
7.2.1	Merge Trees as Landscape Profiles . . . . .	167
7.2.2	Tracking Information of the Complete Function . . . . .	168
7.2.3	Tracking Graphs of Superlevel Sets . . . . .	173
7.3	Conclusion and Discussion . . . . .	176
<b>8</b>	<b>Application: Temporal Clustering</b>	<b>179</b>
8.1	Related Work . . . . .	180
8.1.1	Creating the Time-Varying Density Function . . . . .	181
8.1.2	Analysis of Document Collections . . . . .	182
8.2	Conclusion and Discussion . . . . .	190
<b>9</b>	<b>Thesis Conclusion</b>	<b>193</b>
<b>A</b>	<b>Data Sets</b>	<b>197</b>
A.1	Artificial 2-D Data Set . . . . .	198
A.2	Artificial 100-D Data Set . . . . .	198
A.3	19-D Image Segmentation Data Set . . . . .	200
A.4	25-D Isolet Data Set . . . . .	201
A.5	8-D Italian Olive Oils Data Set . . . . .	202
A.6	4-D Iris Plants Data Set . . . . .	203
A.7	9-D Reuters Data Set . . . . .	204

A.8 4-D Medline Data Set . . . . .	205
------------------------------------	-----



# Chapter 1

## Introduction

The abilities of computers and the human brain to solve complex problems complement each other. Computers are efficient in performing trivial operations on massive amounts of stored data, but they cannot imagine or question what they calculate and they cannot interpret the results. Humans, on the other hand, lack specialized storage and processing skills, but they are able to formulate their problems theoretically and to find suitable representations. Hence, synergy arises when humans feed computers with meaningful data, let them perform meaningful operations automatically and evaluate and interpret the results afterwards. The crux is that computers work only with numbers, while the human eye is naturally trained to distinguish different patterns, sizes, colors, or shapes. Therefore, to leverage stored data and computed results, one has to make knowledge hidden behind the numbers visible to the eye in order to form a mental image of the data. This is literally<sup>1</sup> the definition and the task of *visualization* in computational environments.

This thesis is about visualizing a kind of data that is trivial to process by computers but difficult to imagine by humans because nature does not allow for experience and intuition with this type of information: high-dimensional data. Such data often result from representing observations of objects under various aspects or as single entities with different properties. In many applications, a typical, laborious task for object-based data is to find related objects or to group those that are more similar to each other than to other objects. One classic solution for this task is to imagine the data as vectors in a Euclidean space with object variables as dimensions; a so-called *information space*. Utilizing Euclidean distance as a measure of similarity, in this vector representation, objects with similar properties and values accumulate to groups, so-called clusters, that are exposed by cluster analysis on the

---

<sup>1</sup>Oxford Dictionaries: [verb] visu|al|ize - <sub>1</sub> Form a mental image of; imagine; <sub>2</sub> Make (something) visible to the eye.

high-dimensional point cloud. Because similar vectors can be thought of as objects that are alike in terms of their attributes, the point cloud’s clustering structure and individual cluster properties, like their size or compactness, summarize data categories and their relative importance, respectively. Furthermore, (dis-)similarities among the objects can be derived from their points’ affiliation to particular clusters and the overall clustering quality can be evaluated from cluster separation or the *noise-ratio*—the amount of those points that do not belong to clusters but instead reside between and around them.

Traditional cluster analysis usually means to execute an algorithm that finds a meaningful hierarchy or segmentation of the input data, followed by the inspection of its output, which, depending on the internal representation, typically ranges from pure numbers or tabular summaries to tree-like layouts. Because those summaries primarily focus on cluster segmentation rather than on the depiction of individual cluster properties, a popular alternative is the *attempt* to look directly at the point cloud, exploiting that the human eye is naturally trained in detecting patterns and coherent groups efficiently. That is, striving to preserve all pairwise distances in the high-dimensional space, the input points are projected into the plane so that the analyst can quickly identify clustering structure and individual cluster properties. However, there is a fundamental problem with such a direct visualization. If the point cloud’s intrinsic dimensionality is higher than two, information usually has to be discarded to find a two-dimensional embedding that can be mapped on the eye’s retina. This information loss, called the projection error, can cause occlusions in the visualization and can even suggest structure that is not present. Moreover, because this approach does not distinguish between clusters and noise, for large data sets, the depiction of cluster separation can be distorted or hidden entirely. Of course, this makes it difficult (or even impossible) to identify and compare clusters and leads to false insights about the data.

The contribution of this thesis is a novel visual analysis approach that facilitates exploration of high-dimensional point clouds without suffering from *structural occlusion*. The presented work is based on implementing two key concepts: The first idea is to discard those geometric properties that cannot be preserved and, thus, lead to the typical artifacts. Topological concepts are used instead to shift away the focus from a point-centered view on the data to a more structure-centered perspective. The advantage is that topology-driven clustering information can be extracted in the data’s original domain and be preserved without loss in low dimensions. To this end, the high-dimensional point cloud is abstracted by a simpler representation whose topological description accurately describes the clustering, but which can

still be visualized occlusion-free using a landscape metaphor that shows clusters and their hierarchy as differently shaped hills. Topology-based quality measures describe cluster properties and are mapped on the hills in the landscape so that the analyst can quickly identify and compare significant features. Furthermore, to facilitate annotation and to stimulate further analysis of individual features, the data points of the underlying point cloud are augmented on their corresponding hills. The second key idea is to split the interactive visual analysis process into two phases: A topology-based global analysis and a subsequent geometric local analysis phase. The occlusion-free global overview enables the analyst to identify all features and link selected clusters or arbitrary point sets to other visualization techniques that permit analysis of those properties that are not captured by the topological abstraction, e.g. cluster shape or value distributions in particular dimensions or subspaces (cf. Chapter 2). The advantages of separating structure from data point analysis are two-fold. While the structural view on the high-dimensional clustering allows for an occlusion-free presentation in the first place, restricting local analysis only to subsets of the data also significantly reduces artifacts and visual complexity in traditional visualizations that focus on the data points themselves. That is, compared to visualizing the complete data with direct visualizations, the additional topological layer enables the analyst to identify structure that was hidden before and to focus on particular features by suppressing irrelevant points during local feature analysis.

This thesis addresses the topology-based visual analysis of high-dimensional point clouds for both the time-invariant and the time-varying case. Time-invariant means that the points are *static* (or stationary) in the sense that they do not change in their number or positions. That is, the analyst explores the clustering of a fixed and constant set of points. The extension to the time-varying case implies the analysis of a varying clustering, i.e., high-dimensional points that do change in their number and positions and, thus, cause clusters to appear as new, to merge or split, or to vanish. Such temporal cluster analysis is important for many application domains where analysts study changing categories and objects. Especially for high-dimensional data, both *tracking*—which means to relate features over time—but also visualizing changing structure are difficult problems to solve.

The remainder of this thesis is structured as follows: Chapter 2 provides a brief introduction to cluster analysis, common applications and alternative visualization techniques for high-dimensional point data. It also reveals where competing visualizations have issues with this data type and it motivates how topological ideas can solve these problems. Following that chapter, this thesis' subject matter is presented in two major parts; one for the visual analysis of static point clouds and the other part

for its extension to time-varying data. Part I contains detailed explanations about the topological representation, topology-based visualization, and the development and implementation of an interactive visual analysis framework. Part II explains how these ideas and data structures are extended to abstract time-varying input data and how the topological description of changing clusterings can be visualized at high structural detail but without structural occlusion. In both major parts, the utility and efficiency of the approach are demonstrated based on several example data sets; which are described in further detail in Appendix A.

## 1.1 Remarks

The research results presented in this doctoral thesis build on, improve and develop further preliminary work contained in the author’s diploma thesis. With regard to intersecting content, both theses primarily share basic parts of the topological representation and a prototype implementation of the 3-D topological landscape metaphor as proposed by Weber et al. [169]. As for the topological abstraction, the approach investigated in the diploma thesis—namely using an upsampled Gabriel graph (cf. Chapter 3 in this doctoral thesis)—is now generalized in this doctoral thesis as one possible solution among other neighborhood graphs. The initial approach was not yet optimized to scale for larger data sets, did not support as many feature properties, and did not assist the analyst in finding appropriate parameter values by using intuitive widgets. As for the visualization, this doctoral thesis first describes modifications of the 3-D landscape metaphor to reduce its dimensionality and to provide more information about features and data points. Eventually, the original visualization is completely substituted by a more precise and faster to construct 2-D landscape variation (cf. Chapter 4). The extension to an interactive analysis framework supporting feature selection and linking as well as the whole extension to time-varying data are exclusive contributions of this doctoral thesis.

The work on this thesis was carried out within the scope of a Priority Programme (*“Schwerpunktprogramm (SPP)”*) about “Scalable Visual Analytics: Interactive Visual Analysis Systems of Complex Information Spaces” funded by the German Research Foundation (*“Deutsche Forschungsgemeinschaft (DFG)”*). In particular, this work was conducted in connection with a participating project about topology-based visualization of document data represented in the vector space model (cf. Chapter 2); led by Professor Dr. Gerik Scheuermann and Professor Dr. Gerhard Heyer. For this reason, the development of the topological methods and the visualization were inspired and guided by this target application. However, the solutions are not restricted to

this particular application. In fact, the obtained results make valuable contributions to several research fields, like visualization and analysis of high-dimensional point data, high-dimensional scalar field topology, topology-based visualization, temporal clustering of high-dimensional data, and time-varying scalar field topology in high dimensions. In principle, the solutions are applicable in those scenarios where the analyst wants to learn the structure of high-dimensional point data or high-dimensional scalar fields. To cope with this versatility, the example data used in this thesis cover a variety of different application domains.

Although this thesis is the work of one author, the presented research results originate from intensive collaboration with several colleagues over a couple of years. To appreciate this and to be in line with the familiar publication style of the target audience, the pronoun “we” will be used.

### List of Publications

This thesis recapitulates and unifies conducted research. As typical for PhD students in computer science, the majority of the results has already been published in various journals, books, and conference proceedings. These are the relevant peer-reviewed publications for this thesis (sorted by their date of appearance):

- [P1] P. Oesterling, C. Heine, H. Jänicke, and G. Scheuermann. **Visual Analysis of High-Dimensional Point Clouds using Topological Landscapes.** *IEEE Pacific Visualization Symposium (PacificVis 2010)*. Ed. by S. North, H.-W. Shen, and J. van Wijk. 2010, pp. 113–120
- [P2] P. Oesterling, G. Scheuermann, S. Teresniak, G. Heyer, S. Koch, T. Ertl, and G. H. Weber. **Two-stage Framework for a Topology-Based Projection and Visualization of Classified Document Collections.** *IEEE Symposium on Visual Analytics Science and Technology (VAST)*. IEEE Computer Society, 2010, pp. 91–98
- [P3] P. Oesterling, C. Heine, H. Jänicke, G. Scheuermann, and G. Heyer. **Visualization of High-Dimensional Point Clouds Using Their Density Distribution’s Topology.** *IEEE Transactions on Visualization and Computer Graphics* 17.11 (2011), pp. 1547–1559. ISSN: 1077-2626
- [P4] P. Oesterling, C. Heine, G. H. Weber, and G. Scheuermann. **Visualizing nD Point Clouds as Topological Landscape Profiles to Guide Local Data Analysis.** *IEEE Transactions on Visualization and Computer Graphics* 19.3 (2013), pp. 514–526. ISSN: 1077-2626

- [P5] P. Oesterling, C. Heine, G. H. Weber, and G. Scheuermann. **A Topology-Based Approach to Visualize the Thematic Composition of Document Collections.** *Text Mining: From Ontology Learning to Automated Text Processing Applications*. Ed. by C. Biemann and A. Mehler. Theory and Applications of Natural Language Processing. Springer International Publishing, 2014, pp. 63–85. ISBN: 978-3-319-12654-8
- [P6] P. Oesterling, P. Jähnichen, G. Heyer, and G. Scheuermann. **Topological Visual Analysis of Clusterings in High-Dimensional Information Spaces.** *it - Information Technology* 57.1 (2015). Special Issue: Visual Analytics, pp. 3–10. ISSN: 1611-2776
- [P7] P. Oesterling, C. Heine, G. H. Weber, D. Morozov, and G. Scheuermann. **Computing and Visualizing Time-Varying Merge Trees for High-Dimensional Data.** *Topology-Based Methods in Visualization (TopoInVis)*. (to appear, received “best paper award”). Springer, 2015

The author also contributed to the following publications:

- [P8] G. H. Weber, D. Morozov, K. Beketayev, J. Bell, P.-T. Bremer, M. Day, B. Hamann, C. Heine, M. Haranczyk, M. Hlawitschka, V. Pascucci, P. Oesterling, and G. Scheuermann. **Topology-Based Visualization and Analysis of High-Dimensional Data and Time-varying Data at the Extreme Scale.** *DOE Exascale Research Conference*. LBNL-5691E-Poster. Portland, OR, 2012
- [P9] T. Liebmann, P. Oesterling, S. Jänicke, and G. Scheuermann. **A Geological Metaphor for Geospatial-temporal Data Analysis.** *IVAPP '14: Proceedings of the 5th International Conference on Information Visualization Theory and Application*. SciTePress, 2014
- [P10] P. Jähnichen, P. Oesterling, T. Liebmann, G. Heyer, C. Kuras, and G. Scheuermann. **Exploratory Search Through Interactive Visualization of Topic Models.** *Digital Humanities 2015 (to appear)*. 2015

## Chapter 2

# Thematic Classification, Difficulties and Solution Approach

Given a notion of similarity, finding related objects or grouping similar ones is a typical task in various fields of application. For objects with multiple attributes, finding groups is typically desired at a global scale to find those objects that are alike throughout all variables. In this case, occurring groups describe a categorial segmentation of the data and judging group cohesion, hierarchy, and separability reveals each category's significance and facilitates comparison. One possible solution for this task is to represent objects as high-dimensional vectors in a space with object properties as dimensions, henceforth called an *information space*, and to use the Euclidean distance between two points to specify the similarity between their corresponding objects. Note that, strictly speaking, distance is a measure of dissimilarity because the similarity between two points' objects is inverse to their distance in the information space. However, we refer to the Euclidean distance as a similarity measure because this term is commonly used in the literature. Objects with similar properties and values then accumulate to groups, so-called *clusters*, that are exposed by cluster analysis on the resulting high-dimensional point cloud. A *clustering* essentially is the set of all clusters and describes structure at a global scale. This includes information about the number of clusters, their hierarchy if they are embedded in each other, their separation, or the occurrence of noise and outliers—which are basically those points that do not belong to any cluster and occur separately or only in small groups. The clustering specifies the overall quality of the categorial segmentation and enables the analyst, e.g., to find related objects based on their group affiliation. Clusters themselves specify structural information at a local scale. This typically includes a cluster's number of points and their distribution, a

cluster’s spread or compactness, or its shape. Based on these properties, clusters can be identified and compared using various notions of significance.

Performing cluster analysis on a vector-based representation of domain entities has become a widely-used tool to solve problems in many fields of application. In general, this approach is useful in those scenarios where knowledge can be derived (either directly or indirectly) from the grouping behavior of multiply-attributed objects in their corresponding information space. For example, if text data, image data, or speech sound data are represented as vectors in a space of words, pixel positions, or sonorant features, respectively, the corresponding documents, pictures and vocal tracts cluster if they are about the same topic, scenery, or if they rhyme. Representative for various models to transform real-world data into vector format, Figure 2.1 explains a classic approach to represent documents as high-dimensional points. Another example is the analysis of a data set describing the composition of olive oils with a feature vector consisting of percentages of fatty acids (cf. Appendix A.5). In this example, one is interested if these oils form clusters based on their combination of fatty acids, and whether these clusters correspond, e.g., to geographic growing regions. There are many other popular applications of high-dimensional clustering like analyzing gene expression data [174], comparing cars based on their technical specifications, or studying breast cancer data [132] or wine quality [35, 151]. To accommodate this diversity, the example data used in this thesis also consist of a rich set of real-world objects from various application domains (cf. Appendix A). More data and application examples can be found in the *UCI Machine Learning Repository* [7].

## 2.1 Cluster Analysis: A Brief Introduction

In short, *clustering* or *cluster analysis*, sometimes also referred to as *unsupervised learning*, aims at finding “natural”, “useful”, or “meaningful” grouping of entities, given unlabeled data and a measure of similarity. It has long been used as a tool in a wide variety of applications, including biology, marketing, astronomy, psychology, pattern recognition, genomics, earth-quake studies, and data mining. As already mentioned earlier, we consider a *clustering* (as a noun) as the set of all occurring groups together with noise and outliers, i.e., a segmentation of all the objects in the data. There is no universally agreed upon definition of a cluster [53], but mostly, clusters are described by considering the internal homogeneity and the external separation. Depending on the data distribution and the underlying application, clusters typically represent classes or categories, and a good clustering produces



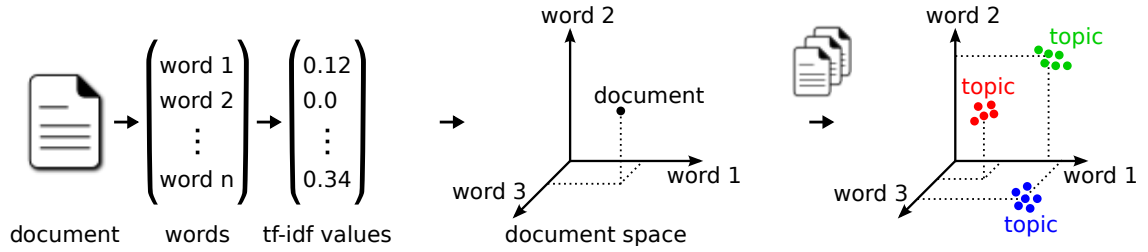


Figure 2.1: Turning text data into point data: To analyze the thematic composition of a text collection, a popular approach transforms documents into high-dimensional vectors using the vector space model [143, 144]. After filtering unimportant words and ignoring grammar and word order, each occurring word is assigned to one dimension of the *document space*. The entries of a particular vector could be the word frequency in this document, or a more sophisticated weighting like, e.g., the *term frequency–inverse document frequency* (tf-idf) [149]. Accumulations in the resulting point cloud represent documents that share vocabulary and word significances—i.e., documents that share topics. The number of clusters reflects topic count, subclusters and their nesting describe sub-topics, cluster size represents the number of documents about this topic, and cluster compactness and separateness indicate a topic’s generality and preciseness, respectively.

clusters with high intra-cluster similarity and low inter-cluster similarity. However, the quality of the clustering mainly depends on the chosen clustering implementation and on the similarity measure used. Typical challenges for cluster analysis are the discovery of clusters with arbitrary shape, the ability to deal with noise and outliers, and scalability with respect to data size and dimensionality. Furthermore, cluster algorithms should make only few restrictions to data attributes, the processing order, and the domain knowledge required to determine parameters.

There are many references for clustering techniques [5, 42, 43, 53, 71, 84, 95, 156] and important survey papers in the literature [175, 9, 17, 55, 85]. Here, we only summarize the key concepts and most important techniques gathered together from the cited literature.

**Distance and Similarity Measures.** For meaningful clustering results, it is vital to have a precise definition of closeness and how to measure the distance (dissimilarity) or similarity between data objects. Since data objects are described by multiple features, it is reasonable to represent them as a multidimensional vector; with individual features that could be quantitative or qualitative, continuous or discrete, or nominal/binary or ordinal.

A *distance* or *dissimilarity function*  $D$  on a set  $X$  is defined to satisfy (1) symmetry:  $D(x_i, x_j) = D(x_j, x_i)$  and (2) positivity:  $D(x_i, x_j) \geq 0$  for all  $x_i$  and  $x_j$ . If also the conditions (3) triangle inequality:  $D(x_i, x_j) \leq D(x_i, x_k) + D(x_k, x_j)$  for all  $x_i, x_j$ , and  $x_k$  and (4) reflexivity:  $D(x_i, x_j) = 0$  iff  $x_i = x_j$  hold, it is called a metric.

Table 2.1: Some typical similarity and dissimilarity measures.

Minkowski distance	$D_{ij} = \left( \sum_{l=1}^d  x_{il} - x_{jl} ^p \right)^{1/p}$ , called the $L_p$ norm
Euclidean distance	$D_{ij} = \left( \sum_{l=1}^d  x_{il} - x_{jl} ^2 \right)^{1/2}$ , for $L_2$
City-block distance	$D_{ij} = \sum_{l=1}^d  x_{il} - x_{jl} $ , for $L_1$
Mahalanobis distance	$D_{ij} = (x_i - \bar{x}_j)S^{-1}(x_i - \bar{x}_j)^T$ , where $S$ is the covariance matrix
Pearson correlation	$D_{ij} = (1 - r_{ij})/2$ , where $r_{ij} = \frac{\sum_{l=1}^d (x_{il} - \bar{x}_i)(x_{jl} - \bar{x}_j)}{\sqrt{\sum_{l=1}^d (x_{il} - \bar{x}_i)^2 \sum_{l=1}^d (x_{jl} - \bar{x}_j)^2}}$
Cosine similarity	$S_{ij} = \cos\alpha = \frac{x_i^T x_j}{\ x_i\  \ x_j\ }$

A *similarity function* is defined to satisfy (1) symmetry:  $S(x_i, x_j) = S(x_j, x_i)$  and (2) positivity:  $0 \leq S(x_i, x_j) \leq 1$  for all  $x_i$  and  $x_j$ . If it also satisfies the conditions (3)  $S(x_i, x_j)S(x_j, x_k) \leq [S(x_i, x_j) + S(x_j, x_k)]S(x_i, x_k)$  for all  $x_i, x_j$  and  $x_k$  and (4)  $S(x_i, x_j) = 1$  iff  $x_i = x_j$ , it is called a similarity metric.

Distance functions are typically used to measure continuous features, while similarity measures are more important for qualitative variables. Some typical measures for continuous features are summarized in Table 2.1. Their selection is problem and application dependent. There are also similarity measures for binary features (their dissimilarity can be obtained from  $D_{ij} = 1 - S_{ij}$ ), like the Jaccard coefficient or the Sokal and Sneath measure [175]. In this thesis, the similarity measure used for the topological analysis defaults to the Euclidean distance. For document analysis, an instance of high-dimensional cluster analysis, it is often convenient to use the cosine similarity in order to reduce the document space to the (surface of the) unit hypersphere in  $\mathbb{R}^d$ .

**Types of clustering algorithms.** Clustering algorithms are primarily distinguished based on their methodology. They are often categorized into being either: “*partitional*” or “*hierarchical*”, which means they either construct a previously known number of independent partitions or a hierarchical level-of-resolution decomposition of the data objects; “*exclusive*” or “*overlapping*”, which means individual objects

can or cannot belong to multiple groups; “*deterministic*” or “*probabilistic*”, which means a definite and percental cluster affiliation; or “*agglomerative*” (or *bottom-up*) or “*divisive*” (or *top-down*), which means the cluster finding starts with single-point clusters or the complete data set.

**Clustering methods.** Many clustering methods were introduced for each available clustering type. Making no claim to be complete in any sense, we only mention a representative selection of the most important and established techniques and refer to the literature cited above for further reading and in-depth explanations.

*Partitioning* (or *flat*) algorithms construct a partition of a database  $D$  of  $n$  objects into a set of  $k$  clusters. Given a number of desired groups  $k$ , the goal is to find  $k$  clusters that optimize a partitioning criterion. Well-known partitioning algorithms are  $k$ -means [116],  $k$ -medoids [95], CLARA [95] or CLARANS [123].  $K$ -means is the best known and its criterion function is based on the sum of squared errors—one of the most widely used criteria. The algorithm is simple: partition the data into  $k$  non-empty subsets; compute seed points as the centroids (mean point) of the clusters of the current partition; assign each data point to the cluster with the nearest seed point; repeat from step two and stop when no more (or only slight) new assignments. This approach has a run time complexity of  $O(Nkd)$  and works well for compact and hyperspherical clusters. Drawbacks, on the other hand, are that there is no universal method for identifying the initial partitions and the number of clusters  $k$ , and that the algorithm is not guaranteed to converge to a global minimum. Moreover, the clustering result varies depending on the initial choice of seeds, all objects are forced into a cluster, and the algorithm is also sensitive to outliers. To overcome the most crucial problems, many variants of  $k$ -means were introduced [175].

*Hierarchical* algorithms create a hierarchical decomposition of the data objects and yield a successive level of clusters by iterative fusions or divisions. This approach does not require the number of clusters  $k$  as an input, but needs a termination condition. The algorithms are mainly classified as *agglomerative* and *divisive* methods. Agglomerative starts with  $n$  single-instance clusters and successively joins the two closest groups until all objects belong to the same group. Divisive clustering proceeds in the opposite way, i.e., it starts with one universal cluster and splits groups recursively. The results of hierarchical clustering are typically depicted by a binary tree or a *dendrogram*—whose root node represents the whole data set and whose leaf nodes represent the data objects. The height of the dendrogram usually expresses the distance between each pair of objects (points or clusters). Thus, it shows several levels of nested partitioning and a clustering can be obtained by cutting the dendrogram at different levels; then each subtree represents a cluster. Based on the different

definitions of the distance between two clusters, there are many agglomerative and divisive algorithms. The most popular methods include *single-linkage* to define cluster distance by the two closest objects in different clusters, *complete-linkage* to define inter-cluster distance by the farthest distance of objects, and *average-linkage* based on the clusters' means. Classic hierarchical clustering lacks robustness and is sensitive to noise and outliers. Furthermore, it cannot correct previous misclassification since each point is handled only once during the process. The computational cost is at least  $O(n^2)$ , which is problematic for very large data sets. Improved variants, like BIRCH [177], of the classic approach have been presented to deal with larger data sets and noise and outliers by using specifically tuned data structures.

*Density-based* algorithms are based on connectivity and density functions. They require that the density in a neighborhood of a data point should be high enough if it belongs to a cluster. The most prominent density-based implementations are DBSCAN [52], OPTICS [6], DENCLUE [77], and CLIQUE [1]. DBSCAN creates clusters based on two user-specified thresholds: *eps*, the maximum radius of the neighborhood, and *minPts*, the minimum number of points in the *eps*-neighborhood of a particular point. Using an R\*-tree data structure for more efficient queries, and building on *eps* and *minPts* to define “core objects”, “density connectedness”, and “density reachability”, DBSCAN finally considers a cluster to be “a maximal set of density-connected points” and can distinguish “core points” from “border points” and “noise points”. OPTICS generalizes DBSCAN for multiple values of *eps* to find clusters of varying density. It produces a (linear) order of the points that, if plotted on the *x*-axis together with a special distance on the *y*-axis, produce a 2-D plot that shows clusters as valleys (cf. Figure 5.12c on page 133). OPTICS does not produce a strict partitioning, but the 2-D plot, from which a hierarchical partitioning can be extracted based on detecting “steep” areas. It has a stronger focus on finding appropriate parameters and concentrates less on the visualization of the clustering. DENCLUE seeks clusters with local maxima of the overall density function and also requires a user-specified threshold to specify the window width of the filter kernel used to construct the density function. The overall density function can be calculated as the sum of the influence function of all data points. Visualizing the result of density-based clustering is a challenge particularly for high-dimensional density functions. The major advantage of density-based clustering methods is their capability to find clusters of arbitrary shape, to handle noise and outliers, and to scan the data only once. A major drawback, however, is their dependence on the adjusted neighborhood size. While a too large neighborhood combines clusters in

the data, a too small size splits clusters and can even assign each data point to its own cluster.

*Grid-based* algorithms use a multi-resolution grid data structure. They basically assign data points to the appropriate grid cell, compute the density of each cell, eliminate cells whose density is below a certain threshold, and compose clusters from contiguous groups of dense cells. The complexity of this approach typically depends on the number of grid cell instead than on the number of objects. A strength of grid-based techniques is that there are no distance calculations involved and that it is easy to determine neighbored clusters. However, cluster shapes are limited to the union of grid-cells and thus the accuracy of the clustering may be degraded at the expense of simplicity of the method. Prominent methods are STING [165], CLIQUE [1], or WaveCluster [150]. WaveCluster employs wavelet transformations for a representation of the data objects and the key idea is that clusters can be easily distinguished in the transformed space.

In this thesis, the terms *cluster*, *clustering*, and *feature* are used a lot. They are typically interchangeable to reflect that clusters are the general subject of interest. This does not only include cluster properties to describe an individual feature found in the data but also the segmentation of the data into clusters. That is, we also consider cluster hierarchy, cluster separation, the number of clusters, and the presence of noise or suspicious accumulations as features of the data that are revealed by cluster analysis. Finding and furthermore visualizing these *features* to the analyst are the main subject of the presented topology-based visual cluster analysis approach.

### 2.1.1 Problems with Clustering High-Dimensional Data

Most techniques listed above lack capabilities to deal with high-dimensional data, i.e. their performance decreases with increasing data dimensionality. To make clustering applicable to large-scale data, parallel algorithms can more efficiently use computational resources and incremental techniques do not require the storage of the entire data. These approaches have challenges and difficulties [175] on their own and are not covered by this introduction. Instead, we address fundamental problems accompanying high-dimensional spaces. Because it is difficult to imagine high-dimensional spaces, it is already complicated to realize how features and algorithms exactly behave in the original space. Furthermore, due to the exponential growth of possible values with each dimension, it becomes intractable to enumerate all subspaces. This becomes problematic because for a large number of attributes, some will usually not be meaningful for a given cluster; while others might correlate. This

is known as the *local features relevance* and refers to the problem that different clusters might be found in different subspaces. While these issues are related to the semantics of the working methodology of clustering techniques for high-dimensional data, there is a more subtle, yet fundamental problem in high-dimensional spaces.

### Distance-Based Similarity Measures

More than fifty years ago, Richard Bellman first spoke about “*a malediction that has plagued the scientist from the earliest days*” [12]. While his statement, basically, refers to the problems caused by increasing the number of independent variables in different fields of application, especially for metric spaces, where the problem often is termed the *curse of dimensionality*, this means an exponential increase of volume and data sparsity with each additional dimension. Particularly for distance-based approaches, it has been shown [157, 14, 76] that depending on the chosen metric, distances between points either depend on the dimensionality ( $L_1$  norm), or approach a constant ( $L_2$  norm) or zero ( $L_{d \geq 3}$  norm). That is, for  $L_2$ , the ratio between the distances to a point’s farthest and closest point approaches one. As a consequence, distance variation vanishes in higher dimensions and some distance-based relationships such as *nearest neighbors* become fragile in those spaces [76, 14]. Of course, if distances become uniform, every distance-based approach is affected by this phenomenon. To illustrate the effects for proximity-based problems like similarity and clustering search, we consider the Medline data set (cf. Appendix A.8). It consists of 1 250 vectors in a 22 095-dimensional space, divided into five equally sized clusters. The black graph shown in Figure 2.2a indicates the distance distribution between any two points. It reveals that even though the distances range from 0.0 to 1.414, around 98.1% of them are greater than 1.37. The key issue is that both the inter-cluster distances (green) and the intra-cluster distances (red), which are obtained from given classification information, show the same behavior. However, for clustered data, such a graph typically shows two peaks: one peak for the distances inside the clusters and another one representing the average distance between the clusters [157]. Consequently, because only one peak is present, any purely distance-based approach will have problems with finding the underlying clustering of this data set.

## 2.2 Visualization of High-Dimensional Data

Since there are many clustering techniques which support different data aspects, features, and various notions of a cluster, the most desired strategy to detect groups would be to simply “look” at the point cloud in its original domain. Such a visual

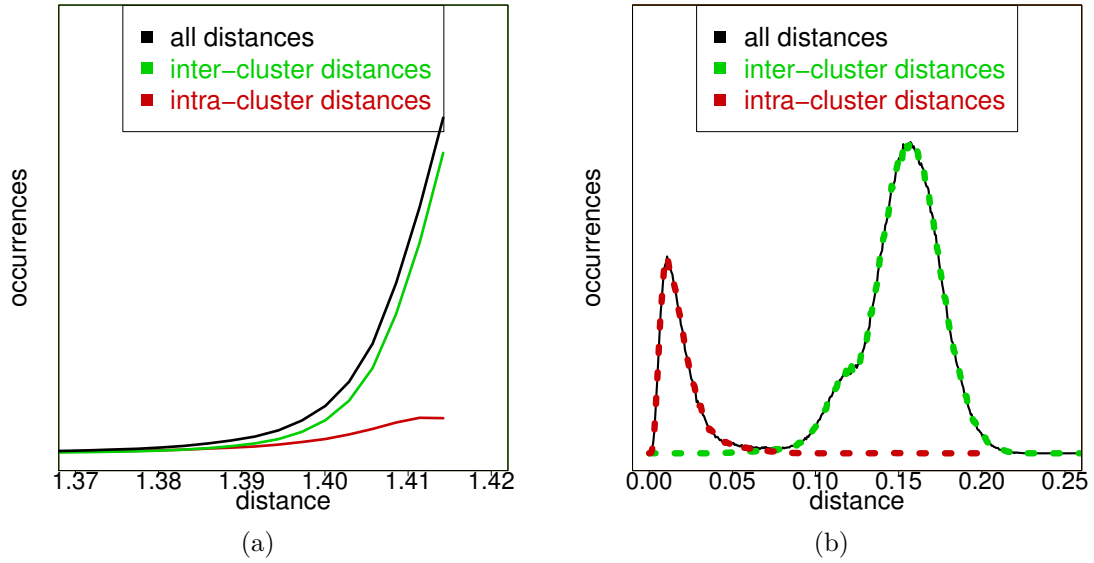


Figure 2.2: Medline data set: Two plots showing the occurrences of all pair-wise distances (black) and their partition into inter-cluster (green) and intra-cluster (red) distances (a) in the original 22 095-D space and (b) after projecting the points into a lower dimensional space to counter the curse of dimensionality.

approach aims at delegating feature identification to the human eye-system, which is naturally trained and very effective in detecting coherent groups and telling apart structure from noise. Hence, to detect structure quickly and at both a global and a local scale, direct visualization of high-dimensional data has become a popular alternative to traditional cluster analysis. Because of its importance for many applications, visualizing high-dimensional data developed to an important and independent research branch affiliated to *Information Visualization* [23, 96, 152] and *Visual Analytics* [98, 100, 163, 99]. Classical challenges in these research areas include dealing with excessive data sizes, large dimensionality, visual complexity, screen resolution, and, most importantly, structural occlusion and overlapping data representatives. Although there are manifold concepts to visualize high-dimensional data efficiently, like, e.g., *pixel-based* methods [97] which unveil trends in the data's attributes by arranging all items in recursive patterns, the remainder of this section focuses on the most prominent techniques to visualize high-dimensional point data: variations of projections and axis-based techniques.

### 2.2.1 Projective Visualization Techniques

In Euclidean geometry, projections are mappings from an arbitrary-dimensional Euclidean space into a subspace of smaller dimensionality. In the context of data visualization and clustering, projections aim to find a lower-dimensional representa-

tion of a point cloud that preserves pair-wise distances and, thus, the clustering in the original space. In practice, the desired target space is typically two- or three-dimensional to show the projected data as a scatterplot [27], e.g., on the screen. To find suitable representations in the projected space, typical optimization criteria to preserve structure include, among others, the aggregation of a maximum of variance, a maximal distance between cluster centroids, or reducing the projection error, e.g., by minimizing the difference of pair-wise distances in both spaces. However, if the intrinsic dimensionality of the data is higher than two, information has to be discarded to find a two-dimensional embedding.

There are many dimension reduction techniques and the most popular ones are often based on singular value decomposition, like principal component analysis (PCA) [92] or latent semantic indexing [39]; on least square approximations or multidimensional scaling (MDS) [107], e.g. Sammon's mapping [145]; or on neuro-computational algorithms like Kohonen's Self-Organizing Maps (SOM) [104]. These approaches are restricted by underlying assumptions about the data's structure, e.g. linearity for PCA, dimensionality of the manifold for SOM, and neglect for the curse of dimensionality in MDS. This often results in distortions which cause illusionary artifacts and clutter. More complex approaches are often based on these basic techniques. Paulovich et al. [136] proposed an MDS projection of a representative subset of the input points through a numerical solution that aims at preserving a similarity relationship given by a metric in the original space. Jänicke et al. [86] use a manifold learning to layout the high-dimensional input points based on a 2-D force-directed layout of the Euclidean minimum spanning tree. Gansner et al. [65] embed high dimensional graphs as geographic-like maps in 2-D. After embedding the graph in 2-D, they perform a clustering on either the graph or the embedded point set to finally create and color the *countries* using Voronoi diagrams. With *Glimmer*, Ingram et al. [80] present a multilevel algorithm for multidimensional scaling designed to exploit modern graphics processing unit (GPU) hardware.

For moderate dimensionality, the scatterplot matrix [10], Prosection View [63] or Hyperbox [4] provide 2-D projections of all possible attribute configurations. However, because these techniques consider only two dimensions at one time, the analyst has to assemble a mental image about the data from different 2-D views. Since projective techniques often do not scale well for large data sets and many dimensions, specific enhancements were introduced to improve their usability: Tatu et al. [161] describe quality measures for scatterplot matrices for automatic analysis that detects potentially relevant visual structures from the set of all possible candidates. Elmquist et al. [51] introduced *Rolling the Dice*, an interactive method to



explore multidimensional data by traversing its scatterplot matrix using animations. Although this approach facilitates better understanding of correlations between single dimensions by animations, in the end, the user always faces only two dimensions of a potentially very high dimensional point cloud at one time. Other contributions to improve scatterplots include continuity [8], illumination [148], or flow [28],

Projections that only work on the raw points and a chosen metric are referred to as *unsupervised* projections. *Supervised* projections, on the other hand, use additional information to optimize the low-dimensional layout. Prominent examples are the orthogonal centroid method (OCM) [88] or linear discriminant analysis (LDA) [62]. LDA uses a segmentation of the input data into  $k$  classes to perform a supervised projection into an optimal  $(k - 1)$ -dimensional space, while maximizing inter-class distances and minimizing intra-class distances. Note that the application of LDA assumes that clusters are related to classes. While this is a rather strong a priori condition, it is still widely accepted in many applications, e.g. for newspaper articles, which are often categorized. Choo et al. [30] report on the combination of unsupervised PCA and supervised OCM and LDA as two-stage projections to minimize the information loss in the final 2-D image. For example, one of their two-stage transformations, termed *LDA+PCA*, first projects the data into the optimal intermediate space preserving its clustering structure, followed by a PCA projection down to 2-D. Due to this split, the projection error of the PCA can be reduced compared to using PCA alone to project the data directly from original space.

Projections were integrated into powerful visual exploration systems for high dimensional data analysis. Jeong et al. proposed *iPCA* [89], a system that visualizes the results of PCA using multiple coordinated views and a rich set of user interactions with the PCA output. With *DimStiller*, Ingram et al. [79] presented a modular system for dimensionality reduction and analysis. As a series of analysis steps, the analyst chains together “operators” into expressions, which finally lead to reusable workflows. *ClusterSculptor* [122] is a visual analytics tool for high-dimensional data that uses many of the techniques mentioned above. Choo et al. implemented *iVisClassifier* [31], a framework in which the analyst explores and classifies data based on supervised LDA projections. With particular attention to document collections, Paulovich et al. presented *HiPP* [135], a hierarchical point placement strategy for displaying, interacting, and organizing large multi-dimensional data sets by defining a hierarchical structure that enables the user to analyze data sets on different levels of detail. Recently, Liu et al. [113] proposed to explore high-dimensional data sets based on subspace analysis and dynamic projections, using transition graphs for flexible navigation through the identified unique 2-D linear projections.

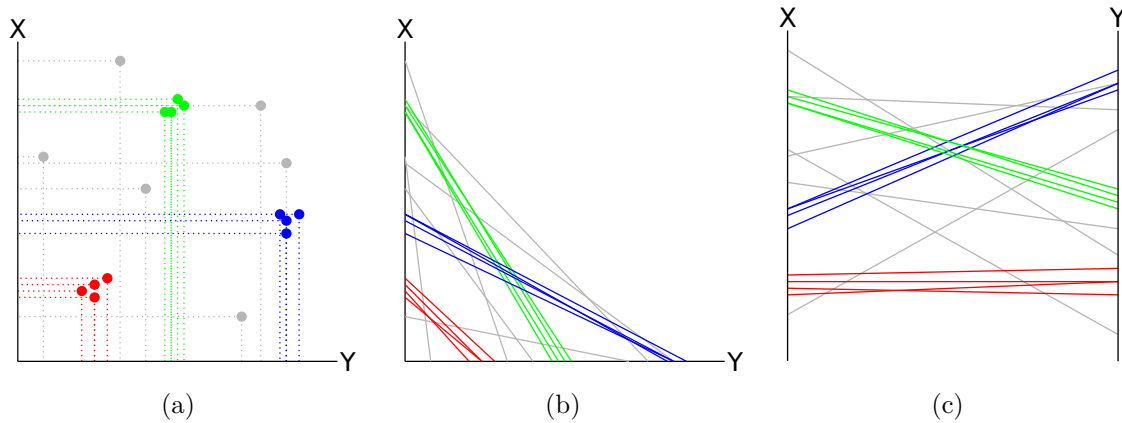


Figure 2.3: Transforming a 2-D scatterplot into parallel coordinates: (a) In a 2-D scatterplot, point coordinates are determined by the values of two variables on the horizontal and the vertical axis. (b) Instead of using the intersection points of the (virtual) lines perpendicular to the axes, each point can also be represented by a line connecting both axes at the values of both variables. This turns clusters into line-bundles. (c) Arranging the axes in parallel, finally, leads to a parallel coordinate plot, which supports visualization of higher dimensional data by adding more axes.

### 2.2.2 Axis-based Visualization Techniques

Axis-based techniques differ from conventional scatterplots in that they do not comply with the Cartesian norm to align axes perpendicularly to each other. Strictly speaking, 2-D scatterplots are also based on (two) axes. However, because these are perpendicular, their number is limited to only two in the plane. Prominent examples for axis-based techniques are *parallel coordinate plots* (PCP) [82], where the axes are arranged in parallel next to each other, or *star plots* [27], where the axes are arranged circularly around a center point. In both cases, individual data points extend to polylines that connect the axes at their corresponding value. Note that without pairwise perpendicularity, potentially many dimensions or attributes can be visualized at once, though not necessarily as intuitive and familiar as in 2-D or 3-D scatterplots. If multiple vectors share similar values and dimensions, i.e. if they belong to accumulations in the high-dimensional space or in particular subspaces, their corresponding polylines also accumulate. This comes in handy to detect clusters as thick polyline bundles that stand out in the visualization (cf. Figure 2.3). In theory, axis-based techniques can visualize arbitrary dimensional data. In practice, however, the screen resolution and the eye's accuracy constitute natural limits to detect structure. Because features in PCPs can sometimes be hard to interpret, Inselberg et al. [81] elaborate on the relation between various line segment constellations and their counterparts in the vector space, and they also

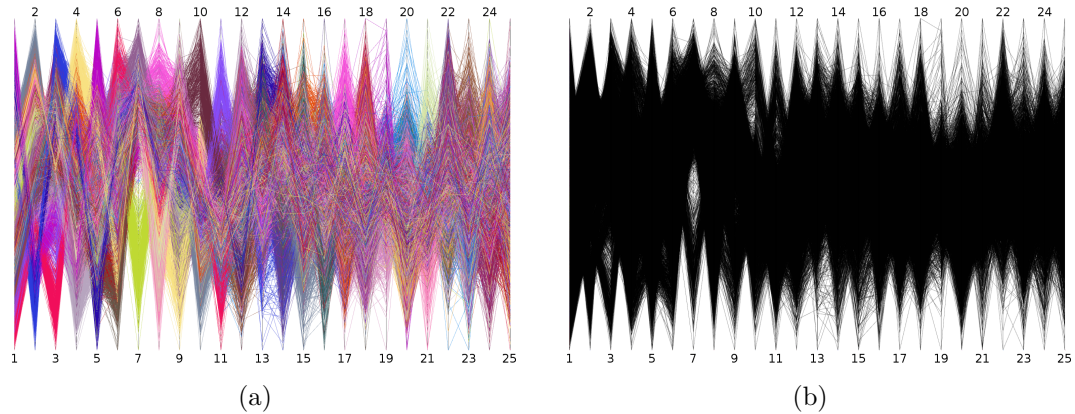


Figure 2.4: Parallel coordinate plot of the 25-D Isolet data set (cf. Appendix A.4): (a) Clustering structure is indicated primarily by line bundles of different colors. While the relation between clusters and classes is actually unknown, single-colored bundles are at least an indicator of potential separation. (b) Without using colors, however, the clustering is unclear for the most part because neither potential groups nor exact value distributions in each dimensions are recognizable. In both cases, identifying and comparing individual clusters is (very) difficult.

provide a rich set of applications. Tatu et al. [161] also specify quality measures for parallel coordinate plots to find a suitable ordering of the axes that reflects structure best. Other suggestions to improve the usability of axis-based techniques include using alpha-blending to emphasize where multiple line segments accumulate, splatting [178], interaction [68], and illustration [120].

### 2.2.3 Problems and Difficulties

Direct visualization of high-dimensional data aims at the identification of patterns and structure in the drawing as a noticeable number of coherent groups of remarkable size, shape, separation, or distance to each other. To this end, the visualization has to preserve pair-wise (dis-)similarities in order to reflect structure suitably. However, with increasing data size and dimensionality, both projections and axis-based techniques also struggle increasingly with accurate depiction of clustering structure. This holds primarily for visual complexity. Since each data point is represented by at least one pixel, the drawings usually suffer from occlusion—at the latest when the data size exceeds the number of available pixels on the screen. Furthermore, because direct visualizations do not distinguish between structure and noise, for large data, noise typically covers the visualization and can distort cluster separation, or even hide it entirely. This results in misleading and false insights about the data. But there are also specific problems with both techniques.

As for axis-based techniques, a polyline for one single data point already requires many pixels. For large and high-dimensional data, this negatively affects visual clutter and occlusions caused by crossing line segments. Furthermore, feature identification is complicated by the order of the axes. While the clustering might be obvious in one particular order, it could be missed in other orders—most likely when discriminable axes are not neighbored. Another major problem is that line bundles of actually separated clusters can still cross frequently. This is demonstrated in Figure 2.4a based on the 25-D Isolet data set (cf. Appendix A.4). Although the PCP suggests multiple clusters as differently colored line bundles, it is still difficult to identify or compare them. The situation becomes even worse for unclassified data, i.e. if all polylines have the same color (cf. Figure 2.4b).

As for projections, if the intrinsic dimensionality of the data is higher than two, information has to be discarded to find a two-dimensional embedding. This information loss, called the projection error, can obscure existent structure and indicate illusionary structure that is not present. In other words, projective visualizations have a fundamental problem: although they rely on the visual identification of structure—as conveyed by distances and closeness—they cannot ensure distance preservation in the first place. Projection artifacts are thus inevitable and occur even for data of only moderate dimensionality. This can be observed for the Reuters data set (cf. Appendix A.7). It consists of 800 vectors with 11 941 dimensions, assigned equally to  $k = 10$  classes (the letters are used in Figure 2.5): acquisitions ('a'), corn ('c'), earn ('e'), grain ('g'), interest ('i'), money-fx ('m'), crude ('r'), ship ('s'), trade ('t'), and wheat ('w'). Utilizing given classification information for a supervised projection, Figure 2.5a shows the result of the *Rank-2 LDA* [30] which consists of two subsequent LDA projections: the first one from the original space into an intermediate  $(k - 1) = 9$ -dimensional space and a second projection down to two dimensions. Although the projection seems to preserve the clustering well, two clusters on the right-hand side ('c','g','w') and in the upper left-hand corner ('i','m') consist of points of different classes. The pivotal question is why? There are two possible explanations: both clusters are indeed mixed in the original domain or the mixture is actually an occlusion artifact caused by the projection error of the second LDA. Because the first LDA preserves the clustering in the  $(k - 1) = 9$ -dimensional space, this intermediate space can be explored with a scatterplot matrix. Figures 2.5b-c acknowledge that the second assumption is true. Looking at the point cloud from the directions of the 7<sup>th</sup> and 8<sup>th</sup> (cf. Figure 2.5b) or the 7<sup>th</sup> and 9<sup>th</sup> (cf. Figure 2.5c) dimensions reveals that the points that are mixed in Figure 2.5a form their own clusters in the intermediate 9-D space. This does not surprise much

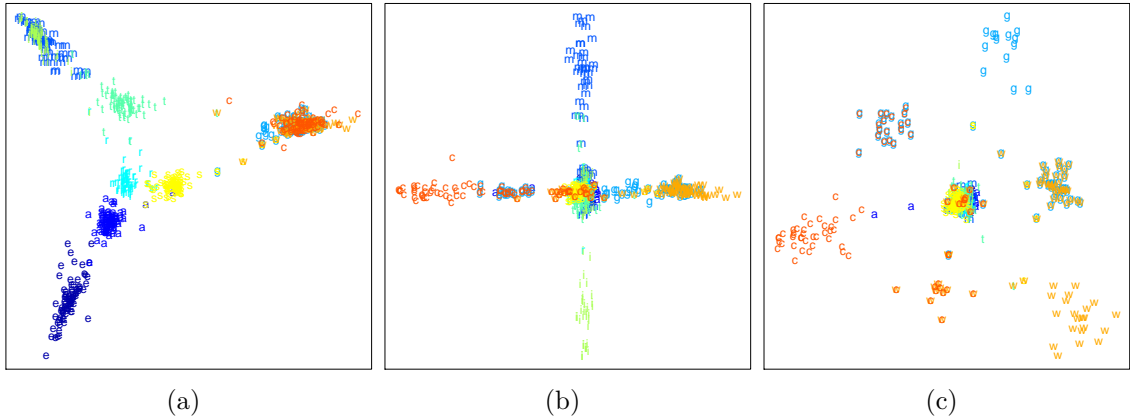


Figure 2.5: Projection of the Reuters data set (cf. Appendix A.7): (a) Scatterplot of the Rank-2 LDA showing separated clusters with points of individual classes and two clusters containing points of mixed classes. Looking at the intermediate 9-D space (b) from the 7<sup>th</sup> and 8<sup>th</sup> and (c) from the 7<sup>th</sup> and 9<sup>th</sup> dimension reveals that both (alleged) clusters actually consist of separated accumulations per class.

given that the second LDA uses only two dimensions to discriminate the classes which contribute most to the optimization criterion. Nevertheless, due to the lack of any knowledge about the intermediate space, the analyst would most likely tend to assume mistakenly that both clusters are really mixed in the original space. Also note that without the opportunity to color the points according to their class, the potential artifacts would have been missed at all. The same is true even for classified data if there is no a priori knowledge about the relation between clusters and classes.

In summary, although they are used frequently for that purpose, projections and axis-based techniques seem suboptimal for visual cluster analysis of large and high-dimensional data. Because they are susceptible to illusionary artifacts, visual complexity, occlusion, and, most importantly, because they do not differentiate between structure and noise, they depend on the data’s benignity for proper depiction of structure. However, this benignity typically decreases with increasing data complexity. That is, if the data becomes more complex or grows in size and dimensionality, the visualization becomes cluttered and difficult to interpret. In other words, the more structure there is in the data, the harder it becomes to identify any structure at all. This is even worse for noisy data or if classification information is unavailable to make use of colors in the visualization. From a clustering point of view, structural occlusion (occurring for whatever reason) simply prevents the analyst from identifying primary clustering information and from comparing individual clusters. Even if clusters are visible, their (illusionary) extent might not be meaningful and to

compare cluster sizes, points or polylines need to be counted manually—which also assumes that there are no duplicates in the data.

## 2.3 Novel Topology-Based Solution Approach

The intended preservation of geometric properties like distances, positions, or cluster shapes in the final visualization is merely a tool to convey structure in high-dimensional data. However, especially for visual cluster analysis, this tool must be questioned if the result is not guaranteed to be correct and if basic cluster properties are difficult to identify and compare [93]. It turns out that knowledge about single points, absolute distances, exact locations or shapes actually describes secondary clustering information that complicates the depiction of fundamental properties in the first place. Moreover, using representatives for every single data point to let them *simulate* high-dimensional proximities is neither promising nor necessarily needed to provide a structural overview. Therefore, the goal is to break the habit of applying non-scalable techniques to large and complex data and to find a more appropriate tool for visual analysis of high-dimensional point clouds.

To come up with another solution, we first have to define what is actually desired. Speaking of clusterings, the primary subject of interest are point accumulations surrounded by sparse or empty regions. That is, one is interested in how many coherent groups there are, whether they are embedded in each other, or whether they are well-separated or surrounded by noise. Quantitative properties of individual clusters typically include the number of points, information about their spread or compactness, or their distribution to derive coherence and intra-cluster similarities. Note that for such a clustering description, neither the points themselves nor their pairwise distances are needed. Taking a closer look, it is actually the information derived from point distances that prevents the analyst from identifying the clustering. Thus, if we could do without secondary properties like cluster distances or a cluster's shape or relative position, we could better focus on primary properties. It appears that preserving global knowledge has to be the first step and that an abstraction of the point cloud into *regions of data appearance* would suffice for this purpose.

Topological tools are efficient at summarizing data prior to visualization. Typically, they segment a domain into parts of equal behavior or properties. We can build on these ideas if we abstract the point cloud and convert it into another form that is suitable to detect clusters as regions of data occurrence. To this end, we consider the point cloud's density function and obtain from topological analysis a segmentation into (nested) dense regions. Regions of high density represent clusters, low-dense

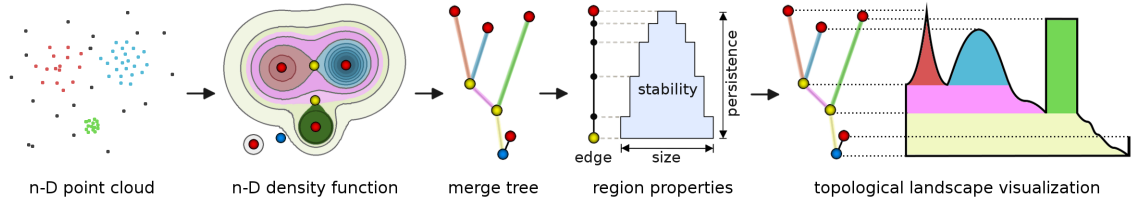


Figure 2.6: Topology-based visual analysis approach: The high-dimensional point cloud is abstracted by its density function to represent clusters as dense regions. Maxima (red) of the function indicate feature candidates and saddle (yellow) densities describe region hierarchy and/or separation. The merge tree captures the function’s topology, i.e. how the regions evolve. Each merge tree edge stores three region properties—a region’s size, persistence and stability (cf. Chapter 3.2.4)—and a list of all points in that region. The structural description provided by the merge tree is visualized as a topological landscape. Each tree edge is represented by a hill that accurately reflects region properties, point distributions and the data points themselves (cf. Chapter 4). Colors in the drawing indicate correspondences among regions, merge tree edges and hills.

regions represent noise, and regions of zero density indicate cluster separation. Furthermore, regions have properties, like the number of points comprised or an absolute density; which is high for compact clusters or lower if they are spread. But we cannot say anything about geometry anymore. Not about a dense region’s shape, its geometric extent, its relative position or how far away it is from another separated region. If we wanted to preserve such information in a 2-D image, we would instantly be back in the realm of projection errors and information loss.

We use the density function’s topology as a tool to simplify the data in its original domain. This abstraction makes it possible to preserve structural knowledge without loss and to visualize it occlusion-free. The topological analysis of the density function yields a tree whose edges describe dense regions and how they merge; hence its name *merge tree*. Each edge is annotated with three region properties and a list of data points together with their densities. To make the complex information provided by the merge tree easier to comprehend, it is represented using an intuitive landscape metaphor. Dense regions show up as (nested) hills and individual cluster properties are indicated by the shapes of the hills (cf. Figure 2.6). The underlying data points will be augmented to the hills of their corresponding clusters.

Such a topological approach has several advantages: First, both the density function and its topology can be calculated for arbitrary dimensional data. That is, independent from the point cloud’s dimensionality, we always end up with a 2-D or 3-D landscape visualization that is free of structural occlusion. Second, the topological description is preserved without loss. This means that every dense region that truly occurs in the original space will clearly be visible in the landscape. Moreover, because

density information is conveyed by height values, noise points do not cover structure, but will be placed separately at the bottom of the landscape. Third, topology-based quality measures provide clustering information accurately and thus facilitate identification of and precise comparison between significant features. Finally, based on the global overview, individual features can be selected and analyzed in further detail using (geometric) techniques that focus on properties that are not captured by the density function's topology. This includes the analysis of *approximated* cluster shape or distances using projections, or analyzing subspaces via parallel coordinate plots. Note that focusing on only a few features reduces the visual complexity of these techniques and that a selection of individual features would otherwise be complicated in the presence of noise or if features overlap.

In summary, because a visualization cannot preserve both structure and geometric details at the same time, we analyze them separately. At first, geometric properties (such as distance and shape) are discarded to provide an appropriate clustering overview that does not suffer from structural occlusion. Afterwards, the analyst can explore the data, annotate single points, or brush-and-link particular features to linked views for further, local analysis. Hence, scalability issues are solved by the assumption that the data contain fewer features than points and that local analysis is applied to only a few features at one time.

The related work and the theoretical concepts involved in each part of the process pipeline of the novel topological approach for both time-invariant and time-varying data will be introduced in the corresponding chapters.



**Part I**

**Visual Analysis of  
Time-Invariant Clusterings**



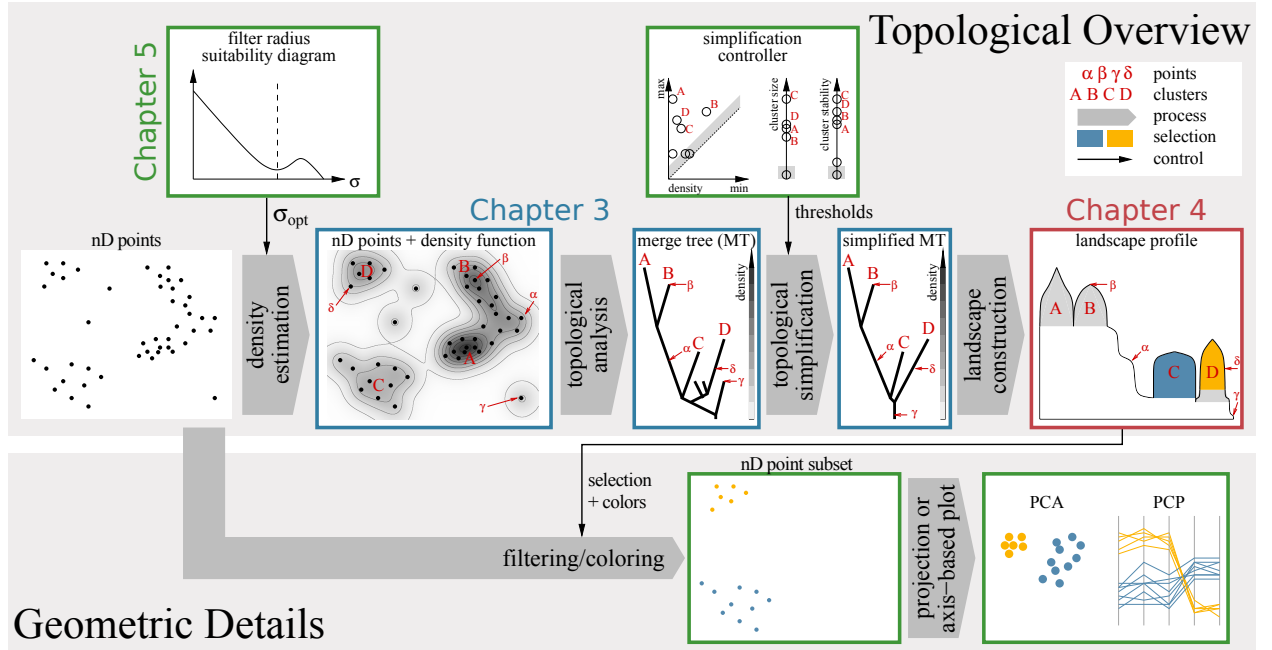


Figure Part 1: Overview of the visual analysis framework for high-dimensional data that do not change over time: The analyst obtains a structural perspective on the data by analyzing the input point cloud’s density function topologically. Clustering structure and individual cluster properties are described by the function’s merge tree. The topological analysis and the complexity of the merge tree depend on parameters for whose selection we present interactive controller widgets. Eventually, the merge tree is visualized as a topological landscape. Hills, their nesting, size, and shape accurately reflect the high-dimensional clustering. Selection mechanisms help the analyst to isolate and link features to projections or parallel coordinates for local analysis of those properties not captured by the topological view.

The first major part of this thesis is about topology-based visual analysis of high-dimensional point clouds that do not change over time. That is, during data inspection both the number of points as well as their coordinates are fixed. Subsequent to the topological analysis, clustering structure will be depicted using a landscape metaphor. However, the static landscape visualization is not final. It is rather a vantage point of an interactive visual analysis loop and is intended to facilitate data exploration top-down, localizing feature investigation on demand. By presenting clustering structure occlusion-free, the analyst can filter individual features by linking clusters and point sets to other views. This permits data inspection under various aspects and at finer granularity using traditional techniques for high-dimensional point data. The final framework supports the analyst with parameter widgets, appropriate selection mechanisms and optimized views to permit convenient clustering exploration without suffering from structural occlusion and visual artifacts.

Figure Part 1 illustrates the analysis pipeline and also provides a visual guideline for the content of the next chapters. As indicated by the blue parts, the algorithmic core of the topology-based approach is introduced in Chapter 3. This chapter contains explanations about the efficient construction of the density function, its topological representation by the merge tree, implemented algorithms and data structures, and also a discussion of runtime and memory aspects. Chapter 4, indicated by the red part, introduces multiple topology-based landscape visualizations to depict clustering structure and the underlying data points. Their usability for high-dimensional clustering visualization is evaluated and demonstrated based on several example data sets. Finally, Chapter 5, as indicated by the green parts, describes the extension of the topological method to an interactive visual analysis framework. This includes the determination of crucial parameters, for which we introduce helpful widgets to support the analyst in finding suitable thresholds, the definition of selection mechanisms for linking subsets to other views, the presentation of a modular prototype implementation, and the application to various real-world data sets.

## Chapter 3

# Topological Representation

The high-dimensional point cloud is abstracted by its density function because the topology of this function can be determined in the original space and be preserved without loss in a low-dimensional visualization. The density estimation can be thought of as reconstructing a probability space from the given samples which is then analyzed topologically to study the data in terms of critical points and dense regions, including their hierarchy and region properties.

This chapter introduces an efficient approximation of a suitable high-dimensional density function and describes the algorithms, data structures, and optimizations required to obtain a suitable clustering description. It also presents supported topology-based quality measures to evaluate cluster significance. An intensive study of all involved parameters, a result section that discusses runtime and memory consumption issues, as well as a final discussion of advantages and drawbacks concludes this chapter about the algorithmic core and the topological representation.

### 3.1 Related Work

In order to determine a suitable density function in a high dimensional space, we need methods to fill the void between data samples and methods to describe closeness between points. Given an appropriate density approximation, topological methods then identify and describe the structure of the point cloud's density function. This section provides an overview of the related and adopted concepts and algorithms.

#### 3.1.1 Neighborhood Description

Let  $\delta(x, y)$  denote the distance of two points  $x, y$  in a  $d$ -dimensional Euclidean space  $\mathbb{R}^d$ , the *Voronoi diagram* [58] of a point set  $P = \{p_1, p_2, \dots, p_n\}$  is a partition of  $\mathbb{R}^d$

into cells  $c_i = \{x \in \mathbb{R}^d \mid \forall j \neq i : \delta(x, p_i) \leq \delta(x, p_j)\}$ . Each cell's points have the same closest point of  $P$ .

A *simplicial complex*  $M$  is a finite collection of simplices (vertices, edges, triangles, tetrahedra; and their generalizations to arbitrary dimensions) such that each simplex' faces are elements of  $M$ , and each intersection of a simplex pair of  $M$  is either empty or a face of both.

The *Delaunay triangulation* [58] of a point set  $P$  in  $\mathbb{R}^d$  in so-called general position is a simplicial complex such that for each simplex there exists an empty circum-hypersphere. There is a well known duality: two cells  $c_i, c_j$  are neighbors in the Voronoi diagram if and only if the Delaunay triangulation contains an edge between  $p_i$  and  $p_j$ . There are algorithms [13] that can compute the Delaunay triangulation directly and they perform well in two and three dimensions. The number of simplices is in  $O(n^{d/2})$  in the worst case [58], giving an exponential lower bound on the runtime with respect to the dimension. We will refer to the vertices and edges of the Delaunay triangulation, omitting all other simplices, as the *Delaunay graph* ( $DG$ ).

A *neighborhood graph* [87], also called a *proximity graph*, is a graph in which *neighbors* vertices are connected by an edge. The Delaunay graph is a suitable neighborhood graph as it maps Voronoi cell neighbors to edges.

The *Gabriel graph* ( $GG$ ) [64] of a point set  $P$  in  $\mathbb{R}^d$  is a graph  $(P, E)$  that contains an edge between two vertices  $u, v$  if and only if the *Gabriel lune* contains no point from  $P$ , except on its border. The Gabriel lune is the smallest hypersphere with both  $u$  and  $v$  on its border. The Gabriel graph can be computed (naively) using  $O(n^3)$  distance calculations by testing for each point pair  $u, v \in P$  whether their Gabriel lune does not contain any other  $w \in P, w \neq u, w \neq v$ . Note that a distance calculation in a high dimensional space usually takes  $O(d)$  time, giving an overall runtime of  $O(dn^3)$ . However, the algorithm can test each point pair independently and thus runs in parallel using up to  $n(n-1)/2$  processors.

The *relative neighborhood graph* ( $RNG$ ) [87] of a point set  $P$  in  $\mathbb{R}^d$  is a graph  $(P, E)$  that contains an edge between two vertices  $u, v$  if and only if their *RNG-lune* is devoid of points from  $P$ . This lune is the union of two hyperspheres being centered around  $u$  and  $v$  and having a radius equal to the distance of  $u$  and  $v$  (cf. Figure 3.1a). The construction of the RNG and the GG differs only in the test for each edge.

A *spanning tree* of a point set  $P$  is a connected, acyclic graph  $(P, E)$ . The *Euclidean minimum spanning tree* ( $EMST$ ) of a point set  $P$  in  $\mathbb{R}^d$  is the spanning tree where the sum of all weights (in this case the Euclidean distance between the edges' endpoints) is minimal compared to all other spanning trees.

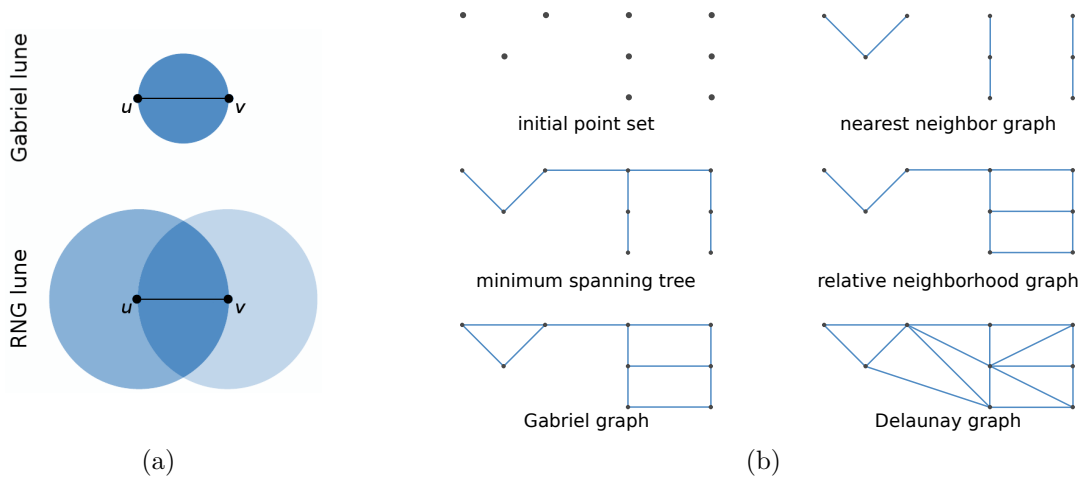


Figure 3.1: Neighborhood graphs of a point set: (a) The Gabriel lune (top) and the relative neighborhood lune (bottom) define the regions of influence for a particular pair of points  $u$  and  $v$  that is required to be empty so that there is an edge between both points in the neighborhood graph. (b) Different neighborhood graphs with an increasing number of edges (from left to right and from top to bottom).

The *nearest neighbor graph* (NNG) [87] of a point set  $P$  in  $\mathbb{R}^d$  is the smallest graph  $(P, E)$  where each vertex has an edge to its nearest neighbor.

A special subset relationship exists between the introduced neighborhood graphs in Euclidean spaces:

$$NNG \subseteq EMST \subseteq RNG \subseteq GG \subseteq DG$$

Because the Euclidean minimum spanning tree is a connected graph, all its supergraphs are connected as well; connectedness will be required for the topological analysis. Being a supergraph of the NNG also implies that the connection of each vertex to its nearest neighbor is also contained. Example illustrations of the neighborhood graphs used for the topological analysis are shown in Figure 3.1b.

### 3.1.2 Scalar Field Topology

In scientific visualization, data are often provided as a discrete scalar function, called a scalar field, describing, e.g., pressure, density or temperature. Studying a scalar function intends to reveal its structure to find out whether it contains (un)expected or suspicious features that require closer investigation. In addition to visual approaches, like volume rendering [111] or extracting isosurfaces using marching cubes [115], computing the topology of a scalar function has proven to be efficient to investigate the data's structure. Topological analysis aims at a complete study of the variation

of the function, described by critical points or entire regions [166], and to simplify the view on the data prior to visualization.

Let the data be given as a point set  $P = \{p_1, p_2, \dots, p_n\}$  in  $\mathbb{R}^d$ , with corresponding scalar measurements  $H = \{h_1, h_2, \dots, h_n\}$ . To interpolate values for points not in  $P$ , the data is extended to the entire space by means of a mesh with vertex set  $P$ , and a continuous function  $f$  that satisfies  $f(p_i) = h_i$ .

A *superlevel set* of a function  $f : \Omega \rightarrow \mathbb{R}, \Omega \subset \mathbb{R}^d$  for a value  $v$  is the set of all points  $x \in \Omega$  where  $f(x) \geq v$ . It may consist of multiple connected components, whose evolution with varying  $v$  is of particular interest. Imagine the function  $f$  as a landscape, initially fully submerged by water, and the value  $v$  as the current water level. When the water is slowly drained, i.e.,  $v$  is decreased, hills will start to emerge from the water, which corresponds to new superlevel sets created at  $f$ 's local *maxima*. When draining the water further, hills/superlevel sets will merge at points called *saddles*. The draining process is stopped when the landscape is free of water, i.e. when  $v$  reaches  $f$ 's *global minimum*. Although the metaphor uses a 2-D landscape, the concepts are applicable in any dimension.

The *merge tree* is a labeled directed graph that encodes changes of superlevel set connectivity. Leaves represent local maxima, inner nodes represent saddles, the root represents the global minimum, and edges connect nodes according to the process outlined. Each node is labeled with the value  $v$  of its corresponding event. Figure 2.6 (on page 23) illustrates these correspondences. Each point in the merge tree, be it a node or on an arc, corresponds to exactly one connected component of a superlevel set for one value  $v$ . For piecewise-linear functions on simplicial grids, Carr et al. [24] give an algorithm to compute the *augmented merge tree*: Initially, each grid vertex is represented by one node in an otherwise empty tree and one set in a union-find data structure. The algorithm processes all grid vertices  $u$  in order of decreasing function value and (1) determines the sets of  $u$ 's *upper link*, i.e., grid neighbors with higher function value, (2) adds an *arc*, i.e., a directed edge, from  $u$ 's node to each set's lowest node, and (3) unites these sets with  $u$ 's set and declares  $u$ 's node as the new set's lowest node. The augmented merge tree consists of different types of nodes: *maxima* are nodes without outgoing arcs, *regular nodes* have one outgoing and one incoming arc, *saddles* are nodes with multiple outgoing arcs, and the unique *root node* has no incoming arcs. The *unaugmented merge tree* can be computed from the augmented merge tree by removing all regular nodes and replacing paths of arcs by *superarcs*. The remaining nodes are referred to as *supernodes*. In this thesis, we refer to the unaugmented merge tree simply as the merge tree and point out the augmented version if needed. Moreover, to ensure a consistent nomenclature, we



henceforth use *points* to refer to input data items, *vertices* and *edges* to refer to items of the neighborhood graph, and *superarcs* and *supernodes* to refer to items of the merge tree. Because we never work directly on the augmented merge tree, we sometimes use *arc* and *node* as acronyms for superarc and supernode, respectively.

To remove ambiguities, the merge tree algorithm requires a function where all maxima, minima, and saddles occur at distinct function values. For functions that violate this condition, called non-Morse functions, *simulation of simplicity* [47] adds small symbolic perturbations to break ties in the order in which vertices are processed. These small perturbations as well as noise in the data can easily complicate the merge tree, but can be detected and removed by *topological simplification*. In this process, superarcs are annotated with a measure of robustness, e.g., the size of the domain subset that corresponds to the superarc, or the superarc’s *persistence* [46]—the difference between the maximum and minimum function value of a region. Repeatedly, the leaf superarc of lowest robustness is removed from the tree, potentially turning a saddle into a regular node, which, together with its two incident superarcs, is then replaced by a superarc. For example, topology-based simplification is helpful to remove topological structures based on local geometric measures in 3-D [26].

The merge tree can be partitioned into *branches*, which are paths that are monotonic with respect to the function values. The branches can be organized in a *branch decomposition* [133, 134], which is a tree. Each of this tree’s nodes is a branch; the root is the branch of highest persistence. One node is a child of another node if the branch it represents has lower persistence than the other node’s branch and both branches are connected by a saddle. It is also possible to construct the branch decomposition for other region properties. Very “deep” branch decompositions with many hierarchy levels can be *re-balanced* [169]. Child branches whose saddles values are close to that of its parent are moved up in the hierarchy and become its siblings. This reduces the depth of the branch hierarchy and in particular removes artifacts due to symbolic perturbation. However, the re-balanced branch decomposition no longer reflects the exact nesting properties of the superlevel sets.

For a *sublevel set* of the function  $f$ , i.e. for the set of all points  $x \in \Omega$  where  $f(x) \leq v$ , the above definitions can be inversed to define the *split tree*; a tree whose leaf nodes reflect the minima of  $f$  and which, hence, describes the splitting behavior of  $f$ ’s features for varying  $v$ . Carr et al. [24] describe an algorithm to merge both trees into the *contour tree* to describe the evolution of *level sets*. A level set of  $f$  at some function value  $v$  is the set  $\{x \in \Omega \mid f(x) = v\}$  and may consist of zero, one, or more connected components. For lower dimensions, these sets of connected components are known as *isolines* (2-D) and *isosurfaces* (3-D). In general, they are

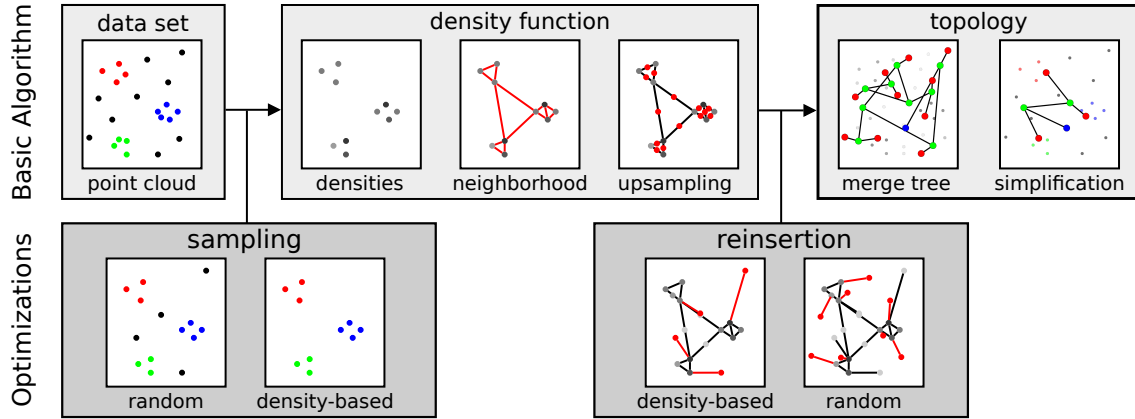


Figure 3.2: Algorithmic pipeline of the topological analysis: Optional sampling represents the input data by a smaller point set of very similar clustering structure. To approximate the high-dimensional density function, we evaluate the densities at the remaining points, construct the neighborhood graph and upsample the density function at the midpoints of that graph’s edges. After an optional reinsertion of non-samples, we compute the merge tree of the upsampled neighborhood graph and remove noisy fluctuations using topological simplification. The simplified merge tree contains a maximum (red) for each dense region/cluster. Saddles (green) and their densities describe cluster hierarchy or separation.

referred to as *contours*. If the function value  $f(x)$  is thought of as time, we can watch the evolution of contours of  $f$  over time, seeing them appear, join, split, and disappear. The contour tree is a graph in which each contour is contracted to a single point and that tracks these topological changes. The contour tree is not necessary for the topology-based visual analysis presented in this thesis. However, because algorithms and visualizations that are applicable to contour trees are also applicable to the less complex merge tree, we occasionally refer to the contour tree at several places throughout this thesis.

## 3.2 Basic Algorithm

Analyzing the clustering of the input data topologically consists of multiple parts. From an implementation point of view, these sub-routines can be thought of as parameterized modules that are linked together as a chain and executed consecutively. Input of the first module is a high-dimensional point cloud and output of the last module is a simplified merge tree encoding the point cloud’s clustering (cf. Figure 3.2). Algorithmic scalability is achieved by inserting or excluding optional modules or by changing parameters of mandatory modules to reduce the accuracy of the topological abstraction. While these optimizations will be explained in Chapter 3.3, this section

describes the mandatory parts required to obtain a merge tree as a structural description of the point cloud.

### 3.2.1 Approximation of the Density Function

Given a set  $P = \{p_1, p_2, \dots, p_n\}$  of points in a fixed-dimensional Euclidean space  $\mathbb{R}^d$ , the implicit density function of  $P$  is typically estimated by applying a filter kernel, e.g. the Gaussian filter

$$f(x) = \frac{1}{n(\sigma\sqrt{2\pi})^d} \sum_{i=1}^n \exp\left(-\frac{\delta(x, p_i)^2}{2\sigma^2}\right),$$

with  $\delta(x, p_i)$  being the Euclidean distance between sample point  $x$  and data point  $p_i$ . The kernel window width  $\sigma$ , henceforth called the filter radius, describes the influence of all points to the density of a particular sample. To accelerate the density estimation, we use a cut-off radius and define that points with a distance from the sample higher than this threshold do not contribute to its density. A consequence of this simplification is that samples can have zero density if their neighborhood defined by  $\sigma$  does not contain any point  $p_i \in P$ .

Because it is infeasible to compute the topology of  $P$ 's density function analytically, a common alternative is to construct a mesh on the given points. In lower dimensions, the function is usually sampled on a regular grid of sufficiently small resolution. In higher dimensions, however, this approach is impractical because regular grids grow exponentially with every additional dimension. Hence, we seek another sampling scheme that is both economic to be applicable to high-dimensional data and accurate to detect all clusters. To this end, we aim for a simpler function  $f'$  of very similar topology that is a piecewise-linear interpolation on a complex of simplices.

The optimal simplicial complex is the high-dimensional Delaunay triangulation. However, using the Delaunay triangulation on  $P$  with  $f'(p_i) = f(p_i), \forall p_i \in P$ , has two disadvantages: a prohibitive runtime of  $O(n^{d/2})$  [59] to construct it and a rather coarse approximation of  $f$  since the density between two points can be lower than at those points. In this case,  $f'$  would lack topological features to reveal those regions that are separated by low density (cf. Figure 3.3a).

Coarseness, the second disadvantage, can be countered by adding all topologically relevant points of  $f$  to the Delaunay triangulation; which are very hard to compute. Fortunately, the topology is merely a tool to identify dense regions and their nesting. So as long as all dense regions are found accurately, we can ignore the exact positions of  $f$ 's topological features. Therefore, we use a heuristic that adds a further sample on the midpoint  $m$  of each mesh edge and require  $f'(m) = f(m)$  if the density at this

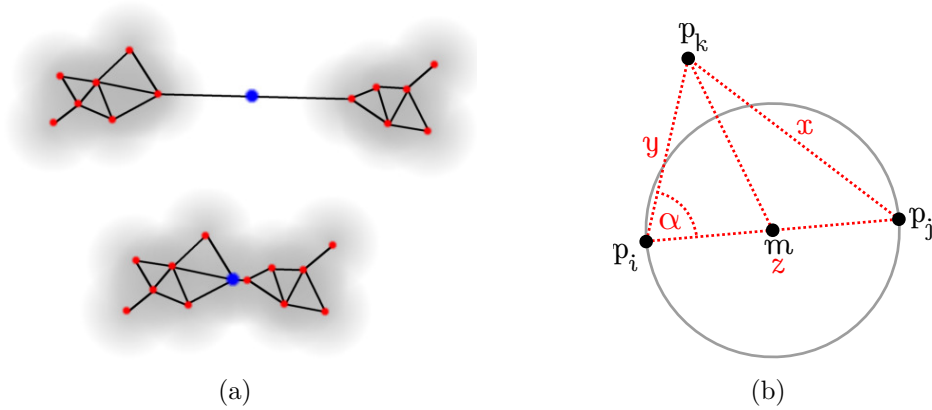


Figure 3.3: (a) Without searching for a potential saddle of low density between two neighbored dense regions, the saddle would lie on one of the regions' borders and separation would be missed. The topological analysis would “see” the lower rather than the upper configuration. (b) Because the triangles  $\triangle p_k p_i p_j$  and  $\triangle p_k p_i m$  share the angle  $\alpha$ , the distance  $\delta(m, p_k)$  can be computed efficiently from the distances between  $p_i, p_j$  and  $p_k$  using the law of cosines in  $O(1)$ .

position is lower than at the edge's endpoints. We refer to this process as *upsampling*. It suffices to add only those midpoints of lower density, as only then they can affect the topology. It is also possible that a new maximum is found at the midpoint of an edge inside a cluster. However, in this case,  $m$  would lack a point  $p_i \in P$  to represent this maximum. This becomes problematic during the later visual analysis, which is intended to segment only the given data points  $P$ . We only split edges if needed, thus keeping the size of the graph small and allowing us to terminate the calculation of  $f(m)$  once it exceeds the minimum of both endpoints. Upsampling also requires the distances from an edge's midpoint to all other given points  $p_i \in P$ . To accelerate the density evaluation at a midpoint, a special property of Euclidean spaces can be exploited: for the midpoint  $m$  between  $p_i$  and  $p_j$ , the distance  $\delta(m, p_k)$  can be computed in  $O(1)$  instead of  $O(d)$  only from the distances  $x = \delta(p_j, p_k)$ ,  $y = \delta(p_i, p_k)$

and  $z = \delta(p_i, p_j)$  observing that  $p_i, p_j, p_k$  span a planar subspace that includes  $m$  and in which the general law of cosines holds (cf. Figure 3.3b):

$$\begin{aligned}
 x^2 &= y^2 + z^2 - 2yz \cos \alpha && (\text{for } \triangle p_k p_i p_j) \\
 \delta(m, p_k)^2 &= y^2 + \left(\frac{z}{2}\right)^2 - 2y \frac{z}{2} \cos \alpha && (\text{for } \triangle p_k p_i m) \\
 \\
 \delta(m, p_k)^2 &= y^2 + \left(\frac{z}{2}\right)^2 - 2y \frac{z}{2} \left( \frac{x^2 - y^2 - z^2}{-2yz} \right) \\
 &= y^2 + \frac{1}{4}z^2 - yz \left( -\frac{x^2}{2yz} + \frac{y^2}{2yz} + \frac{z^2}{2yz} \right) \\
 &= y^2 + \frac{1}{4}z^2 + \frac{1}{2}x^2 - \frac{1}{2}y^2 - \frac{1}{2}z^2 \\
 &= \frac{1}{2}x^2 + \frac{1}{2}y^2 - \frac{1}{4}z^2
 \end{aligned}$$

This means precalculated distances from the preceding density estimation of the input points  $P$  can be reused for upsampling. Because the topological analysis requires pair-wise distances in several of its sub-routines, it is beneficial to store them in a distance matrix. It is also sufficient to work with squared distances to avoid expensive calls of the square root function. Note that storing a distance matrix is basically a trade-off between runtime and memory consumption. Typically memory consumption is less critical and calculating many distances in  $O(d)$  can quickly become a runtime bottleneck for high-dimensional data, especially if they are determined repeatedly.

The prohibitive runtime to construct the Delaunay triangulation in arbitrary dimensions can be mitigated by using subsets instead. Since the edges in the void between dense regions are only needed to look for missing topological events, other neighborhood graphs can also be used as long as they detect the clustering accurately. Approximating the Delaunay triangulation is considered an optimization and will be discussed in more detail in Chapter 3.3.1.

The final approximation of the density function is an upsampled neighborhood graph with  $f'(v) = f(v)$  for the positions of all graph vertices  $v_i$ .

### 3.2.2 Topology of the Density Function

The standard merge tree algorithm [24] is a simple graph algorithm and can be applied to the approximation of  $f$  given by the upsampled neighborhood graph together with the vertex densities. During the merge tree computation, all graph vertices are processed in decreasing order sorted by their density. Utilizing a union-find data

structure, superlevel set evolution is recorded based on the neighborhood information provided by the graph. Superlevel set components are born at the maxima of the density function and they grow in size until they merge with another component of another region. This evolution from birth to potential merge eventually describes the nesting structure and number of dense regions of the density function. The merge tree is assembled simultaneously to the union-find data structure, and thus keeps track of exactly these changes.

The standard algorithm constructs the fully augmented merge tree which contains all regular nodes. Together with the maxima, each regular node represents a given data point  $p_i \in P$ . Hence, for a structural summary of the data set, information about regular nodes is unimportant. To simplify the topological description and to visualize it later on, we require the unaugmented merge tree. Therefore, we remove all regular nodes, but still remember them with their densities as a list of implicit regular nodes for each remaining arc of the unaugmented merge tree. This information will be required to relate the underlying data points to their clusters in the final visualization.

Because the merge tree accurately captures the clustering in terms of dense regions, it is not necessary to investigate the splitting behavior of level sets or to study all minima of the density function. Still, it is worth mentioning that the density function has saddles of zero density for each well-separated dense region. This results from discarding those points with a distance higher than  $\sigma$  during the density estimation, which leads to a non-Morse density function. Depending on the order in which equally valued vertices are processed during the merge tree construction, which is typically defined by simulation of simplicity [47], one of the saddles at zero density will be the global minimum. However, while these saddles of zero density are required to confirm cluster separation, they are still equivalent for the structural summary. Hence, we combine them to a multi-minimum to eliminate topological artifacts and misleading hierarchy caused by simulation of simplicity. This approach also avoids an accentuation of non-existent relationships and structure in the final visualization.

In summary, the topological description consists of the unaugmented, rebalanced merge tree. Superarcs that connect saddles and maxima represent dense regions, superarcs connecting saddles primarily describe region hierarchy, and multiple incoming superarcs of a global minimum with zero density represent well-separated regions. Note that depending on the amount of noise in the data, saddles (or the global minimum) of the density function are typically found either directly on low density noise points, on the borders of the dense regions, or on upsampled midpoints.

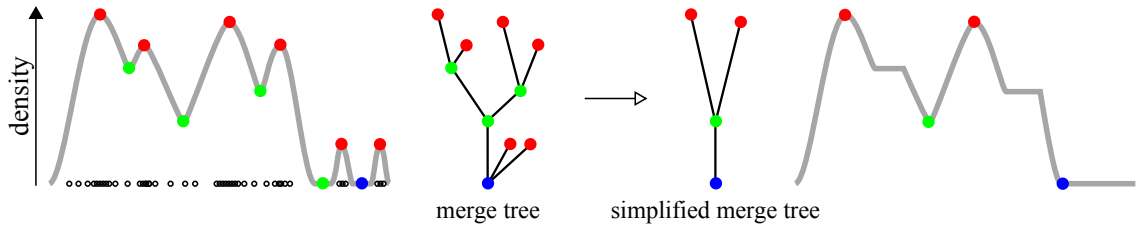


Figure 3.4: Topological simplification based on a 2-D point set: (left) Black dots at the bottom represent the underlying data. The gray graph and the colored dots indicate the density function and its critical points (red=maximum, green=saddle, blue=global minimum), respectively. The merge tree is shown to the right. (right) After topological simplification, only the prominent features remain. The implicit regular nodes of removed features are assigned to their parents with the density of the previously shared saddle. Conceptually, the simplified topology equals that of the function containing plateaus at the places where noisy features occurred before.

### 3.2.3 Merge Tree Simplification

Depending on the kernel window width  $\sigma$ , the estimated density function usually contains little variations caused by outliers or accumulations of only a few points. Compared to accumulations of many points, these minor features occur inside and outside the clusters and are thus considered noise without any special meaning in most applications of cluster analysis. Of course, if the density function is noisy, so is its topology. That is, in the merge tree, structural noise caused by small fluctuations is represented by many small (leaf) superarcs of minor *significance*. These insignificant arcs complicate the tree's structure and also its visualization later on. To focus only on the prominent features and to avoid cluttered visualizations of the merge tree, topological noise is eliminated with topological simplification.

To define a superarc's significance, the analyst specifies minimum thresholds for topology-based properties, like a region's persistence. As will be explained in Chapter 3.2.4, we support three quality measures to tell apart structure from noise. Based on these region properties, insignificant leaf superarcs are removed from the merge tree, leaving a new leaf superarc of higher significance, until only significant superarcs remain. Whenever a superarc is pruned, its list of implicitly stored regular nodes is assigned to the parent superarc and the density of these nodes is changed to the previously shared saddle. Thinking of the density function as a height-field, as illustrated in Figure 3.4 in the 2-D case, topological simplification can be thought of as cropping subhills and leaving plateaus behind [26]. Note that assigning implicit regular nodes to the parent superarc is necessary to preserve the underlying data points for final visualization. There is a special case for the global minimum: to store the implicit regular nodes of simplified superarcs that were children of the

global minimum, we create a symbolic (or virtual) child superarc that hosts noisy data points and outliers. The densities of the nodes on this dummy superarc are all zero, which is the density assigned after topological simplification if these points were children of a global minimum with zero density. The purpose of this symbolic superarc is to preserve the noise points for later visualization, e.g. to place them separately at the height of the lowest possible density, as shown for the right-most accumulation in Figure 4.10 on page 83.

### 3.2.4 Topology-Based Region Properties

Subtrees of the merge tree correspond to (sub)clusters of the point cloud. While important knowledge about the global clustering—like cluster separation, hierarchy, or the presence of noise—are already indicated by the merge tree’s saddles or the overall complexity of the tree, we still lack region properties to evaluate cluster importance or relevance. To help the analyst identify and compare significant features, we support three measures:

A cluster’s size  $size(C) = |C|$  is the number of points it contains. In terms of the density function’s topology, a region’s size reflects the number of those neighborhood graph vertices that are comprised by that region’s corresponding superlevel set component when it merges with another component at a saddle. For a superarc of the merge tree, size is defined as the number of implicitly stored regular nodes; plus one for the maximum in case of a leaf superarc. Note that upsampled points are virtual and never contribute to a feature’s size.

A cluster’s persistence [46]  $pers(C) = \max_{p \in C} dens(p) - \min_{p \in C} dens(p)$  is the absolute difference of the maximum and minimum density of its corresponding region. For a superarc, persistence is simply the absolute difference of the function values of both end-nodes. Relative to its surrounding density, persistence specifies how significant and distinct a region is. For well-separated clusters, where the saddle density is zero, persistence is equal to the maximum density inside the cluster.

Borrowing from the idea of  $d$ -dimensional hypervolume, a cluster’s stability  $stab(C) = \sum_{p \in C} dens(p)$  is defined as the sum of the contained points’ densities. Conceptually, this measure summarizes the function value distribution and represents the amount of energy required to erode the cluster. While a region with many points close to the density maximum is both stable and expensive to erode, a region with many point densities near the saddle density is unstable and cheap to erode. This idea will be picked up and become clearer in the visualization of the merge tree (cf. Chapter 4). A region’s stability is maximal if all cluster points have the same density.



From these region properties, a cluster’s shape or extent can be approximated. While a large-sized cluster of low persistence must be rather spread, a small-sized one of high persistence must be very compact. Because they are defined by simple numbers, these properties can also be preserved in low dimensional visualizations—unlike geometric properties like the shape or the point distribution of a high-dimensional cluster. These topology-driven quality measures are also intuitive for common information spaces and can reveal sophisticated insights. In general, high density reflects data coherence. This is because only similar entities accumulate in the information space and can thus contribute to their mutual density. Then, coherence can be quantified by evaluating compactness, as conveyed by persistence and size, or by judging the intra-cluster distribution using that region’s stability. For example, in a document space, evaluating a cluster’s compactness reveals how precise multiple documents report on a particular topic and evaluating cluster persistences is helpful to compare their corresponding topics’ overall importance. If documents are less alike, they are rather scattered in the information space. In this case, a cluster’s stability also takes the density distribution into account, which is useful to not miss less persistent, but large-sized document accumulations. Similar considerations apply for other applications, e.g. to evaluate and quantify how much images match a particular motif or scenery.

Apart from being used for feature identification and feature comparison, the topology-based quality measures are also important for the merge tree simplification (cf. Chapter 3.2.3). To specify a minimum feature significance, the analyst provides thresholds for persistence, size, or stability to remove superarcs from the tree based on these properties. In Chapter 5, we will present interactive widgets to help finding appropriate thresholds and to simplify the merge interactively.

### 3.3 Optimizations

The basic algorithm described in Chapter 3.2 already generates a merge tree to describe the input point cloud’s clustering structure. Nevertheless, applying the algorithm even to data of only moderate size and dimensionality quickly leads to problems with runtime and memory consumption. This is primarily because of the expensive simplicial complex, i.e. the Delaunay triangulation, whose cells increase exponentially in number and complexity with every additional dimension. To make the algorithm scalable to point clouds of large size and dimensionality, we present strategies to adjust the accuracy of the neighborhood description and to reduce data size with only small effects on the density function’s topology.

### 3.3.1 Approximation of the Delaunay Triangulation

The exponential increase in runtime when computing the Delaunay triangulation in arbitrary dimensions suggests a consideration of other neighborhood graphs as surrogates. However, because smaller surrogates are not simplicial anymore, a precondition of the merge tree algorithm is not met. Yet, this condition is only sufficient and not necessary, as it is trivial to construct a simplicial complex for which the deletion of some edges does not alter the output of the merge tree algorithm. High-dimensional simplices also force each vertex to have an increasingly high edge degree. This results in very dense neighborhood graphs that can quickly have trivial topology; represented by a degenerated merge tree that consists of only one superarc. Using sparser subsets of the Delaunay graph avoids this and also decreases the algorithm's runtime substantially because fewer edges also imply less upsampling.

In comparison to the Delaunay graph (DG), the Gabriel graph (GG), the relative neighborhood graph (RNG), and the Euclidean minimum spanning tree (EMST) increasingly omit long edges of each simplex. The density along these omitted edges is usually lower than the densities along the preserved edges, which is why we make only small mistakes because the saddles of the merge tree are also likely to be preserved. Still, the chance increases to remove an edge that hosts the best saddle.

Regarding runtime-complexity, the Gabriel graph and the relative neighborhood graph can be computed in  $O(n^3)$ . As this can still take very long, a faster computation of these graphs is presented in Chapter 3.3.2. The Euclidean minimum spanning tree is the smallest possible Delaunay graph approximation. Having the fewest edges and an asymptotic runtime of  $O(n^2)$ , it is the most efficient neighborhood graph that is still connected. Note that connectedness is a precondition of the edge-based standard merge tree algorithm [24]. Figure 3.5 shows the DG, the GG, the RNG, and the EMST for an artificial 2-D point set, along with the respective merge trees of the density functions for a fixed filter radius  $\sigma$ . The topological effects of varying the neighborhood graph can be analyzed with a persistence diagram [32]. It shows the saddle and maximum values of each branch of the merge tree's branch decomposition in a scatterplot. Figure 3.6a shows the persistence diagram corresponding to the scenario shown in Figure 3.5. The sixteen separated circles in the upper left part of the diagram indicate that all four clusters can be detected as regions of high persistence with all four neighborhood graphs.

Because changing the underlying graphs does not affect the maxima and the global minima of the density functions, the persistence of the corresponding root branches are also equivalent; these are the four circles in the top left corner. However, with decreasing graph complexity, the circles of the branches corresponding to the

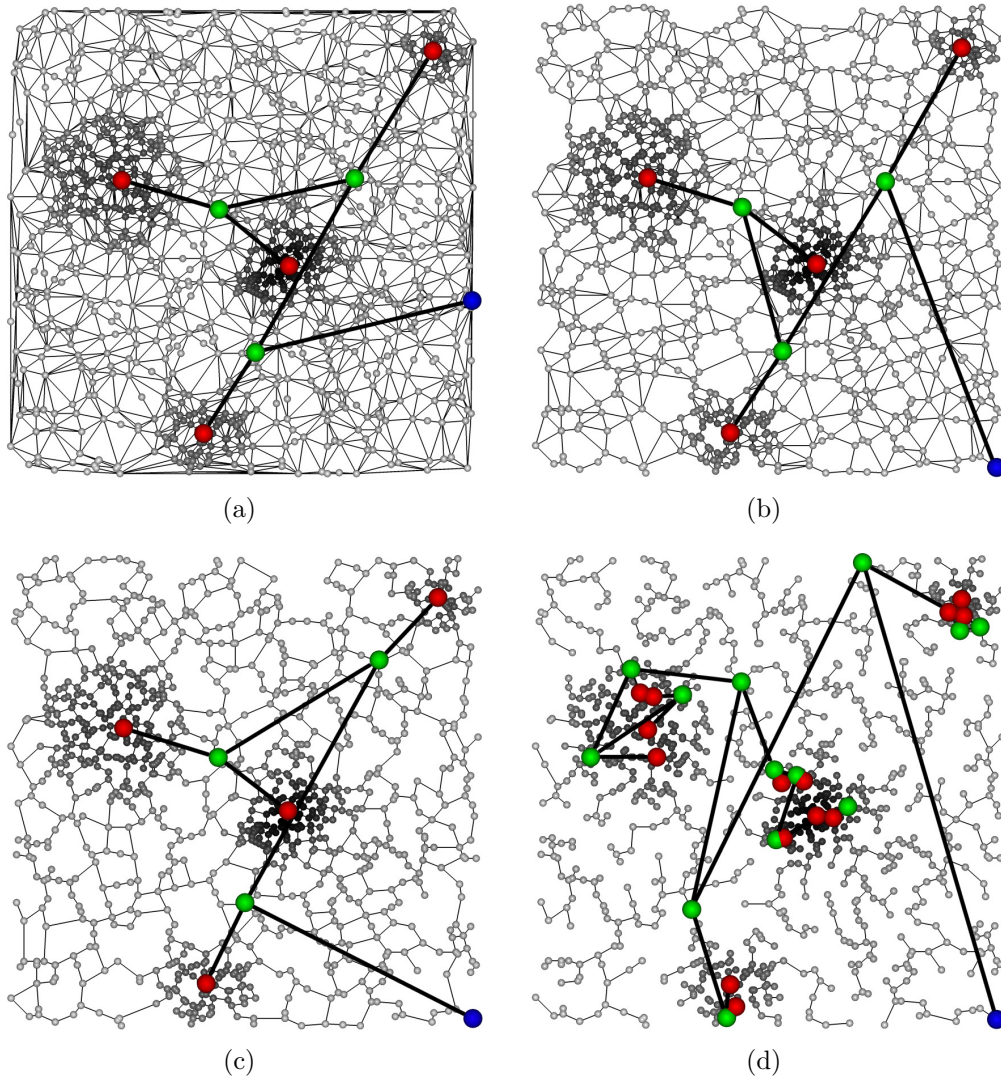


Figure 3.5: Comparison of (a) the Delaunay graph, (b) the Gabriel graph, (c) the relative neighborhood graph, and (d) the minimum spanning tree. The overlaid merge trees (red=maximum, green=saddle, blue=minimum), that were simplified using a 10% persistence threshold, illustrate the changes in the density function's (dark=dense) topology when using less complex neighborhood graphs. Note that the upsampled vertices are inserted after the graph construction.

other three clusters increasingly move to the left. This is a result of less graph connectivity and thus less freedom (i.e. graph edges) to connect the superlevel sets. Because edges vanish from the graph, so do upsampled midpoints, which forces other vertices of lower density to serve as saddle candidates. Another important observation is increasing topological noise, i.e. branches of low persistence represented by circles residing near the diagonal. Two things about noise are important when reducing the number of graph edges: First, while the DG and the GG tend to produce topological noise only in low-density regions (the lower part of the diagonal), the RNG and the

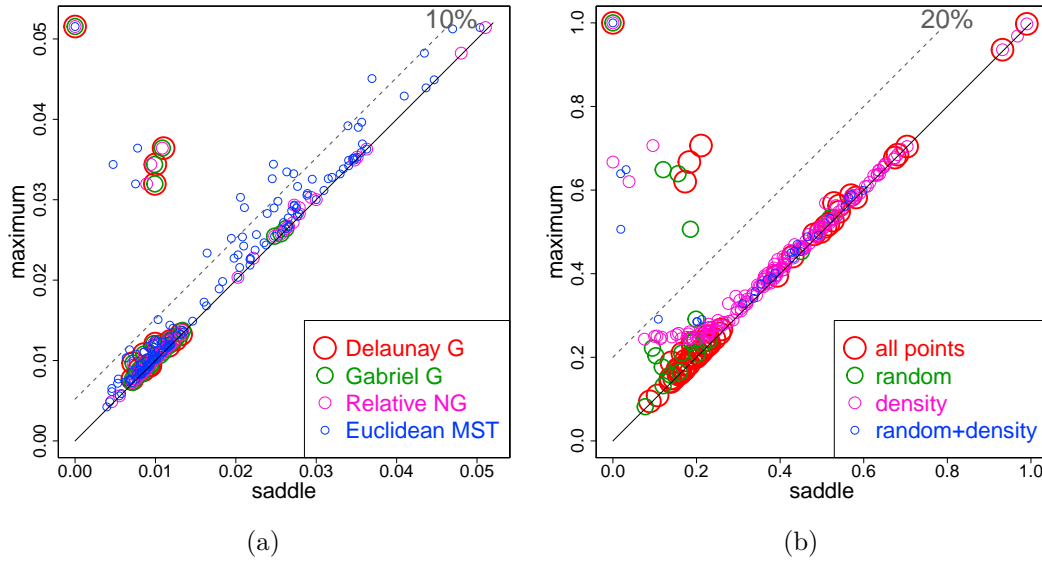


Figure 3.6: Persistence diagrams of the merge trees for (a) different neighborhood graphs (cf. Figure 3.5a-d) and (b) different sampling strategies (cf. Figure 3.7a-c). Each circle corresponds to the persistence interval of a particular branch.

EMST also produce noise in high-density areas, i.e. inside the clusters, as indicated by the distribution of purple circles (RNG) and blue circles (EMST) along the whole diagonal. Second, the persistence of noisy features, as indicated by their distance to the diagonal, also increases; especially for the EMST. The reason for these effects is that the neighborhood graph is increasingly degenerating to a tree and the (initially appropriate) neighborhood description inside the clusters is breaking into separated parts. As a consequence, cluster parts, complete clusters, or even whole regions could be represented by a subtree that is connected to the remaining graph by only one edge. This means that the evolution of superlevel sets becomes more and more dominated and restricted by the graph structure itself. Hence, superlevel sets appear for each subtree and cause topological noise.

Fortunately, additional noise caused by using less complex neighborhood graphs is only moderate and can be countered with topological simplification. As indicated by the dotted line in Figure 3.6a, a 10% persistence threshold removes the majority of topological noise—and leads to the simplified merge trees shown in Figure 3.5. The empty space between the sixteen isolated circles and those representing noise also reveals that there is in fact a possible simplification threshold that would preserve only the four main features for any of the neighborhood graphs. In general, with decreasing neighborhood graph complexity, the simplification threshold has to be increased to isolate only the most prominent features.

---

**Algorithm 1:** Pseudo-code of the improved neighborhood graph computation. Statements colored in red only apply for the Gabriel graph construction.

---

**Input** : a point set  $P = (p_1, p_2, \dots, p_n)$   
**Output** : the relative neighborhood/**Gabriel** graph  $G$

```

1 compute all pairwise distances  $\delta(p_i, p_j)$ 
2 compute each point's nearest neighbor  $NN(p_i)$ 
3 sort  $P$  for decreasing  $\delta(p_i, NN(p_i))$ 
4  $G \leftarrow (P, \emptyset)$ 
5 for  $i \leftarrow 1$  to  $n$  do
6   sort  $P$  for increasing distance from  $p_i$  into  $P'$ 
7   for  $j \leftarrow 1$  to  $i - 1$  do
8     skipedge  $\leftarrow$  false
9      $\Delta \leftarrow \delta(p_i, p_j)^2 - \delta(p_j, NN(p_j))^2$ 
10    for  $x \in P'$  with  $\delta(p_i, x)^2 < \Delta$  do
11      if  $\delta(p_i, x)^2 + \delta(p_j, x)^2 < \delta(p_i, p_j)^2$  then
12        skipedge  $\leftarrow$  true and abort loop
13    end
14    if skipedge = false then
15       $G \leftarrow G + (p_i, p_j)$ 
16  end
17 end

```

---

### 3.3.2 Faster Neighborhood Graph Construction

For a large number of points, the naïve computation of the relative neighborhood graph and the Gabriel graph can be very slow. There are optimizations for the two and three dimensional case, but they do not extend to higher dimensions.

For both graphs, a precomputed distance matrix can be used to store for each point a list of all other points sorted by increasing distance. For the RNG we found that when testing two points  $u, v$ , the innermost loop runs much faster if we test the points in the order of increasing distance from  $u$ , knowing that the loop can be aborted as soon as  $v$  is encountered, because the remaining points all have a distance greater than  $\delta(u, v)$  and therefore cannot lie inside the RNG lune (cf. Algorithm 1).

For the Gabriel graph, the following observations can be made: in Euclidean spaces, each triplet of (non-collinear) points  $u, v, w$  uniquely defines a plane, and the intersection of  $u, v$ 's Gabriel lune with the plane is a circle with center  $c$ . Because of Thales' theorem, which holds in this plane, the test whether  $w$  is outside the Gabriel lune can be written as  $\delta(u, v)^2 \leq \delta(u, w)^2 + \delta(v, w)^2$ . Using this method, we no longer need to compute the circle centers and can determine the Gabriel graph using only the pairwise distances between the original points. Also, the inner loop of

the naïve Gabriel graph algorithm aborts earlier if the points are tested in increasing distance from  $u$ . Compared to the RNG algorithm, we can abort even sooner, namely if  $\delta(u, w)^2 \geq \delta(u, v)^2 - \delta(v, NN(v))^2$  and  $NN(v)$  denotes  $v$ 's nearest neighbor, since the increasing distance from  $u$  must be compensated by a decreasing distance from  $v$  for  $w$  to lie inside the Gabriel lune.

We observed that the innermost loop was finished on average after 5–10 iterations for our data sets: for lower dimensions because a point inside the lune was found quickly and in higher dimensions because the length of the loop decreased drastically due to the “curse of dimensionality”. The runtime of the algorithm becomes dominated by distance calculations and sorting. Since each point can still be treated independently, the algorithm can be parallelized using a maximum of  $n$  processors.

The neighborhood graph construction requires all points to be distinct, but in practice, this requirement may not be met. Therefore, we construct the unique point set from the original data points and remember for each unique point the set of data points it represents.

### 3.3.3 Sampling and Reinsertion

Sampling is an optional phase to reduce the number of input points and, therefore, to reduce the algorithm's overall runtime. As we assume the given points to be samples of an unknown probability density function  $\hat{f}$ , which we approximated with  $f$ , we can consider the effect of a further reduction in samples. The mean integrated square error between  $\hat{f}$  and  $f$  can be approximated ([154]):

$$\epsilon = \frac{1}{4}\sigma^4 \int (\Delta \hat{f}(\mathbf{x}))^2 d\mathbf{x} + \frac{1}{n(\sigma\sqrt{2})^d},$$

where the first term can be thought of as a systematic error resulting from kernel density estimation in general and the second term as a random error based on sampling. There is an inverse linear relation between the random error and the number of samples, and for high dimensions the systematic error dominates. As  $\hat{f}$  is unknown, the error cannot be computed, but it can still be minimized with respect to  $\sigma$  ([154]):

$$\sigma_{opt}^{d+4} = d \left( \int (\Delta \hat{f}(\mathbf{x}))^2 d\mathbf{x} \right)^{-1} \frac{1}{n(\sigma\sqrt{2})^d}.$$

Substituted back into the original formula gives a relation between the total error and the number of samples:  $\epsilon(n) \sim n^{4/(4+d)}$ . This formula can be used to determine the minimum number of samples  $m$  that does not increase the error by more than  $\delta$ :  $m/n \geq \delta^{-d/4-1}$ .

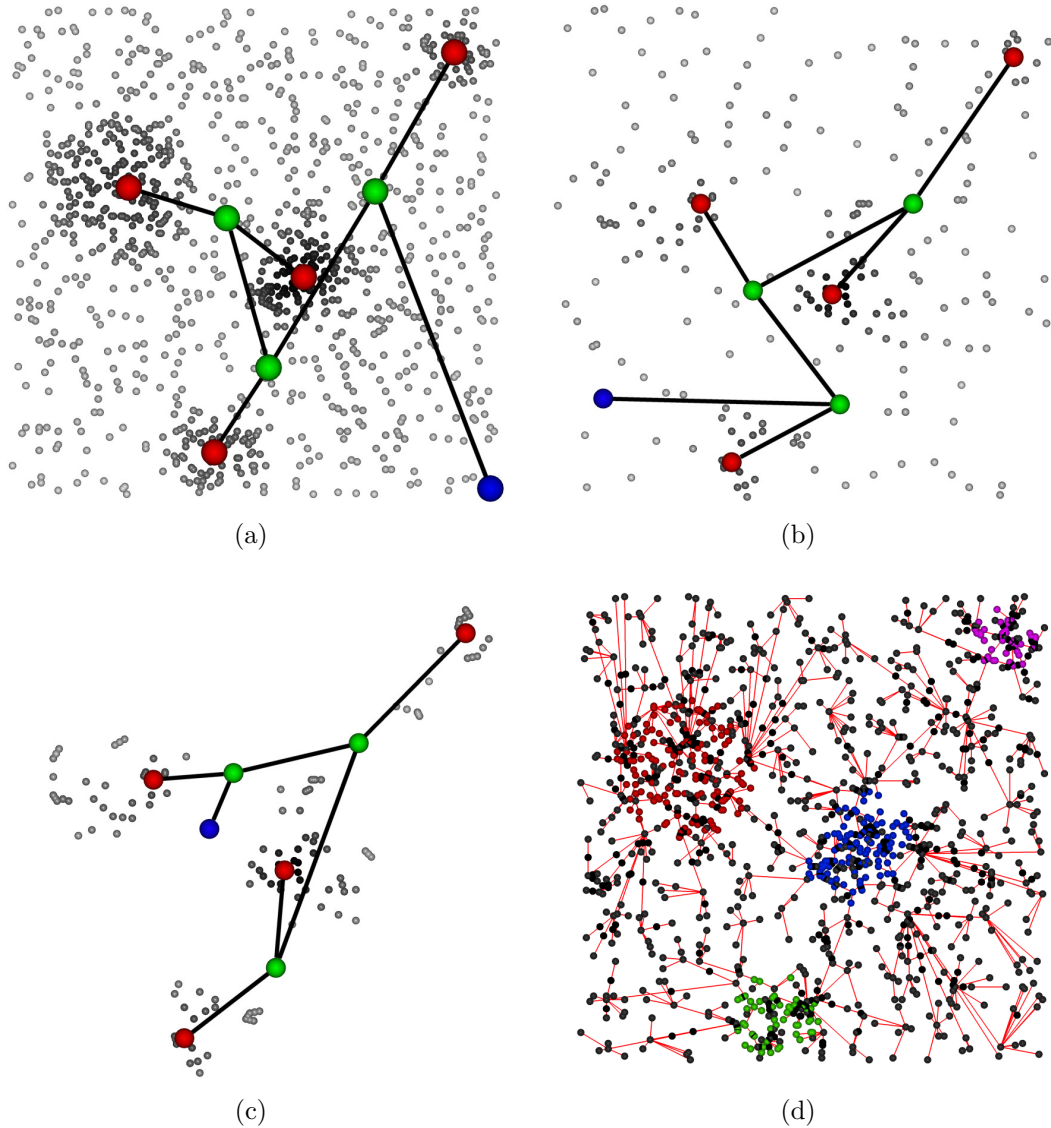


Figure 3.7: Comparison of different sampling strategies based on an artificial 2-D data set: (a) The complete data set. (b) The remaining points after randomly sampling 20% of the input points. (c) The remaining points after additionally sampling only those points that have a density greater than 20% of the maximum density. (d) The relative neighborhood graph after reinsertion of all non-samples.

We use both random sampling as it is very fast and indiscriminate as well as density-based sampling to remove samples with a density lower than a certain threshold. The threshold might be a fixed percentage of the maximum density or a value determined automatically. Figure 3.7 provides an example for both sampling strategies. Figure 3.7b shows the result of randomly sampling 20% of the artificial 2-D point set in Figure 3.7a, along with the density function's merge tree based on the relative neighborhood graph. In Figure 3.7c, an additional density-based sampling step further removes samples with a density lower than 20% of the maximum density.



The persistence diagram in Figure 3.6b illustrates the topological effects for this scenario. Because the maximum density typically changes after sampling, density values are normalized in the diagram. The relevant observation is that all four clusters are detected reliably after random sampling (green), after density-based sampling (purple), and after applying both (blue). The four clusters can also be isolated from noise with a 20% persistence threshold. Note that density-based sampling does not change the maxima. This ensures that prominent features can still be compared in the reduced point set. However, since scattered and vague features in low-dense areas could be eliminated by sampling, this step is virtually related to simplification. Moreover, for noisy data, density-based sampling can increase the persistence of those features that are surrounded by noise. As a consequence, sampling can break subcluster relationships and overemphasize a region's persistence because saddles can be lower than before sampling was applied. Therefore, sampling with high thresholds should be applied with caution as it decreases the signal-noise ratio and can eliminate existent structure or distort relationships among the features.

Although sampling accelerates the topological analysis and still preserves the main features, the final clustering description is incomplete. This is because non-sampled points are excluded from the remaining analysis process once they were discarded. If the analyst needs a visualization of the complete data set later on, non-sampled data points have to be reinserted into the neighborhood graph prior to the merge tree computation. To reverse sampling if both strategies were applied, non-samples are inserted in the inverse order of their removal. Let  $S$  denote the set of sampled points and  $\bar{S}$  the set of non-samples. We approximate the neighborhood for each point  $\bar{p} \in \bar{S}$  by graph edges to its sampled neighbors. Depending on the used neighborhood graph,  $\bar{p}$  may have several valid neighbors. However, because the reinsertion step is not intended to improve the approximation of the density function, but only aims to connect non-samples to their corresponding regions, we add a single edge from  $\bar{p}$  to its nearest neighbor  $NN(\bar{p}) \in S$  to the graph of  $S$ . It is also necessary to test whether the newly inserted edge needs to be upsampled to avoid that noise points are related to nearby clusters that contains the nearest neighbors. A consequence of connecting  $\bar{p}$  with only one edge to the neighborhood graph of  $S$  are *star-like* structures that develop at some sampled points  $p \in S$  (cf. Figure 3.7d). If the non-samples are reinserted for both sampling strategies, these structures develop recursively. Again, the purpose of reinserting non-samples is to ensure that the merge tree contains all input data points on its superarcs. Therefore, if the analyst is only interested in a structural overview rather than visualizing the data points, skipping the optional reinsertion provides a faster structural preview.



Table 3.1: Parameters of the algorithmic core of the topological analysis.

Part of the Algorithm	Requirement	Type	Number
Similarity measure	mandatory	metric	1
Sampling	optional	numeric	2
Filter kernel	mandatory	kernel type	1
Neighborhood graph	mandatory	graph type	1
Upsampling	recommended	boolean	1
Reinsertion	optional	boolean	2
Simplification	optional	numeric	3

### 3.4 Parameters and Runtime

While the algorithmic core of the topological analysis requires only a few mandatory parameters, a couple of additional parameters, of which some are optional, are needed to make the algorithm applicable to more complex data and to control and simplify the accuracy of the topological abstraction. Although the topological approach was designed to work on raw point data, there are still some variable parts that are not apparent at first sight, but whose adjustment could be critical in particular fields of application. This section summarizes all mandatory and optional parameters. Table 3.1 provides an overview of all available parameters, sorted by the order in which they are needed.

**Similarity measure.** Although it defaults to the Euclidean distance, technically speaking, the similarity measure is interchangeable. Using another metric adapted for higher dimensions can be beneficial, e.g., to overcome problems with the Euclidean distance very high-dimensional spaces. The default metric could even be replaced by an application-driven metric that works directly on the domain entities. For example, instead of representing text documents by vectors and using their distances or angles to describe similarity, they could also be compared based on their original content with automatic language processing and sophisticated text mining approaches. In this context, however, it should be mentioned that leaving the vector-based information space could lead to problems with respect to upsampling, which requires to compare real objects with samples at midpoints that do not have a valid representative in the data. Particularly for the text example, it is not immediate how to define a (virtual) document that resides in the middle between two real documents.

**Sampling.** This is an optional step to increase the algorithm’s scalability by working on a representative subset of very similar clustering structure. This phase has two parameters: a threshold for random sampling to define a percentage of the input data that should be kept, and a threshold for density-based sampling to define a percentage of the maximum density that a particular point has to exceed.

**Filter kernel.** The filter kernel used for density estimation is also interchangeable. For the sake of simplicity, it defaults to the Gaussian filter because this one is used frequently for density-based clustering. However, the Gaussian filter is isotropic in that it has the same variance in every dimension, i.e. the region of influence is actually a hypersphere. The estimated density function is also subject to the filter radius  $\sigma$  and requires to evaluate Euclidean distances. As it is conceivable that neither the assumptions about the underlying data, nor data size or dimensionality justify using this kernel type, replacing it by another kernel could be reasonable. Still, it is important to understand that basically every structural insight taken from the topological analysis depends on this parameter and its window width  $\sigma$ . While a too large  $\sigma$  combines actually separated regions, a too small  $\sigma$  splits clusters and can even assign every data point to its own cluster. Because finding a suitable  $\sigma$  is vital, Chapter 5.2.1 presents topology-based strategies and an interactive widget to determine the window width of this parameter effectively.

**Neighborhood graph.** The parameter type is a neighborhood graph to approximate each vertex’ neighborhood. Choosing between the Gabriel graph (GG), the relative neighborhood graph (RNG) or the Euclidean minimum spanning tree (EMST) is a trade-off between runtime and accuracy of the clustering description. Not only are the RNG and the GG worse in runtime complexity, they also contain substantially more edges that require additional upsampling. As a rule of thumb, with increasing data size and dimensionality, less complex neighborhood graphs should be used. In our experiments it turned out that while the GG can handle up to around 30 000 points in some dozens of dimensions, for more complex data, the RNG or even the EMST are recommended. The effect of using sparser neighborhood graphs is increasing structural noise that can be countered with topological simplification. Instead of using the upsampled versions of these graphs, other graphs and (up)sampling schemes are also imaginable as long as they match the target application, i.e. cluster analysis, and can be processed by the merge tree algorithm, which requires connected graphs by default.

**Upsampling.** Strictly speaking, upsampling is rather optional than mandatory. Because omitting this part can heavily distort the clustering description, it is actually *recommended*. Upsampling aims to improve the accuracy of the approximated density

function by detecting missing saddles of zero density that are required to confirm cluster separation. It is important to understand that without upsampling, the number of saddles does not necessarily change. Because a cluster typically exhibits less density at its border than in its center, without upsampling, saddles would likely only “move” from the empty space between the clusters to one of their borders. Although separation is likely missed in this case, the number of clusters found is still the same. For noisy data, upsampling could also be omitted because noise points between the clusters could act as saddle points of low density. Taking such considerations into account, skipping the upsampling step can accelerate the analysis. However, if the semantic and the structure of the data is largely unknown, upsampling is recommended to avoid misleading insights about the data.

**Reinsertion.** Reinserting non-samples is optional and primarily ensures that the merge tree contains all input data points. It requires two boolean parameters to define whether randomly skipped points and low-density points should be reinserted into the neighborhood graph. Because sampling aims to work on a representative subset of the input data, skipping the reinsertion can accelerate the analysis if the focus is primarily on clustering structure rather than on analyzing individual points.

**Topological simplification.** Although the presence of noise might be an interesting insight in some applications, in general, it is recommended to remove small fluctuations of the density function to obtain a clear and precise clustering description. Still, topological simplification is optional and requires up to three parameters to define minimum thresholds for a dense region’s persistence, size, and stability. Note that vague features could be missed if one of these thresholds is too large; especially if the clustering is inhomogeneous, e.g. if some clusters are scattered and thus less persistent and stable, but large in size. These aspects will be discussed in the conclusion at the end of this chapter (cf. Chapter 3.6).

### Parameter changes

As already mentioned in Chapter 3.2 about the basic algorithm, the algorithmic core of the topological analysis follows a straightforward modular design. This implies that parameter changes usually affect subsequent parts of the algorithm. Because changing parameters will play an important role during the visual analysis, an efficient reuse of intermediate computational results is important to facilitate fluent and interactive exploration later on (cf. Chapter 5). Fortunately, the most time-consuming parts will not be changed frequently during the analysis. This primarily holds for computing the distance matrix, which is used by several sub-routines, but also for sampling and the choice of the neighborhood graph. Once these parameters

are adjusted, subsequent parts like the density estimation and upsampling can reuse the pre-constructed neighborhood graph. A parameter that likely requires repeated adjustment is the filter radius  $\sigma$ . As will be explained in more detail in Chapter 5.2.1, the analyst typically has to run the topological analysis for different values of  $\sigma$  in order to detect the clustering suitably. According to the parameter order, as summarized in Table 3.1, changing  $\sigma$  requires to repeat upsampling, the merge tree construction and simplification. Note that the situation changes if density-based sampling is part of the analysis. In this case, changing  $\sigma$  also affects the neighborhood graph construction and the reinsertion of non-samples. The most frequently changed parameters are probably the simplification thresholds. Because simplifying the merge tree is fast and actually the last sub-routine, changing these thresholds is not a runtime bottle-neck and hardly affects the overall runtime. Of course, changing any of the above parameters also updates the merge tree visualization later on (cf. Chapter 4).

### 3.4.1 Runtime Complexity

Although the sub-routines of the algorithmic pipeline have a defined limiting behavior with respect to the input data's size and dimensionality, the expected runtime of a particular execution of the topological analysis also depends crucially on some factors unrelated to the data size. The three most important factors are the data structure itself, its dimensionality, and random factors, e.g., during random sampling. The data structure, i.e., the number of (sub)clusters and their hierarchy, affects the complexity of the topological description which, in turn, affects those steps that work on the merge tree, e.g., the simplification, which depends on the number of leaf nodes. The data dimensionality affects the runtime of the algorithm in that the number of neighbors typically increases with each additional dimension. This affects those operations working on the neighborhood graph edges, like upsampling or the reinsertion of non-samples. Finally, random factors affect the expected runtime because even equally-sized samples of the same data can lead to very different neighborhood graphs (and merge trees). Taking these considerations into account, it is difficult to quantify the expected costs for an arbitrary data set because the expected runtime is not only restricted by the data size. For example, it is easily possible that a specific number of points is faster to analyze in a high-dimensional space than in a two-dimensional space.

Nevertheless, the asymptotic runtime complexity results from the sum of the runtime complexities of each of the individual sub-routines. Taking only the most relevant sub-routines into account and assuming no implementation-specific opti-

mizations, these are the following steps (with  $n$  being the data's size and  $d$  being the dimensionality): making the initial data set unique requires a sort, i.e., linearithmic time  $O(n \log n)$ , and a sweep in linear time  $O(n)$  to remove adjacent duplicates; random sampling requires a shuffle of the input data in linear time; calculating the distance matrix with  $(n^2 - n)/2$  entries takes quadratic time  $O(dn^2)$  in total; estimating the density function in quadratic time; density-based sampling requires a sweep in linear time; constructing the neighborhood graph naively for arbitrary dimensions requires cubic time for the Gabriel graph, cubic time for the relative neighborhood graph, and quadratic time for the Euclidean minimum spanning tree; performing the upsampling takes  $O(en')$ , where  $e$  is the number of graph edges and  $n'$  is the number of sampled points; density-based and random-based reinsertion each in quadratic time, i.e.,  $O(2n'n'')$  to determine in two sweeps for each of the non-samples  $n'' = n - n'$  the nearest neighbored sample and the density of the mid-point of this edge; constructing the merge tree in  $O(\hat{n} \log \hat{n} + N + M\alpha(M))$  [24] where  $\hat{n}$  is the number of neighborhood graph vertices,  $N$  is the number of edges and  $M$  is the number of union-find merges performed; and simplifying the merge tree with an asymptotic cost of mainly  $O(t \log t)$  [25], where  $t$  is the original size of the merge tree.

## 3.5 Examples and Results

In this section, we apply the algorithm to various example data sets introduced in Appendix A. To this end, we use several parameter settings and consider runtime and memory aspects of the analysis pipeline. Note that strategies to set up parameters efficiently as well as visualizing clustering structure and exploring the data is not part of this example section. These issues will be addressed in Chapter 4 and Chapter 5.

The computational platform used for all experiments is a machine with two 2.6 GHz quad-core processors and 32 GB of random access memory. With the exception of the merge tree algorithm, all sub-routines run concurrently whenever possible.

### 3.5.1 Artificial 2-D Data Set

For demonstration purposes, the first example data set is two-dimensional. This should achieve a better understanding of the topological approach because we can illustrate relationships and intermediate results. Of course, the observations and explanations made for this example also hold for higher dimensional data. The example data consists of a noisy 2-D point cloud with clusters of varying shape,

compactness and size. It is illustrated in Figure 3.8a and explained in more detail in Appendix A.1.

Because the data is only two-dimensional, its density function can be imagined as a height field. As indicated in Figure 3.8c, the data can be extended to 3-D by assigning each data point a height value according to its density as determined by density estimation. This causes regions of higher density to stand out as separated hills. To extend the points to a domain suitable for topological analysis, we use the Delaunay triangulation and obtain the terrain shown in Figure 3.8d. The equally distributed isolines on the hills represent superlevel set borders to accentuate height information. Thinking of a single isoline that decreases in its height, it first appears on a peak, merges with another isoline in a valley, and, after it comprised the whole landscape, it (typically) vanishes at zero height. The objective of the topological analysis is to capture these events for all isolines of the landscape and to summarize their evolution in terms of critical points. Figure 3.8e shows all critical points of the height field. It contains a maximum (red) on each peak, a saddle (green) in each valley where superlevel sets merge, and one global minimum (blue) to represent the point of lowest density. The merge tree, which is not shown in the terrains, connects these critical points according to the merging behavior of neighbored regions. Because noise in the data easily complicates the structural description and visual complexity, we eliminate insignificant features with topological simplification. Figure 3.8f shows the critical points that remain after the simplification. There is one density maximum per cluster and saddles of lower density indicate subclusters or cluster separation. The merge for this data set is shown in Figure 3.8b.

A major advantage of the density-based approach is its capability to detect clusters of arbitrary shape, as long as dense regions are separated by regions of lower density. In terms of the density function's topology this means saddle-maximum pairs evolve inside a cluster and take its form until they reach lower density at its border (cf. Figure 3.9a). However, this behavior depends on the selected filter radius  $\sigma$ . While  $\sigma$  must be sufficiently small to discriminate nearby clusters, a small filter radius also increases topological noise and can split those clusters that are larger than  $\sigma$  itself. Although evolving saddle-maximum pairs are the key to find clusters of arbitrary shape, resulting topological noise typically needs to be reduced prior to further exploration (cf. Figure 3.9b).

Upsampling is necessary to detect missing saddles of the density function. However, as illustrated in Figure 3.9b, in case of noisy data, using the noise points as saddles could also suffice. Although a lower saddle might be found at an upsampled position in the noisy areas, its density is unlikely much lower. That is, the

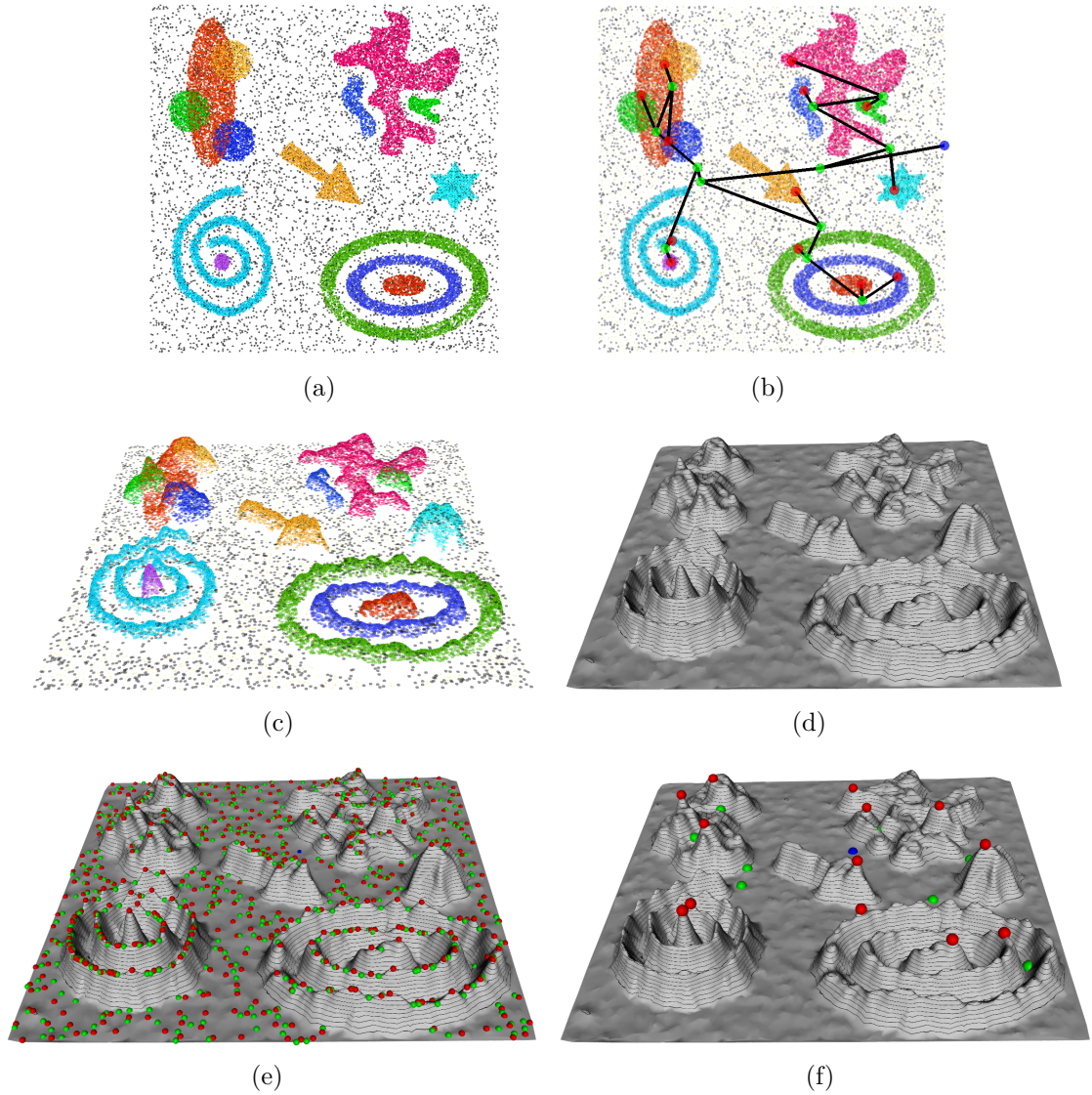


Figure 3.8: Artificial 2-D data set (cf. Appendix A.1) imagined as a height field: (a) Noisy point cloud with clusters of different shape, size, and compactness. Some clusters are intertwined, others contain each other. Colors highlight the relation between clusters and classes; noise points are colored in black. (b) The final merge tree accurately captures the clustering structure. There is one density maximum (red) per cluster and cluster hierarchy and separation are described by connecting saddles (green). (c) The topological analysis can be imagined as analyzing the implicit height field defined by the points' densities. (d) Landscape-like representation of the height field that results from rendering the (2-D) Delaunay triangulation with augmented isolines to indicate some superlevel sets. (e) The large number of critical points (red=maximum, green=saddle, blue=global minimum) reflects a noisy density function. (f) These fluctuations are countered with topological simplification. Given suitable simplification thresholds, one density maximum per (sub)cluster remains. Remaining saddles and their densities indicate cluster hierarchy, separation, or ambient noise between the clusters.



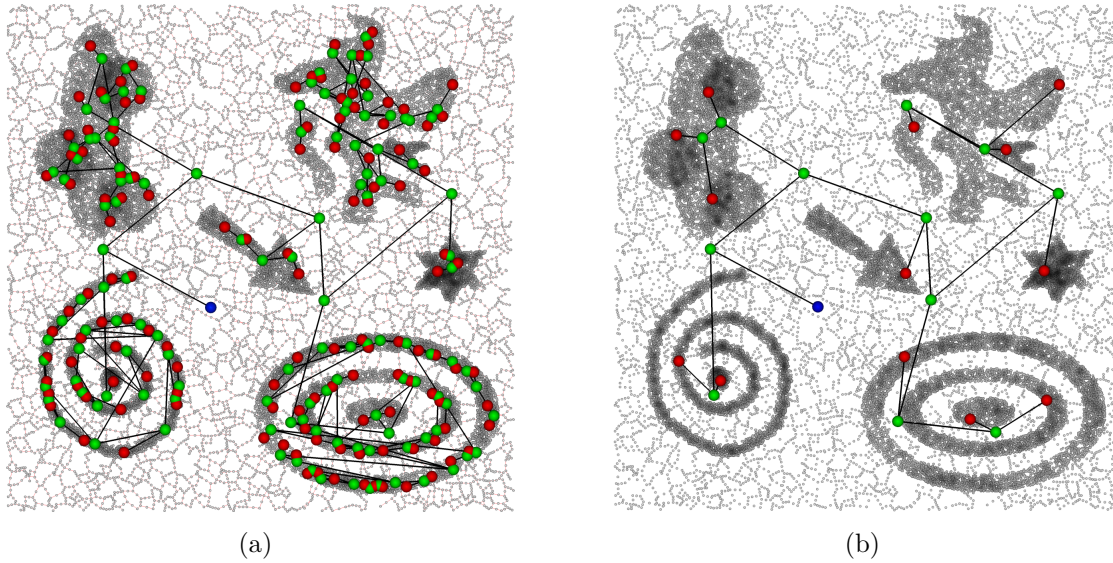


Figure 3.9: Artificial 2-D data set: (a) Height graph consisting of the (upsampled) RNG edges (red) and with vertices colored according to their density (dark=dense). The unsimplified merge tree is augmented. Depending on the point distribution and the selected filter radius  $\sigma$ , arbitrarily shaped clusters are detected as maximum-saddle pairs that evolve inside the clusters until they reach lower density. (b) The large number of insignificant saddle-maximum pairs is reduced with topological simplification. After simplification, only one density maximum per cluster remains.

overall clustering would still be captured by the merge tree and the time-intensive upsampling could be omitted. Nevertheless, because separation could be missed if saddles are non-zero, it is not recommended to rely on noise in unknown data sets. To demonstrate the importance of upsampling, Figure 3.10a shows the 2-D data set after noise removal together with the upsampled Gabriel graph edges and the critical points of the density function. Except for the subcluster hierarchy in the top left corner, all saddles and the global minimum are located on upsampled vertices between the clusters. Note that not all upsampled positions act as saddles between the dense regions and that most of them become regular nodes once two regions were identified to be separated. Moreover, upsampled vertices are not stored as implicit regular nodes because they do not represent real data points. Figure 3.10b shows the same scenario without upsampling. Because there are no virtual upsamples in the height graph anymore, only real data points of lowest density can act as the saddles. This implies that saddles are located on the cluster borders. From a clustering point of view, these non-zero saddles can only describe one big cluster with several sub-structures and, because saddle densities are now higher, the persistence of these features also decreases. This is why some density maxima vanish in Figure 3.10b because they are now considered noise and are removed by topological simplification



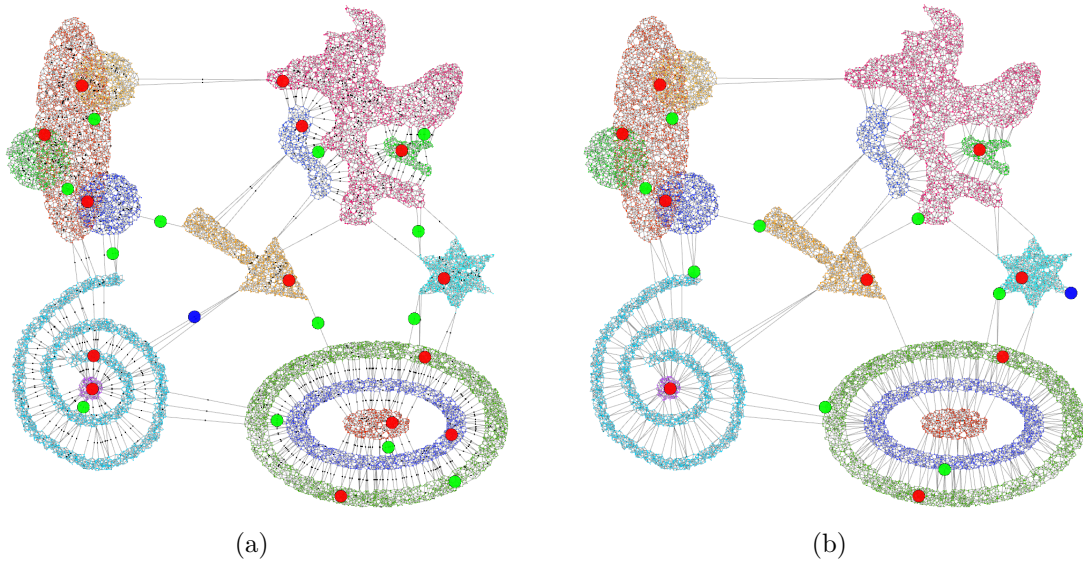


Figure 3.10: The importance of upsampling demonstrated with the artificial 2-D data set: (a) After noise removal, the critical points of the density function are located on the upsampled midpoints (black dots) of the Gabriel graph edges. Cluster separation is detected reliably. (b) Without upsampling, the critical points reside on the cluster borders, which is why cluster separation is missed. From a clustering point of view, the non-zero saddles densities can only reflect one big cluster with several sub-structures.

using the same threshold like in Figure 3.10a. While reducing the simplification thresholds would restore the previously found features, their actual separation would still be missed.

The runtime of the topological analysis depends on the individual choice of parameters. Table 3.2 provides some statistics for several parameter settings. We distinguish primarily between the neighborhood graph and different sampling strategies. In all configurations, the filter radius is fixed to  $\sigma = 30.0$  (the data domain has an extent of 800x800 pixels) and the simplification threshold is fixed to 10% of the maximum persistence—which is typically a good value to remove noise. Note that simplifying by another region property or using other thresholds would slightly change the runtimes for the simplification step. We use the Euclidean minimum spanning tree (EMST), the relative neighborhood graph (RNG) and the Gabriel graph (GG). The used sampling strategies include running the topological analysis (i) without sampling, (ii) with sampling only 20% randomly and density-based, and (iii) with sampling, but without reinserting the non-samples afterwards. The results

<sup>1</sup>Total times include preprocessing like computing a distance matrix or removing duplicates. Duplicates still contribute to the density function and are added correctly to the merge tree.

<sup>2</sup>Randomness causes the graph sizes and the runtimes of subsequent phases to vary a little.

Table 3.2: Statistics for the artificial 2-D data set (times in seconds). The filter radius is fixed to  $\sigma = 30.0$  (pixels) and the simplification threshold is  $thresh_{pers} = 10\%$  of the maximum persistence.

neighborhood graph	number of edges	time for graph	time for upsampling	time for reinsertion	time for merge tree	time for simplification	total time <sup>1</sup>
without sampling							
EMST	31 833	27.37	1.8	-	0.21	1.03	<b>36.07</b>
RNG	44 942	96.01	2.87	-	0.28	0.46	<b>105.59</b>
GG	77 077	98.14	4.94	-	0.91	0.22	<b>111.05</b>
with sampling (20% random, 20% density)							
EMST	5 365	0.57	0.34	11.34	0.29	5.38	<b>18.42</b>
RNG	6 540	2.01	0.34	12.66	0.28	3.69	<b>19.47</b>
GG	10 046	1.80	0.47	12.95	0.27	3.46	<b>19.44</b>
with sampling (20% random, 20% density) without reinsertion <sup>2</sup>							
EMST	5 177	0.51	0.28	-	0.02	0.04	<b>1.33</b>
RNG	6 287	1.73	0.33	-	0.03	0.04	<b>2.59</b>
GG	10 586	2.24	0.54	-	0.04	0.02	<b>3.35</b>

clearly reveal that the total times are dominated by the times required to construct the neighborhood graph and that there are huge differences in the total time depending on the sampling strategy. Because the Gabriel graph produces the most edges it also requires the most time to compute and consumes the highest amount of memory. Furthermore, subsequent upsampling of the graph increases linearly with the number of edges. Compared to the graph constructing and upsampling, creating and simplifying the merge tree is generally fast and takes only a small part of the total time. The runtime bottleneck, hence, is located in the first part of the analysis. This is also the reason why the sampling strategies aim to reduce the costs of these phases. The table reveals that sampling reduces the total time almost linear to the amount of sampling applied. The fastest result and lowest memory consumption can be obtained by applying sampling, but skipping the reinsertion. This approach not only minimizes the runtime and memory consumption, working on a smaller subset also accelerates the merge tree construction and the simplification step.

Changing the filter radius  $\sigma$  also has an effect on the runtime and memory consumption of the topological analysis. Table 3.3 summarizes statistics for different filter radii; using the Gabriel graph and fixing the simplification threshold to a

Table 3.3: Statistics for the artificial 2-D data set (times in seconds). The neighborhood graph type is fixed to the Gabriel graph and the simplification threshold is  $thresh_{\text{pers}} = 10\%$  of the maximum persistence.

filter radius $\sigma$	time for upsam- pling	number of upsam- ples	time for merge tree	number of max- ima	time for sim- plifica- tion	<b>total time</b>
0.1	0.14	77 077	35.39	31 834	0.0	<b>137.87</b>
10.0	7.81	13 049	0.33	3 884	0.53	<b>106.93</b>
30.0	4.64	2 109	0.91	543	0.22	<b>105.85</b>
60.0	5.10	623	1.30	204	0.09	<b>108.65</b>
150.0	10.11	171	2.06	104	0.02	<b>122.58</b>
500.0	32.54	62	1.66	99	0.13	<b>139.49</b>

constant of 10% of the maximum persistence. The filter radius  $\sigma$  varies between the smallest possible value, which is  $\sigma < 0.5$  for two neighbored pixels, and a too large value that cannot separate all clusters anymore. For an increasing  $\sigma$ , the table reveals an inverse relation between the time required for upsampling and the number of upsamples found. This is due to the cut-off radius, which allows us to skip the evaluation of a midpoint's density once it cannot increase anymore because all other points are too far away. As a consequence, for smaller filter radii, less points are relevant for the density estimation of a single evaluation. For  $\sigma < 0.5$ , the filter radius is even smaller than (half of) the shortest possible edge length in this 2-D example data set. This is why the density estimation of a midpoint can be skipped entirely, which also holds in general if the filter radius is smaller than half of the length of the currently processed graph edge. For an increasing filter radius, the effect of this optimization vanishes. Furthermore, the number of upsamples found also decreases with an increasing filter radius because the midpoint densities increase as well and would turn into (uncaptured) regular nodes or local maxima of the density function. The worst-case of the number of required upsamples in this example scenario is for  $\sigma < 0.5$ ; when every graph edge hosts a zero-density saddle. This is also the worst runtime for the merge tree construction, which is otherwise rather constant and independent from the filter radius.

### 3.5.2 High-Dimensional Data Sets

The observations made for the artificial 2-D data set also apply for data of higher dimensionalities. There are only quantitative differences like a more expensive distance calculation in  $O(d)$  in  $\mathbb{R}^d$ , or an increasing number of neighborhood graph edges because high-dimensional points generally also have an increasing number of “neighbors” with each additional dimension. These properties affect the runtime and memory consumption of the neighborhood graph and also the costs of subsequent steps like upsampling and computing and simplifying the merge tree. However, beyond the limitations of the similarity measure and the chosen filter kernel, the algorithm itself does not make any restrictions to the data’s dimensionality. Although the regions of influence used for the neighborhood graph construction also become high-dimensional, i.e. a hyper-sphere for the Gabriel graph and a hyper-lune for the relative neighborhood graph, the inside-region test is still implemented only with simple distance evaluations.

#### Real-World Data Sets

To demonstrate the versatility of the topological approach and its independence from the underlying application domain, we apply it to several high-dimensional data sets that are explained in more detail in Appendix A. Table 3.4 summarizes the results for different neighborhood graphs. The values for the filter radius  $\sigma$  used to create the density function are considered *suitable* to detect the point cloud’s clustering structure. Finding these suitable parameters is not part of this result section, but will be addressed in Chapter 5 about interactive visual analysis.

The total times for all three graph types indicate that the algorithm scales for some thousands of points in higher dimensions, even without prior sampling. However, for the most complex data set, there are significant differences in the size and runtime of the neighborhood graphs. While the number of EMST edges is always below the number of points, the number of GG edges can be a multiple of the data size. As can be seen for the Isolet data set, this can even be some orders of magnitude and reflects the high edge degree of each graph vertex in high-dimensional spaces. This also underlines the importance of using sparser neighborhood graphs to mitigate time-consuming upsampling. Upsampling also gets increasingly inefficient because the number of edges between two regions typically increases with every additional dimension, while only one upsample is required to confirm cluster separation. Note that the number of EMST edges is not exactly  $n - 1$  if the data contains duplicates, i.e. points with identical coordinates. In a preprocessing step, we remove duplicates and increase the multiplicity of the one remaining point at this position to avoid

Table 3.4: Statistics for the real-world data sets in Appendix A (times in seconds). The filter radii were determined manually with the strategies introduced in Chapter 5.

property	Iris (A.6)	Medline (A.8)	Italian oils (A.5)	Reuters (A.7)	Segmen- tation (A.3)	Isolet (A.4)
points	150	1 250	572	800	2 310	7 797
dimensions	4	4	8	9	19	25
filter radius $\sigma$	0.8	0.01	22	0.06	30.0	0.7
EMST	edges	1 245	570	633	2 085	7 796
	graph time	< 0.01	0.02	< 0.01	< 0.01	0.06
	upsamples	12	178	310	55	319
	<b>total time</b>	<b>&lt; 0.01</b>	<b>0.04</b>	<b>0.01</b>	<b>0.02</b>	<b>0.17</b>
RNG	edges	1 816	833	828	2 786	15 995
	graph time	< 0.01	0.08	0.01	0.01	0.22
	upsamples	22	282	527	76	719
	<b>total time</b>	<b>&lt; 0.01</b>	<b>0.11</b>	<b>0.02</b>	<b>0.04</b>	<b>0.36</b>
GG	edges	5 555	4 361	3 975	8 066	766 576
	graph time	< 0.01	0.08	0.01	0.02	0.24
	upsamples	124	1 118	3 553	529	3 704
	<b>total time</b>	<b>&lt; 0.01</b>	<b>0.13</b>	<b>0.05</b>	<b>0.05</b>	<b>0.42</b>

problems with the neighborhood graph. Increasing a point’s multiplicity ensures proper density estimation and a correct merge tree with all input data points.

### Artificial 100-D Data Set

This artificial data set consists of 127 995 points in 100 dimensions and is explained in more detail in Appendix A.2. The complexity of this data set makes it useful to investigate the runtime behavior of the presented algorithm and to demonstrate both the strengths and the limitations of the topological approach.

Table 3.5 shows the results of analyzing the data set with different neighborhood graphs and with different sampling strategies. During these experiments, we fix the filter radius to  $\sigma = 650.0$  and use a 10% threshold for density-based and random sampling. The used sampling strategies are: no reinsertion of non-samples, reinsertion of non-samples discarded by density-based sampling, reinsertion of non-samples discarded by random sampling, and reinsertion of all non-samples. The total

times provided by the table do not include the time required to pre-calculate the distance matrix, which takes around 25 seconds with our implementation.

The table reveals some important aspects regarding the algorithm’s scalability. The first observation concerns the relation between the number of points and their dimensionality: In higher dimensions, the number of neighborhood graph edges typically increases quickly—and so do the computational costs. This implies that in lower dimensions, where fewer neighborhood graph edges exist, more input data points are manageable without a significant increase in runtime or memory consumption. To reduce the overall costs of the topological analysis, we can either reduce the data points or the number of graph edges. These strategies are implemented with prior sampling and by using sparser neighborhood graphs. For the 100-D example, the quickest possible result can be obtained by using the Euclidean minimum spanning tree (EMST) and without reinserting non-samples. In this case, constructing the neighborhood graph and computing the merge tree takes around 27 seconds (including around 25 seconds for the distance matrix). Because density-based sampling without reinsertion does not lead to a suitable representation of the input data set, reinserting the non-samples from density-based sampling takes around 21 seconds and slightly decelerates the merge tree computation, leading to a total of around 48 seconds. A noticeable decrease in runtime can be observed for the reinsertion of the 90% input points that were discarded by random sampling. This process takes around 7 minutes and decelerates the merge tree computation to around 27 seconds. Finally, computing the merge tree after the reinsertion of all non-samples takes around 17 minutes in total for the EMST.

The algorithm takes even longer for more complex neighborhood graphs. While the relative neighborhood graph (RNG) contains around 23 000 edges and takes approximately six seconds, the Gabriel graph (GG) already contains around 13 million edges and takes approximately 2.5 minutes. Moreover, upsampling becomes a critical factor in high-dimensional spaces. While upsampling takes only one second for the RNG, it takes around 17 minutes on the GG and also doubles the number of graph nodes. Note that most of these upsampled positions will be (redundant) regular nodes in the merge tree later on and only result from the large number of neighbors in high-dimensional spaces. Table 3.5 also reveals that even though using the RNG is not much slower than using the EMST, there are critical drops in runtime for the Gabriel graph. Computing the merge tree based on the GG already takes around 2 minutes and reinserting the non-samples discarded by density-based sampling takes around one hour. Applying the analysis to the complete input data

Table 3.5: Statistics for the artificial 100-D data set (times in seconds). The filter radius is fixed to  $\sigma = 650.0$  and sampling is performed with a 10% threshold for both random and density-based sampling.

neighborhood graph			times for reinsertion and merge tree			
type	edges [time]	up- samples [time]	none [merge tree]	density [merge tree]	random [merge tree]	both [merge tree]
EMST	7 830 [1.22]	19 [0.43]	- [0.06]	18.79 [0.24]	432.86 [27.63]	870.19 [156.97]
RNG	23 119 [5.47]	301 [1.27]	- [0.38]	21.71 [0.61]	543.4 [30.15]	803.57 [184.64]
GG	$13.31 \cdot 10^6$ (149.78)	$11.67 \cdot 10^6$ (1 049.85)	- [132.55]	3 650.61 [134.41]	n/a	n/a

set, maybe even without any prior sampling and using the complex Gabriel graph, would certainly take several hours or even days. Therefore, the basic idea of the introduced optimizations is to reduce the data size to some ten-thousand points for around 50-D, or more points for fewer dimensions, and to handle increasing dimensionality with less complex neighborhood graphs.

### 3.6 Conclusion and Discussion

We presented a multiple stage process to approximate the clustering structure of a high-dimensional point cloud. This approximation aims at the study of the data in terms of dense regions, including nesting structure and individual region properties. We introduced optimizations to reduce the runtime of the algorithm: moderate random sampling is applied because it does not change the overall shape of the estimated density function, and therefore preserves its topology. However, since this also increases the impact of noise especially in low-density regions, additional density-based sampling favors regions of high density. Furthermore, instead of using the Delaunay graph, which is inefficient to compute in high-dimensional spaces, sparser graphs can be used to represent neighborhood relations between points. The merge tree’s complexity increases with the “sparseness” of the neighborhood graph, but this can be countered with topological simplification. Still, there is not one neighborhood graph for all cases, as it can be observed that the sparseness may only be increased along with the dimension: while for lower dimensions the Gabriel graph should be used, for medium-dimensional data, it should be sufficient to use the sparser

RNG or even the EMST. To obtain results within seconds or a few minutes, data size is currently limited to approximately 100 000 points in around 100 dimensions. Although bigger data sets could be analyzed by more approximation, eventually, the algorithm's usability for cluster analysis is limited by the curse of dimensionality because it uses a distance-based proximity measure. While this could be countered with an application-driven metric, we opted for an application-independent, multi-purpose approach that works on raw point data. Another solution to overcome problems with distance calculations in very high-dimensional spaces is to project the data to a lower dimensional space that is less affected by the curse of dimensionality. For example, for classified data consisting of  $k$  classes, linear discriminant analysis strives to preserve the clustering structure in an intermediate  $k - 1$ -dimensional space that can then be analyzed topologically.

A limitation of the topological approach is its inability to preserve absolute distances and geometric cluster properties like shape, extent, or local point distribution. However, the density-based notion of clusterings, i.e., clusters are dense regions that are separated by regions of low density, allows us to find clusters without constraining the input data with respect to these properties. Still, some clusterings cannot be captured by this notion. For example, a dense cluster inside a sparse cluster usually leads to only one density maximum, and also close clusters might be combined if their point distribution is anisotropic. The topological analysis of the density function can also not provide information about empty regions in the data set, and, like every density-based approach, the clustering result heavily depends on one parameter: the filter radius  $\sigma$  of the Gaussian kernel. Compared to hierarchical clustering methods, like OPTICS [6], we only consider one segmentation and cannot detect clusters at different density levels or hierarchies. That is, while a large  $\sigma$  is required to detect large clusters, it will combine accumulations of small clusters at the same time. Likewise, a small filter radius  $\sigma$  separates small and nearby clusters, but splits large clusters at the same time. Although topological simplification can mitigate these effects by removing noise caused by using too small filter radii, the clustering found is still only one possible segmentation and is primarily characterized by the adjusted filter radius.

A strength of the topological approach is its capability to detect the clustering and cluster separation in arbitrary dimensions. Moreover, the topological analysis is robust with respect to noise and captures distinct cluster properties, namely persistence, size, and stability, for arbitrarily shaped clusters. By relating these quality measures, cluster properties can be approximated, e.g., whether they are compact or scattered. Note that only by discarding unpreservable geometric properties, we can focus on



key properties to obtain a suitable description of the high-dimensional clustering. Presenting the manifold information provided by the merge tree in a clear and intuitive fashion and guiding the analyst in finding appropriate thresholds to set up important parameters of the topological analysis will be the subject of Chapter 4 and Chapter 5, respectively.



## Chapter 4

# Topological Visualization

The unaugmented merge tree resulting from the topological analysis describes the input point cloud’s clustering structure. Its hierarchy reflects that of the dense regions found in the data and its superarcs reflect quantitative properties like a region’s distinctness, separation, compactness, or size. By storing the underlying data points as implicit regular nodes, the merge tree also relates individual points to their clusters or noisy regions. Based on the point densities, the analyst can evaluate point distributions and the importance of each point for its corresponding feature. The latter can be derived from comparing a point’s density to the density maximum of its cluster to find out whether it is positioned close to the center or at the border.

Presenting this complex information clearly, intuitively and without occlusion artifacts requires an effective visual metaphor to facilitate proper and convenient visual analysis. To achieve this goal, we advance previous work on a terrain-like visualization of scientific data as 3-D *topological landscapes* [169]. This metaphor harnesses the human’s naturally trained ability to read and understand the structure of a terrain and to quantify the importance of individual hills based on their height and extent. We extend this idea for visual cluster analysis to visualize the clustering, including cluster hierarchy and cluster properties, as nested hills of different shape and extent. Furthermore, we augment the landscape with the underlying data points to allow the analyst to relate individual points to features, to annotate them with additional meta-information, and to link them to other views for further local analysis.

This chapter introduces several topology-based landscape visualizations for visual analysis of high-dimensional point clouds. We present variations of the original 3-D topological landscape, which was designed for the more complex contour tree [24], and we introduce a novel 2-D landscape metaphor specifically designed for the merge tree and the targeted clustering application. Finally, we compare advantages and

drawbacks of the landscape variations and we discuss the benefit of topology-based visualization compared to using standard techniques for high-dimensional data, e.g. projections or parallel coordinate plots.

## 4.1 Related Work

**Tree layouts.** Because the merge tree encodes the complete clustering, a traditional tree layout in the plane would apparently suffice to present the clustering structure. There are many tree layouts [67, 75], of which (variations of) dendrograms [69] and Icicle Plots [106] are the most commonly used techniques to analyze structure visually [109]. Tree and graph layouts [139] perform well for rather small and simple data, but these methods present only little information beyond hierarchy and thus provide only a one-sided global perspective. Tree-based layouts have also been developed specifically for topological structures like the contour tree. A prominent example is the 3-D *toporrery* layout [134]. Inspired by the classical design of a mechanical orrery, the hierarchy of stars, planets and moons is replaced with a hierarchy of maxima, minima, and saddles that can be filtered interactively by importance with respect to a given metric. However, because the toporrery is embedded in 3-D, it suffers from general problems like view-dependency and occlusion of geometry for a particular viewing direction. That is, depending on the size and complexity of the tree, the analyst may need to rotate the scene to find an appropriate line of sight. To alleviate occlusion problems when drawing contour trees, Heine et al. [74] propose various aesthetic criteria and present a fast algorithm to visualize the tree in the 2-D plane. Nevertheless, tree layouts are limited regarding their capability to show properties beyond the linkage of nodes. Since one node property is typically represented by the  $y$ -axis or the  $z$ -axis in a 2-D or 3-D layout, respectively, to consider multiple node and edge properties, the layout would have to adhere to several drawing conventions. However, these characteristics could be mutual exclusive. For example, the requirement to minimize occlusions and intersections might contradict the goal to reflect node and edge properties by their extent and breadth. Moreover, using several colors, shapes, and sizes to discriminate nodes and edges can quickly lead to confusion and cluttered diagrams. This is particularly relevant for large trees; like a merge tree describing a clustering with multiple properties per node and edge.

**Topological visualizations.** To provide visual access to the complex information provided by a contour tree, Weber et al. introduced *topological landscapes* [169], an intuitive metaphor for scientific scalar data that utilizes the human’s familiarity

with terrains. Reflecting scalar values as height on the  $z$ -axis, the landscape contains hills and sinks for every maximum and minimum, respectively. A topological landscape, thought of as 2-D function, has the same topology like the input contour tree in the sense that a horizontal plane at an arbitrary height  $\epsilon$  intersects as many hills and sinks as the contour tree has superarcs containing  $\epsilon$ . That is, the terrain has same contour tree (of the height values). Two arc properties can be represented by a hill's/sink's height/depth and its base area in the  $xy$ -plane. Because height information reflects absolute scalar values, the height/depth of a hill/sink denotes a feature's persistence and the base area reflect a feature's volume, which is often approximated by the number of grid vertices in that region. Subsequent work aimed at eliminating limitations in the initial implementation of this metaphor. This primarily concerns its accuracy and usability to compare features. Harvey et al. [72] use a tree-map [153] construction scheme to improve the accuracy of mapping a quality measure to a feature's area in the terrain. However, even though tree-maps always ensure correctly sized and evenly shaped hills, rectangular features can still be hard to compare for very different aspect ratios [105]. Beketayev et al. [11] create a direct correlation between a scalar function and its topological landscape by introducing the notion of geometric proximity into the topological landscapes, reflecting the distance of topological features within the function domain. Demir et al. [40] presented a variation of the original landscape metaphor with an improved layout without the need for (expensive) re-parametrization and supporting dynamic and interactive changes to the terrain to enable focus-and-context style zooming. Takahashi et al. [160] adopted the *Isomap* [162] algorithm and proposed a 3-D arrangement of the input positions that reflects the topology as a tree-like structure. Using an approximated Morse-Smale complex embedded in 2-D space, Gerber et al. [66] propose a method for visual exploration of high-dimensional scalar functions that combines topological and geometric techniques to provide interactive visualizations of discretely sampled scalar fields. Correa et al. introduced *topological spines* [34], a representation of a scalar field that preserves the topological and local geometric structure, including structural cycles that are useful for exposing symmetries in the data.

**Visualizing text data.** The thematic composition of text data represented as high-dimensional vectors (cf. Figure 2.1 on page 9) is often depicted with projections that optimize some general or method-specific criteria. Prominent examples include Sammon's mapping [145], the Text Map Explorer [138] or the Projection Explorer (PEX) [137]. Instead of illustrating text items as points or graphs, more intuitive metaphors show features using heat maps in WEBSOM [94], as landscapes in SPIRE/IN-SPIRE [172], as islands using wavelets in TopicIslands [121], or as height

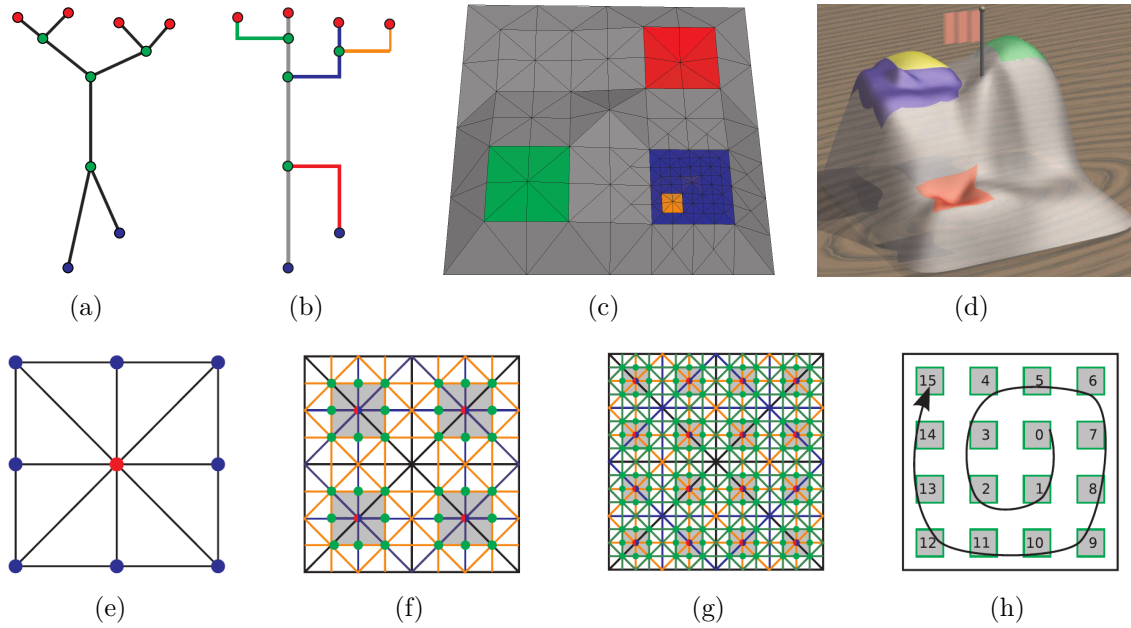


Figure 4.1: Construction of the 3-D topological landscape (images courtesy of [169]): (a) A contour tree. (b) Its branch decomposition (maximizing feature persistence). (c) Topological landscape having the same topology like the contour tree. Hills and sinks represent maxima and minima, respectively. Nested hills preserve the hierarchy of the tree. Colors indicate the relation between branches and their corresponding part of the landscape. (d) Topological landscape after metric-based distortion and smoothing to make it look more natural. (e) Triangle configuration for a single branch. Height information (seen from above) represents the scalar values: the height of the border vertices reflects the branch’s saddle/minimum value, the center vertex reflects the branch’s maximum/minimum. (f)-(g) Subdividing the triangles reproduces the original configuration for child branches. (h) Child branches are positioned on a spiral in decreasing order of their saddle values.

fields in VxInsight [38]. Note that if such metaphors are based on lower-dimensional representations of the data, they still suffer from feature aggregation and thus from information loss.

### 3-D Topological Landscapes Metaphor

Because the original topological landscape metaphor [169] will be extended for cluster analysis, we first explain this visualization in a little more detail.

**Construction.** As illustrated in Figures 4.1a-c, the landscape is constructed based on the contour tree’s branch decomposition. Each branch is represented by a distinct part of the landscape and child branches are placed inside their parent branch’s area to preserve the topology. The landscape is constructed recursively: The root branch is initially described by a simple quad patch divided into eight

triangles (cf. Figure 4.1e). To give this configuration a hill-like shape, the border vertices (marked blue) and the center vertex (marked red) are assigned a height equal to the branch’s minimum and maximum value, respectively. If the root branch has child branches, the eight triangles are subdivided to reproduce the original configurations for each child patch. As indicated in Figure 4.1f, subdividing the triangles four times produces four more child patches. If more child patches are required, further subdivision quadruplicates the number of available patches for every two subdivision steps (cf. Figure 4.1g). Afterwards, in decreasing order sorted by their saddle value, the child branches are positioned on a spiral starting from the center (cf. Figure 4.1h). If a child branch has no children on its own, the vertices of its patch are adjusted according the initial configuration; using the saddle for the border vertices (marked green). Otherwise, the procedure is repeated recursively for the eight triangles of the child patch. The height values of unused patches on the spiral are interpolated linearly between the last child branch’s saddle value and the parent branch’s saddle/minimum value. Vertices of the parent’s triangulation that do not belong to a child patch are also interpolated linearly between the surrounding vertices. To increase the expressive power of the metaphor, an additional smoothing step transforms sharp-edged pyramids into naturally looking hills.

**Metric-based distortion.** The terrain still has a perceptual problem because the sizes of the hills’ base areas (in the  $xy$ -plane) only depend on the complexity of the branch decomposition. That is, they become increasingly smaller for branches deeper in the hierarchy and are, thus, perceived less significant. However, a branch’s significance and its position in the hierarchy are not connected in any way, and the base areas are lacking a precise meaning. Hence, they can be considered a variable to optimize and reflect another branch property. Weber et al. [169] scale it according to the corresponding region’s volume, which is approximated by the number of grid vertices. The authors call this post-process “metric-based distortion” because the triangulation is distorted subsequently to the landscape construction and lets base areas reflect a chosen metric. It is required to assign this property, e.g. the volume, proportionally to all of the branch’s corresponding triangles. For example, in Figure 4.1f, these would be the yellow triangles; including the triangles of unused child patches. Using an iterative process in a coarse-to-fine manner recursively through the hierarchy, all vertex positions are then adjusted without changing their connectivity. Figure 4.1d shows the distorted and smoothed landscape for the input contour tree.

## 4.2 Extended 3-D Topological Landscape

Because the 3-D topological landscape only requires a scalar function's contour tree, it could already be used to visualize a merge tree of a potentially high-dimensional density function. However, if applied to visual cluster analysis, the original landscape design has a disadvantage regarding the metric-based distortion and, because it focuses only on the function's structure, the landscape is not suitable to explore the underlying data points. To alleviate these issues, we will first remediate the problem with incorrect base areas and will then extend the metaphor with additional features to facilitate exploration of clustered high-dimensional point clouds.

### 4.2.1 Modified Metric-based Distortion

While it is easy to compare the base areas of moderately distorted hills, problems arise for nested hills. To evaluate the volume of a hierarchical feature, the base areas of child hills have to be subtracted from the area of the parent. For example, in Figure 4.1 if the base area of the parent hill only consists of the yellow triangles; plus the gray ones of unused child patches. Comparing such perforated base areas of varying shape, however, quickly becomes cumbersome and also inaccurate. Most importantly, by distributing the volume equally to all triangles, the perceived significance of the center hill is affected adversely because an equal distribution does not consider the scalar value distribution. This is a problem for the clustering application, or more precisely for density functions of well-separated clusterings. Imagine a simple scenario with five identical, separated clusters. The saddles of the density function (which are located at upsampled positions) will have zero density and the undistorted landscape of the merge tree's branch decomposition will look like the one shown in Figure 4.2a. If the volume of the root branch, which represents one of the five clusters, is now distributed equally to the gray triangles, the distorted landscape looks like the one shown in Figure 4.2b. Even though the total area of the gray triangles is exactly the same like that of each of the other four colored hills, the base area of the center hill, and thus the volume of its corresponding cluster, is likely perceived smaller or even insignificant compared to the other four clusters of the same size. This representational problem can be solved by considering how the volume is distributed inside the region, i.e. how the scalar values of the grid vertices are distributed on the branches.

Because the saddle values of the density function are vital for the clustering description, a branch's volume distribution is modified as follows: At first, the volume is separated into two partial volumes. The first one,  $v_1$ , represents those grid vertices



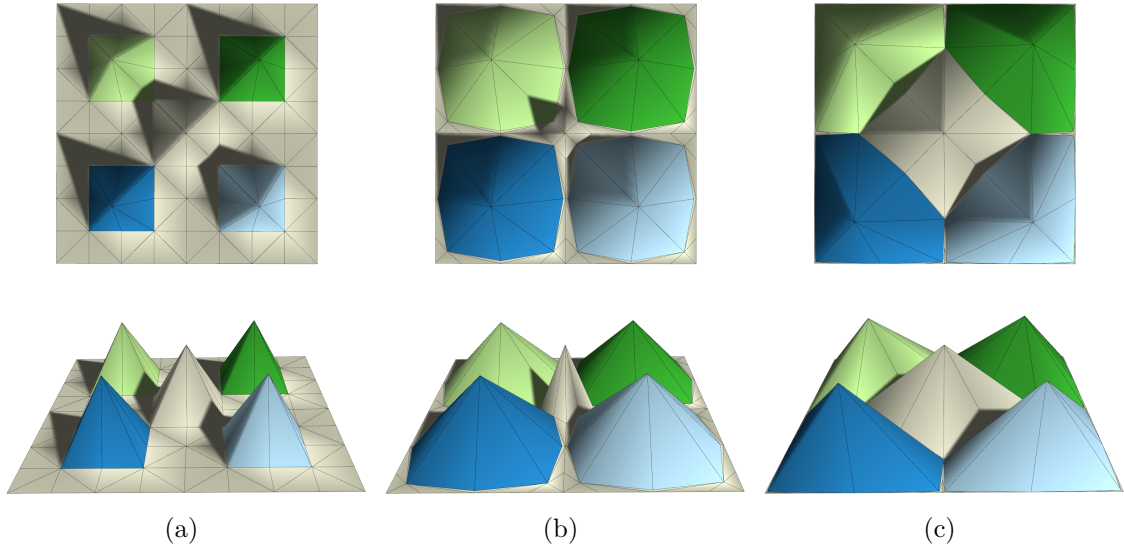


Figure 4.2: Modified metric-based distortion: The scenario consists of five identical, separated clusters. The branch decomposition consists of the root branch with four child branches; each branch has the same density maximum and all saddles and the global minimum are zero. (a) Undistorted 3-D topological landscape. (b) Landscape after the metric-based distortion. Because the volume of the root branch is distributed equally to all gray triangles, the based area of the center hill is smaller. (c) Landscape after the modified metric-based distortion. By distributing the volume according to the density distribution on the branches, only those triangles that contain these height values reflect the partial volumes. Because the implicitly stored regular nodes of the root branch, i.e. the cluster’s points, all have a density higher than zero, only the eight triangles of the center hill reflect this volume.

with a density higher than the saddle  $s_1$  of the branch’s first child branch. The other volume,  $v_2$ , represents the number of grid vertices with a density below  $s_1$ . Afterwards,  $v_1$  is distributed equally to the eight triangles of the center hill of the parent branch’s triangulation and  $v_2$  is assigned equally to all remaining triangles. Figure 4.2c shows the result of the modified volume distribution. The landscape contains five equally sized hills for five equally sized clusters. Note that the volume could be distributed accurately by considering all partial volumes between all saddle pairs and a distribution of these volumes only to those triangles on the spiral that contain these height values. However, for the clustering application, the accentuation of  $v_1$  sufficiently points out the sizes of separated clusters and subclusters.

In the remainder of this chapter, we provide example images based on the Reuters data set (cf. Appendix A.7). Figure 4.3 shows the smoothed 3-D topological landscape before and after the modified metric-based distortion. Using a suitable filter radius  $\sigma$  (whose determination will be addressed in Chapter 5.2.1), the landscape reveals fifteen separated clusters of varying density and size. Cluster separation is indicated

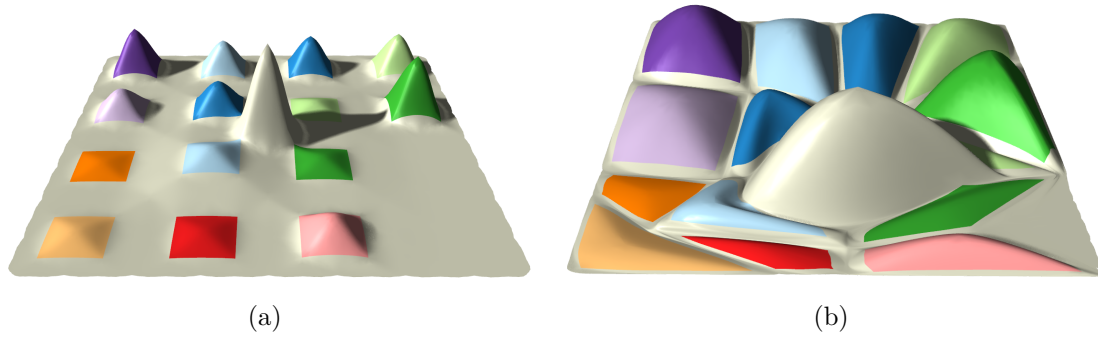


Figure 4.3: 3-D topological landscape of the 9-D Reuters data set (cf. Appendix A.7): (a) The undistorted 3-D landscape reveals fifteen separated clusters of varying density; the hills are colored randomly to discriminate them visually. (b) The base areas of the distorted landscape reveal that the clusters also differ in their size. The precise shapes of the base areas have no meaning and are actually an artifact of the distortion algorithm, which solves an optimization problem iteratively.

by the valleys between the hills, which reflect the saddle densities and are, thus, at zero level for separated dense regions. A cluster’s density and number of points, are reflected by the corresponding hill’s height and base area, respectively. A hill’s relative height also reflects that cluster’s persistence.

### 4.2.2 Data Point Representation

The landscape already illustrates high-dimensional clustering structure. It reflects all merge tree arcs, including their hierarchy and two arc properties. However, it does not consider the implicitly stored regular nodes, i.e. the underlying data points. Still, their visual representation could be important for the analyst to study intra-cluster distributions or to display additional meta-information per data item.

The merge tree stores the data points as a list together with each superarc to relate them to the region where they belong to. This gives a relation between a point and its cluster—and thus to its hill in the landscape. Based on this information, we can place glyphs for all data points at the height of their corresponding density. A simple glyph could be a small sphere that is colored according to possibly available classification information. More complex glyphs can be used to provide more meta-information.

To position a glyph on its hill, it is placed randomly on the contour at the height of its density  $\epsilon$ . Conceptually, this contour results from the intersection of the hill’s triangles with a horizontal plane at height  $\epsilon$ . Because each triangle may have a different amount of intersection with the contour, we select a random triangle using probabilities proportional to the amount of triangle-contour intersection. If each triangle had the same chance of being selected to host the glyph, glyphs would

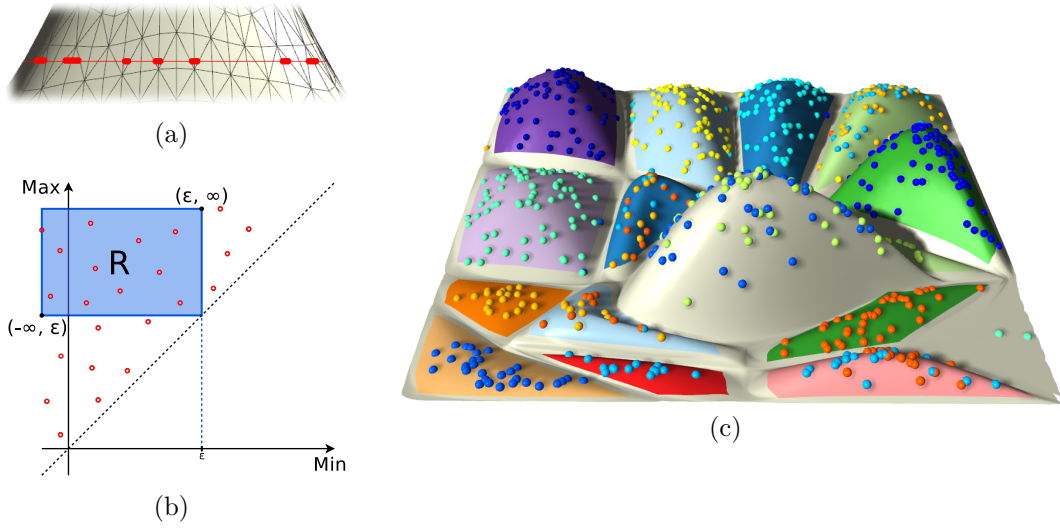


Figure 4.4: Data point representation in the 3-D landscape of the 9-D Reuters data set: (a) A particular glyph is placed randomly on the contour at the height of its density. Because for many glyphs, accumulations would occur at sharp triangles if the location is determined by selecting a random triangle, a triangle is determined proportional to the amount of contour intersection. (b) A range query in the span space efficiently returns only those triangles that could host the glyph at a certain height  $\epsilon$ . (c) The landscape augmented with the data points, which are colored according to their class. The color distribution of the glyphs indicates that clusters primarily correspond to the classes.

accumulate at the corners of sharp triangles, resulting in a distorted impression of the point distribution (cf. Figure 4.4a). To exclude those triangles efficiently that do not intersect a certain contour and therefore have a probability of zero, we use a span space [114] data structure for each branch (cf. Figure 4.4b). In the span space, every triangle is represented by a single point with an  $x$ - and  $y$ -coordinate corresponding to the triangle's minimum and maximum height value, respectively. To place a single glyph, a range query  $R$  on a simple 2D-tree, the two-dimensional specialization of a kD-tree, then quickly identifies those triangles that contain a particular  $\epsilon$ . The construction of a kD-tree is known to take  $O(n \log n)$  and a single range query takes  $O(\sqrt{n} + |k|)$  [13]. To accelerate the placement of all glyphs, they are processed branch-wise for each hill (concurrently), requiring one span space for each branch. Figure 4.4c shows the 3-D landscape with the augmented data points based on the Reuters data set. The glyphs are colored according to their class. Their color distribution on the hills reveals that clusters primarily correspond to the classes. Furthermore, the height distribution of the glyphs reflects the density distribution inside the clusters. The higher glyphs are located on a hill, the higher the average density is inside the cluster (and the more compact the cluster is).

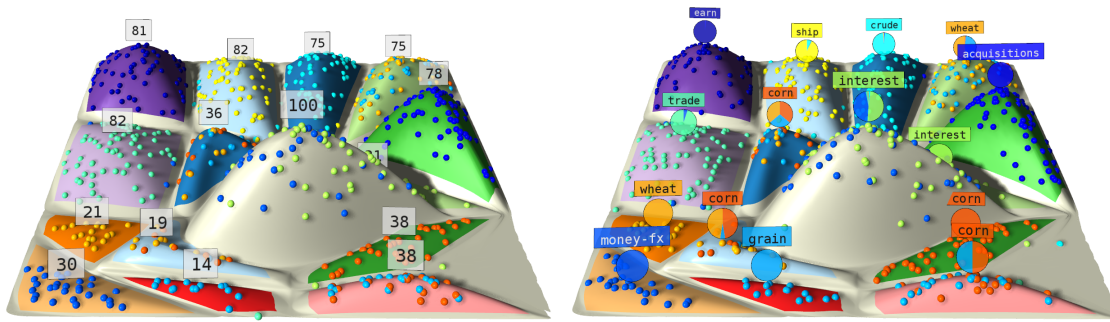


Figure 4.5: Hill-based labeling in the landscape of the 9-D Reuters data set: (a) Annotating the hills with the number of points in these clusters quickly reveals their size and avoids having to look at the landscape from above to evaluate base areas. (b) For classified data, pie-charts above the hills reveal the class distribution inside the clusters and display the name of the most frequent class.

### 4.2.3 Labeling

To provide additional information about clusters and individual points, hills in the landscape as well as data glyphs located on them can be labeled interactively.

**Hill-based labeling.** Labels above the hills provide more information about the clusters. For example, they can present exact cluster sizes, which is helpful to compare similarly sized hills and avoids having to look at the landscape from above to evaluate base areas. If classification information is available, more complex labels like pie-charts could also summarize the class-distribution inside the clusters. Figure 4.5 gives two examples based on the Reuters data set. Depending on the application domain, labels could also display the result of more sophisticated data analysis. For example, in case of text document data, the label above a hill could present statistical results of the documents in that cluster, or a determined topic based on their content. Labels can also trigger an action if clicked on them, e.g. by linking the corresponding points to other views.

From an implementation point of view, determining labels for the hills is just an operator on the merge tree. This operator may require only the merge tree itself, e.g. to determine cluster sizes based on implicitly stored regular nodes, or it may process additional information, e.g. to summarize their properties or content. A label's position is simply that of the center vertex of its hill in the landscape. This position is stored together with each maximum during the landscape construction. Furthermore, labels always face the viewer, i.e. their orientation is updated when the landscape is rotated during data inspection.

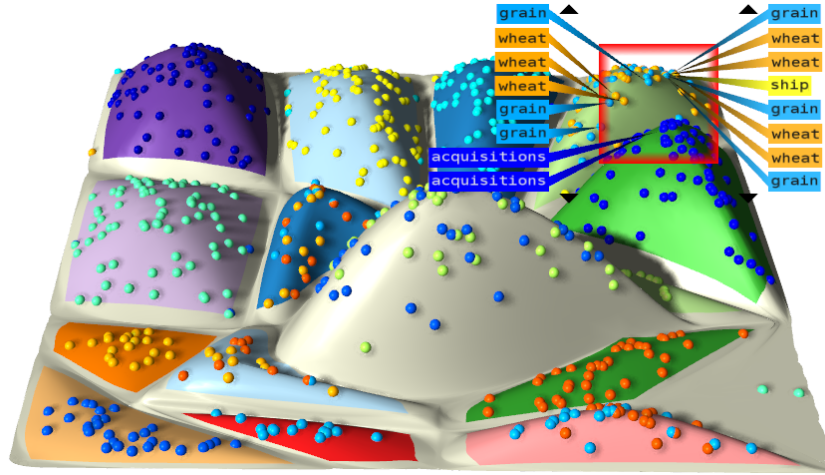


Figure 4.6: Glyph-based labeling in the landscape of the 9-D Reuters data set: Using a movable focus lens, only those glyphs inside the lens are labeled. Depending on their glyph positions inside the lens, labels are displayed in two scrollable lists. For classified data, labels are colored by class. To facilitate interactive inspection, the focus lens is connected to the position of the mouse cursor.

**Glyph-based labeling.** Typically, the analyst also needs information about single points to make sense of the data, e.g. to locate interesting data items or to find out why several points do or do not belong to the same cluster. For this purpose, we annotate data glyphs with meta-information like an entity’s name, id, or class. However, for large data sets, showing labels for all glyphs at the same time quickly occludes other labels and the landscape. Therefore, we implemented the *excentric labeling* [56] to annotate only those glyphs inside a movable focus lens. In its simplest case, the lens has a rectangular or circular shape and is connected to the position of the mouse cursor on the screen. Two scrollable lists to the left and to the right of the lens contain the labels, sorted by the vertical order of the points inside the lens. For classified data, labels are colored by class to be in line with the glyphs. Figure 4.6 shows an example based on the Reuters data set using the class names.

To determine the glyphs inside the movable focus-lens in real-time, the labeling is implemented with a two-dimensional kD-tree. For a fixed viewing direction, i.e. after changing the view on the landscape, a 2D-tree is constructed based on the screen space coordinates of all visible glyphs. While moving the focus-lens, label candidates are then identified by a range query on the kd-tree using the range defined by the focus lens.



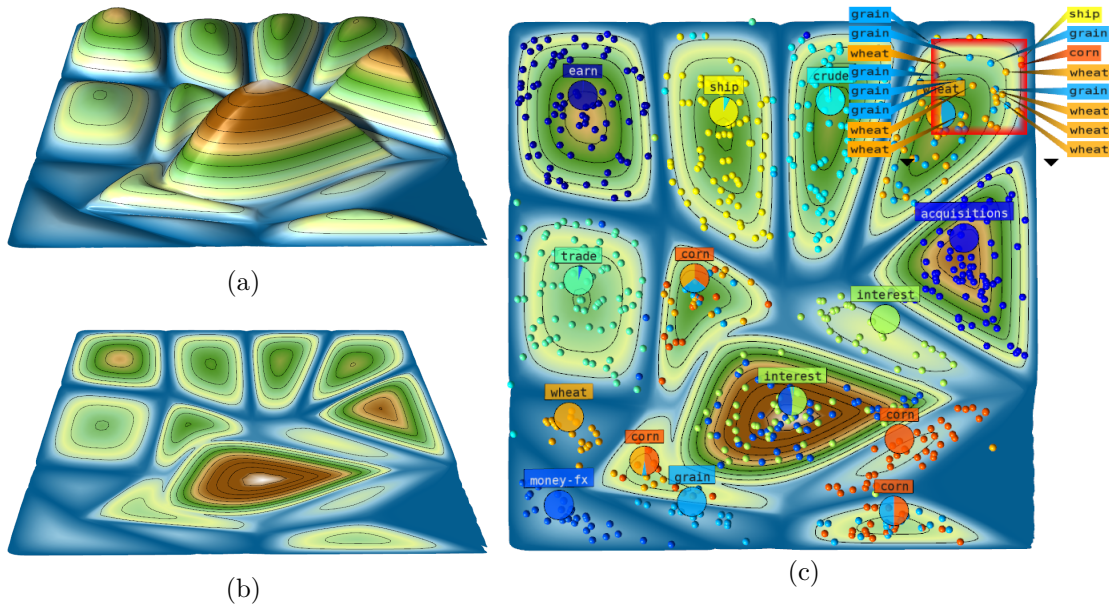


Figure 4.7: 2-D topological atoll of the 9-D Reuters data set: (a) Using a hypsometric tint, the landscape is colored by height values. This already helps compare height values irrespective of perspective distortion. (b) Because density information is now conveyed by colors, height information is redundant and can be discarded. The result is a 2.5-D visualization. (c) Inspecting the visualization from above gives a 2-D atoll-like visualization in which absolute densities, persistence, and feature sizes (base areas) are visible at the same time. Isolines help the analyst compare point densities inside the clusters.

### 4.3 2-D Topological Atoll

Compared to the occlusion problems that projections and axis-based techniques have for high-dimensional data, the 3-D topological landscape is already free of *structural occlusion* in the sense that all clusters in the original domain truly appear as separated hills. However, the landscape suffers from typical problems of 3-D visualizations, like perspective distortion and view-dependent occlusion of geometry. Depending on the camera position, hills in the background appear smaller and are often occluded. Glyphs residing on a hill's back-side are also invisible. If large hills are positioned at the landscape's border, these drawbacks have a negative effect on the visual analysis. It is also impossible to compare height values and base areas at the same time because both properties require to inspect the landscape from different angles.

These problems occur because the landscape's geometry conveys more than one feature property and requires more than two dimensions for this purpose. Distributing the required information more efficiently to the available information channels can

mitigate these issues. One information channel that is still unused is the landscape's color. Since colors only discriminate hills visually, this attribute can be used to display a feature property in order to simplify the landscape's geometry.

By using a hypsometric tint, which indicates elevation by colors as commonly used in geographic maps, we transfer information about densities from a triangle's height attribute to its color. To preserve the expressive power of the landscape metaphor, we use a transfer function that maps naturally occurring colors to different height levels: going from blue (water) and yellow (beach) through green (grass) into brown (mountains) and finally to white (snowy mountain top). Black isolines augmented to the landscape at various height levels help compare density values (cf. Figure 4.7a). Because height information is now redundant, the terrain can be flattened by setting the  $z$ -coordinates of all vertices to zero (cf. Figure 4.7b).

While these explanations illustrate how the 3-D landscape is transformed into a 2-D visualization, a more efficient implementation would directly create a flat triangulation. Understanding the former  $z$ -information as a 2-D scalar field, an extraction of isolines using marching triangles, a specialization of marching cubes [115] for isosurfaces, and using a color map with the transfer function above directly leads to the same result.

The landmasses surrounded by water remind the viewer of a bird's eye view on an atoll rich of islands. This metaphor is useful to depict clusterings whose density function contains saddles of low density. These are represented and perceived easily by the blue area around the islands. Figure 4.7c gives an example based on the Reuters data set. Compared to the 3-D landscape, all clusters and data glyphs are visible in the same view, and cluster sizes and their persistence as well as absolute densities of clusters and points can be identified and compared at the same time.

## 4.4 2-D Topological Landscape Profile

The original topological landscape requires three dimensions to visualize a contour tree because ambiguities need to be solved when representing local minima by sinks and merge saddles by valleys (cf. Figure 4.1b,d). However, these ambiguities do not arise for the less complex merge tree which only captures the appearance and merging behavior of superlevel sets. Consequently, its simpler structure allows the merge tree to be visualized as a 1-D function having the same topology. This 1-D function can be imagined as a cut through a 3-D density height field; hence its name *2-D topological landscape profile*. In this section, we present a novel 2-D landscape metaphor specifically designed for a merge tree. While visualizing a merge tree as a

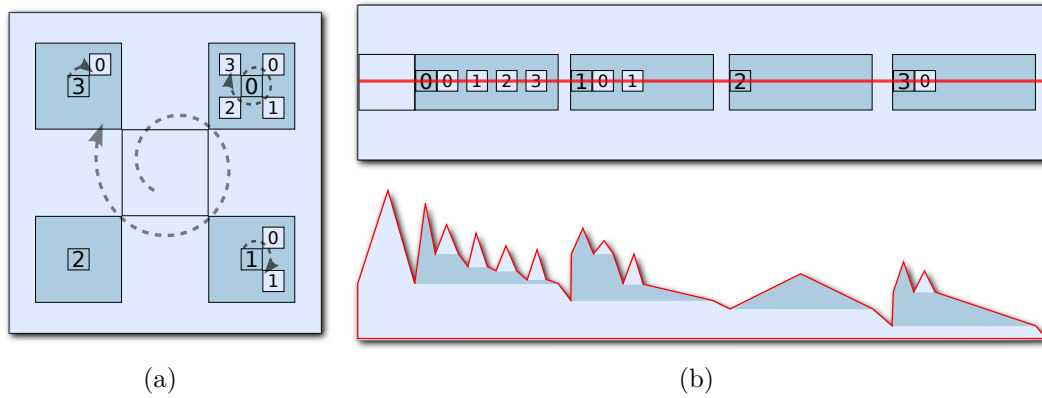


Figure 4.8: Transforming a merge tree’s 3-D topological landscape into a 2-D landscape profile with the same topology: (a) Schematic illustration of the 3-D landscape seen from above. Spiral shaped arrows, alternating colors and numbers indicate the spiral layout of the hills and the hierarchy of the child branches. (b) By unrolling the spiral layout, all child patches are placed next to each other. A lateral cut through the 3-D landscape provides a 2-D view on the hills.

2-D landscape profile is also valuable for other research fields, here, the metaphor is developed particularly for visual cluster analysis.

Conceptually, the 2-D landscape profile could be obtained from a variation of the 3-D topological landscape for merge trees. Although the profile will not be constructed this way later on, it is still interesting to see the relation between both visualizations. We make the following observations (cf. Figure 4.8): The 3-D landscape of a merge tree only consists of hills and valleys belonging to the maxima and saddles, respectively. That is, because there are no sinks and ambiguities, the spiral layout of the child branches can be *unrolled* by placing them next to each other. The topology of this 3-D landscape still reflects that of the input merge tree. However, the analyst still has to inspect the landscape from different angles to compare heights and base areas. This is avoided by depicting a feature’s volume/size by its hill’s width instead of its base area, i.e. by using only one dimension instead of two. Furthermore, the volume/size that was previously assigned to all triangles around the hills is now distributed only to the triangles between the hills. When looking at this landscape from the side, all hills are visible at the same time and, after the metric-based distortion, their width and the gaps between them accurately reflect the volumes/sizes of all individual merge tree arcs. Because depth-information is now redundant, it can be discarded by considering only a lateral cut—a profile—of the landscape.



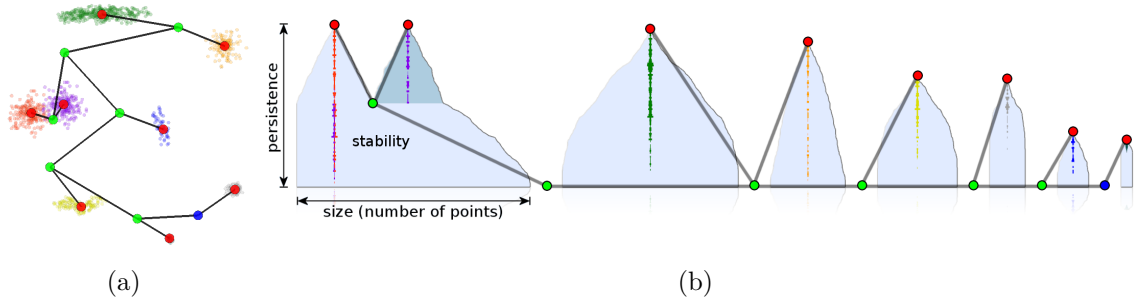


Figure 4.9: 2-D topological landscape profile of an artificial 2-D data set: (a) 2-D point cloud with clusters of varying size (number of points) and compactness. The merge tree encodes the clustering structure. (b) 2-D profile with the merge tree augmented. In the profile, separated clusters are represented by separated hills. A hill’s height, width and area reflects that cluster’s persistence, size, and stability, respectively. The subtrees of saddle nodes are sorted by persistence to place similarly persistent/high hills next to each other. Histograms or stacked bar charts (colored by class) on the hills indicate the point density distribution inside the clusters.

#### 4.4.1 Properties of the Landscape Profile

The 2-D profile allows for a much clearer clustering depiction and a more precise display and comparison of individual cluster properties. It requires less user interaction to explore the data and conveys structure at both a global and a local scale.

**Global structure.** The 2-D profile has the same topology as its input merge tree in the sense that each cut of a horizontal line through the profile intersects as many hills as the tree has superarcs containing this value. It thus quickly reveals the number of clusters and their hierarchy, and whether clusters are separated or surrounded by noise. However, it is important to recall that Euclidean distance does not reflect similarity in the topological context. In a topological landscape, structural information is conveyed only by the hierarchy of the hills and the valleys between them. Figure 4.9 provides an example: If multiple neighbored hills share valleys at zero height, this just indicates that these regions are well-separated. It tells nothing about how far away from each other they are in the original domain. Because the profile’s topology is invariant with respect to changing the position of these hills, we cannot say that a particular cluster is “closer” to that of its neighbored hill than to any other separated cluster. Such information about distances is not captured by the density function’s topology. If at all possible, spatial relations may only be derived from valleys above zero height. In this case, the non-zero density saddles indicate a spatial overlap, e.g. subclusters, in terms of the adjusted filter radius  $\sigma$ . Note that a large filter radius could also combine actually separated regions. Therefore, the first step to reading and understanding topology-based visualizations

correctly is to resist interpreting distances between hills in any way other than based on the valleys between them. While this could appear counterintuitive at the first sight, it is actually this abstraction that allows us to preserve the clustering in lower dimensions.

There is only little space for optimizations without changing the profile’s topology. One possible modification is to accentuate well-separated clusters by increasing the gap between those hills that are separated by a valley at zero height (cf. Figure 4.9b). Another optimization is to sort the subtrees of saddle nodes. This changes the position of the corresponding hills, but still preserves the profile’s topology. For example, in Figure 4.9b, all subtrees were sorted by persistence. This places the most prominent features to the left and gives the profile a global downward trend from left-to-right. Placing similarly persistent hills next to each other facilitates convenient comparison between them. Nevertheless, it is not possible to switch arbitrary hills just by sorting subtrees of saddle nodes—as this would quickly destroy the profile’s topology. Sorting subtrees by cluster size or stability is also possible. Hills could even be sorted by inter-cluster distance in the original domain. However, the advantage is very limited since each hill has only two neighbors. Sorting by topology-driven quality measures better reflects the topological context of the profile itself and these measures can also be preserved without loss in two dimensions.

**Local cluster information.** Similar to the 3-D landscape, hills in the profile describe individual cluster properties. While the values of a hill’s height and width (of its base line) still denote the cluster’s persistence and size, respectively, a hill’s shape additionally reflects the cluster’s stability. This is because the profile is basically a topology-based serialization of the input points on the  $x$ -axis together with their densities on the  $y$ -axis. The shape of the profile only results from ordering implicitly stored regular nodes so that they take the form of a hill for leaf superarcs and that of a slope for superarcs that connect two saddles. The convention is that at each height, the width of a hill reflects the number of points that have at least this density. Therefore, the hill’s shape accurately reflects the density distribution of the points, i.e. the cluster’s stability as defined in Chapter 3.2.4. This implies that the hill of a “stable” cluster, where many points are close to the density maximum, are rectangular-shaped, and less stable hills, with many points close to the saddle density, are more triangular- or peak-shaped. Plateaus at different height levels indicate suspicious subfeatures, but are also a typical effect of topological simplification. The alternating two-tone coloring scheme of the hills accentuates their hierarchy if they are nested.

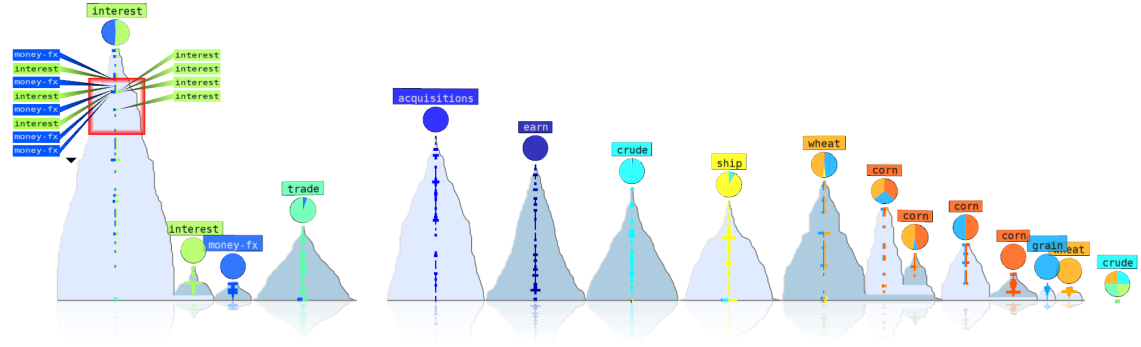


Figure 4.10: 2-D topological landscape profile of the 9-D Reuters data set: Hills in the landscape represent clusters in the data. Their height, width, and area reflect a cluster’s persistence, size, and stability, respectively. Alternating colors represent the hierarchy of nested regions; cluster separation is emphasized by an additional gap between those hills separated by a valley at zero height. Histograms represent the data points at the height of their densities and can be colored by class. Excentric labeling for histogram fragments and hill-based labeling provides additional meta-information.

The data points are augmented to the profile as (horizontal) histograms for annotation and to show the point distributions. Individual points are represented by the fragments of the histograms. The length of all histograms on a particular hill totals to the cluster’s size. If classification information is available, histograms extend to stacked bar charts, one bar and color per class (cf. Figure 4.9b). With this representation the analyst can quickly determine whether classes correspond to clusters. As already mentioned for the Euclidean distance between the hills, the distance between histogram fragments does not indicate spatial similarity. While the fragments of a particular bar do have similar densities, they are probably located at very different positions inside the cluster; likely in a circular fashion around the density maximum (cf. Figure 5.10 on page 131).

**Advantages.** There are several advantages of the 2-D landscape profile over the 3-D landscape: no view-dependent occlusion that hides features, no perspective distortion that complicates feature comparison, no strangely distorted base areas, more accurate depiction of individual cluster properties to simplify feature comparison, another cluster property conveyed by a hill’s shape, no invisible and randomly placed data points, a more compact and discriminable display of the data points as colored histograms, less user interaction required to navigate through the scene, less complex geometry and faster construction scheme, and no expensive metric-based distortion. Drawbacks, on the other hand, are a slight decrease in the expressive power of the landscape metaphor compared to the more natural looking hills in 3-D and a little less efficient screen-space utilization by the layout from left-to-right compared to the

---

**Algorithm 2:** Pseudo-code to construct the 2-D landscape profile.

---

```

Input   : root node of the merge tree
Output : topological landscape profile

1 procedure PaintLandscapeProfile( root)
2    $x \leftarrow 0.0$ 
3   PaintPart( root,  $x$ )

4 procedure PaintPart( node,  $x$ )
5   if HasParentArc( node) then
6      $arc \leftarrow$  GetParentArc( node)
7     if IsLeaf( node) then
8       DrawHill( arc,  $x$ )                // cf. Figure 4.11a
9     else
10      DrawInnerPart( arc,  $x$ )            // cf. Figure 4.11b
11   for  $i \leftarrow 1$  to NumberChildNodes( node) do
12      $childNode \leftarrow$  GetChildNode( node,  $i$ )
13     PaintPart( childNode,  $x$ )
14      $x \leftarrow x +$  SubTreeSize( childNode)
15   end

```

---

spiral layout in 3-D. Figure 4.10 shows a 2-D topological landscape profile of the Reuters data set. Compared to the 3-D landscape shown in Figure 4.3, the profile contains more hierarchical features. This is because we removed vague hierarchies in the 3-D landscape by *re-balancing* (cf. Chapter 3.1.2) the merge tree’s branch decomposition for two reasons: First, we wanted to avoid massively distorted base areas for hierarchical features while giving initial example illustration. The second reason is that these little differences in height values would not have been noticed in the 3-D visualization. Both reason are actually drawbacks of the 3-D landscape metaphor. In contrast, the 2-D landscape profile easily reveals vague hierarchies because even small differences in height values are visible for the valleys, and because this hierarchy is additionally emphasized by alternating hill colors.

#### 4.4.2 Construction and Implementation

The construction of the 2-D landscape profile works directly on the merge tree instead of on its branch decomposition. Because feature properties are mapped to only one dimension each, there is also no metric-based distortion required as an expensive post-process. Hence, the construction is much simpler and faster than that of the 3-D landscape.

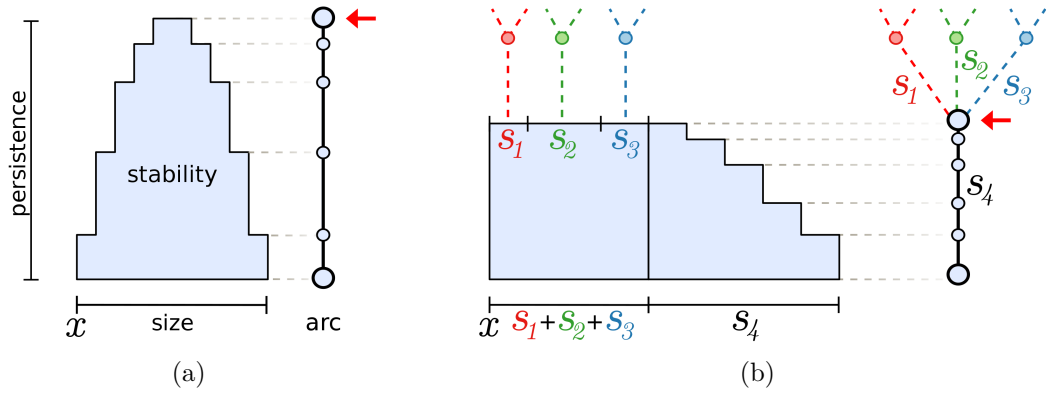


Figure 4.11: Construction scheme of the 2-D landscape profile: (a) A leaf superarc is represented by a hill whose height and width values reflect the superarc’s persistence and size, respectively. The shape of the hill is affected by the distribution of the implicitly stored regular nodes; its area thus reflects the cluster’s stability. The red arrow indicates the currently processed node;  $x$  is the current position on the  $x$ -axis. (b) Non-leaf superarcs reserve space for child superarcs and are represented by a slope to the right.  $s_1, s_2, s_3$  represent the size of the whole subtrees. In both cases, shapes of hills and slopes result from a binning of the density distribution. Changing the binning parameter affects the geometry’s complexity and its accuracy.

The profile is generated with a simple recursive algorithm. Because each superarc of the merge tree is represented by a distinct part of the profile, we just need to traverse the merge tree starting from its root node and consider each superarc’s persistence, number of implicitly stored points, and its stability to create hills and slopes. Algorithm 2 and Figure 4.11 provide a detailed description of the profile construction. For simplicity,  $y$ -coordinates in the profile directly map to density values and the  $x$ -coordinate of the profile is initialized to 0.0. Ranges on the  $x$ -axis also directly map to feature size, i.e. the total width of the profile equals the size of the data set. A node’s *parent superarc* and *child superarc* are its superarcs towards the merge tree’s root node and leaf nodes, respectively. The root node has no parent superarc and the leaves have no child superarcs. The granularity of the histograms and the shape of the hills results from a conventional binning approach, i.e. they depend on a parameter to define the height of the horizontal bins.

The two-tone coloring accentuates the hierarchy of nested regions and is based on the implicit branch decomposition. That is, sub-hill relationships are highlighted by switching the colors for each hierarchy level. For classified data, both used colors should be discriminable from the histogram colors. For aesthetic reasons, the profile’s height is scaled to a fixed percentage of its width and, to focus on the main features in a profile, we also provide the option to shorten long slopes and plateaus by taking

the logarithm of each superarc’s size. Finally, the profile’s appearance is spiced up with a bevel effect and a reflection on the ground.

The runtime complexity of the profile construction depends on the number of points in the data set. Because each data point contributes to the shape of its hill, every (implicitly stored) point is considered once during the merge tree traversal, which gives a complexity of  $O(n)$ .

## 4.5 Examples and Results

In this example section, we apply the presented topology-based visualizations to several data sets. The focus is on their capability to provide an appropriate global clustering overview and to visualize the structure of unclassified data—when colors cannot be used to discriminate data points visually. We discuss these aspects for the different landscape variations and also compare them to standard techniques for high-dimensional data, like projections or parallel coordinate plots.

The structure of this results section is similar to that of Chapter 3.5. We start with the artificial 2-D example again to illustrate relations between the visualization and the underlying data set. Afterwards, topology-based visualization is applied to several real-word data sets and strengths and limitations are addressed based on the artificial 100-D data set. Visualizing the clustering structure of unclassified data and considering document-related applications complete this results section.

The time required to construct the merge tree visualization is negligible compared to the topological analysis. In this example section, the computation times to create the landscapes were typically less than one second. Potential runtime-bottlenecks can be identified primarily for the 3-D landscape if it consists of considerably many triangles. This happens for very complex merge trees or after landscape smoothing. In this case, the metric-based distortion and the glyph placement can take some seconds. For example, small filter radii quickly increase the number of density maxima and thus the number of hills and the triangulation’s complexity. Note that strategies to determine appropriate parameters as well as local analysis aspects are addressed in Chapter 5.

### 4.5.1 Global Overview

An accurate global overview is important to understand the data and to stimulate further analysis. A suitable clustering overview points out all clusters contained in the data and allows the analyst to identify and compare features based on their

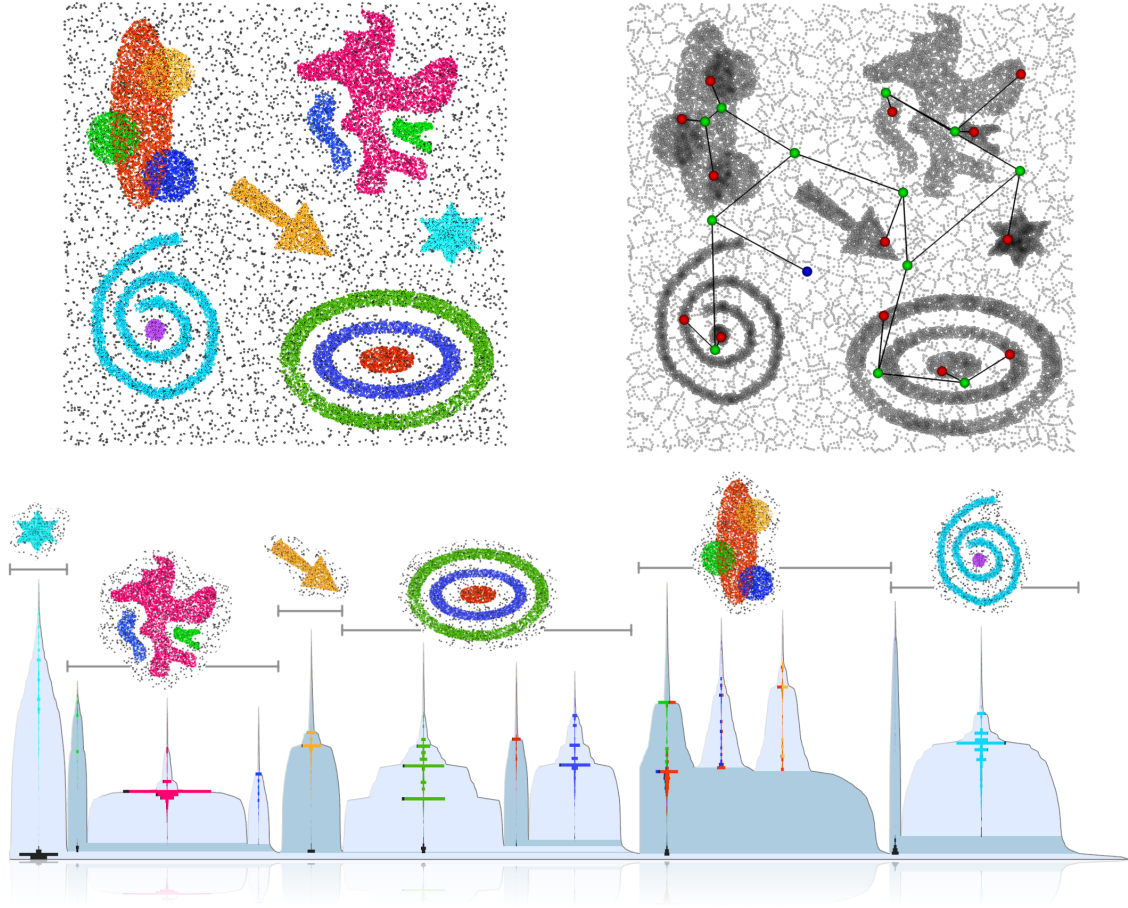


Figure 4.12: Mapping between clusters and hills in the 2-D landscape profile based on the artificial 2-D data set: (top) The data set and a density-colored version augmented with the merge tree. (bottom) 2-D topological landscape profile with hills of varying characteristics. The structure of the profile and the shape of the hills accurately reflect the merge tree and the properties of its superarcs.

properties. Only if structure is presented accurately, the analyst can validate existing knowledge or find new insights that are worth further analysis.

### Artificial 2-D Data Set

The purpose of the artificial 2-D data set (cf. Appendix A.1) is to demonstrate as much features as possible. Figure 4.12 shows the 2-D data set (top left), a density-colored version with the augmented merge tree (top right), and the corresponding 2-D topological landscape profile (bottom).

The profile exhibits thirteen separated hills, one for each maximum of the density function. The order of the hills is defined by the merge tree, but can be adjusted by sorting subtrees of saddle nodes, e.g. to place the highest hills to the left. As already discussed and now clearly visible in this example, topology cannot (and does not

intend to) preserve cluster shapes in the final visualization. However, topology-based quality measures can be preserved and are helpful to approximate properties like compactness and shape. Region properties, like their persistence, size, stability, and density distribution, affect the characteristics of each hill and thus quantify a cluster's significance. The analyst can quickly identify dense or large-sized features by looking for hills of large height or width, respectively. For example, as indicated by the width of their hills, the cyan spiral-shaped cluster contains substantially more points than the purple cluster in its center. Furthermore, a hill's shape reflects the density distribution and provides valuable information about the cluster's compactness and approximated shape. For example, the hill belonging to the cyan star-shaped cluster has a triangular shape to reflect the typical density gradient from high density in the center to lower density at the border. This hill's aspect ratio also suggests a compact cluster because few points produce high density. Valleys and their depth denote cluster distinctness relative to their surrounding density. While valleys at zero height indicate cluster separation, non-zero valleys reflect subclusters or clusters surrounded by noise.

Another important aspect is how the topological approach deals with noise. Not only is the topological analysis robust with respect to noise, noise points also do not hide features or clutter the visualization. Since the data points are positioned at the height of their density, representatives of noise points are typically found at the bottom of the landscape. Figure 4.12 shows noise as black histogram accumulations. Because black points cover the whole data domain, accumulations of black histograms can also be found at the bottom of several hills to reflect their belonging to that region. Note that black histogram fragments occasionally occur on the hills as well. These fragments represent the black data points of higher density inside the clusters. This behavior is just for demonstration purposes as it actually contradicts the definition of clusters and noise.

#### 4-D Iris Data Set

The Iris plants (cf. Appendix A.6) are a rather small and low-dimensional data set containing only 150 points in four dimensions. It is known from this standard data set that one class is separable from the other two; while the latter are not separable from each other. We use this data set to illustrate how this already established knowledge is represented in the topological context of the landscape metaphor. Figures 4.13a-b show a PCA projection and a parallel coordinate plot for the Iris data. The colors of the data representatives reflect their class affiliation. It is visible in both visualizations that the classes form clusters, that the red and green points overlap (as confirmed



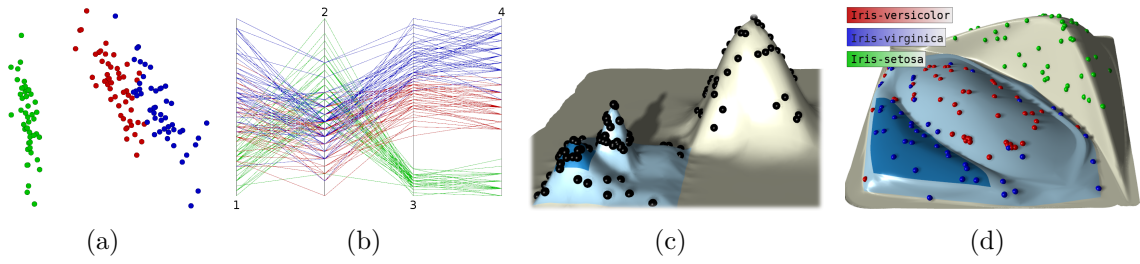


Figure 4.13: Visualizing the 4-D Iris data set: (a) The PCA and (b) the parallel coordinate plot reveal that clusters correspond to classes and that two clusters overlap and one cluster is isolated. Identifying a cluster’s size or compactness is still difficult. (c) The topology of the 3-D landscape reflects the same situation: three density maxima, of which two are separated by non-zero density and one is isolated by zero density. (d) After the metric-based distortion and coloring the data points by class, the landscape accurately reflects the already known information about the clustering and furthermore provides information about cluster size and compactness.

by the PCP), and that the green cluster is isolated. Consequently, the topological 3-D landscape in Figure 4.13c shows an isolated hill and another hilly structure containing two peaks. This configuration describes a density function with three density maxima, of which two are separated by a non-zero saddle density and one is separated from the other by zero density (as reflected by the low valley between the gray and the blue hill). The larger height of the center hill furthermore reflects a higher density of this cluster, which is a typical result if a similarly sized cluster is more compact. Figure 4.13d shows the volumetric distorted 3-D landscape in which the data glyphs are also colored according to their class. As can be identified from the color distribution, the structure of the landscape accurately reflects the already known relationships between the three classes. The ratio between the height and the base area of the hill belonging to the green cluster also reveals that these points are more compact. Another advantage of the topological abstraction is that we can easily identify the overlapping region: these are the data glyphs at the height of the saddle density on the blue hill.

## 8-D Italian Olive Oils Data Set

An interesting aspect of the Italian olive oils data set (cf. Appendix A.5) is to find out whether the oils share combinations of fatty acids and how groups of similar oils relate to the growing regions as specified by classification information. Figure 4.14 shows a PCA-projection, a parallel coordinate plot (PCP), and the three presented variations of the topological landscape metaphor. Although the PCA and the PCP convey structure primarily by color, they still indicate a fairly good clustering. However, the

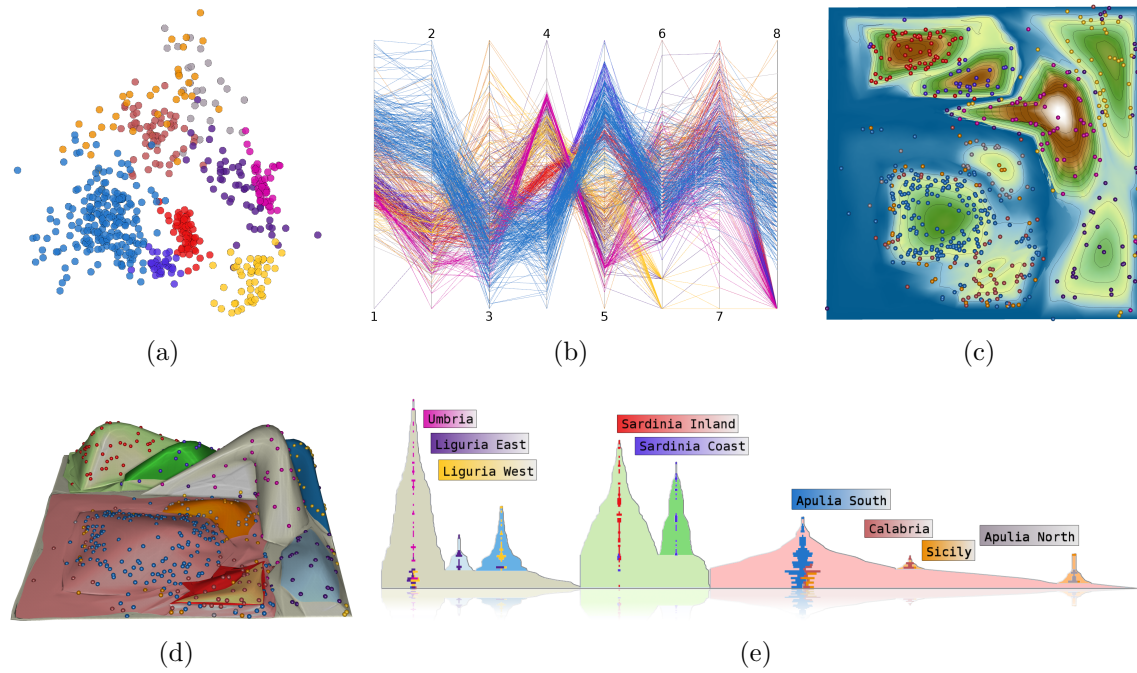


Figure 4.14: Different visualizations of the 8-D Italian olive oils data set: (a) The PCA projection and (b) the parallel coordinate plot (PCP) convey clustering structure primarily based on colors. Cluster properties can only be estimated and hierarchy can only be guessed. Occlusion and the necessity to count points manually make it impossible to compare cluster sizes. (c) The topological atoll, (d) the 3-D landscape and (e) the 2-D landscape profile accurately reveal the clustering structure and individual cluster properties. The profile is the most efficient and accurate variation of the landscape metaphor.

projection error of the PCA is approximately 30%, i.e. the projection preserves only 70% of the variance in the original 8-dimensional space, which is why we can assume that some of the clusters are actually more separated than the projection is able to express. The PCP confirms this to some extent because polyline bundles are spread or even inverse in some dimensions. This behavior indicates separation at least in these subspaces. Beyond these limited results, it is difficult in both visualizations to identify the number of clusters, their hierarchy or individual properties like their sizes.

The topology-based visualizations shown in Figures 4.14c-e accurately describe the clustering structure in the original domain. The very low valleys between the hills in the 2-D landscape profile notify three separated clusters, each with some subclusters. Based on the color distribution of the histograms, we conclude that the three clusters correspond to the three major growing areas on Sardinia and in Northern and Southern Italy. This confirms that the olive oils form clusters based on their combination of fatty acids and that they differ for the growing regions.

Furthermore, the profile reveals knowledge that was misleading or hidden entirely in the PCA and in the PCP. First, cluster separation is not that obvious in both visualizations and it is difficult to identify the most prominent clusters. This is because a cluster's extent does not necessarily reflect its number of points and because occluded points are not visible at all, e.g. to count them individually. Second, structure is misleading even for those clusters that are clearly visible. For example, while the PCA suggests that the yellow "Liguria West" cluster is larger than the pink "Umbria" cluster, comparing the width of both hills reveals the opposite. Also the perceived dominance of the blue "Apulia South" cluster compared to the rather unremarkable purple and red "Sardinia" clusters cannot be confirmed by the profile in which these two hills are actually higher. A capital illusion in the PCA is a wrong subcluster relationship between the purple "Sardinia Coast" and the blue "Apulia South" points. While these clusters seem to overlap in the PCA, the valley between their hills in the profile reveals that these points actually correspond to different clusters that are separated in the original domain. This illusion is caused by the 30% projection error.

Note that the 3-D landscape and its 2-D atoll-like variation basically provide the same structural insights. However, even though some users may favor their expressive power, they are not as precise and efficient as the 2-D profile. For example, cluster sizes are difficult to compare based on distorted base areas and, most importantly, little differences in saddle heights are harder to notice in 3-D. This makes it difficult to identify slight hierarchies, which can be observed for the cluster belonging to Northern Italy, i.e. the gray and blue hills in the profile and in the 3-D landscape. While the atoll in Figure 4.14c somewhat confirms this hierarchy by shallow water, the 3-D landscape hardly reveals this hierarchy. The analyst has to look at the landscape from the side to observe this little decrease in height. In the profile, this hierarchy is clearly visible at the first sight. Nevertheless, their focus on structure instead of on geometric properties and colors makes both landscape metaphors still more useful for cluster analysis than lossy and occlusion-prone direct visualizations.

We conclude from the global overview that olive oils from Sardinia, Northern Italy, and Southern Italy are different from each other and also have different intra-cluster similarities. The latter is indicated by the density distributions inside the clusters, i.e. by differently shaped hills. In fact, a detailed statistical analysis of the features shows that southern oils have much higher eicosenoic acid on average and slightly higher palmitic and palmitoleic acid content. The oils from Northern Italy and Sardinia have some difference in the average oleic, linoleic, and arachidic acids [57].

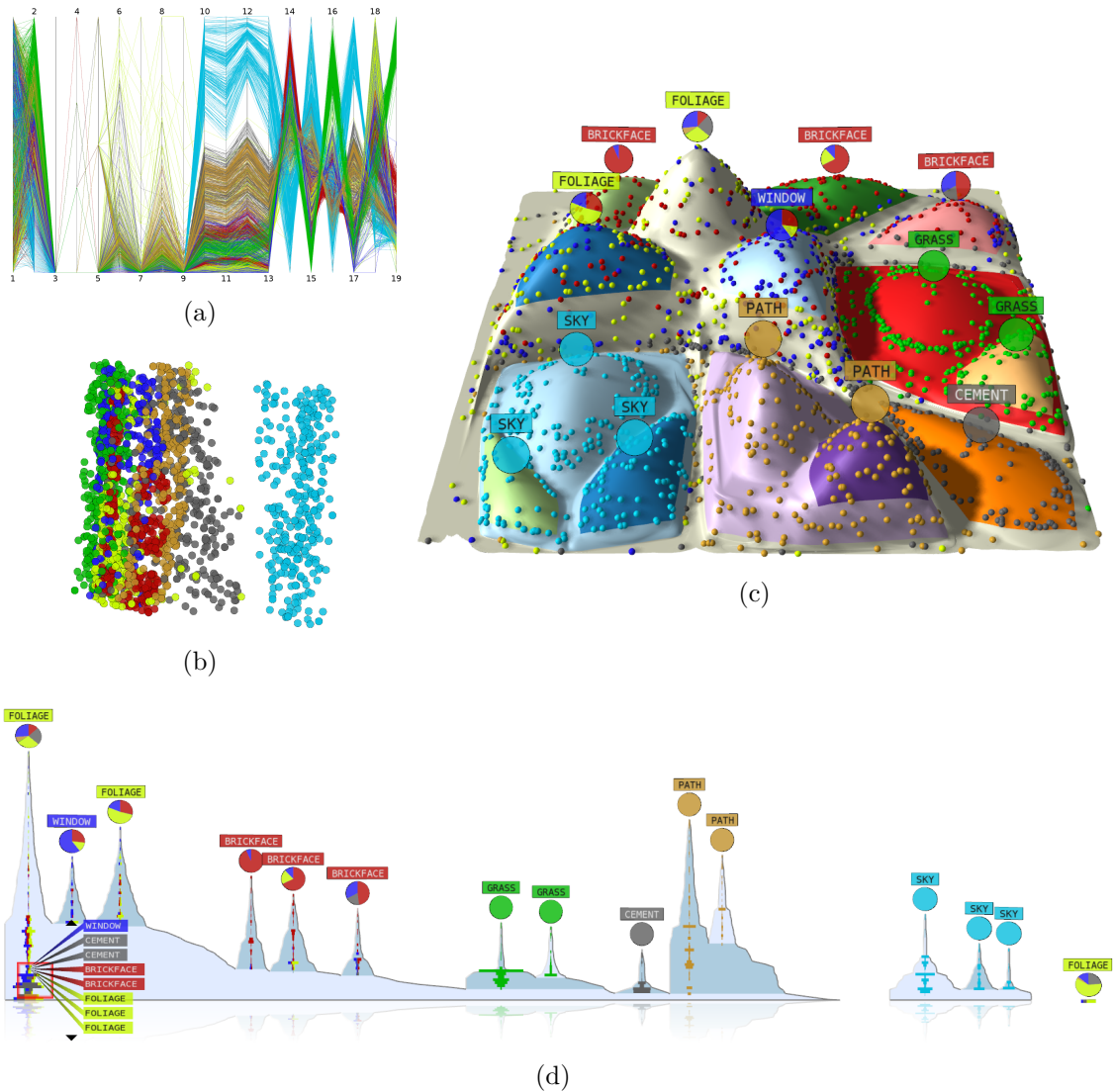


Figure 4.15: Different visualizations of the 19-D image segmentation data set: (a) The PCA projection (the projection error is  $\approx 35.76\%$ ) and (b) the parallel coordinate plot convey structure primarily by color. Although both images indicate a separated cluster (cyan “sky” points), the hierarchy of the remaining clusters is unclear and cluster properties are difficult to identify and compare. (c) The 3-D topological landscape and (d) the 2-D landscape profile accurately illustrate the clustering structure in the 19-D space. While cluster separation and suspicious accumulations of data glyphs are slightly more difficult to identify in the 3-D landscape, the 2-D profile presents structure and cluster properties clearly and accurately.

### 19-D Image Segmentation Data Set

The advantage of the topology-based visualization becomes clearly apparent with increasing data size and dimensionality. Figure 4.15 shows the PCA, a PCP, a 3-D landscape, and a 2-D landscape profile of the 19-D image segmentation data set (cf. Appendix A.3). While the projection error of the PCA and the visual

complexity of the PCP typically increase for high-dimensional data, the appearance of the landscape metaphor is not affected by data size or dimensionality. Although the cyan “sky” cluster clearly stands out in both the PCA and the PCP, clustering structure is conveyed primarily based on the color distribution rather than on separated accumulations of point representatives. Still, the PCP confirms separation in the subspaces starting from the 10<sup>th</sup> dimension, and the PCA suggests three accumulations of red points, which is not visible in the PCP though. As already mentioned, identifying cluster hierarchy or comparing cluster properties is very difficult because both standard techniques suffer from occlusion and visual complexity.

The topological landscapes shown in Figures 4.15c-d show a complex clustering with individual hills for every class. The landscape profile also emphasizes the separation of the cyan “sky” cluster by an extra space to isolate this hill visually. Because cluster separation is harder to detect in the 3-D landscape, the 2-D profile is more efficient to detect this feature of a particular clustering. Both landscapes also confirm three clusters for the red “brickface” points, as was already indicated by the PCA. Two of the easier to identify clusters in the PCA and the PCP are the brown “path” cluster and the cyan “sky” cluster. However, in both images it is difficult to compare them in terms of their compactness or size. While their compactness is indicated by the spatial closeness of the points and polylines, it is impossible to evaluate their size because their extent could be an artifact and because counting the points manually is not an option. In contrast, looking at the profile quickly reveals their precise density and size. Because the hills have almost the same width, but are different their heights, the “path” cluster must be more compact than the “sky” cluster.

The subcluster hierarchy of the cyan “sky” points is slightly visible in the PCA and the PCP. However, this is not the case for the brown “path” points. In contrast, the topology-based visualizations do not only confirm these subclusters with individual hills, they furthermore indicate even more local accumulations based on groups of different heights for the data point representatives. Another advantage of the 2-D profile compared to the 3-D landscape is that the distances between the hills accurately reflect the volume of the regions before they merge at the next saddle. Because this volume is distributed equally between the triangles of the hills in the 3-D landscape, the hills are placed closer to each other and thus these details about intermediate sizes are lost.

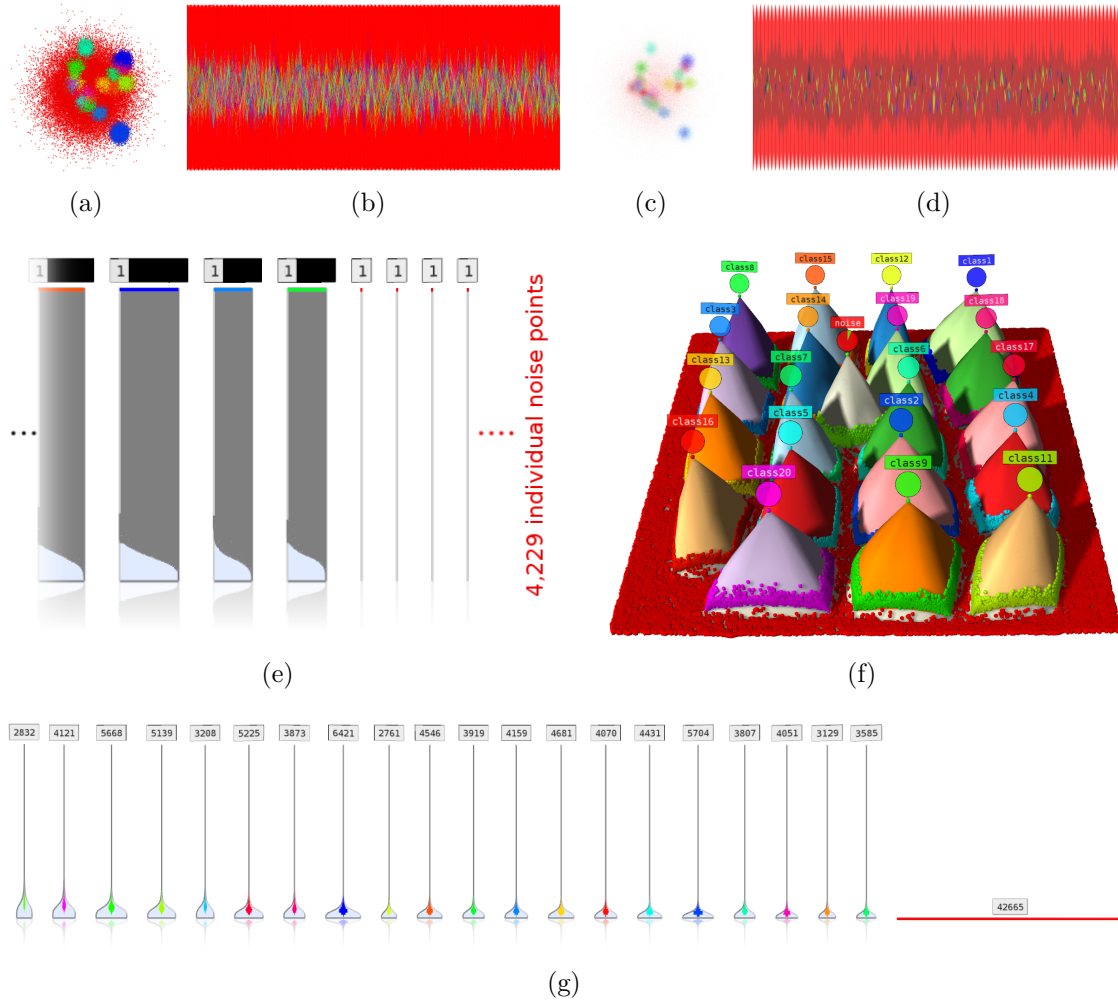


Figure 4.16: Artificial 100-D data set: (a) The PCA projection and (b) the parallel coordinate plot clearly have problems with large data sets of high dimensionality. (c)-(d) A common strategy to suppress noise is to reduce the opacity when drawing individual data points or polylines. Although this works reasonably well for the PCA, the correct clustering or cluster properties cannot be identified in both visualizations. (e) A part of the unsimplified 2-D landscape profile showing that every point actually has its own density maximum and that the clustering is primarily defined by the saddle densities. (f)-(g) The topological landscapes accurately depict the clustering and cluster properties. Noise is placed at the bottom and does not cover the visualization.

### Artificial 100-D Data Set

The artificial 100-D data set (cf. Appendix A.2) was already used in Chapter 3.5.2 to analyze scalability issues for the algorithmic core of the topological analysis. We revisit this data set to demonstrate how its complexity affects the topology-based visualization and to compare the landscape with the output of standard techniques.



Figure 4.16 shows images of the 100-D data set for the PCA projection, a parallel coordinate plot (PCP), the 3-D topological landscape and the 2-D landscape profile. As for the standard techniques, the PCA and the PCP in Figures 4.16a-b quickly reveal that both techniques have severe problems with illustrating the clustering structure of this data set. Because the large amount of noise points typically reduces the quality of these visualizations, we also provide alpha-blended versions in Figures 4.16c-d. In this case, individual points or polylines are rendered with reduced opacity so that only their accumulations, i.e. clusters, are accentuated in the final image. This is a common strategy to suppress the depiction of noise in sparse areas with only little pixel complexity. As can be seen in both images, this strategy works fairly well for the PCA where only dense areas of different color remain. However, increasing transparency in the PCP does not help to accentuate individual polyline bundles because those polylines corresponding to noise still cross frequently and thus increase the pixels' opacity. That is, the transparency effect vanishes if there are too many noise points. Regarding the ability to communicate the clustering of this data set, at least the transparency-optimized PCA projection suggests a clustering. Identifying the number of clusters and their hierarchy, or even comparing clusters in their size, is not possible because the projection error is approximately 95%.

Figures 4.16f-g show the 3-D landscape and the 2-D landscape profile of this data set. The most obvious property is that in both visualizations the depiction of structure is not covered by the 50% noise and that all 20 clusters are clearly visible as separated hills. This demonstrates that the topological analysis and the overall appearance of the landscapes do not at all depend on the dimensionality of the underlying data. Only the distribution of the data glyphs on the hills in the 3-D landscape could complicate inspecting the landscape's structure; but the glyph placement is optional and can be deactivated at any time to inspect only the hills and their hierarchy. For example, in Figure 4.16f, the 42 665 red glyphs make it difficult to inspect the valleys between the hills or to distinguish the noise glyphs from the reddish glyphs of other classes. In the 2-D landscape profile, these issues are no problem at all. As shown in Figure 4.16g, the profile consists of 20 separated hills and the noise points are summarized by one red histogram bar. The individual properties of all clusters are clearly visible and can be compared precisely based on the hill properties and the labels above them.

The clear structure of the landscapes results from topological simplification. Furthermore, the curious shape of the hills as well as the suspicious distribution of the data glyphs indicates interesting behavior during the analysis. As demonstrated in Figure 4.16e based on a 10% randomly sampled subset, the unsimplified landscape

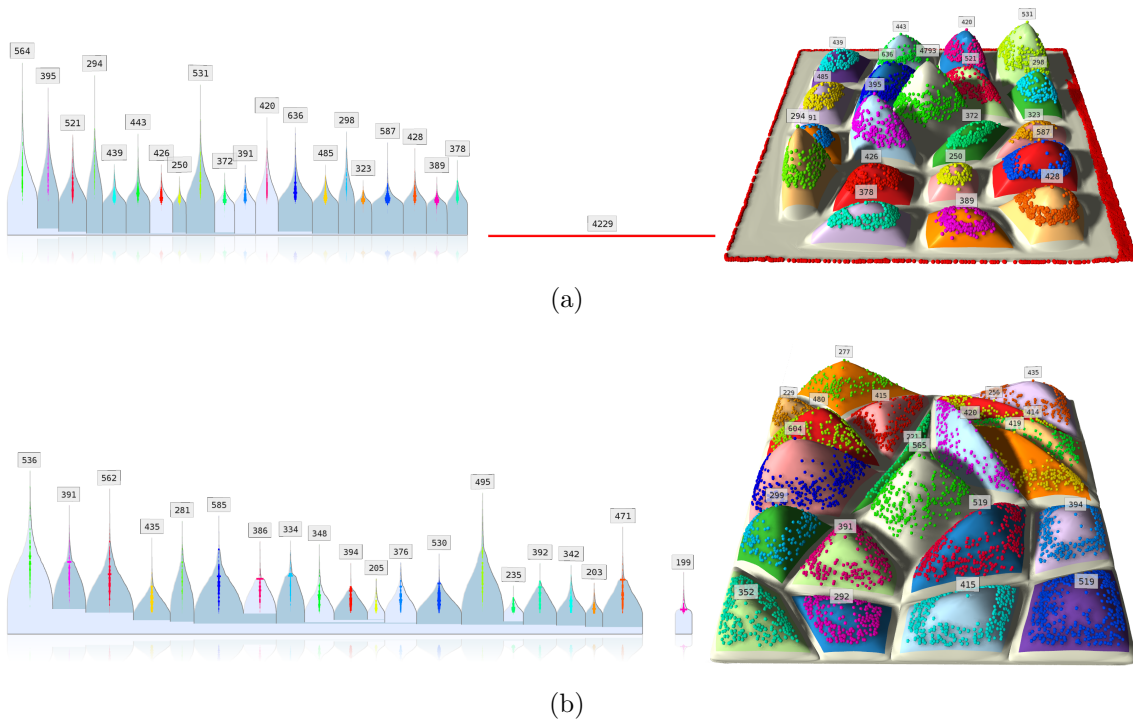


Figure 4.17: Artificial 100-D data set for different parameter settings: In both cases only a 10% randomly sampled subset is used and regions with less than 50 points are simplified topologically. (a) The 2-D landscape profile and the 3-D landscape using the relative neighborhood graph and after reinsertion of non-samples. (b) The 2-D landscape profile and the 3-D landscape using the Euclidean minimum spanning tree and without reinsertion of non-samples to provide a quick structural overview if the presence of noise is unimportant. From a clustering point of view, the overall structure in all three scenarios (cf. Figure 4.16) is almost the same for different parameter settings.

profile actually consists of 20 noisy hills and a single hill for every individual noise point. The gray “areas” on the hills are actually many thin hills for individual points inside the clusters. The interesting aspect is that the height of the hills inside the clusters is only slightly higher than 1.0—which is not obvious in the images, but results from printing statistics about the point densities—and that it is actually the non-zero saddle densities that make the points inside a cluster being related to each other. After topological simplification, only the point with the density maximum remains and the other points are assigned a density of the former saddle value. This is why the hills are peak-shaped and why the other points accumulate at the bottom of the hill. Although it is difficult to imagine the situation in the 100-D space, the fact that all cluster points have almost the same density, but lower density in between, is assumed to result from the clusters being very sparse and having a “tetrahedral”-like shape in the original domain.



Figure 4.17 indicates that the overall structure of the landscapes is also robust to changing parameters of the topological analysis. Compared to showing all data points, here we use only a 10% random subset and simplify clusters with less than 50 data points. Figure 4.17a shows the 3-D landscape and the 2-D profile for using a larger filter radius (which affects the densities), the relative neighborhood graph, and with reinsertion of the non-sampled points. Figure 4.17b shows the landscapes for the Euclidean minimum spanning tree and without reinsertion of non-samples. As can be seen, the overall clustering structure is the same in all three scenarios and accurately reveals 20 separated clusters of varying size. To detect cluster separation, smaller filter radii could be necessary to observe whether hills become separated or split into several sub-hills once the filter radius is smaller than a cluster's diameter. As explained above, in very high-dimensional spaces, reducing the filter radius can somewhat distort the distribution of the point densities inside the clusters. This is a limitation of the Gaussian kernel.

### 4.5.2 Unclassified Data

To demonstrate how unclassified data affect the landscape metaphor compared to the output produced by standard techniques, we revisit some of the example data sets, but discard classification information provided with the data. To this end, we simply color all data glyphs uniformly in black.

The key advantage of the topological approach over direct visualizations is its focus on structure instead of on relying on colors to convey information. As a consequence, the clustering structure learned from the topological landscape is not affected at all by the availability of classification information. This should be demonstrated based on some already known example images: The Rank-2 LDA of the Reuters data set in Figure 2.5 (on page 21) indicated two clusters that contain points of several classes. This observation stimulated further analysis with a scatterplot matrix. Discarding the classification information of the Reuters data set leads to the Rank-2 LDA and the parallel coordinate plot shown in Figures 4.18a-b. Because suspicious accumulations of mixed colors do not stimulate further analysis anymore, we miss subclusters and important cluster hierarchy due to occlusion artifacts caused by the projection error. Beyond the fact that there *are* clusters, the PCP cannot present a clear clustering description either. In contrast, the 3-D topological landscape, the 2-D atoll, and the 2-D landscape profile shown in Figures 4.18c-e have exactly the same structure like before and thus still provide the same insights about the clustering. Individual clusters can still be identified and compared, and the data glyphs for each cluster can still distinguished, annotated and selected for further analysis.

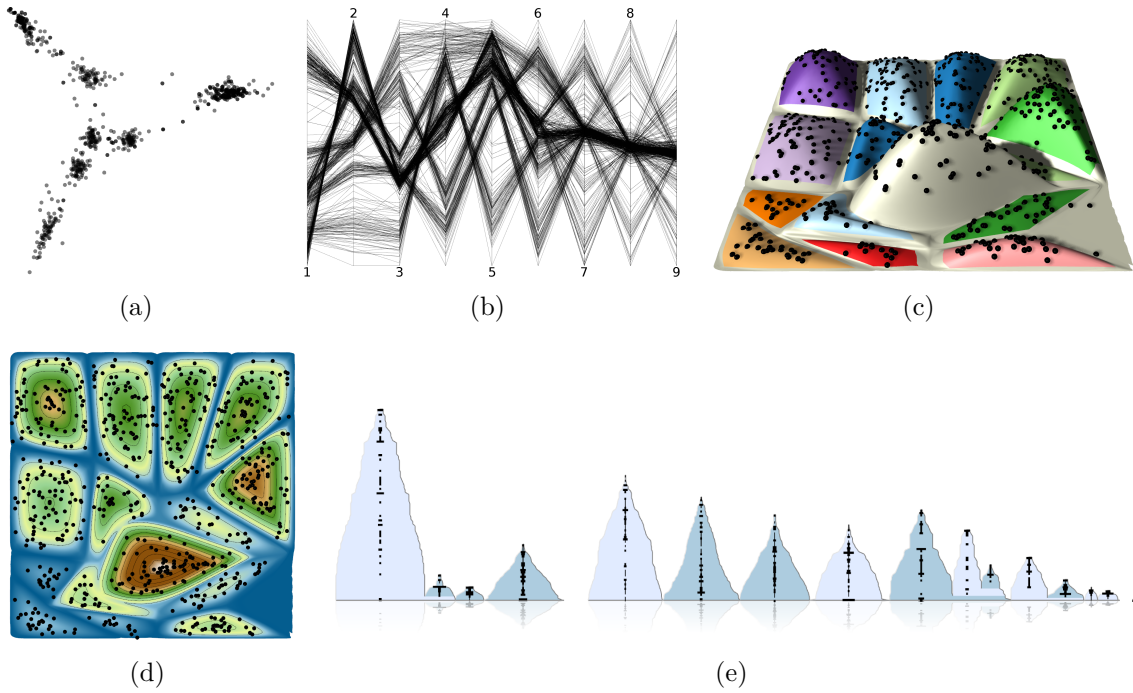


Figure 4.18: 9-D Reuters data set without classification information: (a)-(b) The uniformly colored Rank-2 LDA and PCP do not contain clusters of mixed classes anymore. Hence, further analysis of suspicious features is not stimulated, which leads to wrong insights about the clustering. (c)-(e) Because they only focus on structure, the topological landscape variations provide the same clustering insights if classification information is unavailable.

The analysis of the Italian olive oils data set showed that cluster hierarchy as well as cluster properties were quite misleading in the PCA/PCP and that cluster identification was primarily based on colors (cf. Figure 4.14 on page 90). Without classification information, the single-colored PCA and PCP in Figures 4.19a-b suggest fewer clusters than before in the colored versions and the analyst cannot distinguish these features based on spatial closeness anymore. That is, the point distribution in the projection can only suggest that these points are either in the same cluster or at least in a subcluster relationship. The topological landscape profile in Figure 4.19c, on the other hand, still provides the same insights about the clustering.

The situation gets even worse for noisy unclassified data. As shown for the PCA and the PCP of the artificial 100-D data set in Figure 4.20, the only useful visualization is the transparency-optimized projection. Nevertheless, because color support is missing, the raw point distribution in this image can only indicate half of the truly available clusters. Figure 4.21 provides several landscapes visualizations with and without showing the data glyphs. In both cases, the overall clustering structure and individual cluster properties are clearly visible. While showing all

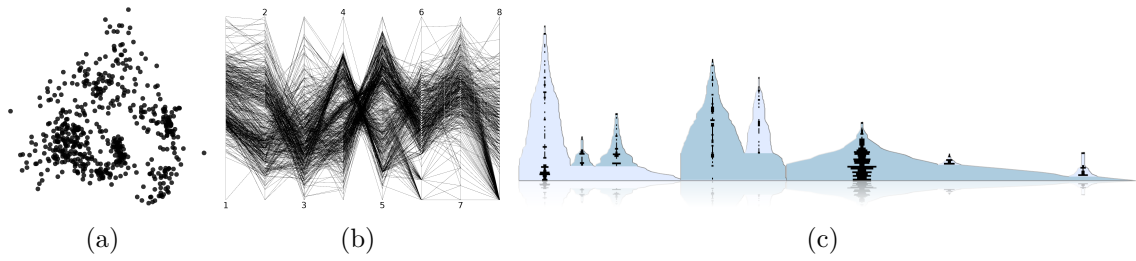


Figure 4.19: 8-D Italian olive oils data set without classification information: (a)-(b) The uniformly colored PCA and PCP do not contain accumulations of different colors anymore. The spatial distribution of the points and polylines can only suggest fewer clusters than before because color is unused as an information channel. (c) Because topology-based visualization does not rely on colors to communicate structure, the landscape profile provides the same structural insights if classification information is unavailable.

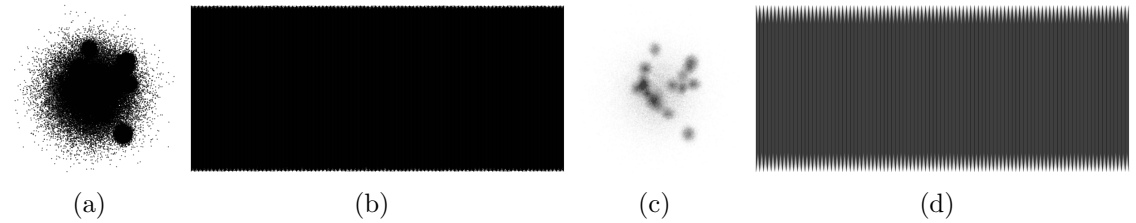


Figure 4.20: Artificial 100-D data set without classification information: (a)-(b) Drawing pixels with 50% transparency for the PCA and the PCP does not reveal the clustering. The projection only suggests one big cluster with a few subclusters. (c)-(d) For 99% transparency the noise vanishes in the PCA, but the projection still shows only half of the clusters contained in the data. The PCP does not benefit from transparency optimization.

data glyphs could make it more difficult to identify cluster separation in the 3-D landscape and the 2-D atoll, structural occlusion or clusters covered by noise are still no problem and the color of the data glyphs is also not required to obtain structural insights. The landscape profile shown in Figure 4.21e provides the most accurate global overview. Changing classification information would only change the color of the histograms.

### 4.5.3 Application 1: Visualization of Document Collections

In Figure 2.1 (on page 9), we already referred to the vector space model to translate text data into high dimensional point clouds. In essence, a high-dimensional document vector is constructed by considering the importance of every word of the total vocabulary for a particular document. Among other approaches, two of the most frequently used strategies are to use the word frequency, i.e., how often a

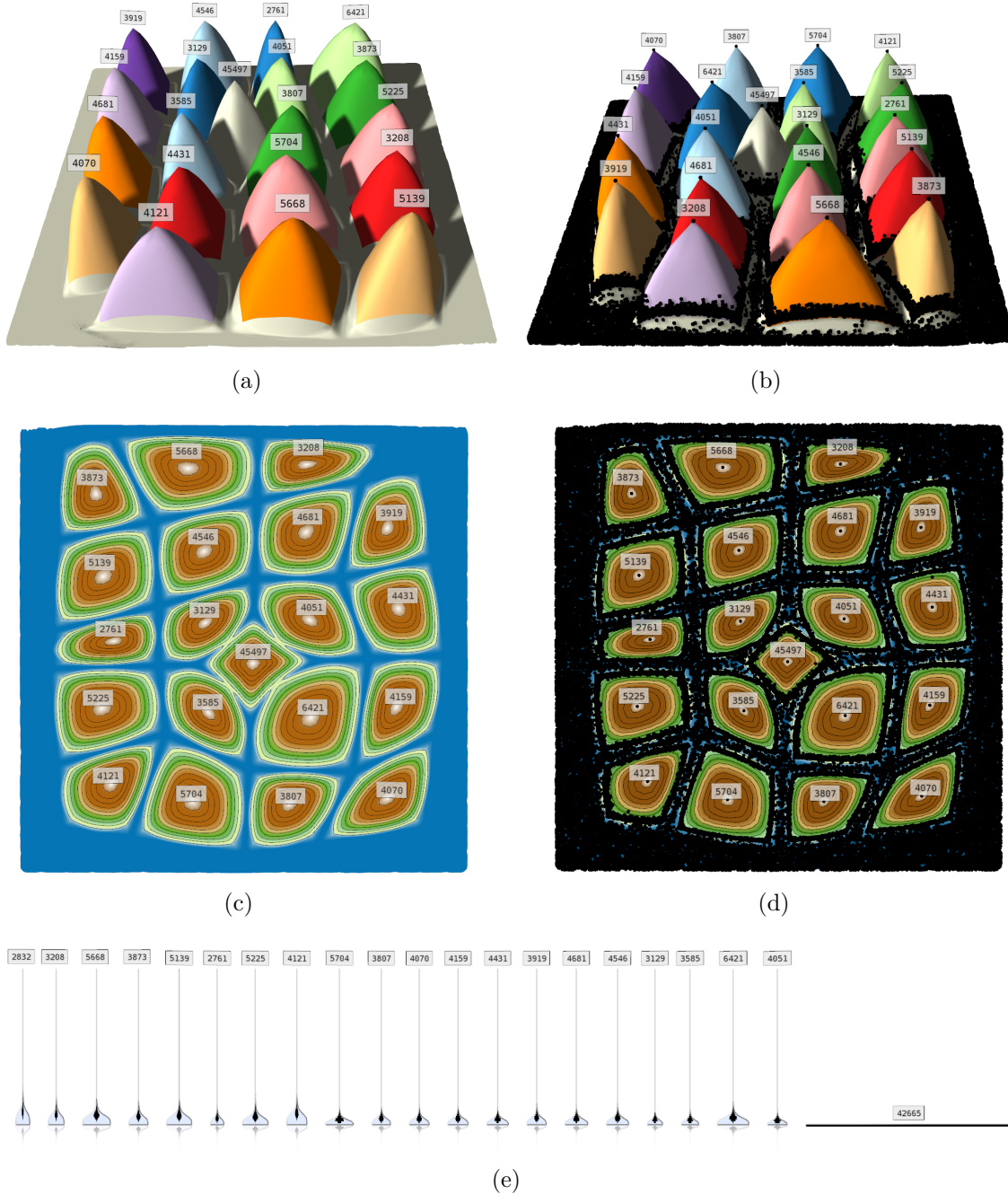


Figure 4.21: Artificial 100-D data set without classification information: (a)-(d) 3-D topological landscape and 2-D atoll visualization with and without augmented data glyphs. While all clusters and their individual properties are clearly visible in both cases, identifying cluster hierarchy and separation is slightly more difficult with the augmented data points. (e) The 2-D landscape profile accurately depicts the clustering structure and individual properties. Augmented histograms do not clutter the visualization.

particular word occurs in the considered document, or to weight the word frequency against the number of other documents that also contain this word; called the *term*

*frequency-inverse document frequency* (tf-idf) [149]. From a semantic point of view, documents cluster in the document space if they share vocabulary and if these words are similarly important for these documents, i.e., if the documents are about the same topic. Clusters and properties like their size, compactness, point distribution, and hierarchy then reveal the number of topics, their overall significance, their generality or preciseness, or the occurrence of sub-topics, like different *ball games* as sub-topics of a *sports*-related topic.

### New York Times Corpus

The *New York Times Annotated Corpus*<sup>1</sup> contains over 1.8 million articles written and published by the New York Times between January 1, 1987 and June 19, 2007 with article meta-information provided by the New York Times Newsroom, the New York Times Indexing Service and the online production staff at nytimes.com. As part of the New York Times' indexing procedures, most articles are manually summarized and tagged by a staff of library scientists.

The example data set consists of documents from the year 2001. We considered 50 random documents per day, but filtered them for 10 different tags; allowing up to 250 documents per tag. This results in 1 896 documents with a vocabulary of 46 393 different words. To avoid issues with the curse of dimensionality in such high-dimensional spaces, but also to accelerate distance calculations and the neighborhood graph construction, we utilize the classification information provided by the tags to project the point cloud with linear discriminant analysis (LDA) to an intermediate  $(t - 1) = 9$ -dimensional space, with  $t = 10$  being the number of tags.

Figure 4.22a shows a visualizations of the documents' thematic composition using the atoll-like 2-D landscape metaphor. Islands correspond to topics and their sizes reflect the number of documents for each topic. The (random) colors of the document glyphs reflect their class association. Instead of using the tag-names to label the islands, a semantic analysis of the documents provides a topic descriptor. To this end, we consider the words shared by all the documents on each superarc of the merge tree and use the two most frequent words as a label. This approach takes into account that a single descriptor does not necessarily reflect a topic sufficiently enough. It also provides an example of an application-dependent labeling, implemented as an operator on the merge tree. Figure 4.22b shows the 2-D topological atoll for a larger filter radius and without metric-based distortion. Although the distance between hills in the landscape metaphor does not communicate spatial closeness between the corresponding clusters in the original domain, merging features with a larger

<sup>1</sup><http://www ldc.upenn.edu>



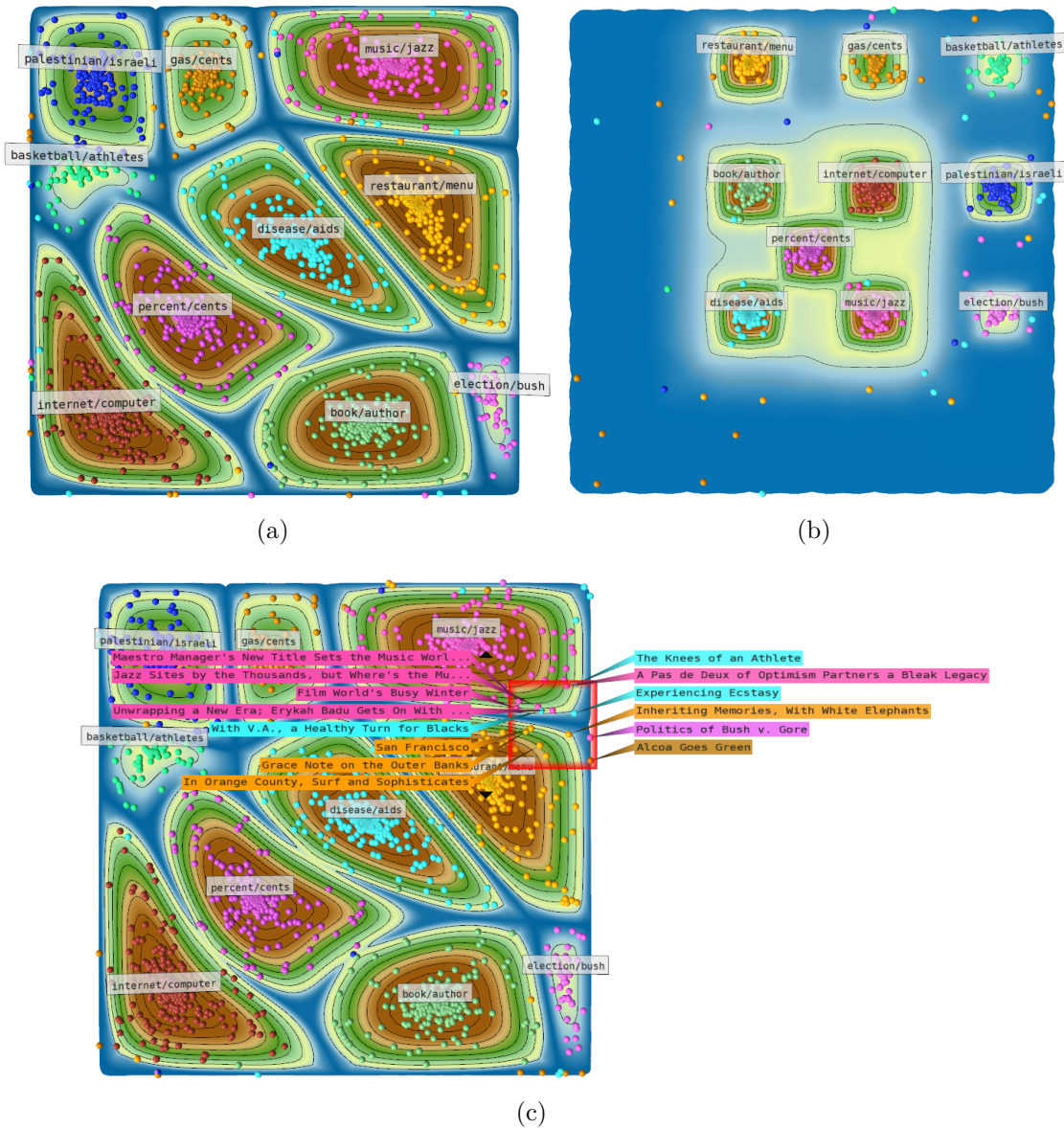


Figure 4.22: Thematic composition of a document collection as a 2-D topological atoll based on the New York Times data set: (a) 2-D atoll with ten islands corresponding to the ten groups of tagged documents. The size of the base areas reflects the number of documents for each topic. Labels indicate the two most frequent words shared by the documents on each island. (b) 2-D atoll for a larger filter radius and without metric-based distortion. The merging behavior caused by using a larger filter radius can approximate which clusters are closer to each other than to other clusters. (c) Excentric labeling of the documents using their original titles.

filter radius can be used to approximate which clusters are closer to each other. As indicated by the shallow water between the islands in the center, i.e. by the non-zero saddle values between these superarcs in the merge tree, these clusters must be closer to each other than to, e.g., the “palestinian/israeli” cluster or the “election/bush”

cluster. Otherwise, they would have merged at the same time or the center clusters would not have merged with for a larger  $\sigma$ . Using different filter radii is a general tool to investigate such issues. However, this approach has to be applied with caution to avoid misleading insights about wrong subcluster relationships; in this example, the clusters are indeed well-separated in the original domain. Figure 4.22c shows an excentric labeling of the documents with their document titles.

### IPC Patent Collection

Access to patent information is of importance for a variety of interest groups today. Besides many other properties, the majority of information describing the nature of a patent is still conveyed through its textual content, therefore making natural language processing a mandatory part of solutions for patent analysis. The sheer mass, complexity, high dimensionality, and heterogeneity of patent data make scalable visual analytics approaches for patent analysis [103] a hard task. One particularly relevant type of meta-information that is available for patent applications is manually assigned classification information. This classification information organizes the vast numbers of patents into predefined classes representing certain technical or functional aspects. Several different schemes for patent classification, such as the International Patent Classification (IPC), Japanese F-terms, and the US classification, exist.

In order to generate an example data set, we test against the IPC comprising more than 70 000 classes, hierarchically organized into sections, classes, sub-classes, main groups, and sub-groups. The final data set consists of 1 552 randomly selected patents from different IPC hierarchies (up to 200 each): A61K...38/17, C12N...1/21, H04Q...7/22, B41C...1/10, C09D...11/00, C09J...7/02, G01N...33/53, H04Q...11/04. We used patent data<sup>2</sup> from the European Patent Office (EPO). As a preprocessing step, the data has been analyzed and the text content was stored in vectorized form within a search index. From this index the tf-idf values for all dimensions of the term vectors have been computed. Linear discriminant analysis is used to project the document vectors into the intermediate 7-dimensional space. First, we examine whether the landscape metaphor reflects the nesting structure of the chosen patent hierarchy. Figure 4.23 shows the 2-D atoll for this data set. Mainly five groups can be identified: The purple and brown glyphs in the upper-right corner, belonging to H04Q patents, clearly address networking, as the labels (and the annotated document titles) relate to *atm*, *address*, *message*, *mail*, and *call*. In fact, the H04Q IPC-hierarchy categorizes patents belonging to “electricity” (H), “electric communication technique”(H04) and “selecting (switches, relays, etc)” (H04Q). Although the group of pink, blue, and

<sup>2</sup>from ‘Text of EP-A documents’ and ‘Text of EP-B documents’

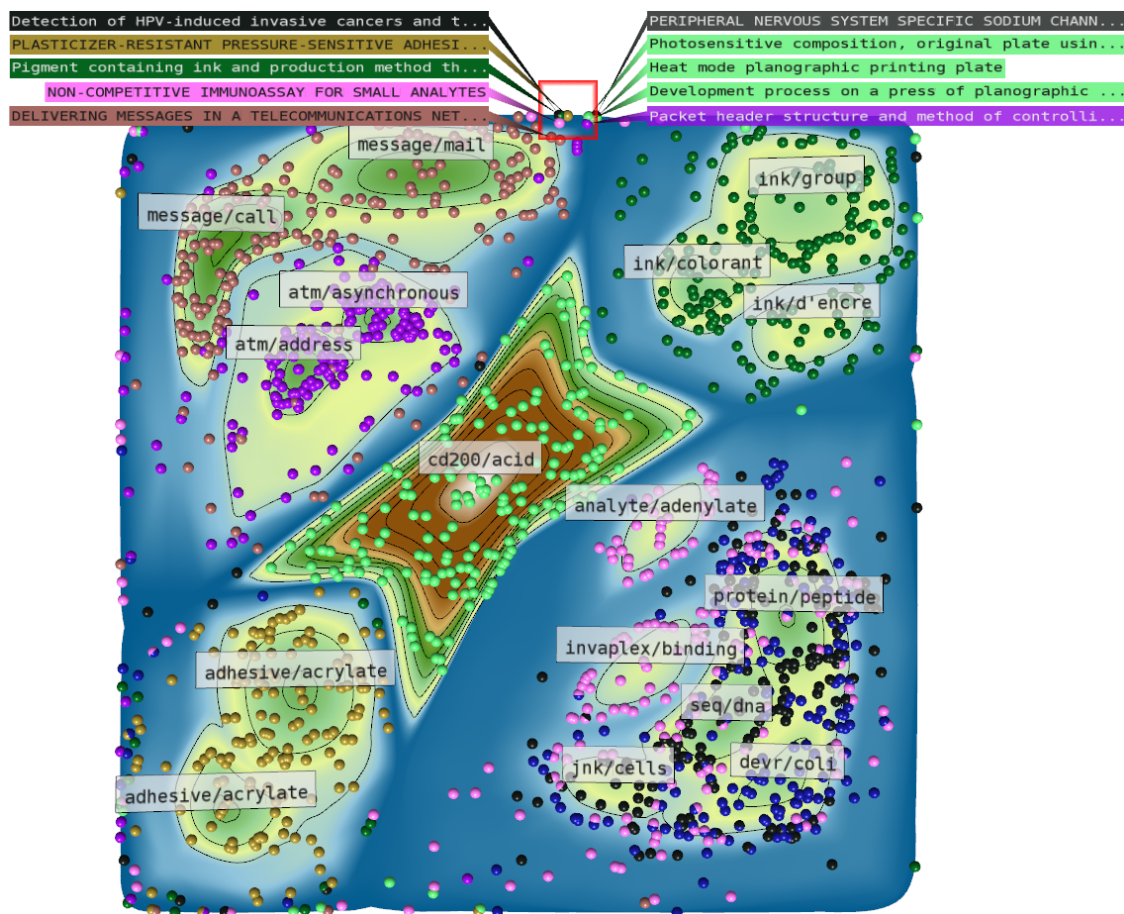


Figure 4.23: Thematic composition of patents as a 2-D topological atoll based on IPC patent data: The nesting structure of the islands in the 2-D topological atoll reflects the IPC hierarchy of the test data set. Five coherent island groups can be identified for patents covering similar sections of the IPC hierarchy. The remarkable shape of the center hill is an artifact of the metric-based distortion of the atoll.

black glyphs on the right-hand side belongs to completely different IPC-sections (A61K, C12N, G01N), the corresponding patents all concern medical issues in their major field: A - Human Necessities, C - Chemistry, G - Physics. Because they share the medical vocabulary, they still constitute one topic in the vector space model. Finally, the centered hill belongs to the B41C cluster and the green and golden glyphs describe clusters related to applications of materials in chemistry and metallurgy (C09D, C09J).

#### 4.5.4 Application 2: Classification of Unclassified Data

Assuming an unclassified entity would match one of the given classes, it is possible to use the topological analysis in combination with a prior LDA projection for classification purposes. Provided that we already have *learned* the LDA projection



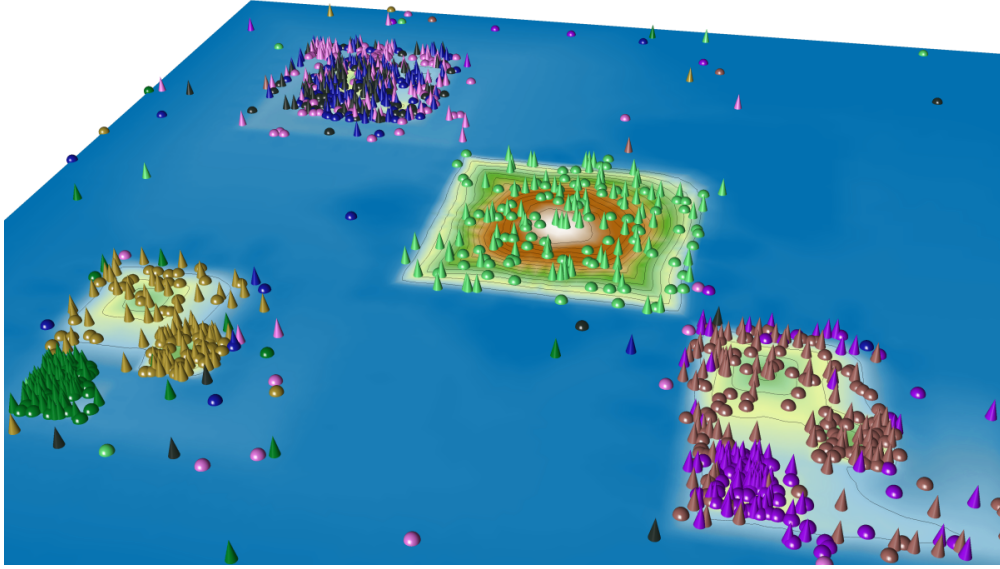


Figure 4.24: Classification of unclassified documents based on IPC patent data: The spheres and the cones represent the training data and the test data, respectively. Both point clouds were projected with the LDA projection matrix that was determined based on only the training data. The (stored) colors of the cones confirm that the test data are projected into the correct regions in the target space. Entities of the test data can be assigned the label of the most occurring class on their islands.

for a given classified training data set, the nature of LDA, as a linear dimension reduction, ensures that similar unclassified vectors are projected into the same target area in the lower dimensional space. While this is an intrinsic feature of a linear projection itself, we can utilize that unclassified data will be part of the clustering of the training data as described by the density function's merge tree. In other words, we can relate unclassified data to the regions in which they are projected using the LDA projection matrix generated for the classified training data set. There are two possible strategies to relate both point clouds: They are either merged into one point cloud or the topological analysis is applied only to the classified point cloud and the topology of the merged point cloud is approximated. Similar to the reinsertion used after sampling, the combined point cloud's topology can be approximated by adding an edge to the neighborhood graph between each unclassified data point and its nearest classified neighbor. After upsampling these edges and determining the densities for the unclassified points, the merge tree is computed as before. In both cases, similar classified and unclassified entities occur on the same superarcs of the merge tree. Hence, we can assign an unclassified entity the label of the most frequently occurring class on this superarc. Another strategy would be to use the reduced set of classified entities on the same superarc and to analyze their content to find the most suitable class in terms of the underlying application.

Since patent offices are interested in automatic classification of new patent applications according to the existing classification schemes, we use patent data to demonstrate the classification. To this end, we split the patents into 50% classified training data and 50% unclassified test data, which means that we henceforth ignore their class label. For demonstration purposes, however, we remember the test data's class association to validate the results in the visualization. After determining the training set's LDA projection matrix, we use it to project both patent sets and use their combination for the topological analysis. Figure 4.24 shows the (undistorted) 2-D topological atoll for the combined data. As reflected by the color distribution of the glyphs, the training data (represented by spheres) and the test data (represented by cones) appear in combination on their own islands. Note that the color of the cones is actually unknown, but was stored for validation purposes. This confirms that patents of a specific class are projected in the same target area if they share vocabulary. In addition to providing means to classify unclassified data, determining the LDA projection matrix on a subset of the data also accelerates the preprocessing necessary to project the input data in a lower dimensional space to avoid problems with the curse of dimensionality. While finding the LDA projection matrix for all documents took 28 seconds in this example, using only 50% of the data to determine the projection matrix took only 12s. To evaluate the quality of the classification, we determine for each branch of the merge tree, i.e. for each island in the atoll, how many of the branch's test data entities were correctly assigned the label of the most occurring training class on that branch. On average, 89.4% of the test data were classified correctly. In particular, in the noisy region of the medicine archipelago, approximately 76.7% were classified correctly and around 99.6% were correct on the remaining branches corresponding to separated clusters.

## 4.6 Conclusion and Discussion

Depending on the complexity and dimensionality of the data, direct visualizations can have severe issues when it comes to accurate depiction of the clustering structure of high-dimensional points clouds. To mitigate problems with structural occlusion, visual complexity, and illusionary (non-existent) structure, we presented topology-based visualizations that build on the topological abstraction of the point data by their density function's merge tree. Utilizing the human's naturally trained ability to read and understand the structure of a terrain, we developed three landscape metaphors: an extended version of the original 3-D topological landscapes [169] suitable to explore and annotate clusters and individual points, a 2-D atoll-like

variation representing clusters as islands and showing more feature properties at the same time, and a novel 2-D topological landscape profile that provides the most accurate depiction of the complete information of the merge tree in one view. The presented variations of the landscape metaphor differ primarily in three aspects: expressive power, accuracy, and computational costs. Unfortunately, these properties seem to be mutually exclusive. While the 3-D landscape is certainly the most natural looking and easiest to comprehend metaphor for quantitative, hierarchical relationships, constructing the landscape and comparing individual hills is more complicated than for the other two variations. Not only is it difficult to compare differently shaped base areas and height information with perspective distortion, the metric-based distortion of the landscape is also pretty expensive and does not always converge to an optimum; leading to strange distortion artifacts for complex data sets. Furthermore, the additional smoothing also increases the triangle complexity and decelerates the data glyph placement. While some of these considerations also apply for the 2-D atoll visualization, this metaphor avoids occluding geometry and makes it easier to compare several feature properties at the same time. The 2-D landscape profile, on the other hand, is probably the least *attractive* or natural looking implementation of the landscape metaphor. However, it conveys the complete information provided by the merge tree (additionally showing a cluster's stability as a third feature property), permits accurate feature comparison, requires the least amount of user interaction to explore the scene, has the simplest and fastest construction scheme, and does not require any post-processing like a metric-based distortion or triangle smoothing step.

Compared to conventional direct visualizations, the presented topology-based landscape metaphor focuses on structure rather than on the underlying data points and on colors provided by classification information. The key idea is to reduce occlusion artifacts by expecting less features than data points to focus on the primary subjects of interest. The intended preservation of high-dimensional distances to let the data points *simulate* high-dimensional neighborhood relationships contradicts this idea because this vast amount of (pair-wise) information can, generally, not be preserved and leads to the typical artifacts. The cost of the topological abstraction, and thus of the topology-based visualization, is that information derived from distances cannot be conveyed by the landscape metaphor. This primarily includes information about absolute positions, inter-cluster distances, cluster shapes, and distances and distributions of the points inside the clusters. This information has to be discarded to obtain an occlusion-free depiction of high-dimensional clustering structure. Still, this loss is considered a price worth paying for if the alternative

is an occlusion-prone visualization and wrong insights. In the end, the major advantage of topology-based visualizations is that geometric information can still be approximated by relating the preserved cluster properties to each other, and that the visualization's clarity is not subject to classification information or the data's benignity and simplicity.

## Chapter 5

# Interactive Visual Analysis

Topological analysis of a point cloud’s density function and the topology-based visualization of the merge tree as a landscape are only core components of a more complex visual analysis framework for high-dimensional point data. Although the landscape metaphor provides an adequate global overview of the clustering and already indicates local details like the density distribution, the visualization itself is only a snapshot of a specific parameter setting. However, suitable parameters are generally not known in advance and need to be determined manually. Furthermore, the analyst is also interested in local details of individual features, e.g. to find out why particular points do or do not belong to the same cluster. The density function’s topology alone cannot provide details about individual dimensions, subspaces, or geometric properties like cluster shape or cluster distances. Therefore, we aim for an interactive analysis tool that utilizes synergetic effects between an accurate, topology-based global overview and local analysis of individual features in linked views to study those data aspects that are not captured by the density function’s topology.

In this chapter, we develop a framework for interactive visual analysis of high-dimensional point clouds and present interactive controller widgets to help the analyst find appropriate parameters to set up a suitable structural overview. We define potential features provided by the landscape metaphor and extend the visualization with selection mechanisms to link subsets to other views for further inspection. Synergetic effects between the topological overview and local analysis in linked views arise from gathering knowledge about individual features that would have been missed by exploring the complete data with the standard techniques alone.

## 5.1 Related Work

**Structural overview and local details.** Separating global structural analysis from local geometric details is in accordance with the classic information-seeking mantra: “overview first, zoom and filter, then details-on-demand”, as proposed by Shneiderman [152]. The utilization of several interactive plots used for brushing and linking of features to other views relates to the concept of multiple coordinated views; Roberts et al. [141] provide an overview. Using these concepts for visual cluster analysis, Fua et al. [61] introduced *structure-based brushes* to navigate through hierarchical cluster trees. They use a special 2-D brushing tool to permit individual selection of subtrees at different levels of detail. In [60] they applied this technique to PCPs by aggregating data as bands of varying translucency. Yang et al. [176] extended this work to support other traditional visualization methods, such as star plots or scatterplot matrices. Johansson et al. [91] proposed the use of pre-clustering to present inherent data structure via high-precision textures and different transfer functions. Novotny et al. [124] used clustering on a binned data representation to combine outliers, trends, and focused data items in an aggregated parallel coordinate plot.

**Visual aids for parameter choice.** The controllers presented in this chapter are based on the idea of *scented widgets* [171], user interface components enhanced with embedded visualizations to quickly judge interesting thresholds. The idea to determine parameters interactively is also in line with *dynamic queries*, as described by Ahlberg et al. [2].

## 5.2 Global Overview and Parameter Widgets

The quality of the topological analysis heavily depends on two aspects: How accurate the density function abstracts the clustering of the input points and how complex the topological description finally is. Both parameters influence the landscape metaphor regarding its correctness and visual clarity—and thus regarding its utility to display the clustering. The accuracy of the density function is controlled by the filter radius  $\sigma$  of the Gaussian filter kernel, whose suitable determination for the topological analysis can be challenging. A too large  $\sigma$  will result in the whole point cloud appearing as one dense region, while a too small  $\sigma$  splits clusters and can even assign each points to its own cluster. Obviously, the correct value is somewhere in between and depends on the data. Another difficulty are noisy data sets, i.e. those containing points and small accumulations outside the clusters. Even if a particular filter radius detects all clusters separately, the density function likely still contains little fluctuations. In the

landscape, this structural noise is represented by small and thin hills that complicate the visualization and make it difficult to identify prominent features.

Because an adequate global overview is crucial for subsequent local analysis, we provide the analyst with interactive widgets to set up the topological view.

### 5.2.1 Filter Radius of the Density Function

If the filter radius  $\sigma$  is too large, the result is a rather blurry density distribution and merges dense regions. Too small, and it creates peaks at small noise accumulations and inside dense regions. Conceptually, we want to find a  $\sigma$  that minimizes the number of hills in the landscape and maximizes their height at the same time. This reflects that each dense region has its own maximum, but noise points do not. One strategy to find a good filter radius is to run the topological analysis for different values of  $\sigma$  and to observe how the landscape changes. This process could start with an optimal  $\sigma_{opt} = 4/(n(d+2))^{1/(d+4)}$  which minimizes the approximate mean integrated square error, if the original point distribution followed a  $d$ -variate normal density [154]. As this assumption is usually not true, the user can then modify  $\sigma$  based on evaluating various properties of the landscape:

1. **The number of hills.** Unless the data actually contains only one cluster, the presence of one large hill or only a few hills is an indicator of a too large  $\sigma$ . Likewise, many small hills with only a few data glyphs on them indicate that  $\sigma$  is currently too small.
2. **The data glyph distribution on the hills.** For separated clusters, the point densities are significantly higher than the saddle densities at upsampled midpoints between the clusters. These points of high density would be located on the upper part of the hills. If many points are arranged at the bottom, i.e. at saddle height, they either represent noise or a too small  $\sigma$  currently splits single regions into several ones.
3. **Suspicious accumulations on the hills.** If the data glyphs on a hill occur in groups at different heights,  $\sigma$  is too large and combines dense regions of different densities. Because a hill's shape also reflects the data point distribution, a too large  $\sigma$  can be expected if a hill contains plateaus at different heights.

Figure 5.1 shows how the 3-D landscape changes for different filter radii based on the image segmentation data set (cf. Appendix A.3). By *reading* the landscape, the filter radius is determined visually: Starting with a too large  $\sigma$  and decreasing it continuously, the analyst watches for noticeable hills of large size and extent

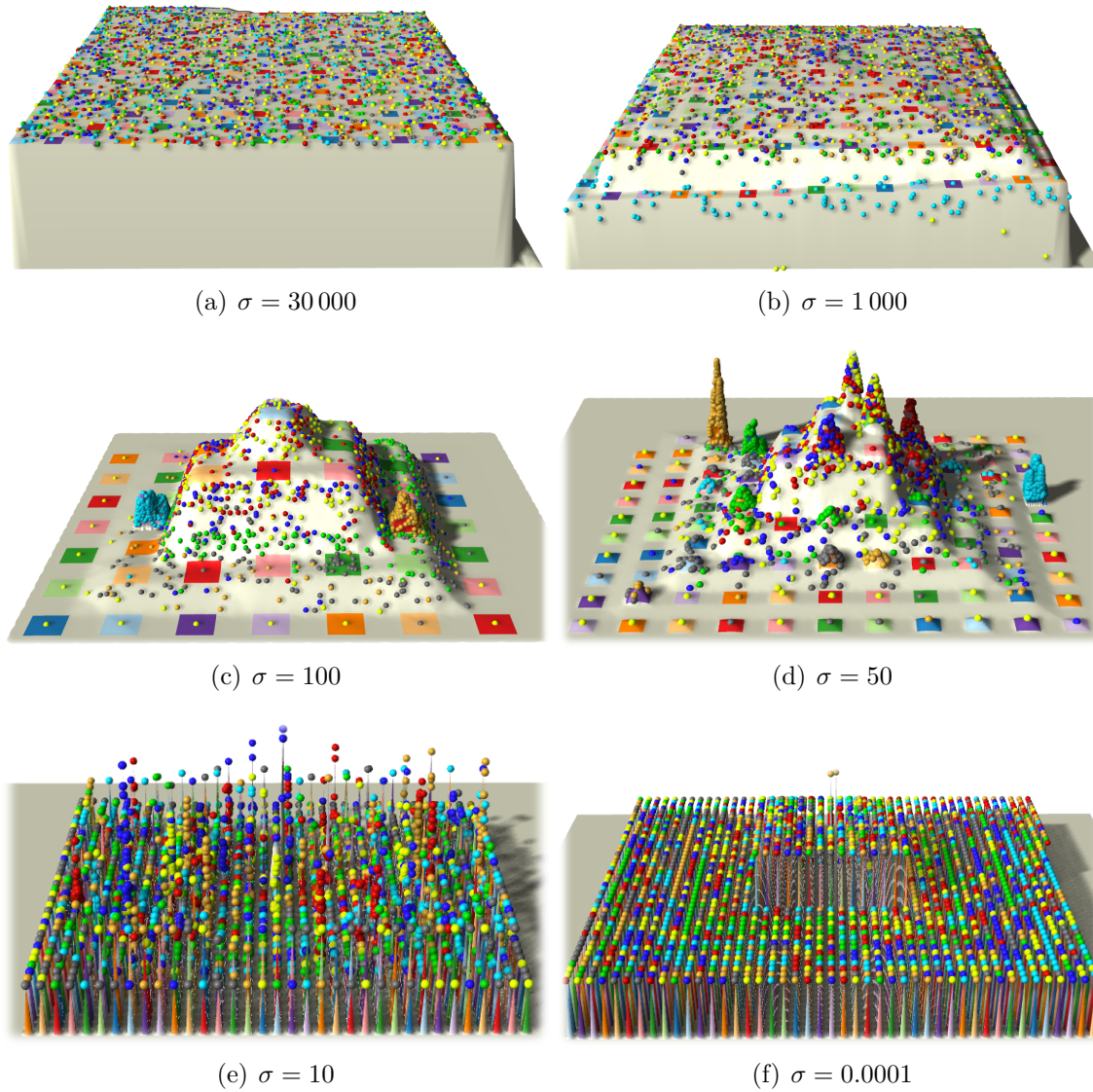


Figure 5.1: Examining the 3-D topological landscape for the 19-D image segmentation data set: (a) A too large filter radius combines all clusters to one dense region. All points have almost the same high density. Flat hills with only one data glyph represent noise. (b)-(c) For clustered data, decreasing  $\sigma$  leads to different densities for clusters and noise. Hills start to appear for the cyan and orange points. (d) A suitable filter radius is found if separated hills occur in the landscape and if a further reduction would only split existing clusters. (e)-(f) For too small values of  $\sigma$ , every data glyph is placed on its own hill, representing that every point is assigned to its own dense region. Note that for aesthetic reasons, height values are rescaled to a percentage of the landscape’s extent. That is, density information is actually decreasing while decreasing the filter radius.

that stand out and are separated by deep valleys from other hills. For the image segmentation data set, this happens for  $\sigma \approx 50$  and is indicated in Figure 5.1d. However, although this strategy is helpful to refine  $\sigma$ , in general, rebuilding the



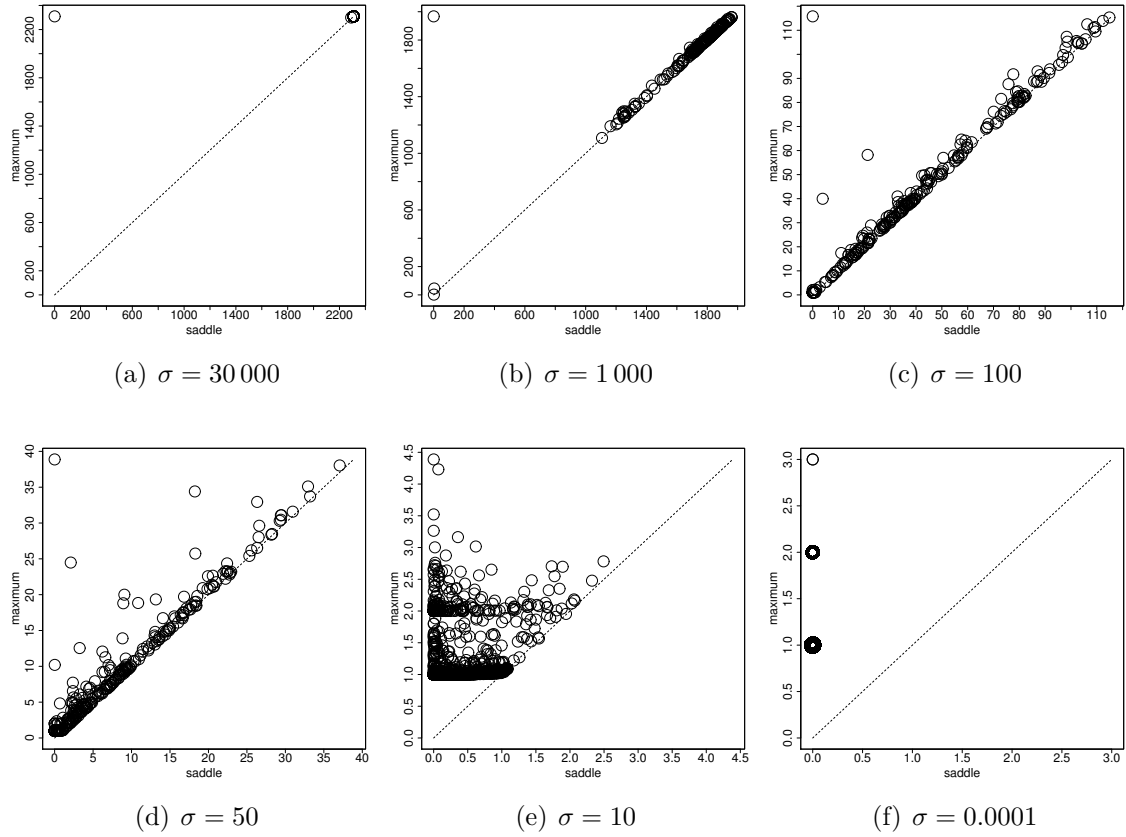


Figure 5.2: Examining the persistence diagram for the 19-D image segmentation data set: (a)-(f) While decreasing the filter radius continuously, branches move along the diagonal and converge to their final position on the ordinate. For clustered data, branches of high persistence depart from the diagonal. Hence, the desired radius is characterized by many persistent branches far away from the diagonal.

landscape several times and inspecting its structure and the glyph distribution is a rather demanding and cumbersome task. Therefore, this strategy is intended for fine adjustment. For convenient determination we aim for parameter widgets that are more intuitive and easier to handle.

### Examining Persistence Diagrams

Instead of evaluating the final landscapes for different values of  $\sigma$ , another strategy is to analyze directly the merge trees of the different density functions. To this end, we consider the merge tree's branch decomposition (cf. Chapter 3.1.2) and display it in a persistence diagram. The changes of the density function's topology are then analyzed by observing the persistence diagram while varying the filter radius between the two extremes. There is a general behavior for clustered data, which is explained based on the image segmentation data set in Figure 5.2.

Starting with a too large filter radius  $\sigma$ , the persistence diagram in Figure 5.2a consists of the root branch represented by a single circle in the upper left corner and multiple noisy branches of near-zero persistence in the upper right corner near the diagonal. After reducing the filter radius in Figures 5.2b-c, the noisy branches start to spread along the diagonal and persistent branches start to depart from the diagonal. This change signifies that clusters become apparent in the point cloud. The low persistent branches indicate very small point accumulations at different density levels and they occur in already found clusters and in those regions where clusters are still combined. A further reduction of  $\sigma$  in Figure 5.2d leads to more branches of high persistence and an accumulation of noisy branches on the lower part of the diagonal. This is a critical situation, as it reflects that separable regions indeed exist. If the data consisted of only one cluster, the circles would just have moved from the right-hand side to the lower bottom without departing from the diagonal. This behavior would reflect that similarly distributed points cannot be separated into several dense regions; unless every point is in its own region. A further reduction of  $\sigma$  (cf. Figure 5.2e) forces the circles to converge towards their final position on the ordinate. Note that the circles do not really move upwards. This is just caused by the changing scale of the ordinate. A circle's final position depends on the multiplicity of the given input points. That is, if multiple points occur at exactly the same position, they would finally have a minimum density depending on the number of duplicates. For example, if there are two points with the same coordinates, both have a density of, at least, two. The diagram in Figure 5.2f reflects the case of a too small filter radius. It assigns each point to its own density maximum and the corresponding circles accumulate at the height of their points' density. For the image segmentation example, the diagram reveals that the data contains many doublets and even some triplets.

Based on these observations, we can provide a guideline for choosing an appropriate filter radius: Varying the value of  $\sigma$  between the two trivial cases, the analyst examines the persistence diagram. If branches depart significantly from the diagonal, this indicates a suitable initial value for the filter radius  $\sigma$ . This value can then be adjusted by analyzing the hills and the glyph distribution in the landscape. Visual determination of  $\sigma$  based on the persistence diagram will be supported by the simplification controller presented in Chapter 5.2.2.

### Filter Radius Suitability Graph

The visual determination of the filter radius with a persistence diagram already yields good results because persistence is usually a good indicator of cluster signifi-

cance. However, the diagram ignores that scattered clusters could also be relevant. These would be missed because large-sized clusters of low persistence would still be represented by circles near the diagonal.

To improve the criterion of a suitable initial radius and to avoid having to inspect several landscapes or diagrams, we construct a function that defines the suitability of a particular filter radius to detect the clustering. A plot of this function then reveals a suitable initial filter radius. The definition of suitability is based on cluster stability. Stability is affected by persistence and cluster size, but also considers the point distribution. Hence, it is more accurate than the product of persistence and size. To evaluate  $\sigma$ 's suitability  $\xi$ , we consider the merge tree  $MT$  of  $\sigma$ 's density function, sum up all superarc stabilities, and normalize this sum by  $\sigma$ :

$$\xi(\sigma) = \frac{\sum_{\forall arc \in MT}^n stab(arc_i)}{\sigma}, \text{ with } stab(arc) = \sum_{\forall r \in R}^n (f(r_i) - f(s)),$$

with  $f$  being the density function,  $R$  being the set of the superarc's implicitly stored regular nodes, and  $s$  being the superarc's saddle node. A plot results from evaluating the suitabilities for different filter radii. Figure 5.3a shows the filter radius suitability graph for the image segmentation data set. Due to  $\sigma$ -normalization, the graph has large values for small radii and small values for large radii. The desired filter radius is characterized by a value near the local minimum of the function, which is  $\sigma \approx 35.0$  for this data set.

The analyst is provided with an interactive graph widget that initially shows the plot for the smallest and largest possible values of  $\sigma$ . These values can be determined based on the shortest neighborhood graph edge and the longest pair-wise distance. The initial graph is a line perpendicular to the diagonal and the analyst refines this graph interactively. The abstraction of the changing topology by a single plot also permits automatic determination of  $\sigma$ . This is achieved by evaluating different radii equidistantly on a logarithmic scale or with a traditional divide-and-conquer approach between the two initial values. The graph is refined automatically at the position where the first minimum is found, i.e. where an evaluation leads to a smaller value than at its two neighboring evaluations. Automatic determination returns either the plot or the value of  $\sigma$  at the local minimum.

### 5.2.2 Simplification of Structural Noise

Structural noise caused by small fluctuations in the density function translates into many small and thin hills in the topological landscape. These features of minor

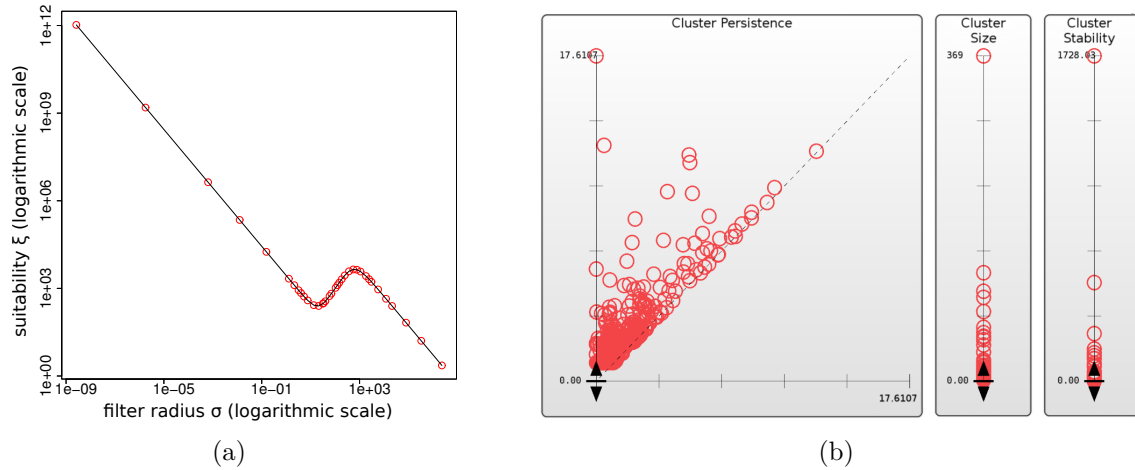


Figure 5.3: Interactive parameter widgets, demonstrated on the 19-D image segmentation data set: (a) In the filter radius suitability graph, the density function’s benignity to capture the clustering is plotted for different filter radii. The desired filter radius  $\sigma$  is characterized by the local minimum of the graph. The analyst refines the plot interactively. (b) Simplification controller with three slider widgets to define minimum thresholds for a feature’s persistence, size, and stability. The distribution of the features in terms of these properties is augmented with circles to the slider widgets. Thresholds are adjusted by moving the sliders on the vertical axes.

persistence and size are typically unimportant and thus simplified topologically to avoid visual clutter and to focus only on the prominent features. According to the simplification process as explained in Chapter 3.2.3, merge tree superarcs are removed if they do not exhibit enough persistence, size, or stability. We provide the analyst with an interactive simplification controller with three *scented* slider widgets for these quality measures.

Scented widgets [171] provide the user with additional information to make an educated guess of an appropriate value for a particular property. For example, such a *scent* could be a value distribution conveyed as histograms augmented to a slider widget to indicate the frequency of different values in the data. Based on this idea, the sliders in the simplification controller display the distribution of the features in terms of their persistence, size, and stability. This allows the analyst to quickly identify interesting thresholds if some features stand out. Because feature properties change during topological simplification, the value distributions are determined based on the merge tree’s branch decomposition. This is helpful because branches already provide a multi-resolution view on the merge tree and thus make the controller’s scent less susceptible to variations between two simplifications. They also indicate which thresholds are necessary to preserve only the prominent features.

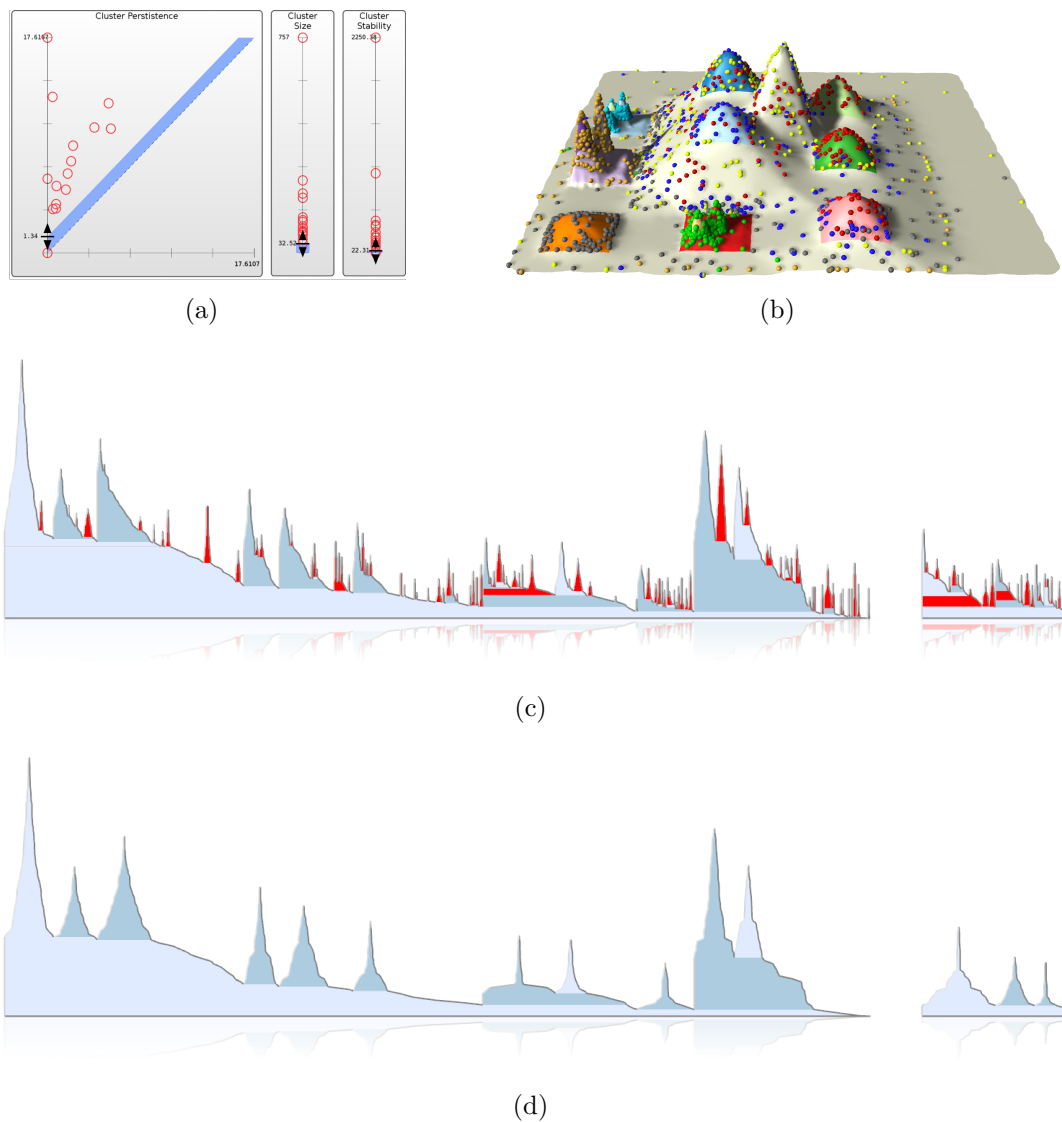


Figure 5.4: Interactive simplification of the 19-D image segmentation data set: (a) Moving one of the sliders in the simplification controller simplifies the merge tree in real-time and updates the value distribution in all three sliders. Currently selected thresholds are accentuated as blue bands. (b) 3-D topological landscape after topological simplification. (c) The simplification controller is also linked to the visualization. If the user drags a slider, the hills that would be removed with the current thresholds are colored red. (d) After releasing the slider, the landscape is reconstructed and shows only the remaining prominent features.

Figure 5.3b shows the interactive simplification controller with the three scented slider widgets. The persistence slider widget is a persistence diagram to distinguish structure from noise by the distance of the circles from the diagonal. The persistence diagram has the advantage to show persistent features for different merge densities along the horizontal  $x$ -axis. This allows the analyst to distinguish persistent subclusters from separated features of lower saddle density.

Simplification thresholds are adjusted by dragging the sliders on the vertical axes. This is demonstrated in Figure 5.4 for the image segmentation data set. Dragging one of the sliders simplifies the merge tree in real-time and updates the value distributions in all sliders accordingly. That is, the circles of preserved superarcs move if their branches vary in their persistence, size, or stability, and the circles of simplified branches vanish from the slider widgets. Current threshold selections are accentuated as blue bands in the widgets. Furthermore, the simplification controller is linked to the landscape visualization. While the analyst moves one of the sliders, those hills that would be removed by simplification are colored red (Figure 5.4c). After releasing the slider, the landscape visualization is reconstructed and shows the structure without noisy features. Figures 5.4b,d show the 3-D landscape and the 2-D landscape profile after topological simplification using the thresholds adjusted in the simplification controller in Figure 5.4a.

The simplification controller is updated whenever the merge tree changes. As this is also the case when the density function changes due to a different filter radius  $\sigma$ , the persistence diagram of the simplification controller is also used to determine a suitable filter radius according to the explanations above (cf. Chapter 5.2.1).

### 5.3 Feature Selection and Local Analysis

The key advantage of the topological analysis over the attempt to preserve high-dimensional relationships in a low-dimensional image is that the density function's topology can be preserved without loss. This preservation facilitates an occlusion-free clustering depiction in 2-D to reveal cluster hierarchy, separation and individual cluster properties. However, the global perspective on the data alone cannot tell *why* clusters have subclusters, or why points of the same class do or do not belong to the same cluster. A hill in the landscape only signifies that *globally* there is a dense region, possibly containing local density maxima. To obtain information beyond this global perspective and to learn more about the semantics of the data, the analyst has to study in which dimensions or subspaces similar data entities differ *locally*.

Reliable identification of truly occurring features and convenient selection of arbitrary subsets are important requirements for proper visual analysis of high-dimensional data. However, these requirements are not ensured when features occlude each other or if they are covered by noise, which is often the case in projections or axis-based techniques. Moreover, if important features are missing, further local analysis is not stimulated, which leads to misleading or wrong insights. From that point of view, the topology-based global perspective on the data provides a suitable

overview to permit local analysis on-demand because all features can be identified, compared and selected reliably and conveniently in one view.

The ability to select individual features permits a linking of subsets to other views. The idea behind this linking is a thorough analysis of those aspects that are not caught by the topological view, e.g. to study a cluster's approximate shape with a projection or to inspect the value distribution in a few dimensions using a parallel coordinate plot. Linking only subsets also improves the quality and reduces the visual complexity of these standard techniques substantially if they can focus on fewer points.

### 5.3.1 Features in the Landscape

Potential *features* provided by the visualization are categorized into two groups: those described by the landscape and those indicated by data glyphs. In the remainder of this chapter, we focus on the 2-D topological landscape profile.

The most obvious feature is a hill representing a (sub)cluster. Hills are always visible separately and stimulate local analysis because the analyst can read and compare their properties efficiently. For example, cluster sizes are compared easily by the widths of their hills, which also include point duplicates. Cluster hierarchy or well-separated clusters can be identified quickly based on inspecting the valleys between the hills. The presence of noise, which is also a potential feature to learn about unknown data sets, is indicated by valleys at non-zero height or as an individual histogram at zero height in the 2-D landscape profile after simplification. Suspiciously shaped hills also indicate a feature. Especially those hills containing plateaus often indicate subclusters of different densities that are still combined by a too large filter radius. Hence, for clarification, the data could be analyzed at higher resolution.

Noticeable accumulations and the overall distribution of the data glyphs on the landscape can also reveal interesting features. Although data glyphs always appear in the 2-D landscape profile whenever a hill grows wider, the occurrence of glyphs is generally easier to perceive than a hill growing in its width. Because height information reflects densities, the user can easily identify accumulations of a certain density. Likewise, the presence of noise is quickly detected at the foot of a hill or at the bottom of the landscape. If classification information is available, potential features are the relation between the hills and single-colored glyphs, or suspicious accumulations of individual or mixed colors, e.g. when points of one class occur at several hills.

### 5.3.2 Selection and Linked Views

There are multiple mechanisms to select interesting features or arbitrary subsets. While most of them apply for the landscape metaphor in general, some are easier to implement for the 2-D landscape profile. Therefore, we introduce and explain the interactive selection based on the 2-D profile variation of the metaphor.

Clusters and subclusters are selected by clicking directly on a hill in the profile. Individual data points or arbitrary point sets are selected by drawing a rectangle around their glyphs. Multiple selections can be concatenated. If classification information is available to color the data glyphs, selected points are linked together with their assigned color. This way, they can be related in different views, e.g. to discover a particular cluster in a projection.

An advanced functionality is to assign different colors to individual selections. This makes it possible to distinguish selections in linked views if the data is unclassified or if subsets of points of the same class should be compared. To indicate the color of a selection in the landscape, either a hill's color changes after clicking on it or the selection rectangle is filled with that color. A set of discriminable colors can be generated algorithmically and should also consider the colors already assigned to the classes. Finally, the histograms and stacked bar charts for classified data support a bar-wise picking to select different accumulations of a particular density and class.

#### Linking to Axis-based Techniques

Axis-based techniques like star plots or parallel coordinate plots allow the analyst to study the homogeneity of the points throughout many dimensions. However, for many points, the visualization quickly becomes complex and suffers from occlusion. This is why the number of polylines has to be limited and subsets have to be accentuated to avoid overloaded drawings. Utilizing the global overview of the topological landscape accomplishes both goals by linking only selected features of interest.

For axis-based techniques, focusing only on subsets substantially reduces the number of crossing polylines and thus the visual complexity. There is an implicit correlation between the topological landscape and axis-based techniques: Hills in the landscape correspond to polyline bundles in axis-based visualizations. This is simply because points of a global cluster necessarily have to share similar values throughout many dimensions. To analyze why a particular cluster consists of subclusters or why points of the same class are not in the same cluster, we link selected features to an axis-based technique, e.g. a parallel coordinate plot.

Figure 5.5 shows an example of local feature analysis with a parallel coordinate plot based on the image segmentation data set. Utilizing the occlusion-free depiction



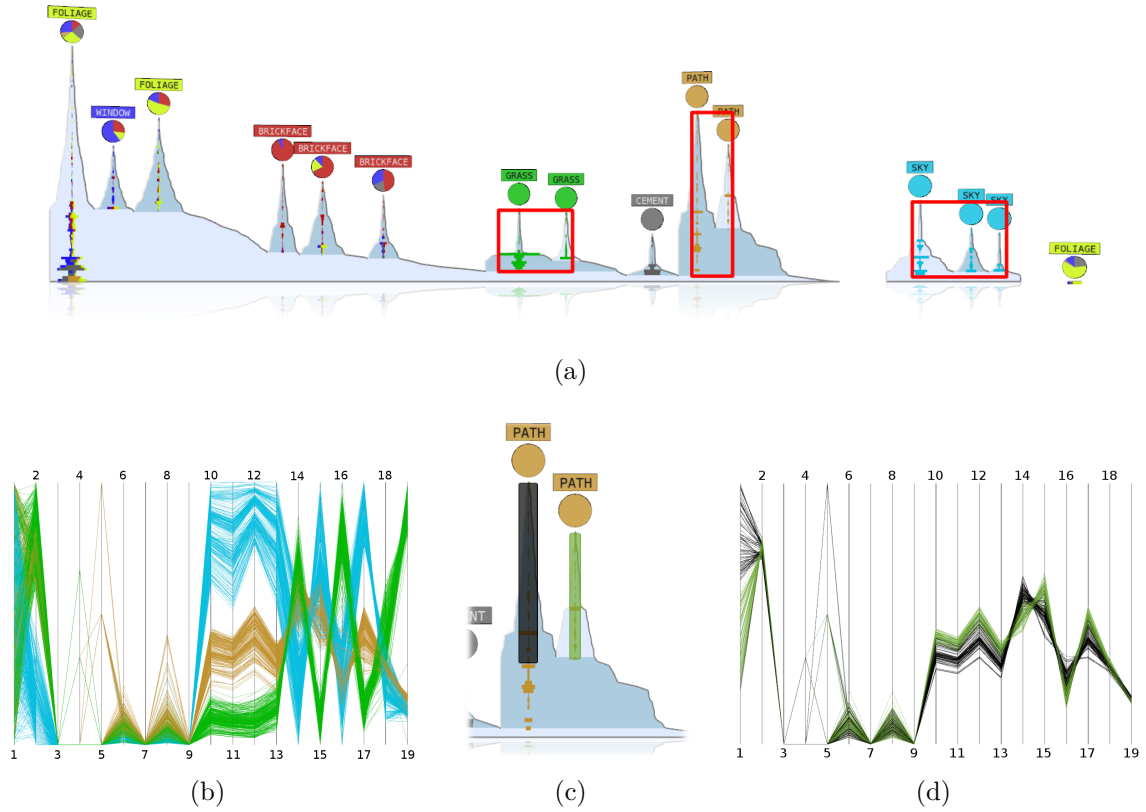


Figure 5.5: Local feature analysis by linking subsets to a PCP-view based on the 19-D image segmentation data set: (a) In the 2-D landscape profile, clusters and point sets are selected by clicking on the hills or by using selection rectangles. (b) Focusing on only a few features substantially reduces the visual complexity of a PCP and thus simplifies feature comparison. (c) Using different colors for individual selections helps to analyze subsets of the same class and reveals (d) in which dimensions the points differ and why the cluster splits into subclusters. This accentuation of polyline bundles also works for unclassified data.

of clustering structure in the 2-D landscape profile, the analyst can easily pick some clusters with individual selection rectangles (cf. Figure 5.5a). As shown in Figure 5.5b, the PCP of only the selected points is much clearer. Because it focuses on only a subset, it is easy to inspect in which dimensions these points differ to understand why they do not belong to the same cluster. To analyze subclusters of a single class, Figures 5.5c-d demonstrate the advantage of using different colors for each selection. By using two different colors for the two hills of the brown “path”-cluster, we can distinguish these selections in the linked view. The PCP reveals that even though these points have the same global trend, in some dimensions their values are widely spread ( $d_1$ ), shifted ( $d_{10}$ - $d_{13}$ ,  $d_{16}$ - $d_{17}$ ) or even inverted ( $d_{14}$ - $d_{15}$ ). Note that for unclassified data, assigning different colors to individual selections is helpful to point out polyline bundles in an otherwise monochrome PCP.

## Linking to Projective Techniques

Unsupervised projections, like the principal component analysis (PCA), consider the input data as a whole and typically try to find an embedding that is best on average for all given points. As a consequence, specifically for the PCA, the largest variance of multiple clusters in one image typically does not reflect the largest variance of each individual cluster. That is, properties like distances and shapes of the features of interest are inaccurate if other points take part in the projection. Even for low-dimensional well-separated clusterings, a 2-D projection likely contains overlapping regions and distorted shapes if the clusters are oriented just differently enough. Therefore, in order to minimize the projection error, the optimization criterion should be applied to only few points. Utilizing the global overview provided by a topological landscape helps to accomplish this goal by selecting only meaningful features and subsets.

Representative for other projective methods, we demonstrate linking selections to a PCA because it is a widely used and intuitive projection. Typically, other projections are also more accurate if unselected points can be discarded. Particularly for the PCA, we can also specify the projection error as the variance that is not explained by the first two principal components. This information quantifies the reliability of the projection between several selections. This is helpful because for high-dimensional data the projection error can still be rather large for only few points. Restricting the data with the topological overview at least keeps this error small.

Local feature analysis with a projection to learn more about a cluster's approximated shape, extent, point distribution or distance to another cluster is demonstrated in Figure 5.6. For example, in the PCA projection of the complete data set (cf. Figure A.3a on page 200), the cyan “sky” points are depicted with only around 84% of their variance in the original domain. To improve the projection quality of only these points, we can simply click on each of the three hills in the landscape (cf. Figure 5.6a) to select and link the corresponding points with different colors to the PCA-view. This increases the explained variance of these points in the PCA to approximately 92% in Figure 5.6b. Due to the different colors of the selections, these subclusters can also be distinguished visually in the projection—even if no classification was available.

The plateaus and histogram accumulations on the leftmost “sky” hill suggest that even more subclusters could be separated. Although the first two principal components account for around 92% of the cyan points' original variance, the first three principal components account for almost 99.3% variance. Because the green points in Figure 5.6b do not reflect local accumulations, this could be a projection

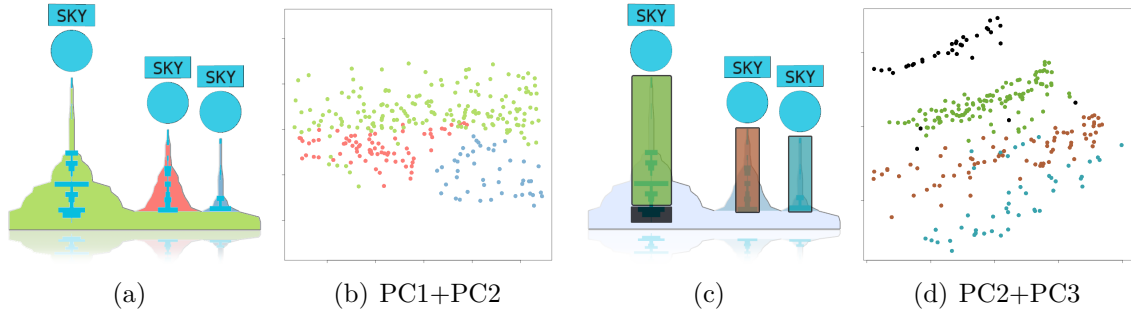


Figure 5.6: Local feature analysis by linking subsets to a PCA-view based on the 19-D image segmentation data set: (a) Clicking directly on the hills selects their points and changes the color of the hills to that of the selections. (b) Because the optimization criterion is applied to only these points, the projection error decreases if less points take part in the projection. The linked points are colored according to the selections, which allows the analyst to distinguish otherwise equally colored points. (c) Suspicious plateaus and histogram accumulations can be analyzed in further detail if they are selected with rectangles of different colors. (d) Projecting these points on the second and third principal component results in a scatterplot in which these points are indeed separated.

artifact and the desired separation must be hidden in the third principal component. In fact, selecting the histograms with different colors (cf. Figure 5.6c) and projecting these points onto the second and third principal component yields a projection that reflects this separation. The PCA did not consider this view on the cyan points because it features less data variance than Figure 5.6b. However, the landscape visualization suggested further investigation to obtain more separation in the linked projection.

## 5.4 Visual Analysis Framework

To facilitate accurate and convenient analysis of high-dimensional point data, we combine the presented algorithms, metaphors, and mechanisms into an interactive visual analysis tool. This framework implements the algorithmic core of the topological analysis, the topology-based visualization of the merge tree, the widgets to support the user in finding appropriate parameters, and the selection and linking mechanisms to explore the data interactively and on demand. This section describes the prototype implementation and the typical workflow of the analysis process.

### 5.4.1 Prototype Implementation

The prototype is implemented in *OpenWalnut*<sup>1</sup> [50, 49], an open source visualization tool intended for scientific visualization and research. Although OpenWalnut’s focus is on multi-modal medical and brain data visualization, its versatile and flexible design and architecture make it useful for a large variety of applications. OpenWalnut follows a simple, yet effective modular paradigm. Modules are responsible for individual tasks performed on input data fed to or loaded by the module itself. Furthermore, modules are connected to a data-flow network. There are three module types: *sources*, which only provide data on their output connector; *filters*, which read data from their input connector, process it and provide the result on their output connector; and *sinks*, which only read data from their input connector and, generally, visualize it. If data on a connector is updated, so are all connected modules—thus triggering updates recursively. Every module can also create GUI-widgets to receive input from the user, like sliders, buttons, item menus, color choosers, or input fields for several numeric types. Modules can also create views to render graphics.

Figure 5.7 shows a screenshot of the visual analysis framework implemented in OpenWalnut. The main window is separated into three major areas: the module graph (top right), the control panel with GUI-widgets for the currently selected module (bottom right), and multiple graphic views maintained by the modules (left). The module graph shows the different module types with different colors and displays the input- and output connectors at the top and on the bottom of the modules, respectively. The currently activated module is highlighted by a dotted frame and the control panel shows the GUI-widgets of this module. Only one module can be selected and configured at a time.

The module graph for the topological analysis is straightforward and requires only few components: A data loader module (blue) reads data from input files and generates high-dimensional vectors with meta-information. This information is passed to the topology module (upper gray box), which constructs the merge tree. This module also has a second output connector to provide intermediate results for potential visualization with other modules. The simplification module (lower gray box) transforms the merge tree into the simplified merge tree. The visualization modules (green) generate the variations of the landscape metaphor based on the simplified merge tree and the meta-information (like labels, colors, etc) provided by the data module. The second output connector of the simplification module provides pseudo-simplification information, e.g. which leaf superarcs would be removed while the sliders are dragged. This information is necessary to provide interactive previews

---

<sup>1</sup><http://www.openwalnut.org/>

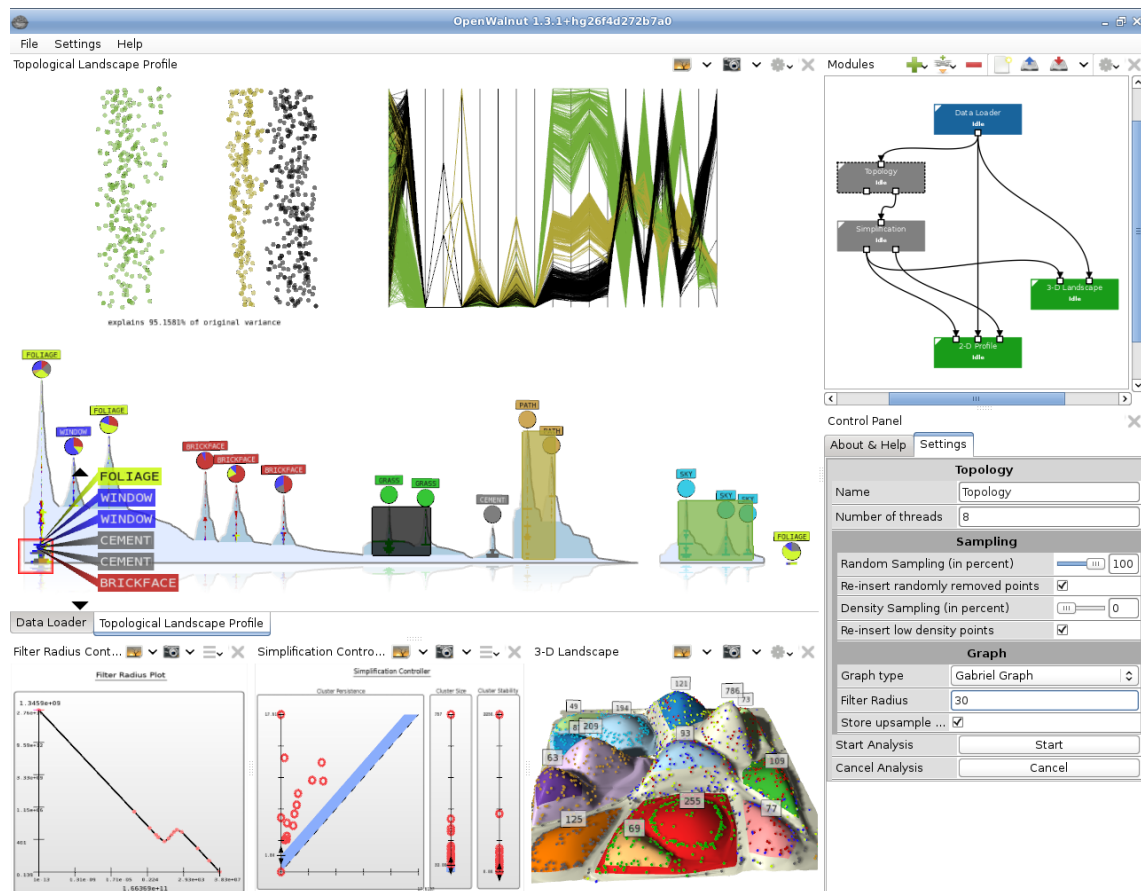


Figure 5.7: Overview of the analysis tool implemented in OpenWalnut: The main window consists of the module graph (top right), the control panel (bottom right) and the graphic views maintained by their respective modules. The module graph consists of a data loader, a topology module, a simplification module, and a 2-D and 3-D landscape visualization module. Arrows indicate the data flow between the modules. The control panel shows GUI widgets for the parameters of the module that is activated in the graph; here the topology module. The filter radius suitability controller and the simplification controller belong to the topology module and the simplification module, respectively. In this implementation, the PCA and the PCP belong to the 2-D landscape view. Selecting any part of the profile updates both linked views. Changing a parameter in any module or interactive controller triggers an (recursive) update in all connected modules.

as demonstrated in Figure 5.4. Note that the simplification module is optional. Because it has the same output type, the unsimplified merge tree from the topology module could also be connected directly to the visualization modules that expect a merge tree on their input connectors.

Module parameters are configured in the control panel. In Figure 5.7, the control panel currently shows the parameters of the topology module. These are basically the parameters explained in Chapter 3.4, i.e. thresholds for sampling and reinsertion,

the graph types or the filter radius. The filter radius and the three simplification thresholds are configured either in the control panel after activating the respective modules, or in the graphic views hosted by these modules. The views of the filter radius suitability graph and the simplification controller are placed at the bottom. Just like changing the value of a GUI-widget in the control panel, interacting with these controllers updates the module's output connector; and thus its connected modules. The main view in the middle belongs to the 2-D landscape profile module and the view of 3-D landscape module is placed next to the simplification controller. In this prototype implementation, the PCA view and the PCP view belong to the profile module. They are only displayed if the analyst selects any part of the landscape profile.

### 5.4.2 Typical Workflow of the Analysis Process

The visual analysis process follows the information-seeking mantra as defined by Shneiderman [152]: “overview first, zoom and filter, then details-on-demand”. At first, the analyst creates an appropriate global overview based on the landscape metaphor. This includes finding a suitable filter radius and thresholds to suppress structural noise with topological simplification. Afterwards, the analyst inspects the clustering and compares significant clusters based on the characteristics of the hills in the landscape. Local feature analysis is performed by linking subsets to standard techniques to learn more about the semantics of the data. The user loops through these steps several times during the analysis.

An exemplary workflow is as follows: The analyst constructs (or loads) the module graph as described above (cf. Figure 5.7). Initially, all graphic views as well as the filter radius controller and the simplification controller are empty (cf. Figures 5.8a-b). After loading a data set, the analyst activates the topology module and specifies the neighborhood graph type and sampling thresholds in the control panel. A suitable filter radius is then determined with the controller widget: At first, the widget is initialized for the current data set by a random click in the controller. This evaluates the suitabilities for the two extreme cases, i.e. a too small and a too large value for  $\sigma$  (determined based on pair-wise distances), which leads to a line perpendicular to the diagonal (cf. Figure 5.8c). The graph is now refined either automatically or manually. The evaluation for a single filter radius can be accelerated by using a sparse neighborhood graph and by using only a sample of the data without reinsertion of non-samples. This reduces the time to find a suitable  $\sigma$  because the topological analysis has to be applied multiple times. Once the approximate position of the local minimum has been found, a few more evaluations at higher accuracy

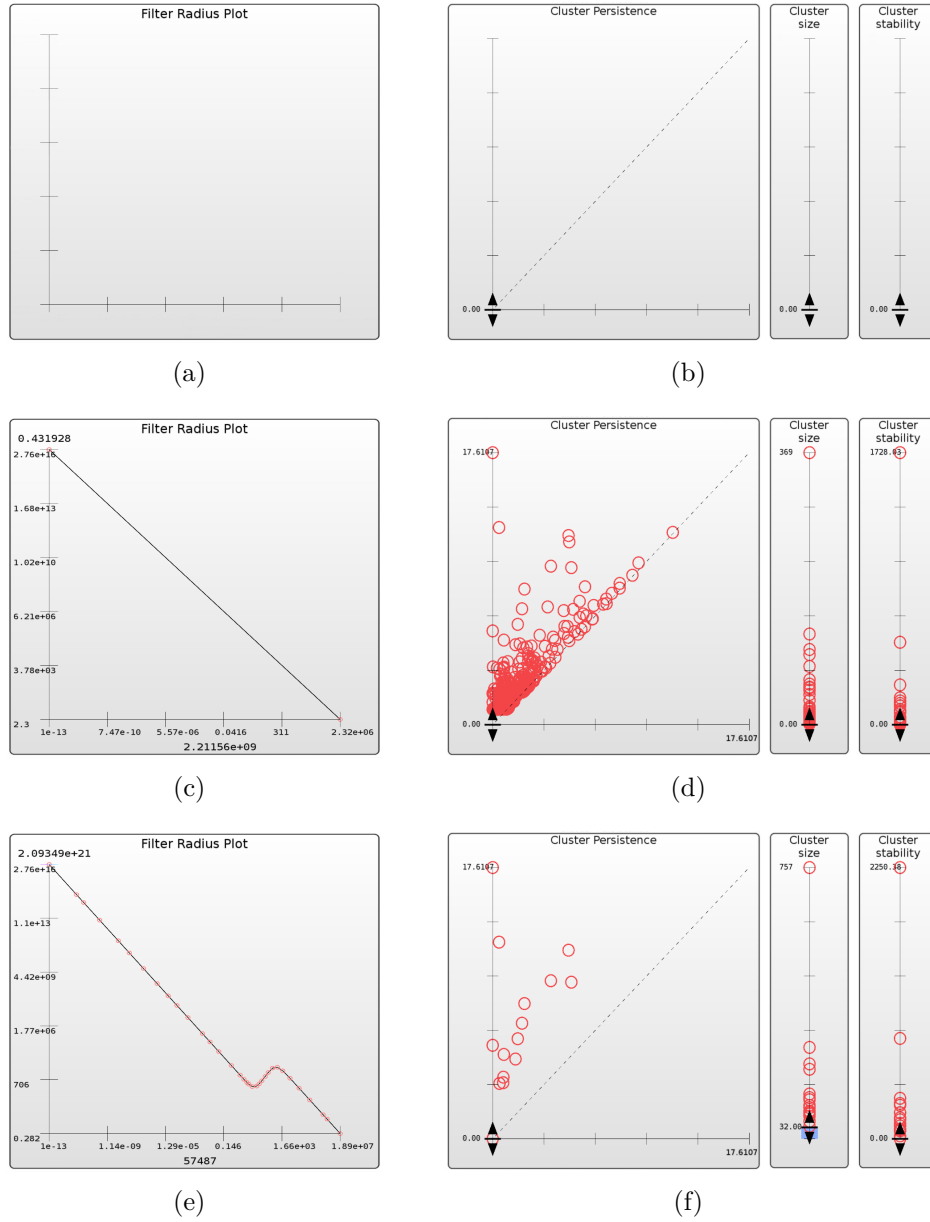


Figure 5.8: Different states of the controller widgets based on the image segmentation data set: (a)-(b) Initially, the filter radius plot and the three sliders of the simplification controller are empty. (c) After loading the input data, the suitability graph is initialized for the smallest and largest possible values of the filter radius  $\sigma$ . (d) When the merge tree changes, the three sliders in the simplification controller are initialized to show the value distribution of the branches for the unsimplified tree. (e) To find the local minimum, the suitability graph is refined automatically or manually for different values on the  $x$ -axis. (f) Dragging one of the sliders in the simplification controller updates the value distribution in all three sliders in real-time. The desired thresholds leave only prominent features with high property values. In the persistence diagram, these are the circles far away from the diagonal.

refine the plot near this location. Alternatively, the graph could be precomputed automatically at higher accuracy if time is not a critical factor. The filter radius controller supports two clicking modes: Right-clicking in the graph only refines the plot without updating the module’s output connector. This avoids consecutive updates of connected modules during the determination of the filter radius. Left-clicking the graph additionally updates the merge tree on the output connector of the topology module. That is, if the user left-clicks on the graph near the local minimum (cf. Figure 5.8e), the connected simplification controller automatically initializes the three sliders with the respective value distributions of the merge tree’s branch decomposition (cf. Figure 5.8d). The focus now switches to this controller to remove noisy features in real-time. The analyst adjusts the sliders so that only those branches with high values for persistence, size, or stability (cf. Figure 5.8f) remain. In the persistence diagram, these are the circles far away from the diagonal. While dragging a slider, the simplified merge tree is updated on the module’s output connector. This triggers an update of the connected visualization module(s). If the landscape was already constructed before, changing the slider in the controller or in the control panels highlights remove candidates by changing the color of these hills to red. If the user releases the slider, the profile is reconstructed for the simplified merge tree. The analyst can refine parameter settings by reading the landscape, e.g. by looking for noticeable plateaus or suspicious data glyph accumulations that suggest a smaller filter radius to split a cluster.

The global clustering overview is stable and robust in that little adjustments of any parameter do not lead to significant changes in the landscape. That is, moderate changes of the filter radius or the simplification thresholds only add or remove some small hills without changing the profile’s overall structure. To find a suitable overview, typically 7-10 refinements were necessary in our experiments to find the local minimum of the plot. Depending on the parameter setting, each single evaluation typically takes around one second for the data sets used in this thesis. However, the evaluation of a  $\sigma$ ’s suitability takes longer for larger data sets or disadvantageous parameter settings. Changing simplification thresholds to reveal or hide small features is typically fast and can be applied in real-time. Once an appropriate overview has been found, the local analysis phase starts with the identification of interesting features and their linking as described in Chapter 5.3. Linking selected subsets to another view and creating the PCA or the PCP is also fast and happens in real-time for medium-sized data sets. Still, projecting high-dimensional data can quickly become expensive depending on the applied projection technique and its implementation.



## 5.5 Examples and Results

In this example section, we focus on the interactive analysis process and on local feature inspection. We revisit some of the real-world data sets from the result section in Chapter 4.5 about the global overview and explore features individually. This aims to achieve a better understanding of the topological view on the data, to reveal features that are invisible in standard techniques, and should demonstrate how the structural view and geometric details complement each other for in-depth analysis of high-dimensional clusterings. After reviewing data analysis aspects, we also consider advantages of the framework for unclassified data.

### 5.5.1 Data Analysis

In a similar vein to the analysis of global overview aspects in Chapter 4.5.1, here we address data analysis aspects of the visualization combined with the introduced selection mechanisms and the presented controller widgets. In particular, we focus on the capabilities of the landscape metaphor to suggest or indicate suspicious features and how the analyst gets to their bottom using selections and linked views. At first, we revisit the artificial 2-D data set to illustrate some of the peculiarities of the topological approach and then we exemplify the typical workflow of the proposed analysis based on a high-dimensional data set.

#### Artificial 2-D Data Set

Being able to select individual features in the artificial 2-D data set helps to better understand the involved topological concepts and to demonstrate the advantages of the density-based approach. Figures 5.9a-b show the 2-D topological landscape profile and the scatterplot of the artificial 2-D data set. One advantage of the topological approach is its robustness with respect to noise. Not only is the detection of clusters insusceptible to noise, the final visualization is also not cluttered by the depiction of noise. Most importantly, it is easy to isolate structure from noise because the latter is typically located at the bottom of the landscape. As demonstrated in Figures 5.9c-d, clusters and noise can easily be selected with two selections: one for the points with low density and another selection for the points with higher density. While the analyst is typically not interested in selecting noise, this is still an easy and convenient way to focus only on the clusters in the data. The artificial 2-D data set is also useful to visualize how the topological approach treats the data internally, i.e. how the merge tree segments the point cloud's density function into coherent regions. For this purpose, selections help us to show how these regions actually look

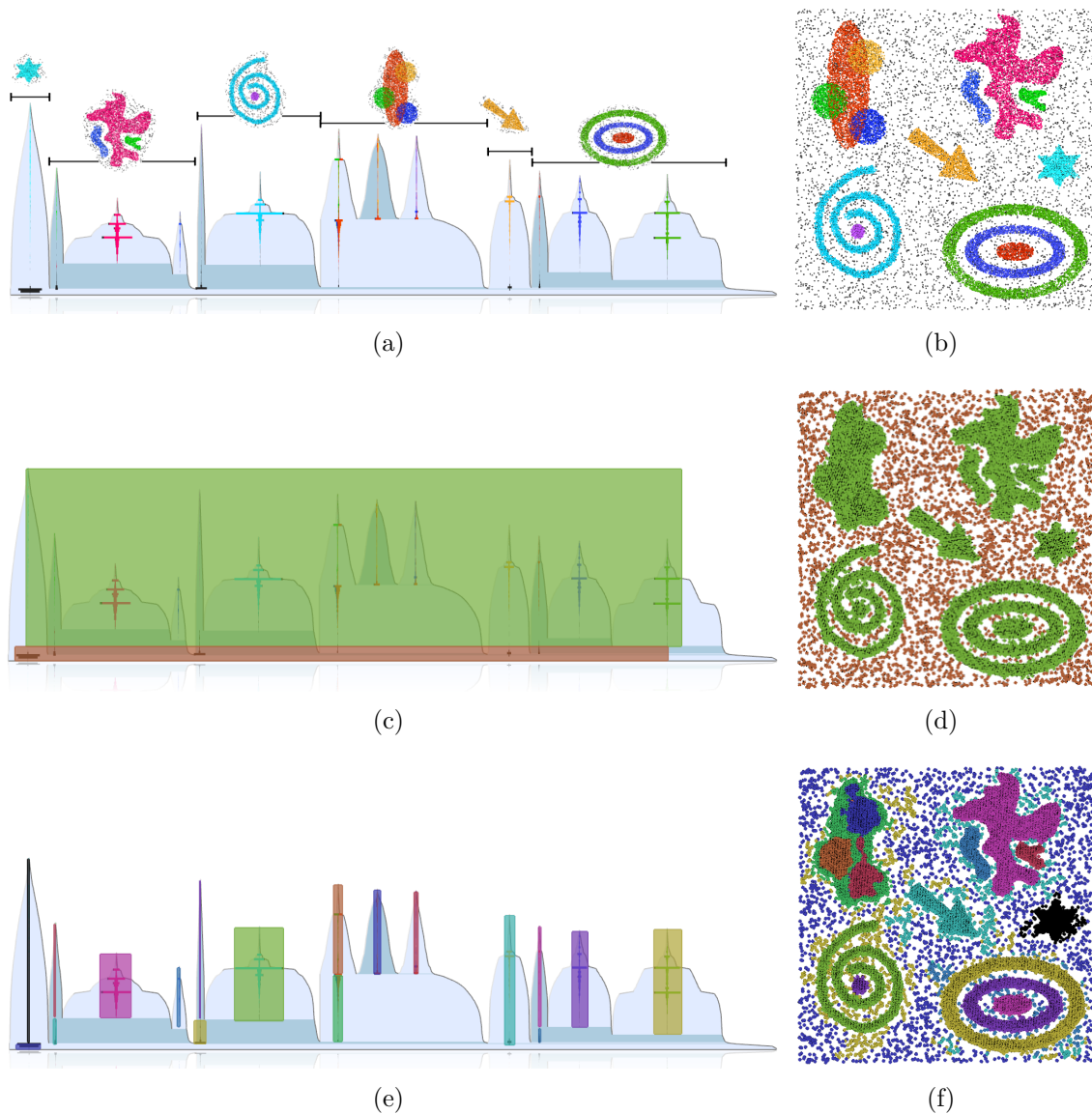


Figure 5.9: Selections in the artificial 2-D data set: (a) The 2-D topological landscape profile for a suitable parameter setting. (b) The artificial 2-D data set. (c) Because densities are encoded by height information in the landscape metaphor, the analyst can easily separate clustering structure from noise with two selections. The lower selection isolates the black histograms at the bottom of the hills. (d) Based on the colors of the selections, the linked scatterplot reveals that structure and noise can be separated perfectly. (e) By selecting hills and inner parts of the profile with different colors, the data can be colored according to the merge tree segmentation. (f) The linked scatterplot illustrates how dense regions grow around the density maxima and merge with neighbored regions. “Noise” points are actually those points that have a low density and are located at the border of the dense regions.

like for a typical clustering. Figures 5.9e-f present a coloring of the data set based on the superarcs of the merge tree. Since hills and inner parts of the profile represent

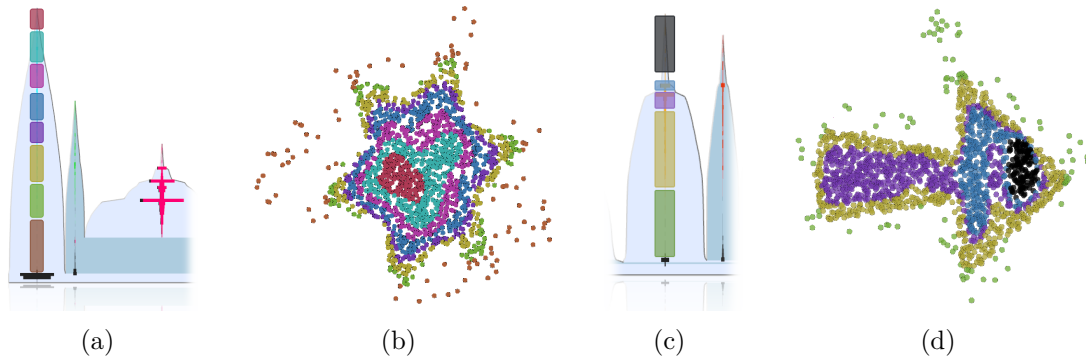


Figure 5.10: Density functions of individual clusters of the artificial 2-D data set: (a) Several selections of different color for the histograms on the hill belonging to the star-shaped cluster. (b) The color distribution in the scatterplot indicates how groups of similar density are located inside the cluster. Typically, the density decreases from the center towards the border, which is why these selections form “rings” around the density maximum. (c)-(d) The same scenario for the arrow-shaped cluster.

superarcs, we can select all corresponding parts of the profile with different colors to see how they relate to the underlying data. Figure 5.9f reveals this segmentation and illustrates that noise is actually part of these regions, which explains why black histogram accumulation occurs at the bottom of multiple hills in Figure 5.9a. In terms of the density function’s topology, a dense region is defined by those points that are comprised by a growing superlevel set until it merges with another one at a saddle. That is, topology cannot tell apart structure from noise, but it considers noise to have a density close to the saddle density of a region. As shown by the colored segmentation, noise points around the clusters have the color of the selected inner parts of the profile, representing this region until it merges with a neighbored region. The largest noise region belongs to the lower histograms on the leftmost hill. This is basically the noise that comprises all the clusters.

Individual selections also allow us to take a closer look at the density function inside the clusters. Figure 5.10 shows several selections at different heights for subsets of the star-shaped and the arrow-shaped clusters. Figures 5.10b,d illustrate that even though the topological analysis cannot preserve arbitrarily shaped clusters, it still detects them reliably. The selections and the linked scatterplots also reveal how data points at similar height actually relate to each other: they typically only have a similar distance to the density maximum of a homogeneous cluster—which is a cluster with a high density peak inside and decreasing density values towards the border. While distance information cannot be preserved, this behavior at least offers valuable clues concerning the distribution of points with similar density. It also gives information about the approximate compactness of a cluster. Note that the

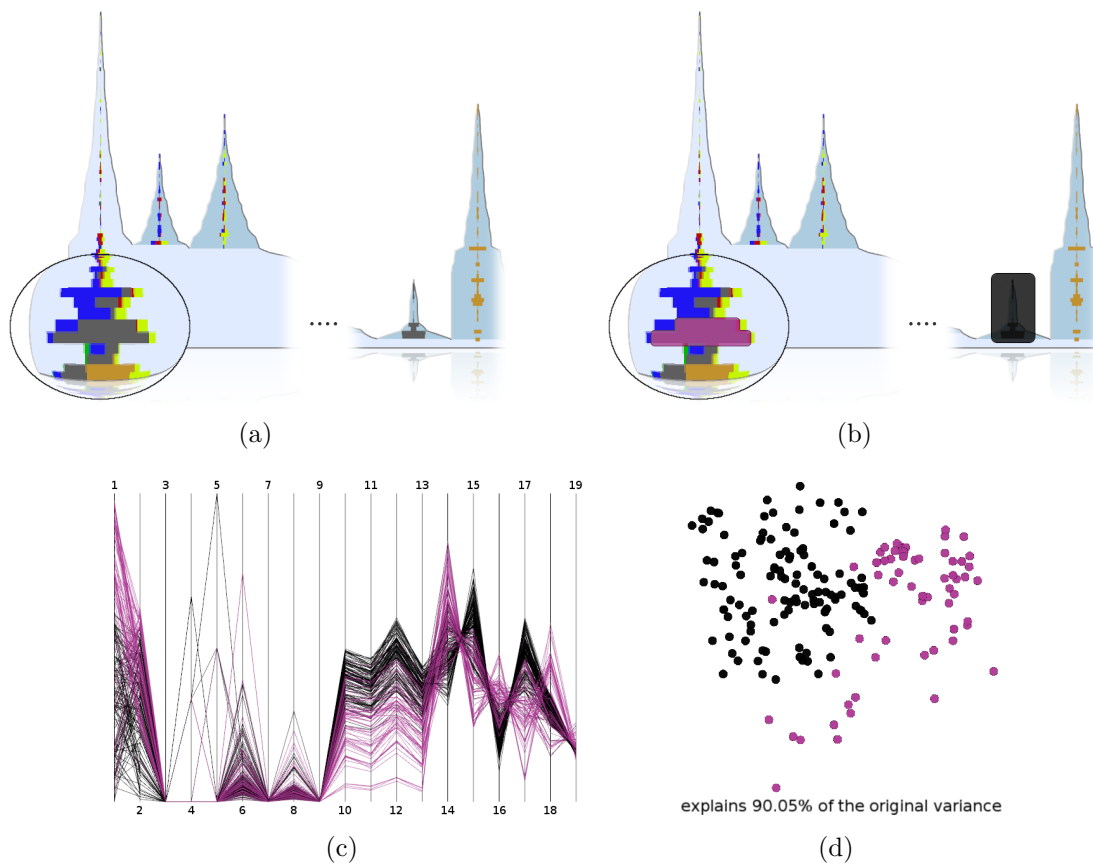


Figure 5.11: Analyzing different accumulations of the same class in the 19-D image segmentation data set: (a) Gray “cement” points have their own hill in the middle of the landscape, but also accumulate on the main hill. (b) To inspect this behavior in more detail, both accumulations are selected with different colors. (c) The linked PCP reveals that even though these points belong to the same class, they still differ pretty much in several dimensions. Interpreting these dimensions semantically would clarify in which aspects these pixel fragments actually differ. (d) The PCA projection also confirms that these points accumulate separately.

histograms at the peak of a hill represent the most similar points. This information is valuable for the underlying application domain, e.g. to identify the most similar documents of a particular topic in a document collection.

## 19-D Image Segmentation Data Set

We already revealed some interesting features of this data set while we introduced linking subsets to axis-based techniques and projections in Chapter 5.3.2. Another suspicious observation that is not apparent at the first sight are accumulations of histograms of the same class at different locations in the landscape. As shown in Figure 5.11a, the gray “cement” points have their own hill in the middle of the profile,

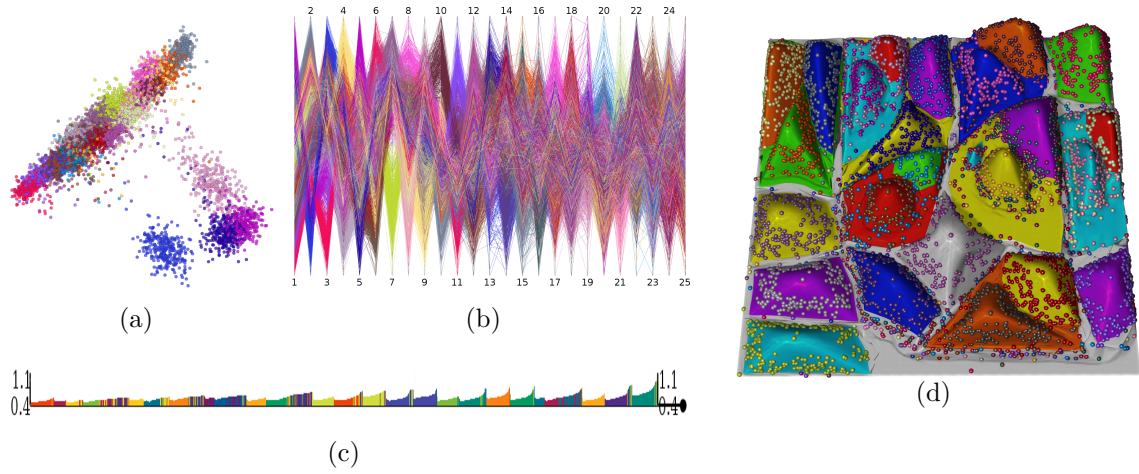


Figure 5.12: Visualizing the 25-D Isolet data set: (a) Rank-2 LDA projection with a projection error of around 58% from the intermediate space down to 2-D. While the point distribution suggests only three or four clusters, the color distribution indicates that clusters could still be combined in the main cluster. (b) The parallel coordinate plot suggests a complex clustering, but the precise structure is hidden and clusters are difficult to identify and compare. (c) Reachability plot as provided by OPTICS. It shows 25 valleys to represent clusters for different reachability distances; a horizontal cut would produce a particular clustering. (d) The 3-D topological landscape shows 26 separated hills of different height and extent and, thus, provides the clearest depiction of the clustering structure in the original space.

but additionally accumulate at a similar density on the main hill. To find out why both accumulations do not belong to the same cluster, we select them with different colors (cf. Figure 5.11b). Linking these points to a PCP, as shown in Figure 5.11c, shows that although they are in the same class, they still differ pretty much in the first and between the 9<sup>th</sup> and 18<sup>th</sup> dimension. The differences in these subspaces prevent the points from being in the same cluster. To interpret this situation semantically and to understand what exactly makes these pixel fragments different from each other in the underlying images, the analyst has to consider which image aspects these dimensions actually describe. The PCA projection in Figure 5.11d also shows that the projection error can be decreased to around 10% by selecting only these points and that they indeed accumulate in separated regions.

### 25-D Isolet Data Set

In the Isolet (Isolated Letter Speech Recognition) data set (cf. Appendix A.4) 150 subjects spoke the name of each letter of the alphabet twice. Trying to learn its clustering structure, we start with visualizing the data with standard techniques. Figure 5.12 shows the Rank-2 LDA projection, the parallel coordinate plot, and a

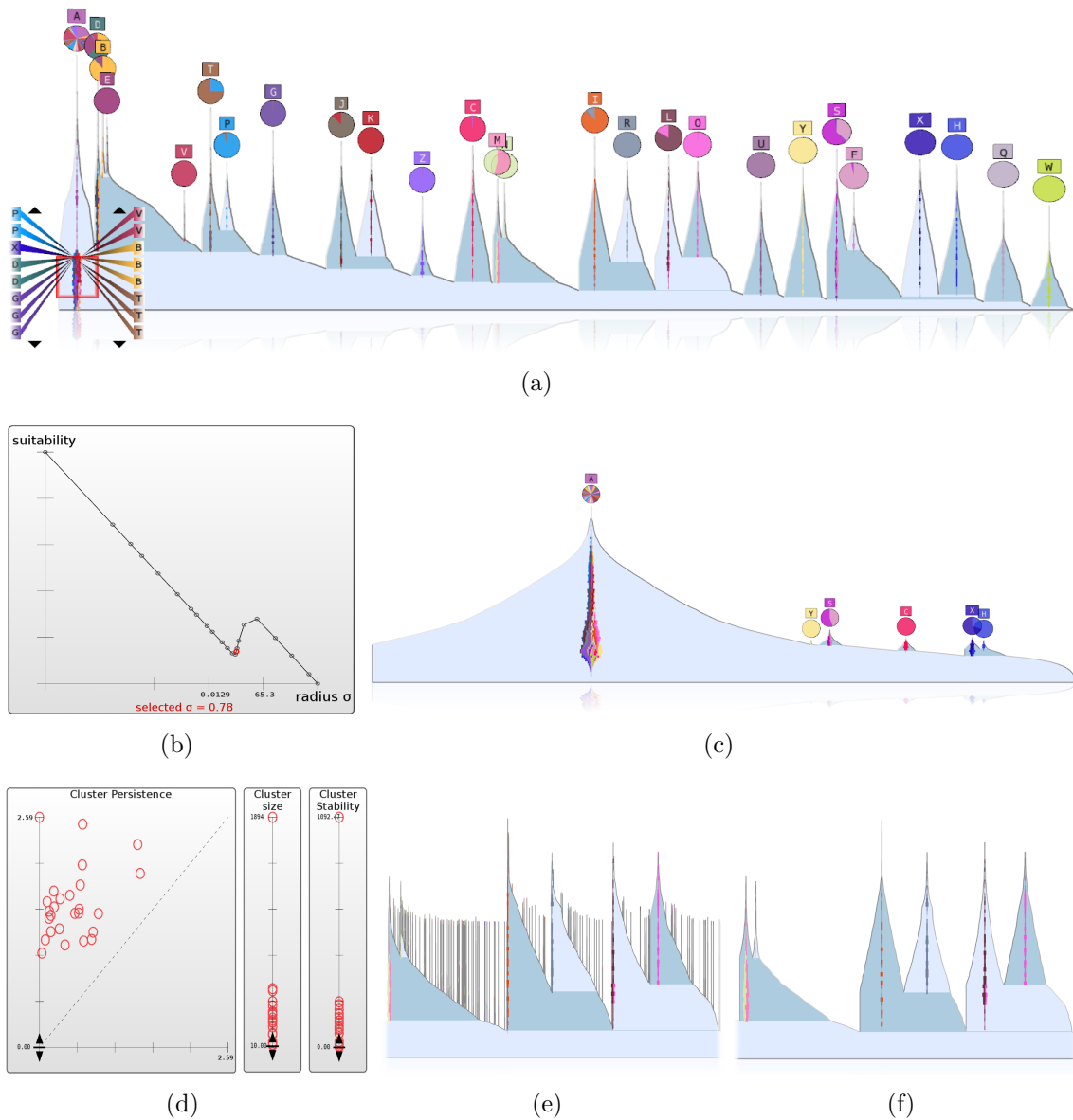


Figure 5.13: Parameter settings for the 25-D Isolet data set: (a) 2-D topological landscape profile showing 26 hills of different height, shape and width. The hills represent clusters in the sonorant information space. These are those recordings that are pronounced similarly, i.e the spoken letters themselves. Cluster hierarchies, furthermore, exist for those letters that rhyme. (b) Filter radius suitability controller with the local minimum at  $\approx 0.78$  used to create the landscape profile in Figure (a). (c) Landscape profile for a too large filter radius showing that many clusters are still combined. (d) Simplification controller with the adjusted thresholds used to create the landscape profile in Figure (a). (e) Part of the landscape profile before topological simplification. Small and thin hills represent fluctuations in the density function. (f) Same part of the profile after simplification.

reachability plot as provided by the OPTICS clustering method (cf. Chapter 2.1 on page 12). The reachability plot shows clusters as valleys to present a hierarchical

partitioning of the data for different values of the density descriptor on the ordinate. This value is similar to the filter radius  $\sigma$  of a Gaussian kernel. The data items are colored according to given classification information. Because there is no a priori knowledge about the relation between clusters and classes, the point distribution in the LDA projection can only suggest three or four clusters. Even if we face occlusion artifacts of truly separated clusters, we cannot tell which of them are closer or even in a subcluster hierarchy. The PCP indicates more separated clusters based on the color and the distribution of the polyline bundles throughout all dimensions. However, individual bundles are still hard to identify, count, and compare. The reachability plot shows many different clusters, but it has a stronger focus on finding appropriate parameters and concentrates less on the visualization of the clustering and individual cluster properties for a particular cluster segmentation.

Figure 5.12d and Figure 5.13a show the 3-D topological landscape and the 2-D landscape profile for the 25-D Isolet data set. The isolated hills in both landscape metaphors easily reveal substantially more clustering structure than the LDA projection or the PCP. The computation time to obtain the landscapes is about three seconds. The interesting aspect of this data set is the semantic behind the clusters and the relation between them. In the information space spanned by various sonorant features for each recording, points accumulate if they share similar vocal aspects, i.e. if they are pronounced similarly and thus sound alike. As can be seen from the labels and the distribution of the colored histograms in the 2-D profile, these point accumulations are the letters themselves. Furthermore, some letters, like “m/n”, “k/j”, “b/e/d”, “t/p”, or “f/s”, are in a subcluster relationship because they sound more similar to each other than to other letters. This is reflected by deep valleys between those letters that sound different and valleys at higher density between those letters that rhyme. In terms of the information space, similar letters are located in the same subspaces and thus accumulate to clusters and subclusters.

To find suitable thresholds for the filter radius  $\sigma$  and to simplify the structural overview, the analyst uses the interactive controller widgets. Figure 5.13b shows the filter radius suitability controller for the Isolet data set. After several manual refinements, which take around 1.5 seconds each and use optimized parameters settings, the local minimum was found at  $\sigma \approx 0.78$ , which is also the value used to create the profile in Figure 5.13a. Figure 5.13c shows an example of a too large filter radius. If  $\sigma$  is larger than the inter-cluster distance, separated clusters are merged and the structural insights about the clustering are misleading. Nevertheless, the histogram accumulations at different heights and the suitability graph suggest that this value is likely inappropriate and should be reduced further. Figure 5.13d shows



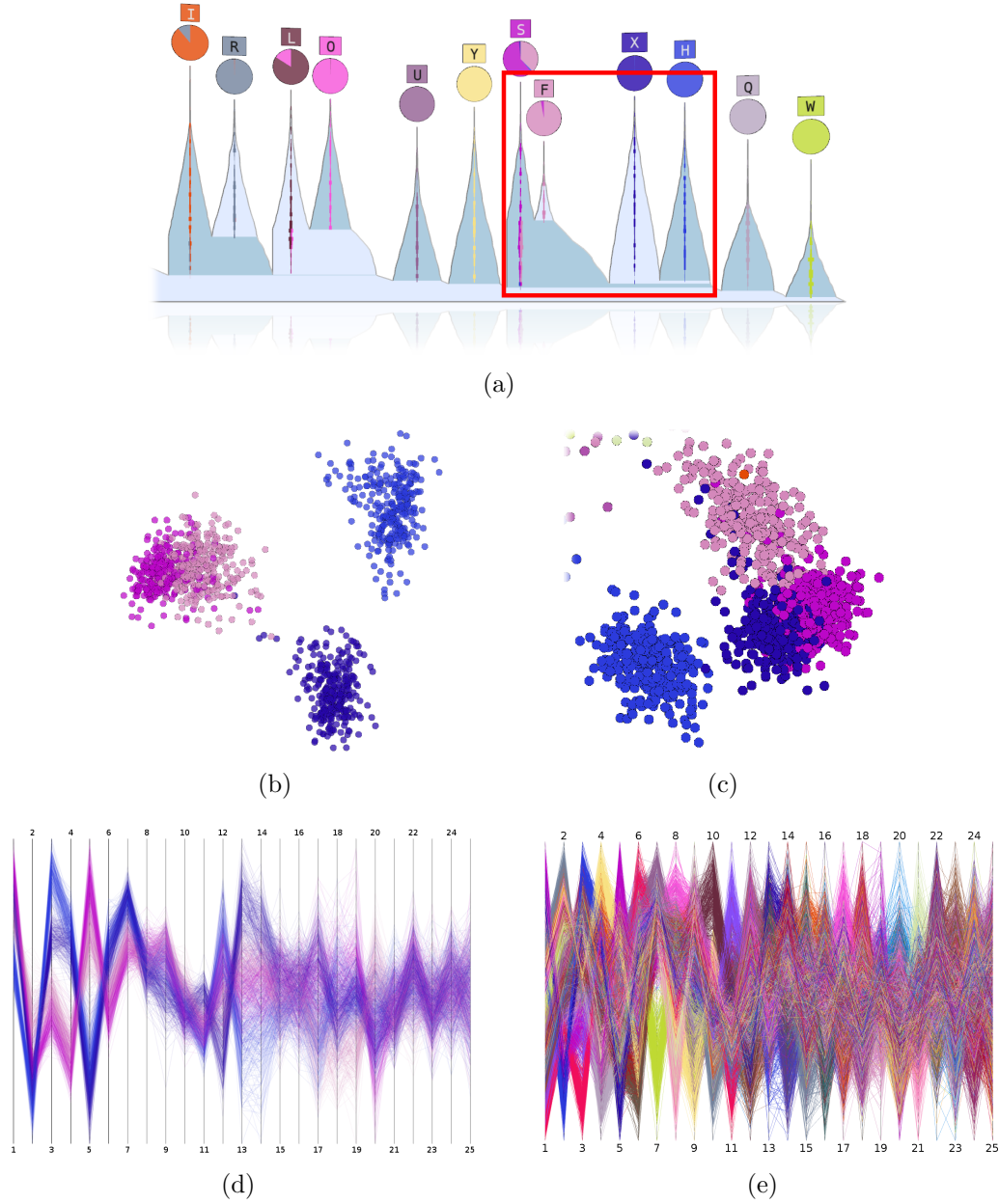


Figure 5.14: Local analysis of the 25-D Isolet data set: (a) Selection of only a few clusters for further inspection in linked views. (b) Rank-2 LDA of only the four selected clusters. (c) Clipped part from Figure 5.12a showing the four selected clusters in the misleading Rank-2 LDA projection of the complete data set. (d) PCP showing only the four selected clusters to inspect in which dimensions these points differ. (e) PCP of the complete Isolet data set in which local analysis of only a few clusters is complicated by the presence of all other points.

the simplification controller with an adjusted minimum cluster size of 10 points. This threshold leaves 26 prominent features and was used to create the profile in Figure 5.13a. Figure 5.13e,f show the effect of topological simplification before and after simplification. While fluctuations in the density function are represented by



small and thin peaks in the unsimplified profile, after simplification, the profile shows only the remaining main clusters.

To demonstrate the advantage of the topological analysis and the synergetic effects caused by linking individual features to geometry-based standard techniques, in Figure 5.14a only the 'S', 'F', 'X' and 'H' clusters are selected with a single rectangle. Although the profile already reveals the correct subcluster hierarchy, the linked LDA projection of only these points (cf. Figure 5.14b) confirms that the occlusion-prone LDA projection of the complete data (cf. Figure 5.14c) is quite misleading. However, the projection is still useful to *approximate* other properties like a cluster's shape or inter-cluster distances. Linking the same subset to a PCP (cf. Figure 5.14d) highlights in which dimensions the points actually differ. This inspection would be difficult in the presence of all other points (cf. Figure 5.14e) and without means to fade them out.

### 5.5.2 Unclassified Data

Another advantage of the interactive visual analysis and the occlusion-free depiction of clustering structure is that the topological overview can be used to accentuate structure in otherwise monochrome standard visualizations. This is demonstrated in Figure 5.15 based on the Reuters data set (cf. Appendix A.7). The profile in Figure 5.15a shows the clustering of the data points without considering the classification information. The labels above the hills display cluster sizes. To highlight individual clusters and to indicate where in the standard visualizations of the complete data these subsets are located, we select three individual features with different colors. The PCA and the PCP in Figures 5.15b-c show the complete data set, but additionally highlight the picked clusters using the color of their selections. This combination of topology-based overview and geometry-based local analysis is helpful in those scenarios where the analyst has to explore features with standard techniques in the context of the complete data set. This would be difficult if features cannot be discriminated visually or if features are missed at all due to occlusion artifacts. Such an approach to highlight features in linked visualizations was also used by Fua et al. [61], who term it *structure-based brushing*, to navigate through hierarchical cluster trees. We improve this concept by providing additional information about feature prominence and by supporting more sophisticated selection in the structural view. Showing cluster quality measures helps the analyst identify interesting features before linking them to other views.

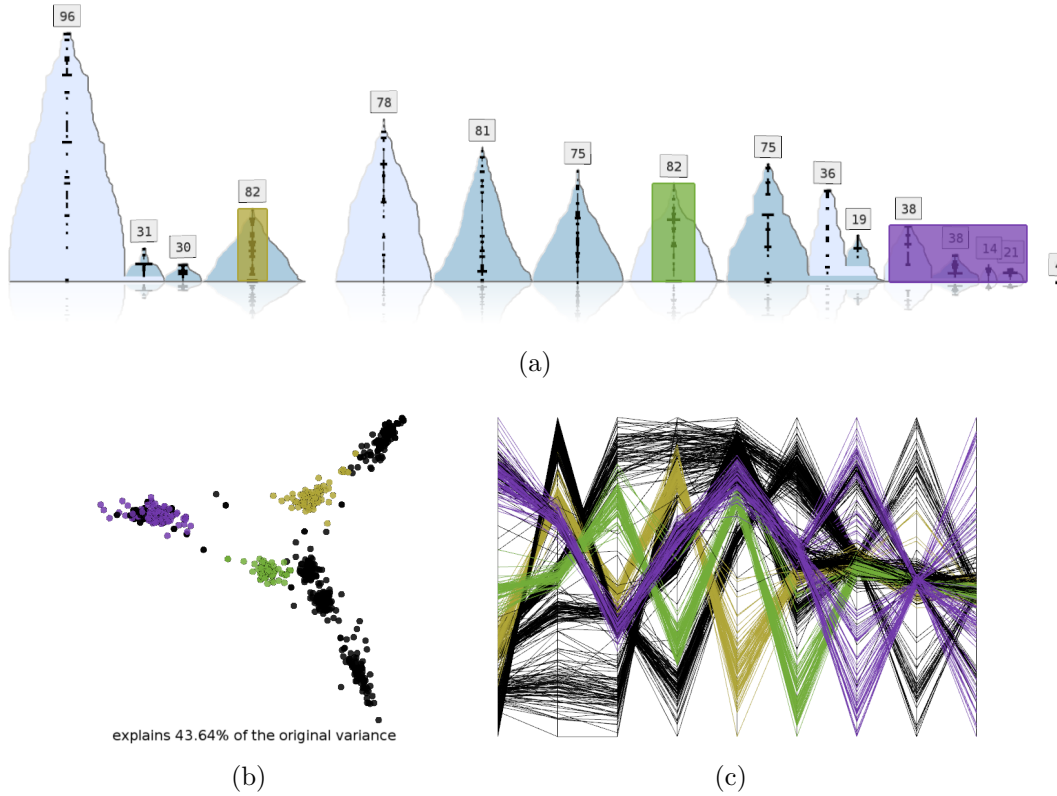


Figure 5.15: Accentuating features in standard visualizations of the complete data set based on the 9-D Reuters data without classification information: (a) 2-D landscape profile with labels showing cluster sizes and three selections of two individual hills and a sub-hill hierarchy. (b) PCA projection of the complete data set with the three selected clusters highlighted by using the color of their selections. (c) PCP showing the complete data set and the polyline bundles of the selected clusters with the color of their selections. This approach helps to compare these features in the context of the complete data.

## 5.6 Conclusion and Discussion

Although standard visualizations like projections and axis-based techniques suffer from information loss, projection artifacts, occlusions, and visual complexity, these methods are still valuable to approximate point distributions and to explore subspaces and individual dimensions. Therefore, to benefit from synergetic effects between the topological approach and these conventional techniques, we propose to split the visual analysis of high-dimensional point clouds into two separated phases: in the global overview phase, we neglect geometric properties to obtain an appropriate presentation of the clustering. This structural overview is independent from the data's dimensionality, is robust with respect to noise, preserves structural information without loss, illustrates features and data points occlusion-free, and provides means to compare, annotate, and select arbitrary features like clusters, subclusters or point

sets. In the local analysis phase, the analyst leaves the global scope and analyzes features individually and in more detail than the topological abstraction could explain. To this end, selected features are linked to standard techniques to explore those properties not captured by the merge tree, e.g. approximated cluster shape, inter-cluster distances, or point distributions in particular dimensions. The key idea is that by focusing on only a few clusters, the artifacts and visual complexity of a projection or a parallel coordinate plot can be reduced substantially. The synergetic effects arise from the fact that these features could not have been selected individually using these standard techniques alone. To help the analyst find an appropriate parameter setting, we also presented intuitive controller widgets and strategies to *read and interpret* the behavior of the landscape while changing parameters.

In addition to the limitations of the topological abstraction (cf. Chapter 3.6 on page 63) and the topology-based visualization (cf. Chapter 4.6 on page 106), disadvantages of the presented framework are limited scalability and restricted feature properties, the necessity to resist interpreting Euclidean distances in the topological context of the landscape metaphor, and certainly a required basic understanding of the topological concepts involved. Although the landscape metaphor itself can convey quantitative and hierarchical relationships, interpreting it and understanding the effects of parameter changes is certainly not immediate for the lay user. However, as already mentioned earlier, studying both the underlying concepts and how the final visualization is meant to be read is considered worth the efforts if the alternative is occlusion-prone and illusionary depiction of high-dimensional point data.



## Part II

# Visual Analysis of Time-Varying Clusterings



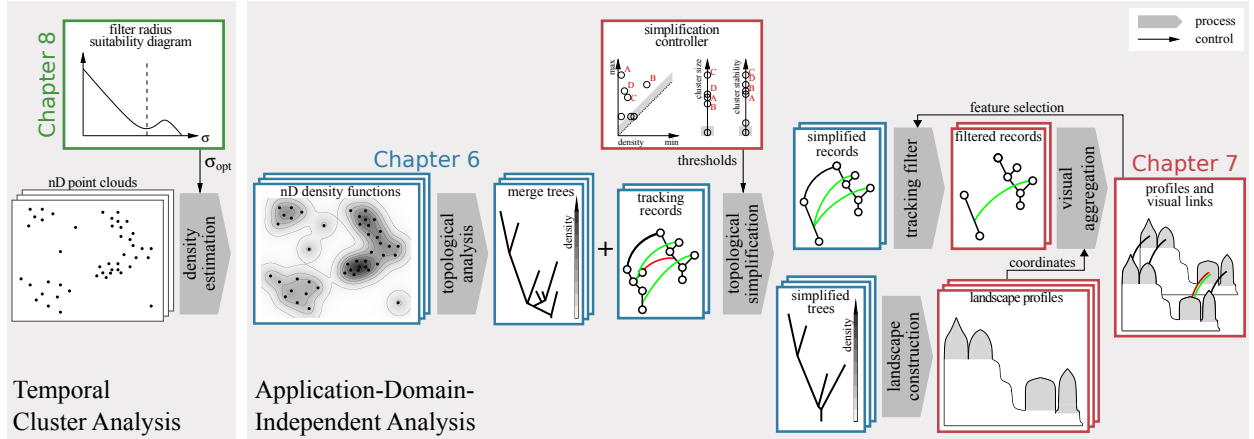


Figure Part 2: Overview of the visual analysis framework for high-dimensional data that change over time: Independent from the application domain, the merge trees of temporally adjacent high-dimensional scalar functions are transformed into each other while storing structural changes of the dynamic tree as *tracking records*. After topological simplification, which removes insignificant superarcs from the trees, tracking information is adjusted accordingly. The complex information provided by the time-varying tree is visualized with a topological landscape profile per time step, augmented with visual links to indicate feature evolution. Temporal cluster analysis is one application for time-varying merge trees and requires a prior transformation of the point data into a time-varying density function. Interactive controller widgets help the analyst to determine crucial parameters, and arbitrary features can be selected and filtered in different directions in time.

The second major part of this thesis is about topology-based visual analysis of high-dimensional, time-varying clusterings. That is, compared to time-invariant point clouds, the number of points as well as their coordinates can change over time. In a similar vein to Part I of this thesis, temporal cluster analysis is considered a specific application of high-dimensional, time-varying scalar field topology. This aims at a study of the clustering in terms of dense regions and their evolution over time. Because existing approaches to compute and visualize superlevel sets and contours of low-dimensional scalar fields do not scale to higher dimensions, we introduce a novel method that identifies and tracks features in arbitrary dimensional scalar functions using *time-varying merge trees*. We analyze the evolution of the function by tracking changes in the merge tree and we relate features by matching subtrees between consecutive time steps. While the analysis of the changing scalar function is independent from the underlying application domain, for temporal cluster analysis, we extend the topological approach described for static point clouds by abstracting a dynamic point cloud by its varying density function. Topology-based quality measures, like persistence, size, or stability, again, describe cluster significance and help the analyst inspect and track significant features. Using the time-varying merge

tree, we develop a structural visualization of the changing function that illustrates both features and their temporal evolution.

Figure Part 2 summarizes the algorithmic pipeline of the time-varying analysis and also provides a visual guideline for the next chapters. In Chapter 6, indicated by the blue parts, we first introduce the time-varying merge tree and the algorithmic core. This includes the representation and topological description of time-varying data, how features are related and tracked over time, and how tracking information is simplified to focus only on significant features. To visualize the complex information provided by the time-varying merge tree, a prototype visualization is described in Chapter 7. Building on the topological landscape metaphor, landscape profiles are linked together to reveal the clustering at the given points in time and visual links are used to indicate the evolution of individual features over time. While the time-varying merge tree and its visualization are independent from the application domain, temporal cluster analysis is considered an instance of high-dimensional, time-varying scalar field topology in Chapter 8. This chapter explains the preprocessing necessary to turn a set of point clouds into a dynamic density function and it applies the time-varying merge tree to topic tracking in time-dependent document collections.



## Chapter 6

# Topological Representation

In many fields of application, time-varying data result from observing or simulating a system over a specific period. To make data storage and processing feasible, the observation process is typically discretized by taking snapshots that represent the system at certain points in time. These *timeslices* are typically sampled on a grid to specify observations at certain locations and different granularities. A time-varying description is obtained by interpolating the samples in both space and time. That is, for each timeslice, cell interpolation extends the sampled grid values to the whole domain, and linear interpolation between consecutive timeslices provides observations for arbitrary points in time. Depending on the dimensionality of the domain, i.e. the dimensionality of the grid vertices, and on the dimensionality of the observed measurements, i.e. the function values at the grid vertices, practical examples for scientific visualization include scalar-, vector-, or tensor fields to describe, e.g., temperature, pressure, velocity, or stress at multiple locations in a 1-D, 2-D, or 3-D space for a fixed moment or over time.

With increasing size and dimensionality, time-varying data become difficult to visualize and analyze. One solution to this challenge is to detect features, i.e. interesting data subsets, at each point in time and to track their evolution, including feature birth, death, join, and split. This form of summary allows more compact and less cluttered visualizations, enables quantitative analysis, e.g. recording the variation of the number of features with time, and provides a vantage point for further data exploration. Just as already demonstrated for high-dimensional, but time-invariant data, topological methods provide efficient data summaries.

If meaningful features of scalar fields are defined by thresholding, the merge tree compactly encodes features for all possible thresholds, as demonstrated by Bremer et al. [20], who studied combustion simulations. However, to track features over time, existing methods often fix a threshold. Changing this value during

the analysis requires expensive recomputation of the tracking information [170] and supporting threshold changes over time or on a per-feature basis depends on specifying a large number of parameters *a priori*. Moreover, existing methods correlate features via spatial overlap in consecutive time steps—an approach that can lead to ambiguities [155] and becomes computationally intractable in higher dimensions. Although topological methods have been applied successfully to analyzing time-varying data, current approaches are only practical for scalar fields defined on two- and three-dimensional domains. Time-varying Reeb graphs [45, 118] require complicated distinction of cases for topological events; their number increases with the dimensionality and no case tables have been presented beyond the 3-D case.

To address these issues, we introduce *time-varying merge trees*—a topological summary of time-varying scalar fields. Instead of using a single threshold, time-varying merge trees track all features for all thresholds over time, supporting threshold selection after and informed by the time-varying merge tree’s visualization. To avoid tracking ambiguities, the algorithm records all necessary changes to the merge tree within the time interval between adjacent time steps, establishing a clean topological foundation for feature tracking. By focusing on merge trees for threshold-based feature tracking, we are able to provide a complete set of cases for arbitrary dimensions and prove their correctness. Compared to the more expressive Reeb graph, the loss in expressiveness is negligible for threshold-based feature tracking; it still captures all application-relevant events.

## 6.1 Related Work

Defining and tracking features [146, 140] is one solution to visualizing large time-varying data sets. Here, we only consider features in scalar fields and refer to Laramee et al. [110] for a survey of vector field methods.

Many feature definitions for time-varying scalar fields are based on isosurfaces. These are surveyed by Mascarenhas et al. [119]. Kettner et al. [102] presented the Safari interface. It shows the properties of a sample of isosurfaces, allowing analysts to make an informed choice of which isosurfaces to inspect. It does not, however, show how the sampled isosurfaces are related in time. Szymczak [159] presented an interface where the analyst can query contours that evolve in particular ways, e.g., split or join between certain time steps, or hit the boundary. To support this operation, the method computes contour trees for each consecutive pair of time steps and annotates them with the subdomains they intersect. Similarly, Sohn and Bajaj [155] track contours over time using a similarity measure that considers spatial

overlap of the inside and outside of contours. The results for one fixed isovalue can be shown as a tracking graph. Ji and Shen [90] use the earth mover’s distance to determine correspondence among contours. Bremer et al. [19] use the Morse-Smale complex to compute burning regions restricted to an isotherm for a range of fuel consumption thresholds. Once an appropriate fuel consumption threshold is identified, they use the Reeb graph of a 4-D space-time isosurface to track these regions over time [19, 167]. In later work, Bremer et al. [20] used the merge tree to compute statistics concerning burning regions within a single time step. Once an appropriate threshold is identified, burning regions are tracked over time via overlap. Keller and Bertram [101] present a method to compute a time-varying isosurface from a so-called hyper-Reeb graph, i.e., a Reeb graph augmented by Betti numbers indicating, among other things, genus changes. While isosurface extraction is applicable for arbitrary dimensions [15],  $d$ -dimensional regular grids of hypercubes and isosurfaces extracted as sets of  $(d - 1)$ -dimensional simplices are quickly becoming impractical in higher dimensions, and Bhaniramka et al. [15] provide only 4-D and 5-D examples; they also report on excessive (i.e.  $2^d$ ) case tables for dimensions greater than four. The variance in these feature correspondence definitions can be explained by the lack of an underlying principle. In contrast, our method observes the topological behavior with respect to linear interpolation in time. In addition to basing feature tracking on a clean topological foundation, this approach also estimates when topological changes take place at sub-time step resolution.

For a family of real-valued functions on a common  $d$ -manifold without boundary, Edelsbrunner et al. [44] define Jacobi sets to compute the time-varying contour tree of a function on the 3-sphere [45]. Their method computes the contour tree for the start time directly and then changes it when topological events of a feature given by the Jacobi sets require it. However, changes to the contour tree require detailed case analysis and the algorithm is difficult to extend to higher dimensions [45, 117]. By restricting considerations to the merge tree, our algorithm considers fewer and simpler cases, and—most importantly—makes it independent of the domain’s dimension. Instead of tracking critical points explicitly, Cohen-Steiner et al. [33] use the stability of persistence diagrams to define *vineyards* for time-varying, real-valued functions: evolving persistence diagrams in which critical value pairs can be traced visually or by computing a matching between them.

## 6.2 Algorithm Overview

We visualize time-dependent, real-valued functions defined on domains of any dimension by studying how superlevel sets evolve with time. Since most applications provide time-varying functions as a sequence of scalar field snapshots, a reasonable approach to encode varying structure is to capture the function's topology at the given time steps and to find out how these features relate to each other between consecutive time steps. The time-varying merge tree supports to identify and track two feature types: Those defined by complete regions of the function, i.e. families of superlevel sets between critical points as represented by the superarcs of the merge tree, and those features defined by thresholding, i.e. the components of a single superlevel set belonging to a particular threshold  $\epsilon$ . For example, in terms of a density function and clustering, features of the complete function are dense regions described by a density maximum and a saddle. Tracking such features is independent from their actual density, i.e. all clusters are tracked irrespective of their density. On the other hand, tracking only a superlevel set of the density function means to fix the density to a certain value  $\epsilon$  and to study how the dense regions of the complete function evolve only regarding this single value. This implies that only those regions containing this density will be captured by the corresponding superlevel set. For cluster analysis, this feature type is useful, e.g., to study the clustering in terms of a minimum or maximum density, or to find those clusters that exhibit a particular density over time. For document analysis, an application of high-dimensional, temporal clustering, analyzing superlevel sets reveals at which times, or for how long particular topics satisfy a user-specified topical importance, as defined by the the clusters' density (cf. Chapter 8).

The algorithm consists of the following steps which can be run concurrently for pairs of consecutive snapshots (cf. Figure Part.2):

- (1) Compute the merge tree for each scalar field in the input sequence.
- (2) Find the sequence of structural changes that transforms each merge tree into its successor and store these changes of the complete function as *tracking records* or changes of a particular superlevel set as linked *level-tracking events*.
- (3) Use topological simplification to remove noise from the merge trees and adjust tracking information accordingly.

————— (cf. Chapter 7 about the visualization) —————

- (4) Represent each tree as a 2-D landscape profile and translate tracking records into visual links connecting related hills, or overlay tracking graphs to show superlevel set evolution in the context of the complete function.

## 6.3 Merge Tree Transformation

To determine related features between two consecutive time steps, we compute the merge tree evolution by continuously transforming the first merge tree into the second. At any point during this transformation, the merge tree correctly represents the topology of the linearly interpolated function. For example, a varying 2-D function can be imagined as a fluctuating height field where increasing and decreasing function values cause grid vertices to transpose—or swap—in their height value. All transpositions take place in a distinct order, and each pair of grid vertices transposes exactly once or never. The merge tree varies according to the changing structure of the fluctuating height field.

The merge tree construction algorithm implies that the tree can only change if the order of the grid vertices changes in terms of their function values. That is, instead of considering infinitely many merge trees, the transformation can be discretized by considering only the finite number of potential changes that arise from transposing vertices. Moreover, it is immediate from the merge tree construction that only transpositions of grid vertices in the same region can affect the tree's structure. Therefore, it suffices to observe changes to the tree whenever two nodes joined by an arc transpose and their common arc *collapses*: only after these transpositions can the number and the order of superarcs change. It is also necessary to work on the fully augmented version of the merge tree because otherwise we could miss when a regular node becomes critical through a transposition. For example, a regular node can turn into a maximum node after transposing with another regular node or with a saddle.

We imagine a dynamic tree, whose arcs grow, shrink, and collapse whenever their end-nodes transpose in their height values. The transformation terminates when each node reaches its final position in the tree; we obtain the merge tree of the second function.

### 6.3.1 Algorithm

The basic idea of the merge tree transformation is to let the arcs collapse when their end-nodes transpose. If multiple transpositions occur simultaneously, we order them based on the lexicographical order of their incident nodes; a straightforward

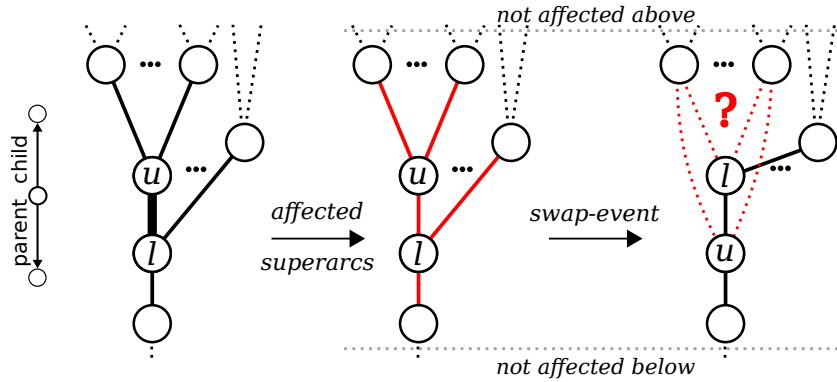


Figure 6.1: Transposition of a single merge tree arc: (left) If nodes  $u$  and  $l$  transpose in their height value, as defined by linear interpolation, the merge tree needs to be reconstructed with the correct superarcs. (center) Only the arcs incident to  $u$  and  $l$  (red) are affected, whereas the rest of the tree remains unchanged. (right) While some arcs (black) are implicitly correct, others (dotted red) need to be validated.

extension of simulation of simplicity [47] to ensure that at any time there is a single superarc whose nodes are to transpose next. Because reconstructing the whole merge tree from scratch after every single arc collapse would be computationally expensive, we consider how the merge tree can change after a single transposition. Figure 6.1 illustrates a transposition for arc  $(u, l)$ , with  $u$  and  $l$  being the upper and lower nodes, respectively. There are two types of arcs: those not affected by the transposition, and thus implicitly preserved in the tree; and those that may change, and, therefore, require additional validation.

### Implicitly Correct Merge Tree Arcs

The correctness of implicitly preserved arcs after the transposition follows from the following lemma.

**Lemma 1** (Arc Lemma). *For a fixed domain and a fixed vertex order  $F$ , there is an arc  $(u, l)$ , with  $F(u) > F(l)$ , in the merge tree of  $F$  if and only if the component of  $l$  in the domain restricted to the vertices  $w$  with  $F(w) > F(l)$  contains the vertex  $u$  and does not contain any vertex  $w$  with  $F(u) > F(w) > F(l)$ .*

The proof of the lemma follows immediately from the algorithm used to construct merge trees. It follows from the Arc Lemma that the transposition of  $u$  and  $l$  affects the merge tree only locally.

**Property 1.** *Any arc that does not contain  $u$  or  $l$  remains in the merge tree.*

*Proof.* Since the only change in the order  $F$  is the transposition of vertices  $u$  and  $l$ , if the two properties of the Arc Lemma hold for an arc  $(x, y)$ , with  $x, y \notin \{u, l\}$ ,

before the transposition, they continue to hold after the transposition. (And if they do not hold before, they do not hold after.)  $\square$

This property implies that the merge tree remains unchanged above all of  $u$ 's and  $l$ 's children, as well as below  $l$ 's parent and thus only arcs incident to  $u$  and  $l$  need further validation. While it is immediate that  $u$  and  $l$  are still connected after the transposition, the Arc Lemma also implies an arc between  $u$  and  $l$ 's parent node.

**Property 2.**  *$u$  inherits  $l$ 's parent node.*

*Proof.* Let  $p$  be  $l$ 's parent node. The component of  $l$  before the transposition is the same as the component of  $u$  after the transposition. Since the order of nodes between  $l$  and  $p$  before, and  $u$  and  $p$  after the transposition are the same, the Arc Lemma implies that we have an arc  $(u, p)$  in the tree after the transposition.  $\square$

The validation of  $u$ 's and  $l$ 's child arcs depends on their connection in the underlying grid. The nodes' links do not change, but their upper links do, and thus require analysis. For node  $l$ , the change of its upper link is limited.

**Property 3.**  *$l$  retains the merge tree arcs to all the components that remain in its upper link.*

*Proof.* For any arc  $(w, l)$ , the two properties of the Arc Lemma hold before the transposition. After the transposition, the first property holds because if the component of  $w$  remains in  $l$ 's upper link, then  $u$  belonged to a different component of  $l$  than  $w$  (so its removal could not have disconnected  $w$  from  $l$ ). The second property holds because there is one less node between  $w$  and  $l$ . In other words,  $(w, l)$  remains an arc after the transposition.  $\square$

### Merge Tree Arcs That Need Validation

To understand the changes to  $u$ 's children, we need to determine how its upper link is affected by the transposition.  $l$  can become a new upper link component; it can become part of an existing upper link component (a regular node); or  $l$  can combine an arbitrary number of  $u$ 's previous upper link components. We need to check which of  $u$ 's upper link components are in  $l$ 's upper link, once  $l$  is higher than  $u$ . In other words, we determine whether  $l$  is connected to some of  $u$ 's upper link components and thus becomes a regular node or a saddle. If  $l$  is not connected to any of  $u$ 's upper link components, it becomes a new maximum within the upper link of  $u$ .

To this end, we start a traversal towards the merge tree's root from each grid node  $x$  in  $l$ 's upper link. The traversal follows the unique path from  $x$  to the root of

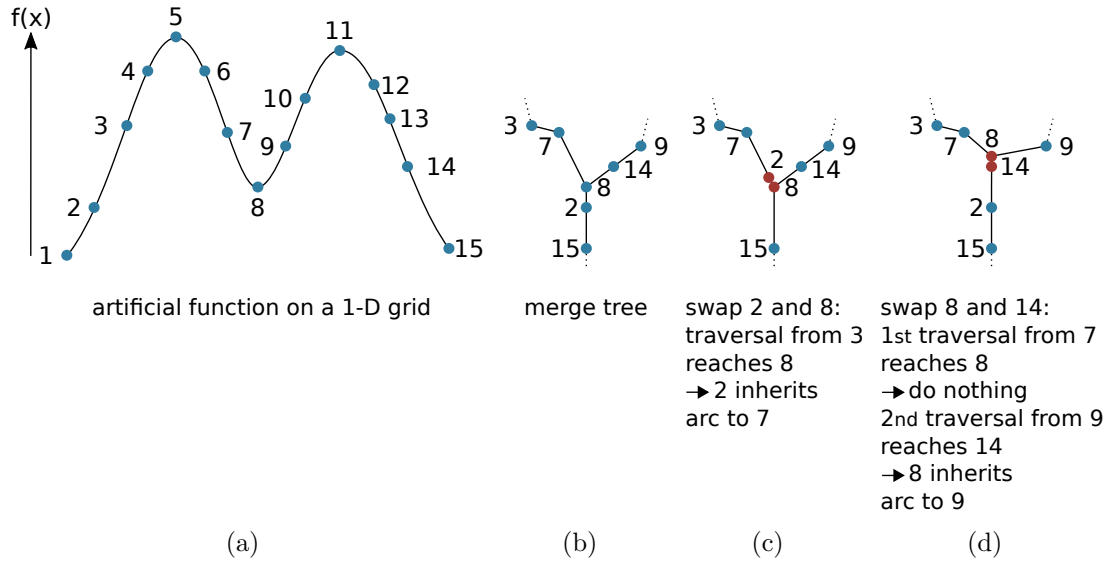


Figure 6.2: Two examples of a collapsing superarc: (a) An artificial function with two maxima on a 1-D grid. (b) Inner part of the merge tree showing the nodes around the function's saddle. (c) Example 1: When node 2's function value rises above node 8's function value we start a traversal from node 2's upper link, i.e. from node 3, towards the merge tree's root node. Because this traversal reaches node 8, both neighbored nodes must belong to the same upper link component (of node 8). Therefore, node 2 inherits the superarc connection to node 7 in the intermediate merge tree at the time of this swap event. (d) Example 2: When node 8's function value rises above that of node 14 we start two traversals from node 8's upper link, i.e from nodes 7 and 9. The first traversal from node 7 reaches node 8, which means that no superarc needs to be redirected. However, because the second traversal from node 9 reaches node 14, node 8 inherits the superarc connection to node 9.

the tree. Node  $l$  lies on this path since  $x$  belongs to the superlevel set component of  $l$ . Node  $u$  lies on this path if and only if  $x$  falls in the superlevel set component of some node  $y$  in  $u$ 's upper link (possibly, with  $x = y$ ). The component of  $y$  in the upper link of  $u$  is represented, without loss of generality, by an arc  $(y, u)$ . In this case,  $l$  inherits the arc  $(y, l)$  after the swap. Indeed, both conditions of the Arc Lemma are satisfied. If  $u$  does not lie on the path from  $x$  to the root, then, after the transposition, there is no connection in the superlevel set component of  $l$  between  $l$  and  $u$ 's former upper link. In this case, no arc is redirected to  $l$ . Figure 6.2 illustrates the results of two example traversals based on an artificial 1-D function.

### 6.3.2 Implementation

The implementation of the merge tree transformation is straightforward. Our implementation starts with the augmented merge tree of the first time step and two



lists of the grid vertices sorted descending by their values at the first and second time step. The first list will be re-ordered over time for correct determination of a node's (changing) upper link. The second list is to determine potential transpositions of new arcs that occur after a transposition. The time of an arc collapse is inferred from the linear interpolation between the vertices' values in the first and the second time step. To process all transpositions in the correct order we keep them in a priority queue. The algorithm ensures that if the nodes of an arc transpose, we insert them into the priority queue before the transposition takes place. In other words, we do not miss any arc collapses. Algorithm 3 provides a pseudo-code implementation of the transformation process. The code already contains commands necessary for feature tracking (cf. Chapter 6.4). Because feature tracking is independent from the transformation itself, these commands are commented out (using a double slash `//`).

### Optimization

Although arc transpositions may change the structure of the merge tree, it turns out that many of them are topologically neutral and do not change the tree's structure. This is especially true if the participating end-nodes of the collapsing arc are both regular nodes. Because it is necessary to process the fully augmented merge tree to not miss when regular nodes turn into critical nodes, for topologically neutral events between regular nodes, the traversal to validate  $u$ 's only child arc is a waste of time: the intermediate merge tree would be correct if we just switched nodes  $u$  and  $l$ .

For transposing regular nodes, which is by far the most frequent event, we can determine if a new maximum can appear without performing a traversal to validate  $u$ 's only child arc: a maximum can be born only if the vertices are neighbors in the grid. While this condition is necessary, but not sufficient, for the lower node to annihilate its upper link, its contrapositive—if the two vertices are not neighbors—ensures that a new maximum cannot be born. We insert the neighbor-condition into Algorithm 3 (Step 8) and quickly swap both regular nodes if they are not neighbored in the grid. The pseudo-code of this optimization is given in Algorithm 4.

### 6.3.3 Runtime Complexity

The number of arc collapses during the transformation depends on how many node pairs change their relative order in the sorted sequences of both time steps. That is, a transformation's complexity depends on the structural variation of the function. In the worst case, if the vertices reverse their order, their number is bounded by  $O(n^2)$ ,

---

**Algorithm 3:** Transformation of the merge tree of time step  $i$  into the merge tree of time step  $i + 1$ . Optional parts are commented out.

---

**Input** : A merge tree  $MT$  for time step  $i$   
           A list  $orderNow$  of sorted grid vertices at time step  $i$   
           A list  $orderThen$  of sorted grid vertices at time step  $i + 1$   
**Output**: A merge tree for time step  $i + 1$  and tracking information

```

1 for each arc  $a = (u, l)$  in  $MT$  do
2   if  $u$  is behind  $l$  in  $orderThen$  then
3      $\lfloor$  Push  $a$  on  $priorityQueue$  with priority  $swaptime \in [0, 1]$ 
4 while  $priorityQueue$  is not empty do
5   Pop  $a = (u, l)$ . ( $u$  is the upper node and  $l$  is the lower node closer to the root node)
6   // Handle outstanding superlevel tracking events, see Chapter 6.4.2
7   Put  $u$  behind  $l$  in  $orderNow$ 
8   // Optional optimization, see Algorithm 4 in Chapter 6.3.2
9   Remove  $a$  from  $MT$ 
10  if  $l$  is not root node of  $MT$  then
11    Let  $a'' = (l, p)$  be the arc incident to  $l$  towards the root
12    Remove  $a''$  from  $priorityQueue$ 
13    Remove  $a''$  from  $MT$ 
14  (Determine which of  $u$ 's children become  $l$ 's children)
15  Let  $N$  be the set of grid neighbors of  $l$ 
16  for each neighbor  $n \in N$  do
17    if  $n$  is in front of  $l$  in  $orderNow$  (i.e. in the upper link) then
18      if traversal from  $n$  towards  $MT$ 's root node leads to  $u$  then
19        Let  $a' = (u, u'_n)$  be the arc incident to  $u$ 
20        Remove  $a'$  from  $priorityQueue$ 
21        Remove  $a'$  from  $MT$ 
22        Add arc  $(u'_n, l)$  to  $MT$ 
23        if  $u'_n$  is behind  $l$  in  $orderThen$  then
24           $\lfloor$  Push  $(u'_n, l)$  on  $priorityQueue$  with priority  $swaptime$ 
25          Stop traversal
26        else if traversal from  $n$  towards  $MT$ 's root node leads to  $l$  then
27          Stop traversal
28  Add arc  $(l, u)$  to  $MT$ 
29  if  $l$  was not root node of  $MT$  then
30    Let  $p$  be the parent node of  $l$  before the swap
31    Add  $arc(u, p)$  to  $MT$ 
32    if  $u$  is behind  $p$  in  $orderThen$  then
33       $\lfloor$  Push  $(u, p)$  on  $priorityQueue$  with priority  $swaptime$ 
34  // Store tracking information for this swap event, see Chapter 6.4.1
35  // Update outstanding superlevel tracking events, see Chapter 6.4.2
36 // Handle outstanding superlevel tracking events, see Chapter 6.4.2
37 // Assemble tracking information for features of the complete function, see Chapter 6.4.1

```

---

for  $n$  tree nodes. Potential push-updates to the priority queue (Steps 24 and 33) take  $O(\log a)$ , for  $a$  arcs in the queue. For non-optimized events, the traversal to validate node  $l$ 's upper link depends on the number of tree nodes on the paths between  $l$ 's

---

**Algorithm 4:** Optimization if regular nodes pass each other (to be inserted at Step 8 in Algorithm 3).

---

```

1  if isRegular( $u$ ) and isRegular( $l$ ) then
2      if  $u$  and  $l$  are not neighbors in the grid then
3          Let  $a' = (u, u')$  and  $a'' = (l, p)$  be the arcs incident to  $a$ 
4          Remove  $a'$  from priorityQueue
5          Remove  $a''$  from priorityQueue
6          Swap node  $u$  and  $l$ 
7          if  $u'$  is behind  $l$  in orderThen then
8              Push  $(u', l)$  on priorityQueue with priority swaptime
9          if  $u$  is behind  $p$  in orderThen then
10             Push  $(u, p)$  on priorityQueue with priority swaptime
11             Goto Step 4 of Algorithm 3

```

---

upper link nodes and  $l$  itself. Generally, this number is bounded by  $n$ , but depends on the function itself and on the grid's granularity. In our experiments (cf. Chapter 7 and Chapter 8), however, we observed that the number of arc collapses is usually less than 5% of  $n^2$  and that the 90th percentile of the traversal lengths is around 10% of  $n$ ; while no traversal was longer than 30% of  $n$ .

## 6.4 Feature Tracking

By using a continuous transformation, we can identify structural changes of the augmented merge tree and we know the exact time and the correct order of all events. This allows us to compile tracking information event-by-event and to understand feature tracking as an independent extension of the transformation process. Although it is necessary to transform the fully augmented merge tree to notice all topological events, for feature tracking, it suffices to track only the unaugmented merge tree, i.e. the superarcs. This is because once we keep track of all regions accurately, the regular nodes neither affect the number of features, nor their hierarchy or properties. Because regular nodes are stored implicitly together with the superarcs, quality measures can still be determined for each feature.

As already mentioned in the algorithm overview in Chapter 6.2, the time-varying merge tree supports feature tracking in terms of the superarcs for the complete function, or in terms of the components of specific superlevel sets. For both cases, we present *operators* to extend the merge tree transformation.

### 6.4.1 Tracking the Complete Function

Between two time steps, superarcs may be born, die, or match with a superarc of the following merge tree. For birth- and death-events we keep track on which parent superarcs they take place because the tree may undergo many changes. For example, a newborn superarc may move within the tree, or it may give birth to other superarcs. Similarly, a superarc on which another superarc died may move or die. In rare circumstances, an arc can also give birth to the arc it dies on afterwards. Therefore, we have to store tracking information recursively. To display superarc relations later on in the visualization, each superarc needs a representative. We use its upper supernode for this purpose. Leaf superarcs are thus represented by their maxima, and inner ones by their upper saddle node. For each transposition, we verify if the superarcs of the tree are affected. If necessary, we create or destroy tracked superarcs accordingly, and we record on which superarcs these changes take place. If superarcs do not change, we still record when regular nodes change their association to a superarc, and we update a superarc's representative if required. Updating the representative is important because a moving feature can be represented by a completely different set of grid vertices in the next time step.

#### Implementation

For every superarc we maintain a *tracking record* that stores the following information: the initial representative, the current representative, the superarc born from, and the superarc died on. Initially we determine all superarcs of the merge tree, set their representative, but leave the entries for superarc “born from” and “died on” empty. We also store for each node to which superarc it belongs. Regarding the nomenclature, being *born on* another superarc means that the lower node which becomes a new critical node after the transposition changes its superarc affiliation to the newly created superarc. That is, the new superarc gets born on the superarc the lower node belonged to before the transposition. The opposite defines the meaning of a superarc *dying on* another one.

Tracking features of the complete function happens event-by-event and is thus an independent part of the transformation process (Algorithm 3, step 34). One possibility to track superarcs, including their precise place of birth/death and correct times, is to consider the node types before and after an arc collapse and to handle this event according to the case table shown in Figure 6.3. The table summarizes all possible configurations of an arc collapse. They result from taking all combinations, but neglecting symmetrical events (e.g. “a saddle node rises above a maximum” versus “a maximum falls below a saddle”) and impossible events (e.g. “two maxima

<p> <span style="color: orange;">■</span> this node is the new representative of its superarc  <span style="color: magenta;">■</span> this node changes superarc affiliation  <span style="color: blue;">■</span> this and all nodes until the next saddle change superarc affiliation  <span style="color: red;">@node2</span>  <span style="color: red;">D<sub>node1</sub></span> node1's superarc dies on node2's superarc  <span style="color: green;">@superarc / node</span>  <span style="color: green;">B</span> new superarc that is born on superarc / node's superarc </p>		<p> none <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">0</span>    one <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">1</span>    more than one, not all <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">1+</span>    all <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">V</span> </p> <p>number of upper node's children that become lower node's children after collapse</p>
<p>regular passes maximum <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">0</span></p>	<p>saddle passes maximum <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">0</span></p>	<p>only two children <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">0a</span> <span style="color: red;">@b</span> <span style="color: red;">D_s</span></p>
<p>regular passes regular <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">0</span> <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">V</span></p>	<p>saddle passes regular <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">0</span> <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">V</span></p>	<p>only two children <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">0a</span> <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">V</span></p>
<p>regular passes saddle <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">0</span> <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">1</span> <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">1+</span> <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">V</span></p>	<p>saddle passes saddle <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">0</span> <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">1</span> <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">1+</span> <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">V</span></p>	<p>only two children <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">0a</span> <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">V</span></p>

Figure 6.3: Case table to track features of the complete function: The case table summarizes how tracking records are updated according to the node types of the swapping nodes before and after a single transposition. For every possible configuration (dotted boxes), the bold arc is about to collapse and all possible results are shown to the right. In principle, superarcs die when supernodes pass each other, and a new superarc is born when the passing node becomes a supernode.

swap”). Possible results of a configuration reflect possible changes of  $l$ 's upper link. Furthermore, the table indicates in which configurations superarcs are born or die and in which cases a node's superarc affiliation changes.

The tracking consists of creating, destroying, or updating affected tracking records after every arc collapse. For example, if before a transposition the lower node is a saddle  $s$  and the upper node is a regular node  $r$  (cf. Figure 6.3, right column, second

row) and both nodes are saddles after the transposition (case 0), a new superarc is created with  $s$  as its current representative and  $s$ 's previous superarc as the “born from” entry. Then we set  $r$  to be  $s$ 's previous superarc's new representative and update the superarc connecting both nodes. For simplicity, events relating to the minimum of the tree are considered to be either regular or saddle events.

### Post-Processing

Once the transformation, and thus feature tracking, finishes, we know for each original superarc whether it matches a superarc in the final merge tree or whether it dies. For dying and newborn superarcs, we also know where exactly the death and the birth happen; this information is recorded in the “born from” and “died on” entries of the tracking records.

Because a function's main features naturally appear as maximum-saddle pairs of significant persistence, size, or stability, we restrict further processing only to leaf superarcs. To create pairwise relations between original and final superarcs, we post-process the tracking records (Algorithm 3, step 37) as follows: If the “died on” entry for an original leaf record is empty, we associate that record's initial representative with its current representative, and store this as a *match record*. If the “died on” entry of an original leaf record is not empty, we recursively follow it until we reach the record where “died on” is empty. We associate the original record's initial representative with the found record's current representative and store this as a *death record*. Finally, for each new leaf record, we recursively follow the “born from” entry, associate the found record's initial representative and the new record's current representative, and store this as a *birth record*. This produces a set of records, classified into either match, birth, or death events, that describe how features of the complete function, as described by leaf superarcs, relate to each other between the original and the final tree.

#### 6.4.2 Tracking Superlevel Sets of the Function

In addition to the analysis of the complete function, analysts also study the evolution of threshold-based features. In this case, one is interested at which times superlevel set components appear, join, split, or vanish; or if components belong together in both time steps. Note that these events do not relate to the homonymous events of the complete function: while a birth/death event in the complete function means the appearance/extinction of a new/existent superarc in the tree, a superlevel set component gets born/dies if an existent superarc just starts/finishes containing a

particular value  $\epsilon$ . That is, even if the structure of the merge tree does not change at all, a superlevel set can exhibit manifold changes just because nodes of the dynamic tree change in their values. To notice all topological changes of the superlevel set, we need to find out at which times superarcs of the time-varying merge tree start or finish containing  $\epsilon$ . This requires awareness of structural changes of the complete function because newborn superarcs could affected the superlevel set.

When a superarc starts or stops containing the threshold  $\epsilon$ , we store the event's type, its time, and the participating superarcs as a *superlevel tracking event*. Successive events of each component are then linked together to obtain a complete description of the changing superlevel set. Source of these linked events is either a new component that gets born during the transformation, or a component that already existed in the first time step. If a superarc contains  $\epsilon$  throughout the whole transformation, the superlevel set component directly matches in both time steps.

### Implementation

Tracking a superlevel set also happens simultaneously to the merge tree transformation. However, the transformation is *only* based on arc collapse events, i.e. on those events that potentially change the tree's structure. Because this implies that we could miss that one or more tree nodes pass value  $\epsilon$  between two consecutive transpositions, we have to extend the transformation's granularity to cope with tracking events of superlevel set components.

Before the transformation starts, we determine for all critical tree nodes the time when they pass  $\epsilon$  according to linear interpolation. Pass-times are then added to a priority queue that is used during the transformation at two places: Before performing an arc collapse, we handle "outstanding" tracking events in the queue, i.e. those with a pass-time smaller than the current arc collapse time. After the arc collapse, we update the priority queue based on the node type of both involved arc nodes. If a regular node becomes a critical node we determine the time when it passes  $\epsilon$  and potentially add it to the priority queue. Likewise, if a critical node becomes regular after the arc collapse, we remove it from the priority queue. Note that once the transformation has finished, the queue could still contain pass events that occur subsequent to the last arc collapse.

To handle a critical node  $n$  when it passes value  $\epsilon$ , we identify the superlevel tracking event based on  $n$ 's node type, e.g. whether it is a saddle or a maximum, and whether  $n$  passes  $\epsilon$  from below or above. Figure 6.4 summaries all possible configurations and their corresponding superlevel tracking event types. Newly generated superlevel tracking events are stored together with  $n$ 's adjacent superarcs

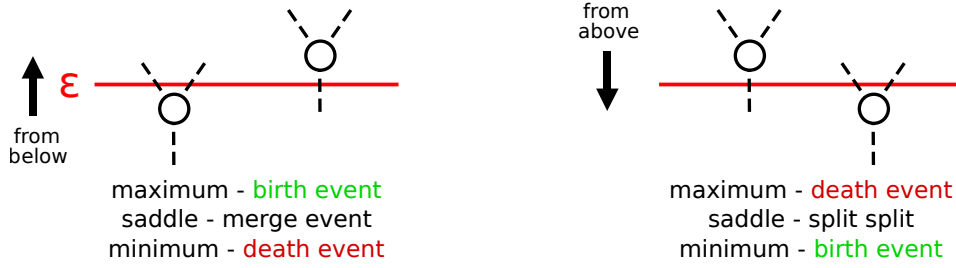


Figure 6.4: Event types for tracking superlevel set components: Superlevel set components appear, join, split, and vanish when superarcs of the varying merge start/stop containing threshold  $\epsilon$ . The type of the superlevel set component's topological change depends on the type of the critical node passing value  $\epsilon$  (red line) and whether it passes the threshold from below or from above.

to connect them with their future events. The transformation's pseudo-code shown in Algorithm 3 also indicates at which places the transformation needs to be extended in order to track superlevel sets (Steps 6, 35, and 36).

## 6.5 Simplification

Similar to topological simplification of the merge tree of a time-invariant scalar function (cf. Chapter 3.2.3), we also eliminate fluctuations and noisy structure of the dynamic function. To focus only on the prominent features of the high-dimensional, temporal data, we simplify the structure within a time step and adjust the tracking information that is related to removed features. To help the analyst find appropriate simplification thresholds, a simplification controller as described in Chapter 5.2.2 can be used to remove noise from the merge trees belonging to the given time steps. Note that changing simplification thresholds for the merge tree at time step  $i$  requires to adjust the tracking information between time steps  $i - 1$  and  $i$ , and between time steps  $i$  and  $i + 1$ .

### Pre-Simplification

The timeslices can be simplified prior to the topological analysis. To this end, we determine the merge trees for all given scalar functions, simplify them individually based on user-specified thresholds, and adjust function values in the grid according to their simplified values in the trees. Conceptually, this approach removes fluctuations of the density function and leaves plateaus behind. As a consequence, noisy regions are not tracked anymore, and the transformed merge trees, the tracking records of the complete function, and linked superlevel tracking events are automatically consistent. However, this also implies that changing simplification thresholds requires



to repeat the whole analysis, i.e. the transformations and feature tracking. For larger data, repetitive transformations constitute a runtime bottleneck if simplification thresholds are changed frequently. To alleviate these issues we can use the following post-processing methods instead.

### Merge Trees

The merge trees at the given time steps are simplified in exactly the same way like in the time-invariant case. That is, based on the branch decomposition of the merge tree at time step  $i$ , the sliders of the interactive simplification controller indicate the value distributions of the branches in terms of their persistence, size, and stability. The merge trees can be simplified with the same thresholds for all time steps, or with different thresholds for each individual time step. The latter is helpful for local analysis to increase or decrease the structural granularity only at a certain point in time, or to understand which noisy features eventually evolve to prominent features.

### Tracking Information of the Complete Function

If a superarc is removed from the tree, we also have to adjust those tracking records that have this superarc as their origin or target, i.e. as their initial or current representative. To this end, we redirect the tracking record by replacing the removed superarc by its parent superarc. Tracking records may become redundant by this process. We remove the record if: (a) it is a birth record, but the target was simplified; (b) it is a death record, but the origin was simplified; and (c) if it is a match record, but both the origin and the target superarcs were simplified.

### Tracking Information of Superlevel Sets

Superlevel tracking events occur if superarcs of the dynamic merge tree start/finish containing value  $\epsilon$ . Because the quality measures of these superarcs change during the transformation, and because it is possible that these superarcs no longer exist after the transformation, we cannot know whether superlevel set components temporally belong to noisy features of the complete function. Therefore, we simplify superlevel set events based on *time persistence*, which is the time interval between two critical events, i.e. a user-specified lifetime for a component. Because linked superlevel tracking events can easily be imagined as a graph, we can recursively peel off those arcs whose end-nodes (i.e. tracking events) differ in time by less than a user-specified threshold. Note that whole connected components of the graph could be removed by simplification. It is also necessary to remove source events that represent simplified

superarcs at the given time steps, i.e. those belonging to structural noise of the given functions.

## 6.6 Conclusion and Discussion

We introduced time-varying merge trees as a compact description of time-varying scalar fields and described an algorithm that computes the augmented and the unaugmented version of these trees. To track features over time, we transform the merge trees of consecutive time steps into each other, record all changes of the varying tree, and finally relate subtrees between the given time steps. Supported features are those described by complete superarcs or by superlevel set components of the changing function. Because the transformation works only on the grid edges and on the merge tree itself, it does not require complex case distinctions and, thus, scales to higher dimensions.

Because the time-dependent merge tree algorithm works on-line, i.e., new time steps do not affect the result for previous time steps, the algorithm can be used with streaming data. However, a requirement of the current implementation is that the number of grid vertices must not change. This is generally the case for scalar fields provided on a grid, but limits the application for high-dimensional data where regular grids are impractical and more flexible grids are needed. The current approach would also benefit from a topological simplification that works directly on the time-varying merge tree. The current solution—to simplify each merge tree independently—potentially removes “weak indicators”: tiny structures that in time grow to features of interest. Because of merges and splits it is difficult to define an equivalence of structures over time, which would be the first step before telling apart features and noise.

The complexity of a transformation, and thus its runtime, depends on the topological variance between two time steps. But what is the lower bound on the algorithmic complexity of both the augmented and unaugmented version of the problem? While we know that the events we consider in the augmented case are both necessary and sufficient, the unaugmented version may need to process fewer events; though we are currently unable to characterize which ones. Hence, a potential improvement is to investigate how the storage of topology changing events of an initial transformation of the augmented tree could lead to a persistent data structure that allows quick extraction of superlevel sets and optimal runtime of subsequent transformations based on the (smaller) unaugmented tree. Asymptotically, the algorithm could benefit from a data structure that optimizes the operation of

identifying region adjacency after an arc collapse. In practice, however, the run time of the algorithm is mostly affected by the large number of changes to the event queue. As most inserted arc events get canceled, a form of deferred event handling would help.

We did not yet study the extension of the transformation to time-varying contour trees. Because it is trivial to adapt our algorithm to give the time-varying split tree, it is possible to extract the augmented merge and split tree for a given time and combine them into the contour tree. But this does not immediately give a tracking of the contour tree's superarcs.



## Chapter 7

# Topological Visualization

Visualizing the structure of a high-dimensional scalar function is already challenging for time-invariant data. As demonstrated in Part I of this thesis, topological analysis and topology-based visualization based on the merge tree are efficient tools to solve typical problems for visualization of high-dimensional data, like structural occlusion and visual complexity. However, for time-dependent data, the time-varying merge tree is even more complex and provides complex information: the hierarchy as well as three quantitative properties for all features at the given time steps; tracking information to relate features over time; and the types and exact times for all events regarding the features of the complete function and those defined by a single superlevel set.

This chapter introduces a visualization of the time-varying merge tree that is based on the topological landscape metaphor. It shows the structure of the scalar fields at the given time steps and indicates feature correspondence and structural changes across time using visual links. To reduce the clutter caused by showing many features and links over time, we add an additional layer of visual simplification and give special attention to feature groups that do not change their topology, representing them by one visual link rather than many. A variation of the visual design also supports analysis of single-threshold feature evolution that, compared to related work, suggests visually how to pick and adapt thresholds. The result is an interactive visualization framework that supports in-depth analysis of time-varying features in high-dimensional scalar fields.

### 7.1 Related Work

Visualizing tracked features can be challenging. Bremer et al. [18] survey methods and applications of topological feature tracking in molecular analysis, combustion

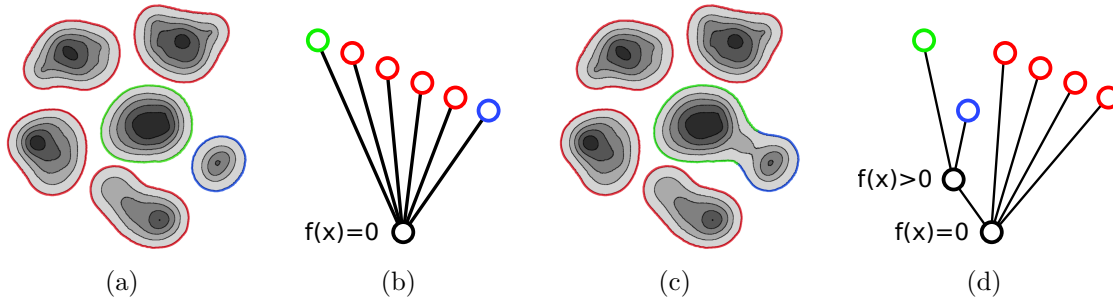


Figure 7.1: Strong variation in the topological description caused by a small variation of the underlying scalar function: (a) Color-coded density function (dark=dense) showing six separated regions of varying density. (b) The merge tree of the density function with a multi-saddle and one maximum per dense region. Without loss of generality, leaves are sorted decreasing by their density value. (c) Two regions join by increasing density in the void between them. (d) The merge tree has a non-zero density saddle for the joined regions and a multi-saddle for the remaining separated regions. The blue leaf node covered the maximal distance in the sorted order. Consequently, every visualization of the time-varying merge tree would have to reflect this complex change, which likely leads to intersections and occlusion artifacts.

simulations, and porous materials analysis. The results are shown as feature tracks embedded in the original domain. Similarly, Chen et al. [29] showed an embedding of the Reeb graph in the original data to track level sets of particle data. Other approaches store the tracking in a graph that is then shown using graph layout techniques. Widanagamaachchi et al. [170] present tracking graphs that update quickly when the analyst changes the isovalue of the tracked features. Because we compute tracking information for all features up front, we need to compute the layout of the tracking information only once. In contrast to these techniques, our method is able to show additional feature properties, e.g. their size and robustness. More related work about the topological visualization aspect is also part of the papers cited in Chapter 6.1 (on page 146).

## 7.2 Visualization Design

Topological concepts have proven to be efficient in summarizing, simplifying, and visualizing otherwise difficult to illustrate high-dimensional data. However, for time-varying applications, visualizing a dynamic merge tree quickly becomes complicated for a subtle reason: the tree can change substantially even for small changes of the underlying scalar function. This is demonstrated in Figure 7.1 based on a 2-D density function whose merge tree changes dramatically by adding only a single point to the underlying point cloud. It is obvious that this *instability* of the changing topological

description has to be reflected by any visualization of the dynamic merge tree. For example, picking up the 3-D topological landscape metaphor, one possibility to illustrate the time-varying merge tree could be to represent each time step by its own layer of earth in a 3-D geological sediment metaphor with individual landscapes conceptually stacked on top of one another. A problem with such an intuitive metaphor would be that a sudden and strong change of the merge tree could force a particular feature to appear at very different locations in adjacent sediments. Problems with intersection and occlusion would then be inevitable, leading to strange and unintuitive artifacts that are not explainable by a geological metaphor itself. In consideration of these facts, we have to come up with a simple and yet informative and easy to read visual representation of the changing tree.

The visual design of the time-varying merge tree is based on two requirements: First, because we aim for an application-independent visual analysis of high-dimensional scalar fields, the visualization should convey the complete information provided by the time-varying merge tree. While it could suffice for certain applications to limit this information, e.g. by discarding feature hierarchy or some properties, we still seek a drawing that shows all supported properties of the time-varying merge tree to make the topological approach applicable to many applications. Second, the visual complexity and the amount of interaction required to explore the data should be kept small. Provided that one dimension in the final drawing will be needed to depict time, the remaining information of the time-varying merge tree is certainly too complex to base the visualization on a *continuous* design, like a sediment- or flow-like illustration in 3-D. Such a design would certainly introduce massive occlusion, visual clutter and (self-)intersection of hierarchical features.

To address these issues, we use a discrete design that separates the depiction of structure from that of temporal evolution to avoid unnecessary problems with occlusion while tracing features visually. To display the structure of the high-dimensional scalar fields occlusion-free at the given time steps, we adopt and extend the landscape profile metaphor. To reduce intersections and visual complexity, we display temporal feature evolution with visual links between consecutive profiles.

### 7.2.1 Merge Trees as Landscape Profiles

Because we focus on (static) merge trees, which can be visualized in two dimensions as a landscape profile, the third dimension is still available to convey the time aspect of the data. We take the set of simplified merge trees at each time step, construct their landscape profiles and arrange them one after another according to their time stamp. Furthermore, we render the scene with orthographic projection to facilitate

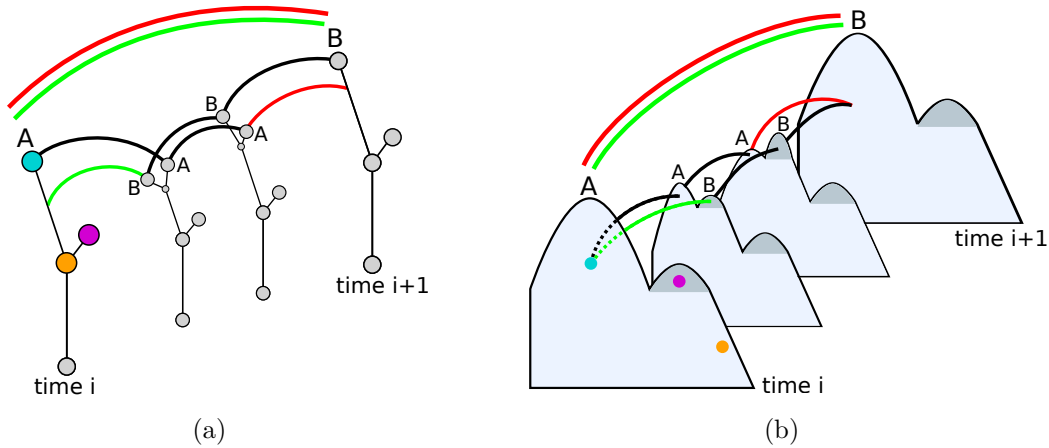


Figure 7.2: Exemplification of a complex topological event occurring *between* two consecutive time steps; explained with single merge trees and their corresponding 2-D landscape profiles: (a) Two superarcs (A and B) are connected by a green (birth) and a red (death) visual link if the first superarc (A) gives birth to another superarc (B) on which it dies afterwards. This often happens for moving features or if the maximum inside a regions changes to another grid vertex. A green and a red link between two hills can also result from simplification, i.e. when a parent hill inherits a visual link from a removed child hill. (b) The colored dots in the first profile indicate the coordinates used for the representative supernodes of the corresponding tracking record's visual link.

feature comparison between the different time steps. For aesthetic reasons, height values are normalized by the maximum height of all profiles, which is also restricted to a percentage of the maximum width of all profiles. To focus on the main features in a profile, we also provide the option to shorten long slopes and plateaus by taking the logarithm of each superarc's size. A variation of ambient occlusion [48] furthermore enhances the perception of depth between the profiles with shadows.

## 7.2.2 Tracking Information of the Complete Function

As illustrated in Figure 7.2, every superarc of a merge tree is represented by a distinct part in the landscape profile. Because every tracking record associates two superarcs of two consecutive merge trees, we create a visual link between the corresponding parts of the profiles. For each record, we identify both superarcs by the record's initial and current representative, identify their areas in the profiles, and use these areas' centroids as the visual link's origin and target coordinates. In its simplest form, a link is a line. To indicate the type of the structural events, we use black, green and red links to represent match, birth, and death records, respectively.



Without simplification, every tracked leaf superarc either matches to, or dies on another superarc in the transformed merge tree of the next time step. Hence, each hill has either one black or one red outgoing link. Similarly, for a newborn superarc, the corresponding hill in the target profile has a green incoming link. Note that hills can be the origin of more than one green link because a superarc can give birth to many other superarcs. Likewise, a hill can also be the target of many red links if several superarcs die on the same one. After topological simplification, parent hills inherit redirected visual links from their former children.

If examined from above, the profiles in combination with the visual links look similar to isotracking graphs as used by Sohn et al. [155], or Bremer et al. [20]. However, in our case the links show the evolution of features of the complete function rather than only of individual isocontours. Furthermore, the width of the hills specifies a feature property and the topology-based layout of the hills also ensures that hierarchical features are close to each other and can be sorted, e.g., by persistence, size, or stability. If examined from a side view, the profiles also describe hierarchy and feature properties and can thus help the analyst identify interesting subsets.

### Tracking Information Filter and Interactive Selection

The analyst can filter tracking information by selecting arbitrary parts of the profiles. Visual links are then filtered either forwards, backwards, or in both directions in time. The filter preserves only those links that emerge from a user-selected hill or that are (recursively) reachable from a user-selected hill by a traversal through the 'born from' or 'died on' entries of the corresponding tracking records. More sophisticated analysis is achieved by combining simplification and interactive selection. For example, small features could be excluded from the simplification if they are related to the evolution of user-selected features—e.g. if noise evolves to a prominent user-selected feature.

### Link Reduction and Minimization of Link Intersections

The presence of many related features translates into many visual links between adjacent profiles (cf. Figure 7.3a) and makes it difficult to trace structural evolution visually. Although topological simplification is the primary tool to remove irrelevant hills (cf. Figure 7.3b), we further reduce the remaining links to increase readability and visual clarity. We use the following strategies:

1. *Link aggregation* is applied to those hills with multiple incoming or outgoing links. The number of links is reduced by grouping them by type/color. To this end, the involved links are split into smaller segments that are combined if the

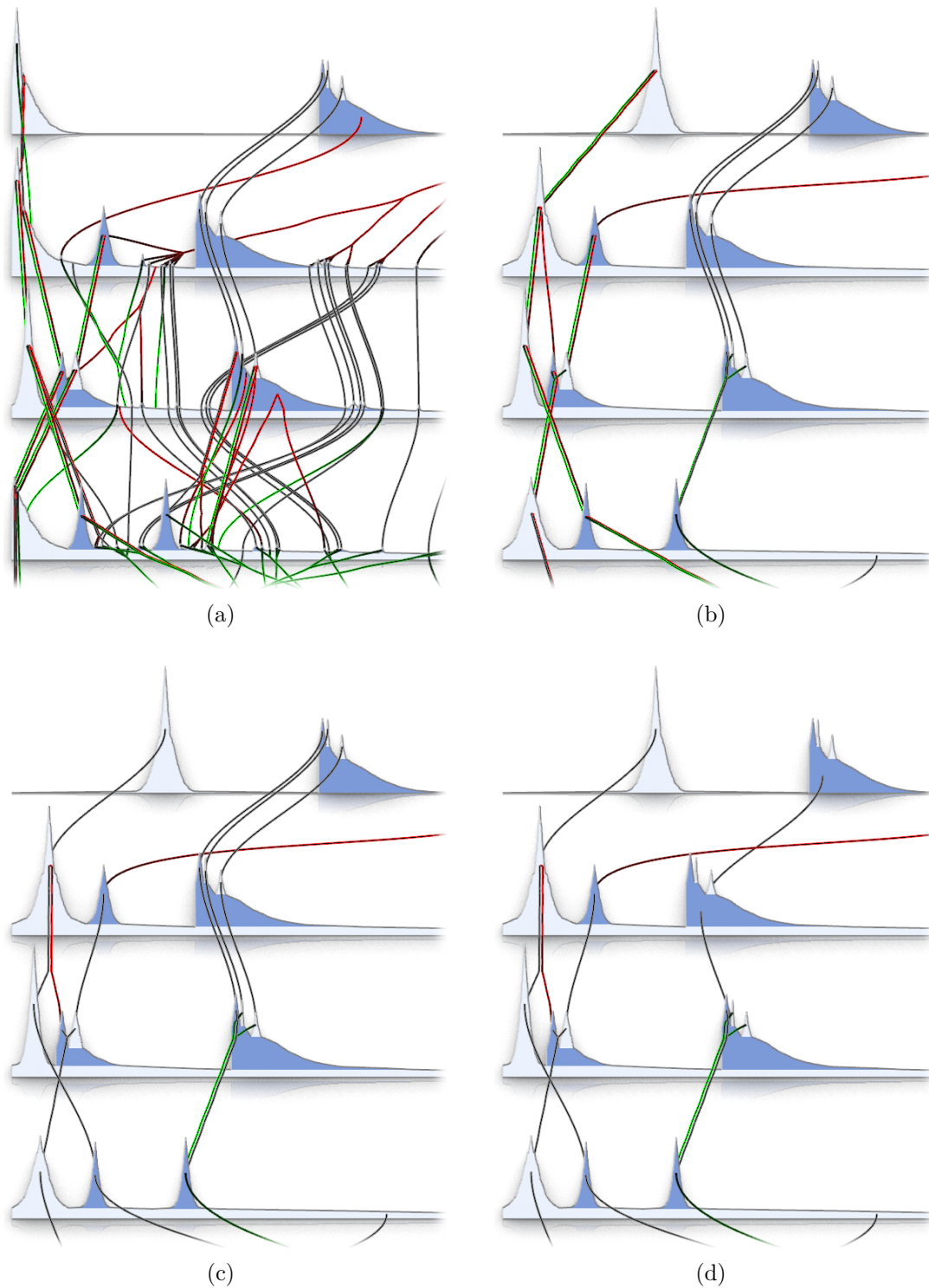


Figure 7.3: Link reduction strategies for the time-varying merge tree visualization of the scenario explained in Figure 7.4: (a) Landscape profiles and visual links without topological simplification and no application of any link reduction strategy. (b) After topological simplification (minimum region size is 60 grid vertices). (c) After *link unification*. (d) After *match-link combination*.

angle between close-by segments of the same type/color is only small. As a result, every hill has at most one incoming or outgoing link per type/color, which forks if necessary so that all aggregated links reach their targets.

2. *Link unification* is applied if links of different type/color have exactly the same origin and target hills. Such link configurations represent special topological events occurring in one and the same region, e.g. if a superarc gives birth to another one on which it dies afterwards (cf. Figure 7.2a). This often happens for moving features, where two maxima exist in the meantime: one maximum at the old location that vanishes and the maximum at the new location that is rising during the merge tree transformation. Another reason for multiple links with the same origin and target is that parent hills can inherit redirected links of simplified child hills. If such detailed insights go beyond the scope of the intended analysis, such relations can be treated as a simple match by unifying the involved links into one (black) match link. This is demonstrated in Figures 7.3b-c for the hills on the left-hand side.
3. *Match-link combination* is applied to hierarchical features that do not change between two time steps. In this case, individual (black) match links belonging to subfeatures are combined into a single match link to connect the hills at the lowest shared saddles. This strategy is illustrated in Figures 7.3c-d for the hills on the right-hand side. A single black link between two profiles indicates that the structure of the function remains constant entirely.
4. *Re-sorting* saddle node children in the merge trees can be used to change the positions of the hills without changing the topology of the profiles. This strategy was already introduced to sort the hills from-left-to right either by persistence, size, or stability. In the time-varying case, changing the positions of the hills could be used to optimize the number of crossing links. However, note that switching subtrees does not allow to switch arbitrary hills; and can, thus, also increase the number of crossing links.

It is also possible to implement other visual links and aggregation strategies. For example, bifurcations in the aggregation could reflect the exact times of the structural changes. However, depending on the variance of the dynamic scalar function, this might contradict our primary goal of reducing the number of visual links and their crossings between the profiles.

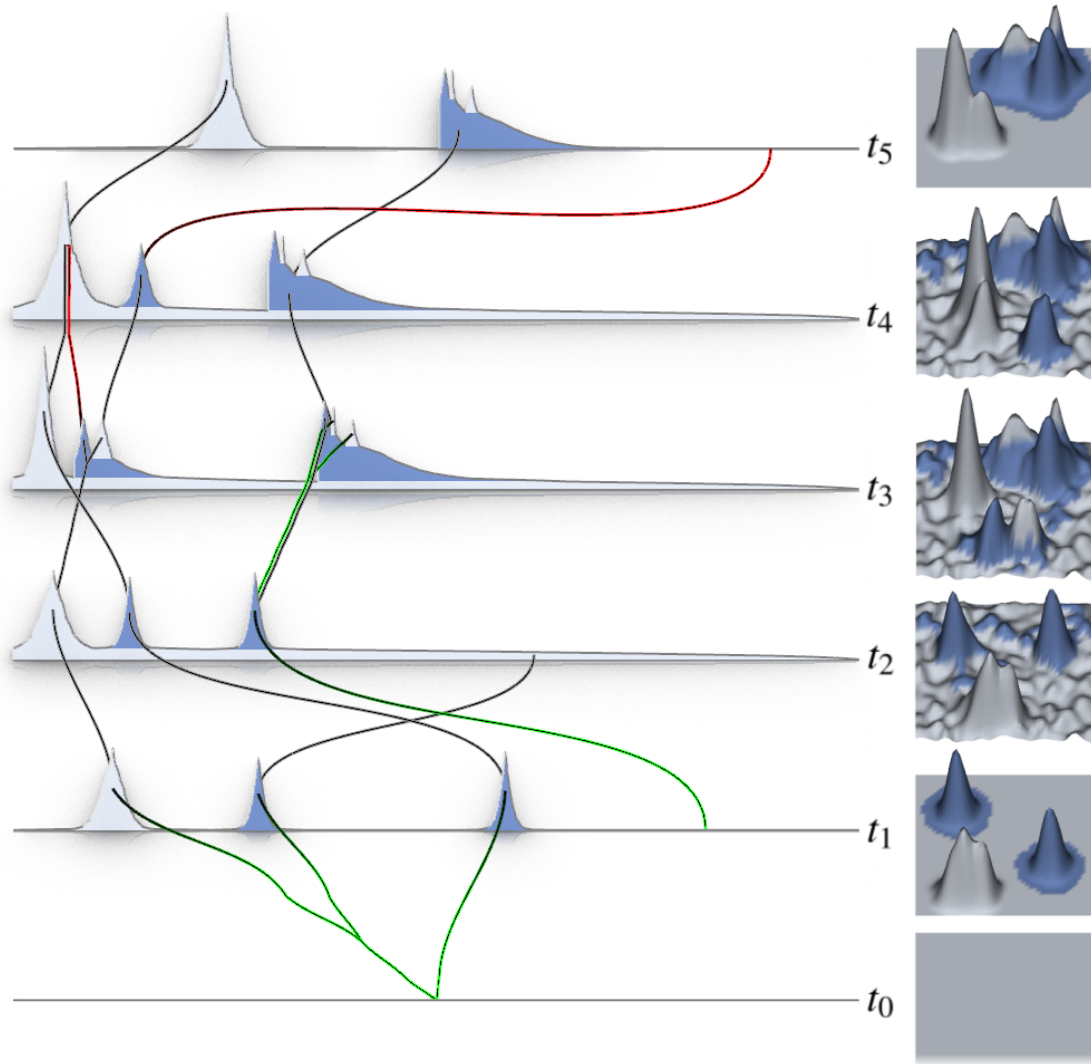


Figure 7.4: Visualization of the time-varying merge tree based on an artificial 2-D function (shown to the right) after topological simplification:  $t_0$ : flat function.  $t_1$ : three main features are born.  $t_2$ : all features move and noise is added.  $t_3$ : all features move, one splits, the other two grow in size and persistence, respectively.  $t_4$ : features move and join.  $t_5$ : one feature dies, another is rotated, noise is removed.

### Artificial 2-D Function

Figure 7.4 shows the visualization of a time-varying merge tree, for better understanding, based on an artificial function that represents a density-based clustering in 2-D; the scenario is described in the caption. The sample grid contains 2 500 vertices and the computation of the topologically most complex time step takes less than a second on our machine with two 2.6 GHz quad-core processors. Because the transformations between consecutive timeslices are processed concurrently, the total time to obtain the image, including topological simplification to remove noisy features, is approximately one second.

As revealed by this example scenario, the tracking is robust with respect to noise and feature shape, and moving features are recognized if their spatial difference is small enough. Typically, a moving feature is identified as a superarc whose implicit regular nodes (i.e. the grid vertices) only change, or as a superarc that gives birth to a new one on which it dies afterwards. A counter-example is given in time steps  $t_1$  and  $t_2$ . Because the spatial difference of the right feature in the function is too big, a new feature gets born and the old one vanishes. A higher time resolution would be needed to infer this as a fast moving feature. In general, the old feature would die (like in  $t_5$ ), but here it matches to one of the maxima produced by the added noise in that area. Because noise was removed by topologically simplification, the link target correctly points to the removed hill's parent hill. If a feature splits, the new one typically matches to one of the former maxima (cf.  $t_3$ , second hill) or a new feature is born (like on the right hill in  $t_3$ ). Likewise, if features join, they first share a higher saddle and then one feature would die on the other (like on the first hill in  $t_4$ ). In  $t_3$ , the first two hills change their order because the hills in the profiles are sorted by persistence to put the highest hills to the left.

### 7.2.3 Tracking Graphs of Superlevel Sets

Tracking information about a superlevel set is inherently affected by the variation of the complete function. Hence, it is desirable to illustrate changes of a superlevel set in the visual context of the complete function to help the analyst find interesting thresholds. Because scalar values, and thus a particular threshold  $\epsilon$ , are encoded by height information in the landscape profiles, superlevel set components can be identified as the intersection of a profile with a horizontal line at height  $\epsilon$ . That is, intersection lines on the hills and their individual width specify the number and the approximated size of the superlevel set components. Arranging landscape profiles next to each other, thus, facilitates visualization of tracking graphs at the corresponding height of the adjusted threshold to illustrate how superlevel set components evolve over time. To focus on the tracking graph, while still maintaining the context of the complete function, we only show the silhouettes of the profiles above the adjusted value. Threshold modification is enhanced visually by showing bold intersection lines to point out the superlevel set components (and their size) at the given time steps.

Figure 7.5 shows an example tracking graph based on the artificial 2-D function (cf. Figure 7.4). Topological simplification was omitted for demonstration purposes, which is why the profiles still contain small hills corresponding to noise. The adjusted threshold for the superlevel set is accentuated by the bold line segments on the hills. Above the black line segments, the context of the profiles is only suggested

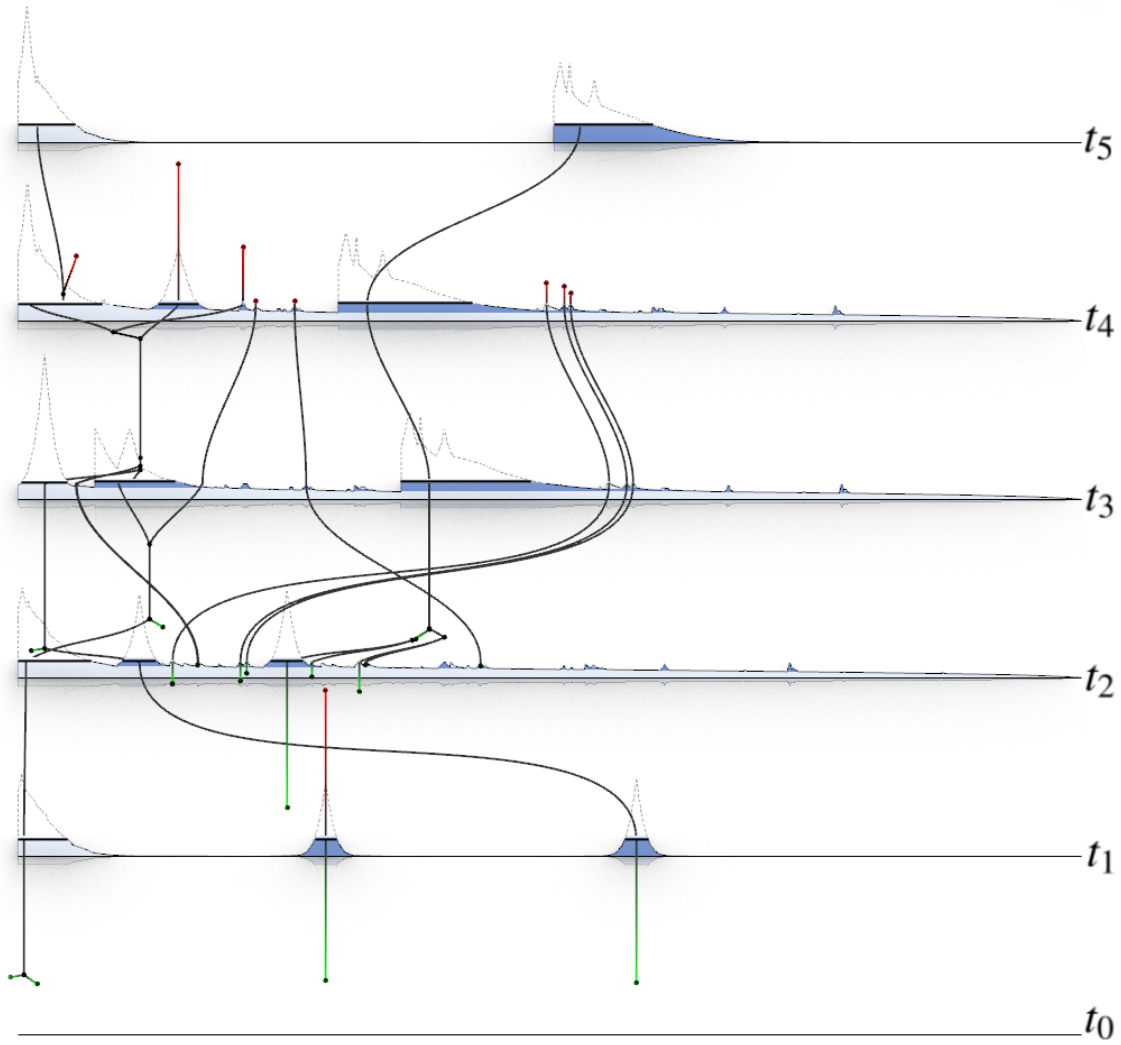


Figure 7.5: Visualizing superlevel set tracking information based on the unsimplified function of the scenario described in Figure 7.4: Superlevel set components are highlighted as bold line segments on the hills. The context of the complete function is indicated by the silhouettes of the profiles above the adjusted threshold. Birth, death, and match events reveal at which times features of the complete function start/stop containing value  $\epsilon$ . The green birth and the red death event of the second hill between  $t_1$  and  $t_2$  confirm that this dense region cannot be matched in both time steps. Instead, the superlevel set component belonging to the new region is born shortly after  $t_1$ , and the component belonging to the old region dies shortly before  $t_2$ .

by dotted silhouettes. The threshold  $\epsilon$  can be changed interactively to identify the superlevel set components at the given time steps for other height values. However, the tracking graph is, generally, not updated in real-time, as this requires to repeat the transformations for another  $\epsilon$ . This drawback will be addressed in some more detail at the end of this chapter. Regarding the 2-D example scenario, we can make several observations: The coordinates of the green birth events between  $t_0$  and  $t_1$ , and the red death events between  $t_4$  and  $t_5$  reveal at which times ( $z$ -coordinate) and

in which order these superlevel set components appear and disappear, respectively. Conceptually, these are the times, when the hills, indicated by the silhouettes, of the next/previous time step have exactly this height value according to linear interpolation. Between  $t_1$  and  $t_2$  we can see that most hills belonging to noise do not have a corresponding superlevel tracking event. This is because their height is still below the adjusted threshold. Moreover, the red death event and green birth event of the second hill confirm that, because it is moving too far for only one time step, these features cannot be matched. It is also visible from the parallel red and green link that there were two density maxima for the most time; the new one that is born shortly after the first time step and the old one that dies shortly before the second time step. The transformation from  $t_2$  to  $t_3$  shows how the superlevel set component on the right main hill grows in size by aggregating surrounding noise. This reflects how the corresponding dense region belonging to this maximum also grows because the surrounding density peaks die in this region while the density increases (also see the transition from  $t_2$  to  $t_3$  in the function in Figure 7.4). It is also visible from the merge events in this area that the noisy features must have died on the main component *after* their saddles passed the adjusted threshold  $\epsilon$ ; the opposite would have been indicated by red death links instead of the shown merge events. The tracking graph of the remaining time steps primarily reflects how the features die when their density in the complete function falls below the selected threshold  $\epsilon$ . Independent from the currently chosen threshold, the analyst can always identify meaningful values of  $\epsilon$  from the visual context of the profiles to analyze the tracking graphs of prominent or suspicious features.

### Graph Layout and Implementation

To obtain the graph layout, it is necessary to determine 3-D coordinates for all topological events to connect them with an edge. Note that there is only one degree of freedom in 3-D because an event's time and the threshold  $\epsilon$  itself already fix two dimensions. In order to embed the tracking graph in the context of the complete function, an event's remaining  $x$ -coordinate is determined based on the unique position of its involved superarc on the profiles. That is, events are placed in front of the hills on which the superlevel set component can be found, or which contained the superlevel set component at the event's time. It could happen that superarcs of intermediate events have no counterpart on the profiles because they (dis)appeared during the transformation. In this case, we use the 'born from' and 'died on' entries of the corresponding tracking record to find the related superarc and its position on the profile.

Because superlevel set tracking information is available for each pair of consecutive time steps, the tracking graph can be constructed concurrently for each pair of consecutive profiles. We iterate through the set of source events, i.e. those describing initially existing or newborn superlevel set components, and recursively handle their linked events. With this strategy, the graph is generated *path*-wise and we can skip recursive calls if paths “cross” at merge events that were already handled. For each event, we determine its own position to place a glyph and to draw edges. For existing and newborn superlevel set components, we determine position based on the involved superarc’s position on the first profile. Positions for split, merge, and death events are determined based on the hill’s coordinates in the second profile. Glyphs and edges belonging to birth and death events are colored green and red, respectively. For aesthetic reasons and to increase its readability, the layout is modified at some places: We spread linked incoming/outgoing events around merge/split events; we use straight lines for birth/death events that are not connected to merge/split events; and the position of a superlevel set component at a given time step is set to the midpoint of the line segment on its hill.

## 7.3 Conclusion and Discussion

To enable inspection and exploration of time-varying, high-dimensional scalar fields, we visualize time-varying merge trees by combining landscape profiles—suitable for single merge trees—with a tracking information overlay. Visual clutter caused by showing many hills and links is countered by implementing another layer of abstraction to aggregate, unify, and combine visual links and by opting for a discrete design rather than for a continuous visualization. The presented prototype visualization is capable of presenting the complete information provided by the time-varying merge tree, thus making it versatile and also independent from the underlying application domain. To focus on particular features, the analyst can filter and trace them individually and in different directions in time. The granularity of the topological abstraction can be changed at the given time steps with interactive controllers to define appropriate values for the filter radii and the simplification thresholds.

Concerning visual aspects, the visualization is currently limited to showing a few time steps in a single image because the representation is not yet optimized. This limitation results from showing the complete function and more properties per feature and time step than competing techniques—which show tracking graphs of single isolevel sets without any feature properties—and from adapting existing approaches rather than using a visual representation optimized for time-varying trees.



The optimal depiction of time-varying merge trees, i.e. dynamic trees with several properties for nodes and edges, remains an open question. However, the proposed visualization design is flexible in that it can easily be modified. For example, a less complex visualization, e.g. in two dimensions, could be obtained by discarding some of those supported properties that are less relevant for a particular application domain.

For tracking superlevel sets of the input scalar function, the context of the complete function, as indicated by the silhouettes of the profiles, is helpful to find interesting thresholds. Nevertheless, a drawback of the current implementation is that changing the threshold  $\epsilon$  requires to repeat the merge tree transformations in order to update the tracking graph for another superlevel set. This is an indirect drawback of the transformation itself, and could be countered by a persistent data structure to accelerate superlevel set extraction (cf. Chapter 6.6 on page 162). Changing  $\epsilon$  and updating the intersection lines on the hills to identify and compare other superlevel sets and the approximated size of their components at the given time steps still happens in real-time.



## Chapter 8

# Application: Temporal Clustering

The computation and visualization of the time-varying merge tree are independent from a specific application. The required input only consists a of set of scalar functions defined on an arbitrary dimensional grid. One example application for the topology of high-dimensional, time-varying scalar fields is temporal cluster analysis. Similar to reducing the problem of studying a high-dimensional, but time-invariant point cloud to the study of a scalar field given by the point cloud's density, this approach can be extended by abstracting a dynamic point cloud by its varying density function and analyzing its merge tree topology over time. Exemplary for various applications that use a vector representation to encode multiple properties and semantic behavior of domain entities, the challenging task then is to determine and track clustering structure in the high-dimensional space and to illustrate the evolution of high-dimensional features appropriately on the screen.

This chapter explains how timesliced point data can be turned into a dynamic density function to be analyzed topologically with the time-varying merge tree. We use categorized documents and consider document analysis as an instance of temporal, high-dimensional cluster analysis. Building on the ideas to analyze time-invariant text data, documents assigned with meta-information about time are represented in the vector space model to analyze changes in the thematic composition over time. Starting with an initial set of documents, we visualize topic evolution—reflected by changing clusters—by continuously adding new documents. Contained topics as well as their evolution and the relation between them are finally visualized with the landscape metaphor augmented with visual links as described in Chapter 7.

## 8.1 Related Work

For point cloud data, Turkay et al. [164] present a system to explore time-varying clusters along with their structural properties, but this splits information into multiple views that need to be integrated in the mind of the analysts. In contrast, our visualization technique shows all relevant metrics in the same view as the cluster evolution. In subsequent work, von Landesberger et al. [108] present a system for visualizing spatially-referenced dynamic categorical data, i.e., trajectories that have a time-varying category. This method, representative of a much larger body of 2-D trajectory and univariate time-series data, usually defines clusters between trajectories, but our document data is modeled as single points in high-dimensional space over time.

Topic tracking developed several visual designs that are able to indicate feature properties. ThemeRiver [73] and stacked graphs [22] show the varying importance of document clusters over time. However, in the underlying model, clusters cannot join or split, e.g., for diversifying topics. TextFlow [36] extends the stacked graphs concept to show topic evolution as river-flows that join and split. But hierarchical relations among topics cannot be communicated and showing more than one property per topic has not been attempted, yet. Building on TextFlow, Cui et al. developed RoseRiver [37] which allows users to progressively explore the complex evolution of hierarchical topics. In a similar vein, Xu et al. [173] and Sun et al. [158] use flow-like depictions with splitting and merging topics augmented with tag clouds for visual analysis of topic competition and topic *coepitition* (a combination of “cooperation” and “competition”) of social media data over time. These designs are based on a psychological model that assumes humans would put each real-world object into exactly one category. Experiments have shown that there are no clear category boundaries, e.g., that there is a smooth transition from cup to bowl, that categories may nest, that each category has an ideal instance, and that instances have a certain degree of membership with their category [142]. Studying the merge tree of a point cloud’s density function seems to better match this model: it can capture category nesting, has an integrated measure of membership degree, and local maxima represent ideal instances. ParallelTopics [41] combines parallel coordinate plots, scatterplots, and flow-like depictions of topics over time to permit effective full text analysis of larger documents that may contain multiple topics based on the probabilistic topic model Latent Dirichlet Allocation.

### 8.1.1 Creating the Time-Varying Density Function

Because regular sampling is impractical in high-dimensional spaces, we approximate the density function in a similar way like in the time-invariant case (cf. Chapter 3). That is, to obtain a sufficiently accurate sampling scheme to identify region count, nesting, and region properties, we first determine the densities at all input point positions. After connecting these points with a neighborhood graph, missing saddles are upsampled, i.e. inserted at the midpoint of a graph edge if the density at this point is lower than at the edge's end-points.

For time-varying data, this sampling scheme is extended as follows: To make the computation of the time-varying merge tree tractable, we construct a grid that remains unchanged over time, but still supports sampling the density functions of all time steps with sufficient accuracy. First, we merge all input point clouds into a single set of points whose positions act as grid vertices. This makes it possible to find all regions of high density in any time step. To detect separated regions, we connect the vertices with a neighborhood graph, determine vertex densities and upsample midpoints of grid edges if necessary. The density estimation of the global graph depends on a filter radius  $\sigma_{global}$ . Note that a too large filter radius combines clusters, while a  $\sigma_{global}$  too small splits clusters and can even assign every data point to its own cluster. Because the choice of this parameter is vital, we provide the analyst with a filter radius suitability controller (cf. Chapter 5.2.1) to find an appropriate threshold efficiently. Using the global, up-sampled graph as the constant grid, we iterate through all time steps and determine the individual density functions based on those points with earlier time stamp.

Note that any change in scalar function at time step  $i$  requires to repeat the merge tree transformations between time steps  $i - 1$  and  $i$ , and between time steps  $i$  and  $i + 1$ . While our implementation defaults to using the same filter radius for all time steps, it is also possible to use different filter radii for each time step, e.g. for local analysis of a particular feature at finer granularity, or to combine multiple features to reduce visual complexity if a particular time step is more complex than others. However, if the density functions for each timeslice are estimated with different filter radii, the density function of the global grid has to be estimated with  $\sigma_{global} = \min(\sigma_i)$  to ensure both a constant grid for the merge tree transformations and that the missing saddles are found in all time steps.

### 8.1.2 Analysis of Document Collections

The literature for natural language processing knows many models and techniques to define *text features* and to describe, process, and finally visualize otherwise difficult to analyze, unstructured text data. One popular approach, called the vector space model [144], represents a document collection as vectors in a space with words as dimensions. Documents that share vocabulary will accumulate to (sub)clusters and thus their topical composition is mapped to the clustering structure of the high-dimensional point cloud (cf. Chapter 4.5.3 on page 99). Specifically for time-dependent text data, analyzing the vector space model aims at studying how topics appear, grow or shrink, how they join as a result of documents that combine word usage that was topic-exclusive before, or how topics split because new documents contain subsets of topic-related words and thus accumulate to groups in particular subspaces, e.g. if a new product is presented for an established technology branch.

#### New York Times Data Set

The data used in the first experiment consists of ten days of news data from September 2001 and comes from the *New York Times Annotated Corpus* [147]. It comprises 2 610 documents with  $V = 346\,996$  unique words. Preprocessing includes normalization of number and currency occurrences as well as removal of the most frequent stop words. To counter issues of the Gaussian kernel with the curse of dimensionality in too high-dimensional spaces, we use Latent Dirichlet Allocation [16] to project the document's dimensionality  $V$  down to a space of  $K = 30$  latent semantic concepts. Latent Dirichlet Allocation is a hierarchical Bayesian model that describes documents as multinomial distributions over  $K$  topics and each topic as a multinomial distribution over the  $V$  unique terms. A separate Dirichlet *prior* is placed on both distributions. More precisely, we used an online version [78] of the original algorithm to successively fit each day of news data to the model and re-inferred the document distributions of each day after learning the latent topics. This process results in  $K$ -dimensional document vectors that are assumed to accumulate in the vector space model if they share one or more semantic concepts.

Using the relative neighborhood graph to approximate the vertex neighborhood, we create the time-varying density function with  $\sigma_{global} = 0.5$  as described in Chapter 8.1.1. The global grid consists of 4 862 vertices (including 2 358 upsampled positions) and 5 696 edges. The merge tree transformation times range from 0.2 to 1.3 seconds on our machine with two 2.6 GHz quad-core processors. Because the transformations run concurrently, the total time to analyze the data and to construct the visualization takes around 2.5 seconds. The topologically most complex trans-

formation (from the first to the second time step) requires to process up to 650 000 superarc collapses, of which around 490 000 represent (optimized) transpositions of two regular nodes that do not change the number of superarcs, i.e. the density function's topology as described by the merge tree.

Figure 8.1 shows an image of the time-varying merge tree applied to the 30-D density function. The ten landscape profiles reflect the thematic composition of the documents from September 7<sup>th</sup> until September 16<sup>th</sup>. Hills represent dense regions in the topic space, i.e. accumulations of documents that share thematic concepts. Based on a hill's shape, the analyst can approximate the extent of the corresponding high-dimensional cluster: if documents are very similar, they accumulate to compact groups of high density. Furthermore, the ratio between a cluster's size and its density can approximate the cluster's spread and thus its topical preciseness or generality. These properties are conveyed by the hills' width and height values. Because at each day only those documents with an earlier time stamp contribute to this time step's density function, the width of the profiles increases over time to reflect increasing cluster sizes. The main insight taken from this example is that until the *9/11-attacks* appeared in the news on September 12<sup>th</sup>, the topical composition was stable and clusters primarily grew in terms of their size and density. This is because newly arriving documents are mostly added to already existing topics. The sudden structural shifts in the profiles starting on September 12<sup>th</sup> back up what would be expected to happen in the *topic space*: while some topics unrelated to the attacks stop growing, a few other topics dominate. While some topics join into new hierarchies because their context suddenly changes, also new topics appear and become very prominent over time. An example is indicated by the hill marked with a star in the last landscape profile in Figure 8.1. This hill initially appears on September 12<sup>th</sup> on another hill and then grows continually, while its parent hill stops growing. A further inspection of the corresponding documents reveals that the documents belonging to the hill until September 12<sup>th</sup> are about *obituary*. While this topic was also relevant for the daily newspaper before the attacks, starting from September 12<sup>th</sup>, obituary-related documents also start to contain words closely related to the attacks or companies and people that resided in the towers. This extension in word usage causes new document vectors to spread into a subspace of the obituary documents. However, because they still share dimensions (words) with other vectors of that topic, they accumulate to a subcluster of the *obituary* topic, represented by a sub-hill on the parent topic's hill in the landscape profile.

The semantic insights provided by the vector space model are rather limited. For features that are more complex than the thematic composition, the application of

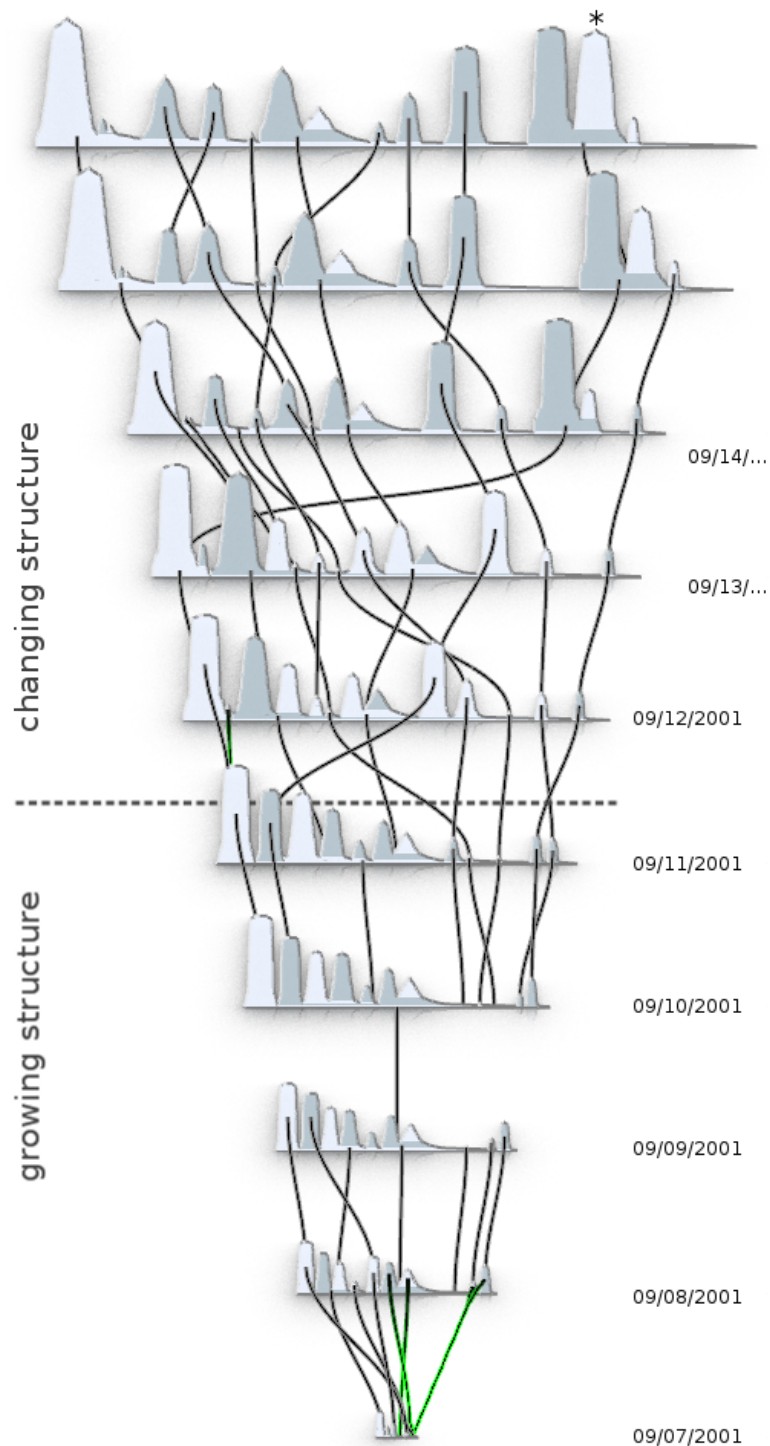


Figure 8.1: Ten days of New York Times newspaper data from September 2001 showing the changes in the thematic composition for the days before and after the 9/11-attacks: Hills on a profile describe clusters of documents that share one or more concepts and thus accumulate in the topic space. The increasing width and height of the profiles reflects that data size, and thus individual densities, increases over time because documents are added on a daily base. While topics are stable and primarily grow until September 11<sup>th</sup>, from September 12<sup>th</sup> structure apparently changes.



competing topic models and adapted visualizations, such as TextFlow [36], certainly permits deeper insights regarding language processing aspects. Nevertheless, since visual analysis of document data represented in vectorized format indeed is a typical use case, the topological approach has distinct advantages over competing techniques, like projections: First, the occlusion-free clustering depiction of the landscape profiles supports to illustrate temporal feature evolution clearly; a task difficult to perform with occlusion-prone scatterplots, especially for time-dependent point data. Second, the topological view provides an implicit and meaningful topic alignment by respecting their hierarchy and by placing related topics next to each other. Arranging the topics is also a problem for river-like visualizations like ThemeRiver [73] and TextFlow [36]; which furthermore do not reflect hierarchy and can display only one topic property. Finally, the topology-based definition, preservation and illustration of cluster quality measures allows the analyst to inspect changing topics at higher detail and to simplify the structural view based on these properties.

As for the visualization, the analyst can filter the evolution of single topics by interactive selection of individual hills. To focus on a particular topic, the granularity can also be adjusted by using different thresholds for the filter radius and to simplify the structural view on the data. Specifically for document analysis, a sophisticated annotation could be implemented to provide additional information. For example, thematic descriptors could be placed above the hills—which, however, is not trivial to implement in the above example because topics are multinomial distributions over the vocabulary and because documents can also belong to multiple thematic concepts. Furthermore, the visual design is still flexible to discard some topological information. Although the properties defined by the time-varying merge tree aim to make the topological analysis generally applicable, specifically for document data, the visualization could be adapted by discarding some properties. For example, by focusing only on topic hierarchy and on the number of documents per topic, hills in the profiles can be reduced to showing only the baselines. That is, by discarding the information about a hill’s shape and its height, it is possible to reduce the dimensionality of the visualization to two dimensions. This would also reduce clutter caused by the visual links between the profiles and would allow for another link representation. For example, colored *rivers* per topic could be used for a flow-like metaphor to indicate the evolution of topics and their size over time, like in TextFlow [36]. However, while the merge tree hierarchy could define a meaningful topic layout in this case, the problem of crossing rivers would still remain. In fact, because rivers would have a larger breadth than simple lines, the amount of occlusion

and clutter caused by differently colored rivers of varying width could also complicate tracing them visually.

## Reuters Data Set

The second example data consists of categorized documents from the Reuters-21 578 collection [7] that appeared on the Reuters newswire in 1987. This time-varying Reuters data set is similar to the static Reuters data set (cf. Appendix A.7) in that it originates from the same source of documents but additionally considers the articles' time of appearance to group them into timeslices. To demonstrate accurate feature tracking in high-dimensional spaces, we extract documents for ten economy-related categories (“acq”, “earn”, “money-fx”, “grain”, “trade”, “interest”, “ship”, “wheat”, “corn”, “crude”). The documents are preprocessed by removing stop words and using the *tf-idf* [143, 144] document-term weighting to define word importances for each dimension in the vector space model. To avoid problems with the curse of dimensionality in spaces with ten-thousands of dimensions, we use Linear Discriminant Analysis (cf. Chapter 2.2.1 on page 15), a supervised projection that uses given classification information per document to minimize information-loss, and project the data to a ( $numClasses - 1 = 9$ )-dimensional space. The data set finally consists of 5 309 documents, manually divided into eight time slices, 20 days each, from 02/21/1987 to 10/19/1987.

Figure 8.2 shows a visualization of the time-varying merge tree of the Reuters data set. The global filter radius is  $\sigma_{global} = 0.3$  and the total time to obtain the image is around one second. The most complex transformation required processing approximately around 33 000 superarc collapses. Because classification information is available for the documents, we can augment the data points as colored histograms on the hills to indicate how documents of different classes are distributed across the clusters. As indicated by the color distribution of the histograms, dense regions primarily match to documents of a single class, while some documents of related classes are in a subcluster relationship. The valleys between the hills of unrelated topics are typically low because they reflect the density between the corresponding clusters. Likewise, for related topics, like “grain”, “wheat”, and “corn”, the subspace spanned by the vocabulary used in these documents is more scattered and also contains less specific documents that lead to higher saddle densities. The rectangular shape of the hills suggests that clusters are compact in the sense that document densities are close to the cluster's maximum density. This is also indicated by the placement of the histograms close to the hilltops and reflects that these documents are fairly similar in terms of their content. The landscape in Figure 8.2 was constructed

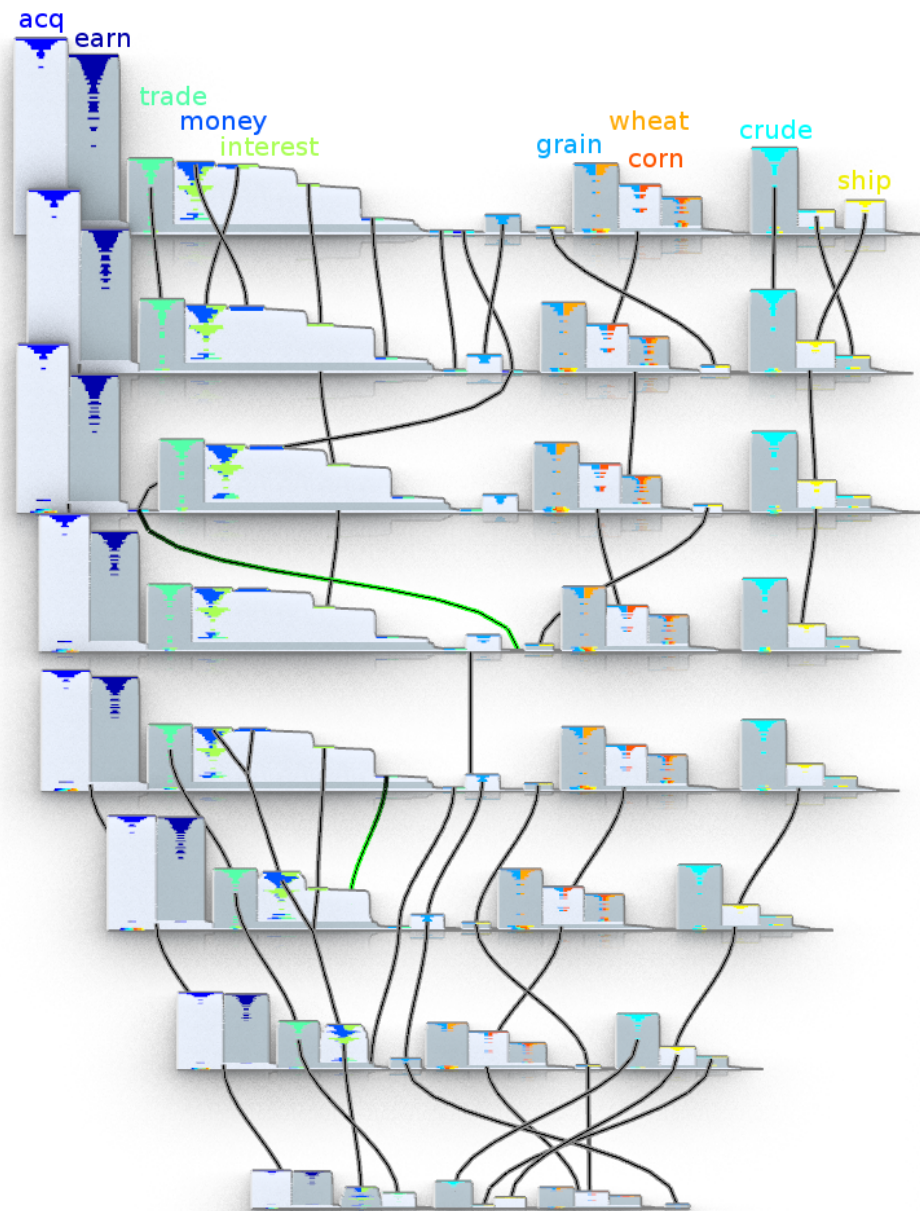
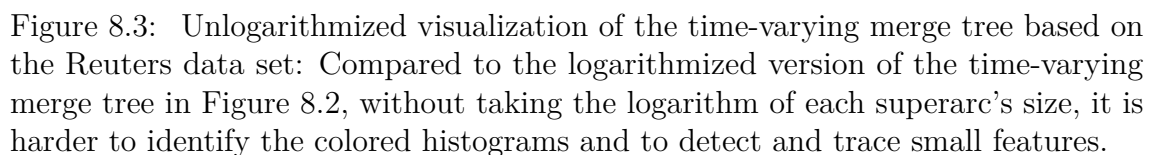


Figure 8.2: Logarithmized visualization of the time-varying merge tree based on the Reuters data set: The hills and the visual links between the profiles indicate how document groups evolve over time. The colored histograms indicate that documents primarily accumulate by class and form subclusters for related categories. The document collection is *stable* in that, over time, document groups primarily grow in size and density, while some subclusters break apart and grow individually.

by taking the logarithm of each superarc’s size to narrow the hills and the slopes. This can be useful to obtain a more convenient overview if the data contains many features of very different size. Taking the logarithm then accentuates small features and attenuates large ones. In contrast, Figure 8.3 shows the same visualization without taking the logarithm. Although the correct depiction of feature size facilitates more



accurate feature comparison, it also becomes harder to identify the data glyphs and to detect and trace small features visually. Regarding the temporal evolution of the features, dense regions are primarily stable, but grow in size and persistence. Furthermore, some clusters split occasionally into more subclusters that become increasingly prominent. This reflects that, over time, newly arriving documents are primarily added to the already existing topics, while some documents slightly deviate from the main topics and create sub-topics on their own.

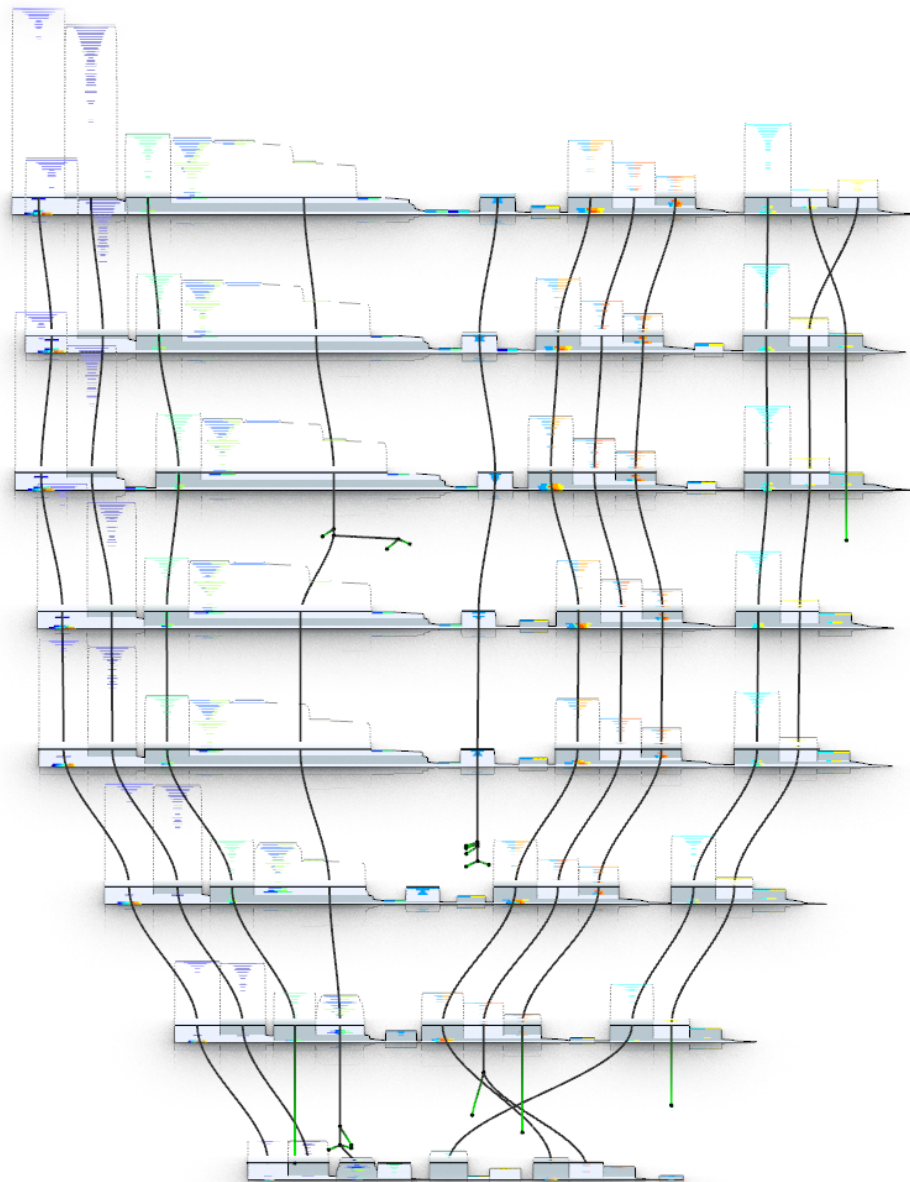


Figure 8.4: Tracking graph of a superlevel set based on the Reuters data set: Analyzing superlevel set components of the density function in the topic space is useful to analyze those documents groups that are similar enough to produce a predefined thematic relevance. The tracking graph then reveals at which times document groups start/stop containing this threshold. The visual context of the complete function, furthermore, helps to identify interesting thresholds.

Tracking superlevel sets in the topic space helps the analyst to track features regarding a predefined density threshold. In terms of the vector space model this means to analyze those topics and sub-topics whose documents are similar enough to produce a minimum thematic relevance. This happens when similar documents are close to each other in the topic space and produce accumulations of high density.

Then, a region's density can be thought of as topic's distinctness or significance and analyzing superlevel sets of the function segments the thematic composition relative to the selected relevance threshold. This can be used to identify and track those topics that are not necessarily persistent, but whose documents are still sufficiently similar to satisfy a defined minimum importance. Figure 8.4 shows a tracking graph of the function and reveals how the document groups evolve in terms of the pre-selected minimum group coherence. The tracking graph and the individual superlevel set components disclose that not all features exhibit this minimum coherence in the first time steps. However, over time, superlevel set components grow in size and more and more topics also pass this threshold. The visual context of the complete function still reveals those features that are not captured by the adjusted threshold. Based on this information, the analyst can change the threshold to focus on these features. The context provided by the silhouettes also summarizes how the topics are distributed for other thresholds. This is useful to find out the thresholds for only the most prominent features. Finding these interesting and relevant thresholds in the first place is difficult for other techniques that only track and show the temporal evolution of single isolevel sets without additional knowledge about the context of the complete function.

## 8.2 Conclusion and Discussion

We showed the utility of the time-varying merge tree using two real-world text data sets and tracked topics in document collections that change with time. This required the transformation of high-dimensional point clouds into a time-varying scalar function. Although the time-dependent merge tree algorithm can work with streaming data on-line, this is not the case for the preprocessing necessary to turn point clouds into scalar fields. Here, we currently unify the neighborhood graphs of all time steps to have a consistent domain. This is necessary because the number of nodes in the merge tree must not change during the transformation. It would be attractive to remove this restriction of the merge tree algorithm to constant domains so that, for the application of temporal cluster analysis, we can also have a streaming solution. However, this would also require an optimal solution to update the neighborhood graph for new points efficiently; which is still an open problem.

Tracking topics in document collections and visualizing their evolution over time is a reasonable application to demonstrate the utility of the time-varying merge tree for high-dimensional, temporal clusters analysis. However, even though the vector space model has a long standing in language processing, the semantic insights of

---

this document representation are rather limited. This implies that the semantic knowledge taken from the topological analysis are limited as well, which is also due to the fact that this model was chosen by availability rather than being state of the art. Therefore, instead of extracting new knowledge by analyzing the information space topologically, our major goal was to demonstrate that the time-varying merge tree can indeed identify, track, and finally illustrate otherwise hard to detect and visualize high-dimensional scalar field structure. From these considerations it follows that, even though the landscape metaphor can depict topics as hills, domain experts may require additional instruction and guidance in steering and reading the output of the time-varying merge tree. Still, since providing a thematic overview is a common vantage point for detailed document analysis, the topological analysis in combination with the vector space model can still be helpful to reveal the thematic composition over time, and to analyze other semantic aspects with competing topic models on demand in a linked view.





## Chapter 9

# Thesis Conclusion

The contribution of this thesis is a novel topology-based approach for visual analysis of high-dimensional point clouds for both time-invariant and time-varying data. By abstracting the input data by a simpler representation, namely the point cloud's density function, the focus is shifted away from the points themselves to a more structure-based perspective that is easier to determine and describe. Furthermore, the topological description of this surrogate permits a more accurate and occlusion-free depiction of global clustering structure; compared to competing standard methods like, e.g., lossy projections or complex and sometimes difficult to interpret parallel coordinate plots. These techniques are prone to visual complexity and structural occlusion in the sense that actually separated features often overlap in the final visualization. This easily leads to wrong and delusive insights about the data on the user's side. The introduced topology-based approach makes effective and elegant contributions for showing the structure of high-dimensional point data from the *information visualization* and the *visual analytics* point of view. As for information visualization, the global overview as a landscape extends the set of available techniques in that it shows the clustering as it occurs in the original domain. This improves the quality and the reliability of the visualization. Regarding visual analytics, which often focuses on interactive and incremental analysis loops performed manually by the analyst, we also extended the topology-based global overview to an interactive analysis tool. This framework helps users in finding appropriate parameters and also allows for local feature inspection on demand in linked views. These advancements expand the set of available tools for visual and interactive exploration of high-dimensional clustering structure. We also showed how the presented approach can be combined with supervised projections to simplify the data's complexity and how this combination can improve competing two-stage projections, such as the one proposed by Choo et al. [30]. While this thesis did not

focus much on the application-dependent “insights” described by the clustering, its contribution is rather the potential to make this structure visible in the first place; making the approach useful for many applications. With document analysis, we gave one example application for high-dimensional clusterings. However, the vector space model was only picked for availability rather than being state of the art, as it is not necessarily the most suitable representation for text data when it comes to explanatory power and interesting insights (content-wise). Beyond this specific application, the topological approach makes valuable contributions to other fields, like visual analysis of high-dimensional point data, visualizing high-dimensional scalar field topology, topology-based visualization metaphors, temporal clustering of high-dimensional point clouds, and studying the changing topology of time-varying, high-dimensional scalar fields. In general, the presented solutions are applicable when analysts desire the clustering structure of high-dimensional point clouds or the topology of high-dimensional scalar fields, as defined by the merge tree.

Concerning scalability issues, the topological approach admittedly does not scale well for “big data”, i.e. millions of points in thousands of dimensions. This limitation primarily results from the necessity to approximate high-dimensional neighborhood relationships with rather expensive neighborhood graphs, and from relying on similarity measures that have intrinsic problems with the data’s dimensionality (i.e., the “curse of dimensionality”). However, for the supported data complexity, which is specified to approximately 100 000 points in around 100 dimensions (including optimizations), the main advantage of the topological approach over competing techniques is that for these data sizes, the final depiction of clustering structure is more accurate and, most importantly, independent from the data’s size and dimensionality. Nevertheless, there is still huge potential for improvements. This primarily includes the usage of more sophisticated sampling strategies to reduce the input data, faster neighborhood graph construction, using sparser graphs (like the unconnected nearest neighbor graph), applying other filter kernels to control the accuracy of the density function, utilizing increased parallelism (maybe on the GPU to exploit extreme numbers of processors), or supporting other topology-based or even geometric cluster properties, like approximated shape.

The extension of the topological approach to time-varying data also makes valuable contributions for the analysis of varying high-dimensional scalar functions without being dependent on the data’s dimensionality itself. Related work about time-varying scalar field topology is also advanced by extending it to higher dimensions and by tracking and visualizing both features defined by superlevel set components and those defined by complete superarcs. Previous work only considered isolevel sets in low

dimensional spaces. Concerning the visualization, the depiction of changing structure as concatenated 2-D landscape profiles admittedly requires some familiarization with the topological concepts involved. However, the design is not yet optimized for a specific application, but instead aims to convey the complete information provided by the time-varying merge tree. The proposed prototype is still flexible enough to be adjusted for a particular application, e.g., by discarding some properties to obtain a simpler or more instinctive depiction. There is also room for improvements regarding the construction and visualization of time-varying merge trees. This primarily concerns the restriction to constant domains during the transformation, but also the visual metaphor used to depict changing structure. While solving the latter problem likely requires more research on the information visualization side, i.e. by finding novel techniques to visualize dynamic trees with multiple properties per edge and node, solving the restriction to constant domains requires to find new strategies to update the high-dimensional neighborhood graph and the density function efficiently for each additionally added data point.

Despite these limitations and the outlined future work (also see the conclusions of the individual chapters), the research results presented in this thesis were very much appreciated by the scientific community. The underlying publications were cited frequently by other leading groups and were presented in the most relevant journals and at reputable conferences of the target community. Earlier work of the project also stimulated other scientists to pick up this problem and to conduct successful research based on the key ideas of our initial results; like approximating high-dimensional neighborhood efficiently with neighborhood graphs as a viable alternative to impractical regular grids.



# Appendix A

## Data Sets

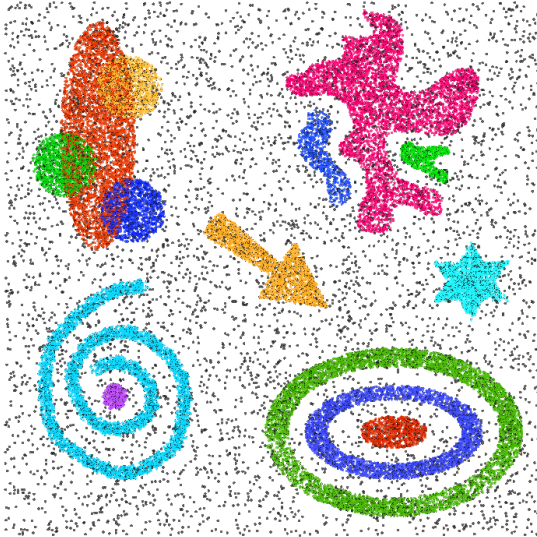
The topological visual analysis presented in this thesis is demonstrated and validated based on various example data sets of different size and dimensionality. The used data sets originate from different application domains and their thematic diversity aims to expose the algorithm’s utility for high-dimensional data in general. The data sets were selected primarily based on their characteristics in order to demonstrate the advantages of the topology-based visualization over competing techniques, to discuss runtime and memory consumption issues, and to reveal limitations regarding the input data’s size and complexity. While the real-world data sets used in this thesis were gathered together from the relevant online repositories, we also generated some data sets manually for demonstration purposes. In case of artificial 2-D data, we used a standard graphics editor to directly draw the desired point clouds as an image. For high-dimensional data sets, one can easily make use of convenient generators [3, 21, 70] to create synthetic clusterings. This appendix introduces all data sets used in this thesis and provides further information about their semantics, origin, and properties. Table A.1 summarizes their relevant characteristics.

Table A.1: Summary of the data sets and their quantitative properties.

name	points	dimensions	classes
artificial 2-D data set	31 834	2	15
artificial 100-D data set	127 995	100	21
image segmentation data set	2 310	19	7
Isolet speech recognition data set	7 797	25	26
Italian olive oils data set	572	8	9
Iris plants data set	150	4	3
Reuters newswire data set	800	9	10
Medline medical text data set	1 250	22 095	5

## A.1 Artificial 2-D Data Set

Figure A.1: Artificial 2-D data set.



The purpose of the artificial 2-D data set is to describe a clustering scenario with as many feature properties as possible. The data set was generated manually with a graphics editor and the data points are stored in an image as non-white pixels at the location of their corresponding pixel positions. Pixel colors indicate the cluster affiliation of the respective data points. The data set consists of 31 834 points and contains several clusters of different extent, number of points, compactness and shape. While

some of the clusters are partially intertwined, others contain each other. Furthermore, equally distributed black pixels represent noise in the clustering. Although black pixels inside the clusters actually contradict the definition of noise, their distribution is primarily for demonstration purposes. The point cloud has an extent of 800x800 pixels; the smallest possible distance between two points is one pixel. The major advantage of the artificial 2-D example data set is that we can illustrate intermediate results to improve the reader's understanding of the involved topological concepts. This data set is used in Chapter 3.5.1 (on page 53), in Chapter 4.5.1 (on page 86), and in Chapter 5.5.1 (on page 129).

## A.2 Artificial 100-D Data Set

The artificial 100-D data set was created with a clustering generator described by Handl et al. [70]. The generator uniformly arranges multinomial distributions in an arbitrary dimensional space, allowing different standard deviations for each dimension. Each data point is provided with meta-information about the name and the color of its corresponding cluster. The point cloud contains 20 clusters of varying size (number of points) and extent; having a total number of 85 330 data points. Furthermore, we added an extra 50% amount of (red) noise points; leading to a final data size of 127 995 points in 100 dimensions. The purpose of this artificial data set is to demonstrate both the strengths and limitations of the topological approach and to compare our topology-based visualization with competing techniques for high-dimensional point data. Figure A.2 shows the principal component analysis

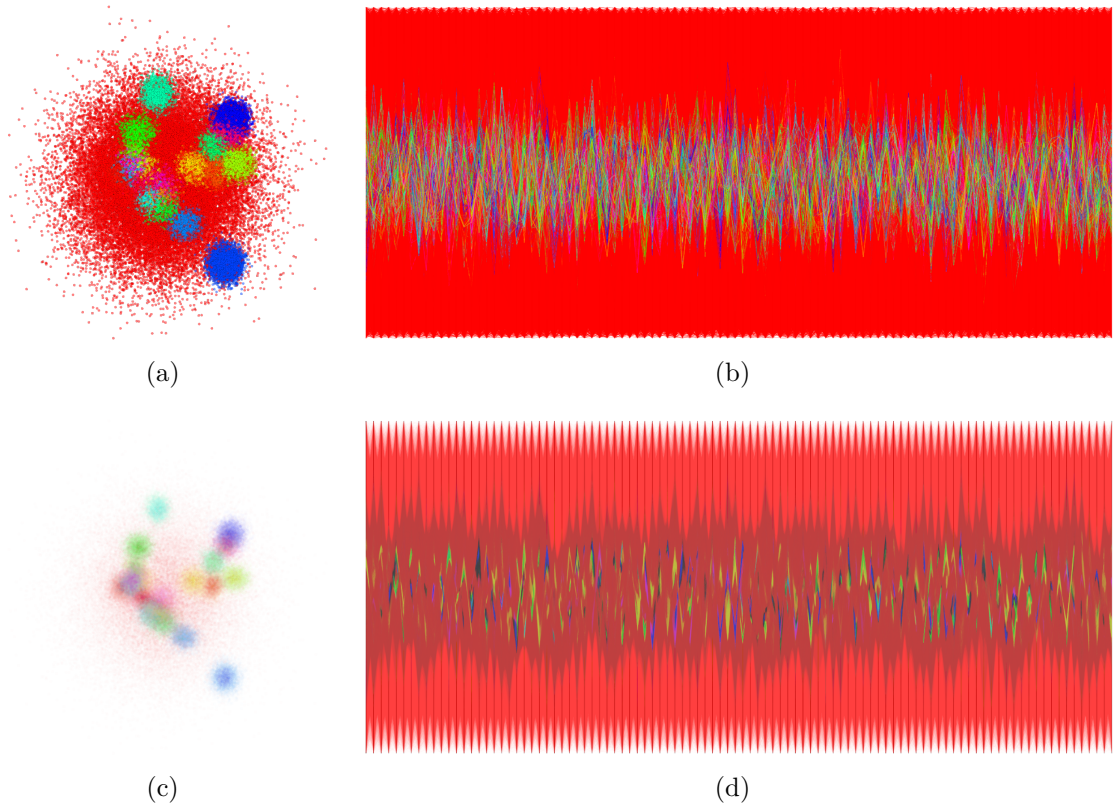























Figure A.2: Artificial 100-D data set: (a) PCA projection with a projection error of approximately 95% and (b) a parallel coordinate plot showing the data points with their corresponding colors. In both visualizations, clusters are difficult to identify and compare. (c)-(d) Rendering geometric primitives (i.e., points and lines) with lower opacity per pixel accentuates the dense regions. This strategy slightly increases the number of visible point accumulations but still shows only around half of the truly existing clusters. The clustering structure conveyed by both visualizations is still misleading.

Table A.2: Properties and meta-information of the artificial 100-D data set.

	number of points							
20 classes	 class1	6 421	 class2	5 704	 class3	4 159	 class4	3 208
	 class5	4 431	 class6	3 807	 class7	3 585	 class8	3 919
	 class9	5 668	 class10	2 832	 class11	5 139	 class12	2 761
	 class13	4 681	 class14	3 129	 class15	4 546	 class16	4 070
	 class17	5 225	 class18	3 873	 class19	4 051	 class20	4 121
	total: 85 330							
noise	 42 665							
total: 127 995								

(PCA) and a parallel coordinate plot (PCP) using different transparency thresholds.

It is obvious that these standard techniques have severe issues with illustrating the clustering structure of this data set appropriately. Table A.2 provides meta-information about the individual clusters and the composition of the complete data set. This data set is used in Chapter 3.5.2 (on page 61), in Chapter 4.5.1 (on page 94), and in Chapter 4.5.2 (on page 98).

### A.3 19-D Image Segmentation Data Set

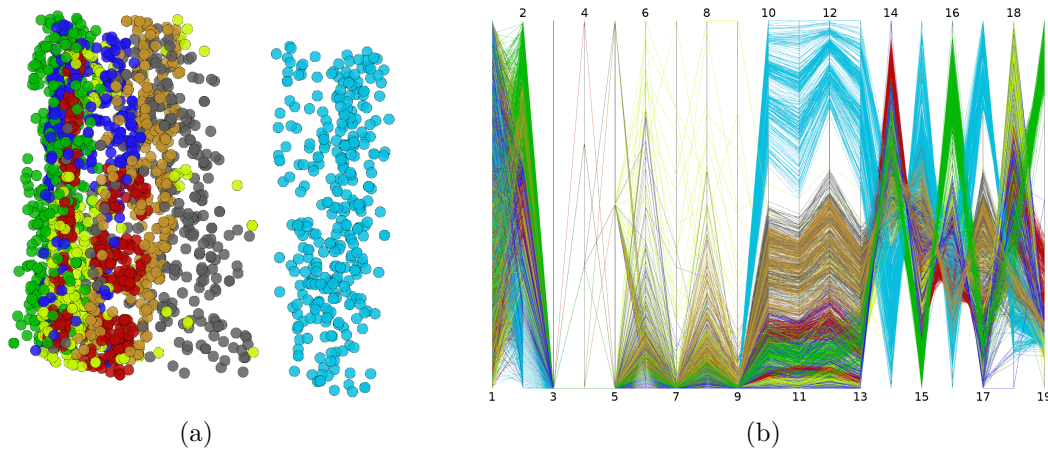


Figure A.3: 19-D image segmentation data set: (a) PCA projection with a projection error of approximately 35.76%, i.e., preserving only 64.24% of the point cloud’s variance in the original domain. (b) A parallel coordinate plot of the same data set. The axes are sorted according to the attribute order as defined by the data set.

In the image segmentation data set from the UCI machine learning repository [7], each instance represents a 3x3 pixel region drawn randomly from a database of seven outdoor images. The images were handsegmented to create a classification for every region using the following classes: “brickface”, “sky”, “foliage”, “cement”, “window”, “path”, “grass”. Each instance has 19 attributes that describe various features like the position of a region’s center pixel, line densities, edge detection, contrast, and color values. There are 330 instances per class, which makes a total of 2310 instances. Figure A.3 shows a PCA projection and a parallel coordinate plot for this data set. While both techniques reveal a few clusters primarily as accumulations of different colors, the exact clustering remains largely unknown. This data set is used in Chapter 3.5.2 (on page 60), in Chapter 4.5.1 (on page 92), in Chapter 5.2 (on page 110), in Chapter 5.4.2 (on page 126), and in Chapter 5.5.1 (on page 132).



## A.4 25-D Isolet Data Set

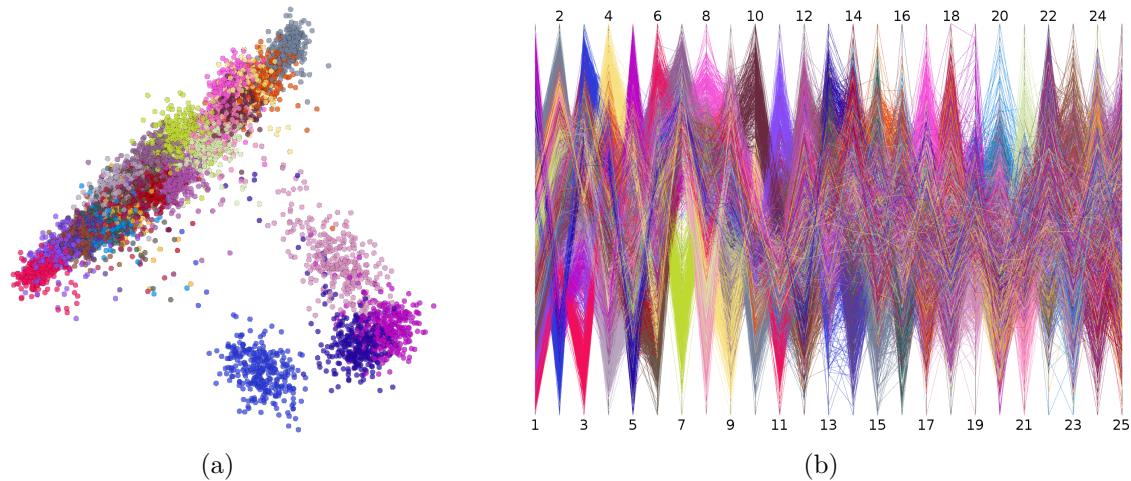


Figure A.4: 25-D Isolet data set: (a) The Rank-2 LDA projection with a projection error of approximately 58%. Although the color distribution suggests multiple clusters, the spatial point distribution can only reflect four or five clusters. (b) The parallel coordinate plot suggests more clusters than the projection, but individual clusters and their properties are still difficult to identify and compare.

In the *Isolated Letter Speech Recognition* (Isolet) data set from the UCI machine learning repository [7], 150 subjects spoke the name of each letter of the alphabet twice; thus producing 52 samples for each speaker. From the 7 797 recordings—three examples are missing due to difficulties in recording—a total of 617 attributes were extracted from the pronunciation of each letter. These attributes include spectral coefficients, contour features, sonorant features, pre-sonorant features, and post-sonorant features. The collection of all attributes is described in more detail in the paper by Cole and Fanty [54]. All attributes are continuous, real-valued attributes scaled into the range  $[-1.0, 1.0]$ . The exact order of appearance of the features is not known. One additional attribute indicates the class, i.e. the spoken letter, of each sample record. However, this attribute is not used in the vector representation of the samples.

To cope with the curse of dimensionality (cf. Chapter 2.1.1) in high-dimensional spaces, but also to enable visual comparison with competing techniques, we do not use the Isolet data set in its original 618-dimensional space. Instead, we project it into a lower-dimensional space using linear discriminant analysis (LDA), a supervised projection that uses classification information to minimize information-loss in an optimal  $(numClasses - 1)$ -dimensional space (cf. Chapter 2.2.1). LDA strives to preserve clustering structure by minimizing intra-class distances and maximizing

inter-class distances in the reduced dimensional space. Having a total of 26 classes, the projected Isolet data set finally consists of 7 797 points in a 25-dimensional space. Figure A.4 shows the Rank-2 LDA projection (cf. Chapter 2.2.1) and a parallel coordinate plot of the Isolet data. This data set is used in Chapter 2.2.3 (on page 19), in Chapter 3.5.2 (on page 60), and in Chapter 5.5.1 (on page 133).

## A.5 8-D Italian Olive Oils Data Set

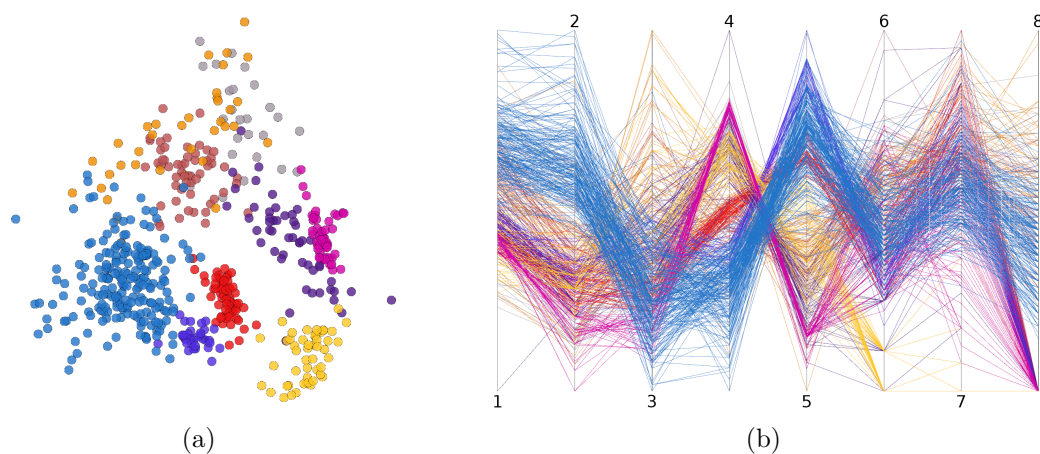


Figure A.5: 8-D Italian olive oils data set: (a) In the PCA projection (with a projection error of approximately 29.59%) and in (b) the parallel coordinate plot, clusters are indicated primarily based on equally colored accumulations whose hierarchy and spatial relation is otherwise hard to identify due to the projection error and occlusion artifacts. Individual cluster properties can only be estimated because it is infeasible to identify and count all data items manually.

In the Italian olive oils data set, 572 olive oil samples from different growing areas in Italy were analyzed chemically. There are nine collection areas from three primary regions: four from Southern Italy (North and South Apulia, Calabria, and Sicily), two from Sardinia (Inland and Coastal), and three from Northern Italy (Umbria, East Liguria, and West Liguria). The data consists of the percentage composition of eight fatty acids (palmitic, palmitoleic, stearic, oleic, linoleic, linolenic, arachidic, eicosenoic) found in the lipid fraction of the olive oils. An analysis of this data is given by Forina et al. [57]. A statistical investigation of the data in the case study “STAT 503X Case Study 2: Italian Olive Oils”<sup>1</sup> reveals that southern oils have much higher eicosenoic acid on average eicosenoic and slightly higher palmitic and palmitoleic acid content. The north and sardinian oils have some difference in the

<sup>1</sup><http://www.public.iastate.edu/~dicook/stat503/05/cs-olive.pdf>

average oleic, linoleic, and arachidic acids. Among the southern oils, there is some difference in most of the averages. Northern oils have some difference in most of the averages. Figure A.5 shows the PCA projection and a parallel coordinate plot for this data set. The images reveal some clusters but hide the correct relationship between them. We use this data set to demonstrate that our topological analysis is able to detect and confirm that the olive oils, or more precisely the composition of their fatty acids, cluster based on their growing regions. This data set is used in Chapter 3.5.2 (on page 60), in Chapter 4.5.1 (on page 89), and in Chapter 4.5.2 (on page 97).

## A.6 4-D Iris Plants Data Set

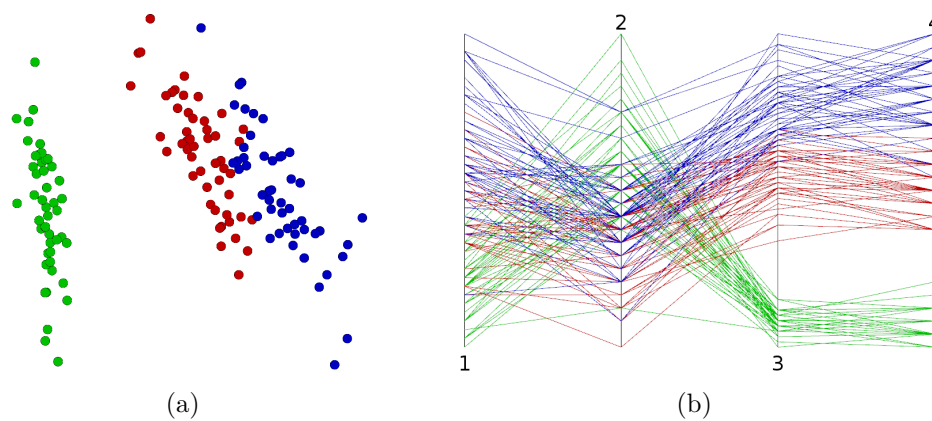


Figure A.6: 4-D Iris flowers data set: (a) The PCA projection (with a projection error of only 3%) and (b) the parallel coordinate plot reliably reflect the already known relationship between the tree clusters of the Iris flowers. Nevertheless, comparing individual cluster properties, like compactness or size, is difficult because points would need to be counted manually and because value distributions need to be evaluated on the PCP axes.

The *Iris flower-* or *Fisher's Iris* data set (named by Sir Ronald Fisher who introduced it) [7] is one of the most popular examples in pattern recognition. The data consists of 50 instances from each of three types of Iris flowers: *Iris setosa*, *Iris virginica* and *Iris versicolor*. Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. It is known that one class is separable from the other two; while the latter are not separable from each other. Having only 150 instances and only four dimensions, this is a rather small data set. However, because it is cited frequently in the literature and often used to compare competing visualizations for higher-dimensional data, we also use this data set to

demonstrate how already established knowledge can be recovered and visualized with the presented topological approach. Figure A.6 shows the PCA projection and a parallel coordinate plot. Although both visualizations reflect the described structure reasonably, it is still difficult to determine and compare individual properties like cluster size. This data set is used in Chapter 3.5.2 (on page 60) and in Chapter 4.5.1 (on page 88).

## A.7 9-D Reuters Data Set

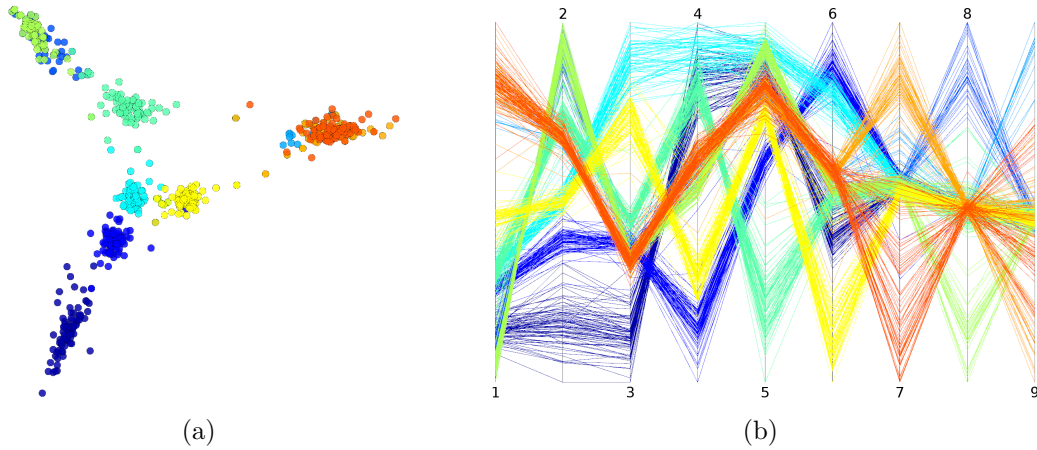


Figure A.7: 9-D Reuters data set: (a) The Rank-2 LDA projection (with a projection error of approximately 56% from the intermediate 9-D space) of a subset of the original Reuters-21 578 data set. The projection reveals a fairly good clustering, but also shows two clusters that contain points of mixed classes. (b) The parallel coordinate plot reveals that some of the mixed accumulations, like the one containing the orange points, are actually separated. This is not visible in the projection. Because the correct relation between the clusters is hidden in both visualizations, it is difficult to identify and compare individual clusters.

The Reuters-21 578 Text Categorization Collection data set is available in the UCI machine learning repository [7]. The documents appeared on the Reuters newswire in 1987 and were assembled and indexed with categories manually. For the sake of comparison with other techniques, we use a reduced version of this collection that was preprocessed and kindly provided by Choo et al. [30], who used this data to demonstrate their introduced supervised projection techniques for high-dimensional, clustered point clouds. The authors extracted 800 documents for ten economy-related categories (“acq”, “earn”, “money-fx”, “grain”, “trade”, “interest”, “ship”, “wheat”, “corn”, and “crude”). After applying language processing techniques to turn the raw text data into vector space format, they end up with a point cloud consisting of

800 points in a 3 907-dimensional space. In this thesis, we use a 9-D LDA-projected version of this data set to demonstrate the advantages of our topological approach over the combination of supervised projection techniques like, e.g., the Rank-2 LDA to project the data down to 2-D (cf. Chapter 2.2.1). The data finally consists of 800 points in a 9-dimensional space and each point is categorized with one of the classes mentioned above. Figure A.7 shows the Rank-2 LDA projection and a parallel coordinate plot. This data set is used in Chapter 2.2.3 (on page 20), in Chapter 3.5.2 (on page 60), in Chapter 4.2 (on page 72), in Chapter 4.3 (on page 78), in Chapter 4.5.2 (on page 97), and in Chapter 5.5.2 (on page 137).

## A.8 4-D Medline Data Set

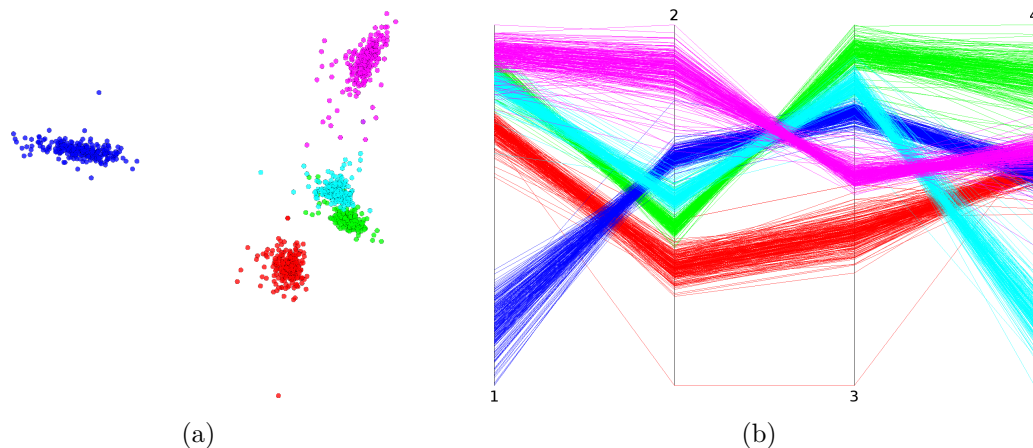


Figure A.8: 4-D Medline data set: (a) PCA projection with a projection error of approximately 38% from the intermediate 4-D space (that results from the LDA projection from the original space). (b) A parallel coordinate plot showing in which dimensions the clusters differ. Although the data set is quite small and simple, the 2-D projection still only reveals four of the five existing clusters.

The Medline data set is a text collection related to medical science from the National Institutes of Health. The data was generated manually and kindly provided by Choo et al. [30], who selected 500 documents for five topics (“heart attack”, “colon cancer”, “diabetes”, “oral cancer”, and “tooth decay”). After performing natural language processing techniques to turn the text data into vector format, the data set finally contains 1 250 points in a 22 092-dimensional space. In this thesis, we use two versions of this data set. The original data to demonstrate the effects of the curse of dimensionality in very high-dimensional spaces, and a Rank-2 LDA projected version to evaluate runtimes issues of the presented topological analysis. This data set is used in Chapter 2.1.1 (on page 14) and in Chapter 3.5.2 (on page 60).



# List of Figures

2.1	Turning text data into point data . . . . .	9
2.2	Pair-wise distances in the Medline data set . . . . .	15
2.3	Transforming a 2-D scatterplot into parallel coordinates . . . . .	18
2.4	Parallel coordinate plot of the 25-D Isolet data set . . . . .	19
2.5	Projection of the Reuters data set . . . . .	21
2.6	Topology-based visual analysis approach . . . . .	23
3.1	Neighborhood graphs of a point set . . . . .	31
3.2	Algorithmic pipeline of the topological analysis . . . . .	34
3.3	Upsampling and efficient midpoint calculation for each graph edge . . . . .	36
3.4	Topological simplification based on a 2-D point set . . . . .	39
3.5	Different neighborhood graphs of an artificial 2-D point set . . . . .	43
3.6	Persistence diagrams for different neighborhood graphs and different sampling strategies . . . . .	44
3.7	Comparison of different sampling strategies . . . . .	47
3.8	Artificial 2-D data set imagined as a height field . . . . .	55
3.9	Critical points and simplified topology of the artificial 2-D data set . . . . .	56
3.10	The importance of upsampling . . . . .	57
4.1	Construction of the 3-D topological landscape . . . . .	70
4.2	Modified metric-based distortion . . . . .	73
4.3	3-D topological landscape of the 9-D Reuters data set . . . . .	74
4.4	Data point representation in the 3-D landscape . . . . .	75
4.5	Hill-based labeling in the landscape of the 9-D Reuters data set . . . . .	76
4.6	Glyph-based labeling in the landscape of the 9-D Reuters data set . . . . .	77
4.7	2-D topological atoll of the 9-D Reuters data set . . . . .	78
4.8	Transforming a merge tree's 3-D topological landscape into a 2-D landscape profile . . . . .	80
4.9	2-D topological landscape profile of an artificial 2-D data set . . . . .	81
4.10	2-D topological landscape profile of the 9-D Reuters data set . . . . .	83



4.11	Construction scheme of the 2-D topological landscape profile . . . . .	85
4.12	Mapping between clusters and hills in the 2-D landscape profile . . . .	87
4.13	Visualizing the 4-D Iris data set . . . . .	89
4.14	Different visualizations of the 8-D Italian olive oils data set . . . . .	90
4.15	Different visualizations of the 19-D image segmentation data set . . . .	92
4.16	Visualizing the artificial 100-D data set . . . . .	94
4.17	Artificial 100-D data set for different parameter settings . . . . .	96
4.18	9-D Reuters data set without classification information . . . . .	98
4.19	8-D Italian olive oils data set without classification information . . . .	99
4.20	Standard techniques for the artificial 100-D data set without classification information . . . . .	99
4.21	Topological landscapes for the artificial 100-D data set without classification information . . . . .	100
4.22	Thematic composition of a document collection as a 2-D topological atoll based on the New York Times data set . . . . .	102
4.23	Thematic composition of patents as a 2-D topological atoll . . . . .	104
4.24	Classification of unclassified documents based on IPC patent data . . .	105
5.1	Examining the 3-D topological landscape for a 19-D data set . . . . .	112
5.2	Examining the persistence diagram for a 19-D data set . . . . .	113
5.3	Interactive parameter widgets . . . . .	116
5.4	Interactive simplification of the 19-D image segmentation data set . .	117
5.5	Local feature analysis by linking subsets to a PCP-view . . . . .	121
5.6	Local feature analysis by linking subsets to a PCA-view . . . . .	123
5.7	Overview of the analysis tool implemented in OpenWalnut . . . . .	125
5.8	Different states of the controller widgets . . . . .	127
5.9	Selections in the artificial 2-D data set . . . . .	130
5.10	Density functions of individual clusters of the artificial 2-D data set .	131
5.11	Analyzing different accumulations of the same class . . . . .	132
5.12	Visualizing the 25-D Isolet data set . . . . .	133
5.13	Parameter settings for the 25-D Isolet data set . . . . .	134
5.14	Local analysis of the 25-D Isolet data set . . . . .	136
5.15	Accentuating features in standard visualizations without classification information . . . . .	138
6.1	Transposition of a single merge tree arc . . . . .	150
6.2	Two examples of a collapsing superarc . . . . .	152
6.3	Case table to track features of the complete function . . . . .	157



---

6.4	Event types for tracking superlevel set components . . . . .	160
7.1	Strong variation in the topological description caused by a small variation of the underlying scalar function . . . . .	166
7.2	Exemplification of a complex topological event occurring <i>between</i> two consecutive time steps . . . . .	168
7.3	Link reduction strategies for the time-varying merge tree visualization	170
7.4	Visualization of the time-varying merge tree based on a 2-D function	172
7.5	Visualizing superlevel set tracking information . . . . .	174
8.1	Ten days of New York Times newspaper data from September 2001 showing the changes in the thematic composition . . . . .	184
8.2	Logarithmized visualization of the time-varying merge tree . . . . .	187
8.3	Unlogarithmized visualization of the time-varying merge tree . . . . .	188
8.4	Tracking graph of a superlevel set . . . . .	189
A.1	Artificial 2-D data set . . . . .	198
A.2	Artificial 100-D data set . . . . .	199
A.3	19-D image segmentation data set . . . . .	200
A.4	25-D Isolet data set . . . . .	201
A.5	8-D Italian olive oils data set . . . . .	202
A.6	4-D Iris flowers data set . . . . .	203
A.7	9-D Reuters data set . . . . .	204
A.8	4-D Medline data set . . . . .	205



# Bibliography

- [1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. **Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications**. *ACM SIGMOD Record* 27.2 (1998), pp. 94–105. ISSN: 0163-5808.
- [2] C. Ahlberg, C. Williamson, and B. Shneiderman. **Dynamic Queries for Information Exploration: An Implementation and Evaluation**. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '92. ACM, 1992, pp. 619–626.
- [3] G. Albuquerque, T. Lowe, and M. Magnor. **Synthetic Generation of High-Dimensional Datasets**. *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 2317–2324.
- [4] B. Alpern and L. Carter. **The Hyperbox**. *Proceedings of the 2nd Conference on Visualization '91*. San Diego, California: IEEE Computer Society Press, 1991, pp. 133–139. ISBN: 0-8186-2245-8.
- [5] M. R. Anderberg. **Cluster Analysis for Applications**. New York: Academic, 1973.
- [6] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. **OPTICS: Ordering Points To Identify the Clustering Structure**. *Proceedings ACM SIGMOD International Conference on Management of Data*. ACM Press, 1999, pp. 49–60. ISBN: 1-58113-084-8.
- [7] K. Bache and M. Lichman. **UCI Machine Learning Repository**. University of California, Irvine, School of Information and Computer Sciences. 2013. URL: <http://archive.ics.uci.edu/ml>.
- [8] S. Bachthaler and D. Weiskopf. **Continuous Scatterplots**. *IEEE Transactions on Visualization and Computer Graphics* 14 (2008), pp. 1428–1435. ISSN: 1077-2626.
- [9] A. Baraldi and L. Schenato. **Soft-to-hard Model Transition in Clustering: A Review**. Tech. rep. Technical Report TR-99-010, 1999.

- 
- [10] R. A. Becker and W. S. Cleveland. **Brushing Scatterplots**. *Technometrics* 29.2 (1987), pp. 127–142. ISSN: 0040-1706.
  - [11] K. Beketayev, G. H. Weber, D. Morozov, A. Abzhannov, and B. Hamann. **Geometry-preserving Topological Landscapes**. *Proceedings of the Workshop at SIGGRAPH Asia*. WASA '12. New York, NY, USA: ACM, 2012, pp. 155–160.
  - [12] R. Bellman. **Adaptive Control Processes: A Guided Tour**. Princeton University Press, 1961.
  - [13] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. **Computational Geometry: Algorithms and Applications**. 3rd ed. Springer, 2008.
  - [14] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. **When Is "Nearest Neighbor" Meaningful?** *ICDT '99: Proceeding of the 7th International Conference on Database Theory*. Vol. 1540. Lecture Notes in Computer Science. London, UK: Springer Berlin Heidelberg, 1999, pp. 217–235.
  - [15] P. Bhaniramka, R. Wenger, and R. Crawfis. **Isosurface Construction in Any Dimension Using Convex Hulls**. *IEEE Transactions on Visualization and Computer Graphics* 10.2 (2004), pp. 130–141. ISSN: 1077-2626.
  - [16] D. M. Blei, A. Y. Ng, and M. I. Jordan. **Latent Dirichlet Allocation**. *Journal of Machine Learning Research* 3 (2003), pp. 993–1022.
  - [17] H. Bock. **Probabilistic Models in Cluster Analysis**. *Computational Statistics and Data Analysis* 23.1 (1996), pp. 5–28.
  - [18] P.-T. Bremer, E. Bringa, M. Duchaineau, A. Gyulassy, D. Laney, A. Mascarenhas, and V. Pascucci. **Topological Feature Extraction and Tracking**. *Journal of Physics: Conference Series* 78.1 (2007), p. 012007.
  - [19] P.-T. Bremer, G. H. Weber, V. Pascucci, M. Day, and J. B. Bell. **Analyzing and Tracking Burning Structures in Lean Premixed Hydrogen Flames**. *IEEE Transactions on Visualization and Computer Graphics* 16.2 (2010), pp. 248–260.
  - [20] P.-T. Bremer, G. H. Weber, J. Tierny, V. Pascucci, M. Day, and J. Bell. **Interactive Exploration and Analysis of Large-Scale Simulations Using Topology-Based Data Segmentation**. *IEEE Transactions on Visualization and Computer Graphics* 17.9 (2011), pp. 1307–1324.

- [21] S. Bremm, M. Heß, T. von Landesberger, and D. W. Fellner. **PCDC-On the Highway to Data-A Tool for the Fast Generation of Large Synthetic Data Sets**. *EuroVA 2012: International Workshop on Visual Analytics*. The Eurographics Association. 2012, pp. 7–11.
- [22] L. Byron and M. Wattenberg. **Stacked Graphs-Geometry & Aesthetics**. *IEEE Transactions on Visualization and Computer Graphics* 14.6 (2008), pp. 1245–1252.
- [23] S. K. Card and J. Mackinlay. **The Structure of the Information Visualization Design Space**. *Proceedings of the 1997 IEEE Symposium on Information Visualization (InfoVis '97)*. INFOVIS '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 92–99. ISBN: 0-8186-8189-6.
- [24] H. Carr, J. Snoeyink, and U. Axen. **Computing Contour Trees in All Dimensions**. *Computational Geometry* 24.2 (2003), pp. 75–94.
- [25] H. Carr, J. Snoeyink, and M. van de Panne. **Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree**. *Computational Geometry* 43.1 (2010). Special Issue on the 14th Annual Fall Workshop, pp. 42–58. ISSN: 0925-7721. URL: <http://www.sciencedirect.com/science/article/pii/S0925772109000455>.
- [26] H. Carr, J. Snoeyink, and M. van de Panne. **Simplifying Flexible Isosurfaces using Local Geometric Measures**. *Proceedings of the Conference on Visualization '04*. IEEE Computer Society. IEEE Computer Society Press, 2004, pp. 497–504.
- [27] J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Tukey. **Graphical Methods for Data Analysis**. The Wadsworth Statistics/Probability Series, 1983.
- [28] Y.-H. Chan, C. Correa, and K.-L. Ma. **Flow-Based Scatterplots for Sensitivity Analysis**. *IEEE Symposium on Visual Analytics Science and Technology (VAST)*. 2010, pp. 43–50.
- [29] F. Chen, H. Obermaier, H. Hagen, B. Hamann, J. Tierny, and V. Pascucci. **Topology Analysis of Time-Dependent Multi-Fluid Data using the Reeb Graph**. *Computer Aided Geometric Design* 30.6 (2013). Foundations of Topological Analysis, pp. 557–566. ISSN: 0167-8396.
- [30] J. Choo, S. Bohn, and H. Park. **Two-stage Framework for Visualization of Clustered High Dimensional Data**. *IEEE Symposium on Visual Analytics Science and Technology (VAST)*. 2009, pp. 67–74.

- [31] J. Choo, H. Lee, J. Kihm, and H. Park. **iVisClassifier: An Interactive Visual Analytics System for Classification based on Supervised Dimension Reduction**. *IEEE Symposium on Visual Analytics Science and Technology (VAST)*. 2010, pp. 27–34.
- [32] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. **Stability of Persistence Diagrams**. *Discrete Computational Geometry* 37 (1 2007), pp. 103–120. ISSN: 0179-5376.
- [33] D. Cohen-Steiner, H. Edelsbrunner, and D. Morozov. **Vines and Vineyards by Updating Persistence in Linear Time**. *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*. ACM. 2006, pp. 119–126.
- [34] C. Correa, P. Lindstrom, and P.-T. Bremer. **Topological Spines: A Structure-preserving Visual Representation of Scalar Fields**. *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 1842–1851. ISSN: 1077-2626.
- [35] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. **Modeling Wine Preferences by Data Mining from Physicochemical Properties**. *Decision Support Systems* 47.4 (2009). Smart Business Networks: Concepts and Empirical Evidence, pp. 547–553. ISSN: 0167-9236.
- [36] W. Cui, S. Liu, L. Tan, C. Shi, Y. Song, Z. Gao, H. Qu, and X. Tong. **TextFlow: Towards Better Understanding of Evolving Topics in Text**. *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 2412–2421. ISSN: 1077-2626.
- [37] W. Cui, S. Liu, Z. Wu, and H. Wei. **How Hierarchical Topics Evolve in Large Text Corpora**. *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 2281–2290.
- [38] G. S. Davidson, B. Hendrickson, D. K. Johnson, C. E. Meyers, and B. N. Wylie. **Knowledge Mining with VxInsight: Discovery through Interaction**. *Journal of Intelligent Information Systems* 11 (1998), pp. 259–285.
- [39] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. **Indexing by Latent Semantic Analysis**. *Journal of the American Society for Information Science* 41.6 (1990), pp. 391–407.

- [40] D. Demir, K. Beketayev, G. H. Weber, P.-T. Bremer, V. Pascucci, and B. Hamann. **Topology Exploration with Hierarchical Landscapes**. *Proceedings of the Workshop at SIGGRAPH Asia*. WASA '12. Singapore, Singapore: ACM, 2012, pp. 147–154. ISBN: 978-1-4503-1835-8. URL: <http://doi.acm.org/10.1145/2425296.2425323>.
- [41] W. Dou, X. Wang, R. Chang, and W. Ribarsky. **ParallelTopics: A probabilistic approach to exploring document collections**. *IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE Computer Society, 2011, pp. 231–240.
- [42] R. O. Duda, P. E. Hart, and D. G. Stork. **Pattern Classification**. 2nd ed. New York: Wiley, 2001.
- [43] B. Duran and P. Odell. **Cluster Analysis: A Survey**. Springer-Verlag, New York, 1974.
- [44] H. Edelsbrunner and J. Harer. **Jacobi Sets of Multiple Morse Functions**. *Foundations of Computational Mathematics* (2002), pp. 37–57.
- [45] H. Edelsbrunner, J. Harer, A. Mascarenhas, V. Pascucci, and J. Snoeyink. **Time-varying Reeb Graphs for Continuous Space-time Data**. *Computational Geometry* 41.3 (2008), pp. 149–166.
- [46] H. Edelsbrunner, D. Letscher, and A. Zomorodian. **Topological Persistence and Simplification**. *Discrete Computational Geometry* 28.4 (2002), pp. 511–533.
- [47] H. Edelsbrunner and E. P. Mücke. **Simulation of Simplicity: A Technique to Cope with Degenerate Cases in Geometric Algorithms**. *ACM Transactions on Graphics* 9.1 (1990), pp. 66–104.
- [48] S. Eichelbaum, M. Hlawitschka, and G. Scheuermann. **LineAO — Improved Three-Dimensional Line Rendering**. *IEEE Transactions on Visualization and Computer Graphics* 19.3 (2013), pp. 433–445. ISSN: 1077-2626.
- [49] S. Eichelbaum, M. Hlawitschka, and G. Scheuermann. **OpenWalnut: An Open-Source Tool for Visualization of Medical and Bio-Signal Data**. *Biomedical Engineering / Biomedizinische Technik*. Ed. by O. Dössel. 2013.
- [50] S. Eichelbaum, M. Hlawitschka, A. Wiebel, and G. Scheuermann. **OpenWalnut - An Open-Source Visualization System**. *Proceedings of the 6th High-End Visualization Workshop*. Ed. by W. Benger, A. Gerndt, S. Su, W. Schoor, M. Koppitz, W. Kapferer, H.-P. Bischof, and M. D. Pierro. 2010, pp. 67–78.

- [51] N. Elmqvist, P. Dragicevic, and J.-D. Fekete. **Rolling the Dice: Multi-dimensional Visual Exploration using Scatterplot Matrix Navigation.** *IEEE Transactions on Visualization and Computer Graphics* 14 (2008), pp. 1141–1148. ISSN: 1077-2626.
- [52] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. **A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.** *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining.* AAAI Press, 1996, pp. 226–231.
- [53] B. Everitt, S. Landau, and M. Leese. **Cluster Analysis.** London: Arnold, 2001.
- [54] M. A. Fanty and R. Cole. **Spoken Letter Recognition.** *Advances in Neural Information Processing Systems (NIPS).* 1990.
- [55] D. Fasulo. **An Analysis of Recent Work on Clustering Algorithms.** Tech. rep. 01-03-02. Department of Computer Science and Engineering, University of Washington, 1999.
- [56] J.-D. Fekete and C. Plaisant. **Excentric Labeling: Dynamic Neighborhood Labeling for Data Visualization.** *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* CHI '99. Pittsburgh, Pennsylvania, USA: ACM, 1999, pp. 512–519. ISBN: 0-201-48559-1.
- [57] M. Forina, C. Armanino, S. Lanteri, and E. Tiscornia. **Classification of Olive Oils from their Fatty Acid Composition.** *Food Research and Data Analysis* (1983), pp. 189–214.
- [58] S. Fortune. **Computing in Euclidean Geometry.** Ed. by D.-Z. Du and F. Hwang. 2nd ed. Lecture Notes Series on Computing 4. World Scientific Press, 1995. Chap. Voronoi Diagrams and Delaunay Triangulations, pp. 193–233.
- [59] S. Fortune. **Voronoi Diagrams and Delaunay Triangulations.** *Handbook of Discrete and Computational Geometry.* Ed. by J. E. Goodman and J. O'Rourke. Boca Raton, FL, USA: CRC Press, Inc., 1997, pp. 377–388.
- [60] Y.-H. Fua, M. O. Ward, and E. A. Rundensteiner. **Hierarchical Parallel Coordinates for Exploration of Large Datasets.** *Proceedings of the Conference on Visualization '99.* San Francisco, California, United States: IEEE Computer Society Press, 1999, pp. 43–50. ISBN: 0-7803-5897-X.



- [61] Y.-H. Fua, M. O. Ward, and E. A. Rundensteiner. **Structure-Based Brushes: A Mechanism for Navigating Hierarchically Organized Data and Information Spaces.** *IEEE Transactions on Visualization and Computer Graphics* 6 (2000), pp. 150–159. ISSN: 1077-2626.
- [62] K. Fukunaga. **Introduction to Statistical Pattern Recognition (2nd ed.)** San Diego, CA, USA: Academic Press Professional, Inc., 1990. ISBN: 0-12-269851-7.
- [63] G. W. Furnas and A. Buja. **Prosection Views: Dimensional Inference through Sections and Projections.** *Journal of Computational and Graphical Statistics* 3 (1994), pp. 323–385.
- [64] R. K. Gabriel and R. R. Sokal. **A New Statistical Approach to Geographic Variation Analysis.** *Systematic Biology* 18.3 (1969), pp. 259–270.
- [65] E. Gansner, Y. Hu, and S. Kobourov. **GMap: Visualizing Graphs and Clusters as Maps.** *IEEE Pacific Visualization Symposium (PacificVis 2010)*. 2010, pp. 201–208.
- [66] S. Gerber, P.-T. Bremer, V. Pascucci, and R. Whitaker. **Visual Exploration of High-Dimensional Scalar Functions.** *IEEE Transactions on Visualization and Computer Graphics* 16 (2010), pp. 1271–1280. ISSN: 1077-2626.
- [67] M. Graham and J. Kennedy. **A Survey of Multiple Tree Visualisation.** *Information Visualization* 9.4 (2010), pp. 235–252. ISSN: 1473-8716.
- [68] P. Guo, H. Xiao, Z. Wang, and X. Yuan. **Interactive Local Clustering Operations for High-Dimensional Data in Parallel Coordinates.** *IEEE Pacific Visualization Symposium (PacificVis 2010)*. 2010, pp. 97–104.
- [69] J. Han and M. Kamber. **Data Mining: Concepts and Techniques.** The Morgan Kaufmann series in data management systems. Elsevier, 2006. ISBN: 9781558609013.
- [70] J. Handl and J. Knowles. **Cluster Generators for Large High-Dimensional Data Sets with Large Numbers of Clusters.** <http://dbkgroup.org/handl/generators>. 2005.
- [71] J. Hartigan. **Clustering Algorithms.** New York: Wiley, 1975.
- [72] W. Harvey and Y. Wang. **Topological Landscape Ensembles for Visualization of Scalar-Valued Functions.** *Computer Graphics Forum* 29.3 (2010), pp. 993–1002. ISSN: 1467-8659.

- 
- [73] S. Havre, E. Hetzler, P. Whitney, and L. Nowell. **Themeriver: Visualizing Thematic Changes in Large Document Collections**. *IEEE Transactions on Visualization and Computer Graphics* 8.1 (2002), pp. 9–20.
- [74] C. Heine, D. Schneider, H. Carr, and G. Scheuermann. **Drawing Contour Trees in the Plane**. *IEEE Transactions on Visualization and Computer Graphics* 17.11 (2011), pp. 1599–1611. ISSN: 1077-2626.
- [75] I. Herman, G. Melançon, and M. S. Marshall. **Graph Visualization and Navigation in Information Visualization: A Survey**. *IEEE Transactions on Visualization and Computer Graphics* 6.1 (2000), pp. 24–43. ISSN: 1077-2626.
- [76] A. Hinneburg, C. C. Aggarwal, and D. A. Keim. **What Is the Nearest Neighbor in High Dimensional Spaces?** *VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 506–515.
- [77] A. Hinneburg and D. A. Keim. **An Efficient Approach to Clustering in Large Multimedia Databases with Noise**. *Knowledge Discovery and Data Mining*. 1998, pp. 58–65.
- [78] M. Hoffman, D. M. Blei, and F. Bach. **Online Learning for Latent Dirichlet Allocation**. *Advances in Neural Information Processing Systems* 23 (2010), pp. 856–864.
- [79] S. Ingram, T. Munzner, V. Irvine, M. Tory, S. Bergner, and T. Moeller. **DimStiller: Workflows for Dimensional Analysis and Reduction**. *IEEE Symposium on Visual Analytics Science and Technology (VAST)*. IEEE Computer Society, 2010, pp. 3–10.
- [80] S. Ingram, T. Munzner, and M. Olano. **Glimmer: Multilevel MDS on the GPU**. *IEEE Transactions on Visualization and Computer Graphics* 15 (2009), pp. 249–261. ISSN: 1077-2626.
- [81] A. Inselberg. **Parallel Coordinates: Visual Multidimensional Geometry and Its Applications**. Springer-Verlag New York, Inc., 2009.
- [82] A. Inselberg and B. Dimsdale. **Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry**. *VIS '90: Proceedings of the 1st Conference on Visualization '90*. 1990, pp. 361–378. ISBN: 0-8186-2083-8.
- [83] P. Jähnichen, P. Oesterling, T. Liebmann, G. Heyer, C. Kuras, and G. Scheuermann. **Exploratory Search Through Interactive Visualization of Topic Models**. *Digital Humanities 2015 (to appear)*. 2015.

- 
- [84] A. K. Jain and R. C. Dubes. **Algorithms for Clustering Data**. Prentice-Hall, 1988.
  - [85] A. K. Jain, R. P. W. Duin, and J. Mao. **Statistical Pattern Recognition: A Review**. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.1 (2000), pp. 4–37.
  - [86] H. Jänicke, M. Bottinger, and G. Scheuermann. **Brushing of Attribute Clouds for the Visualization of Multivariate Data**. *IEEE Transactions on Visualization and Computer Graphics* 14 (2008), pp. 1459–1466. ISSN: 1077-2626.
  - [87] J. Jaromczyk and G. Toussaint. **Relative Neighborhood Graphs and their Relatives**. *Proceedings of the IEEE*. Vol. 80. Issue 9. 1992, pp. 1502–1517.
  - [88] M. Jeon, H. Park, and J. B. Rosen. **Dimensional Reduction based on Centroids and Least Squares for Efficient Processing of Text Data**. *Proceedings for the First SIAM International Workshop on Text Mining*. 2001.
  - [89] D. H. Jeong, C. Ziemkiewicz, B. Fisher, W. Ribarsky, and R. Chang. **iPCA: An Interactive System for PCA-Based Visual Analytics**. *Computer Graphics Forum*. Vol. 28. 3. Blackwell Publishing Ltd. 2009, pp. 767–774.
  - [90] G. Ji and H.-W. Shen. **Feature Tracking using Earth Mover’s Distance and Global Optimization**. *PG’06: Proceedings of the Pacific Graphics 2006*. 2006.
  - [91] J. Johansson, P. Ljung, M. Jern, and M. Cooper. **Revealing Structure within Clustered Parallel Coordinates Displays**. *Proceedings of the 2005 IEEE Symposium on Information Visualization*. IEEE Computer Society, 2005, pp. 125–132.
  - [92] I. T. Jolliffe. **Principal component analysis**. Springer, 2002.
  - [93] R. Kanjanabose, A. Abdul-Rahman, and M. Chen. **A Multi-task Comparative Study on Scatter Plots and Parallel Coordinates Plots**. *Computer Graphics Forum* 34.3 (2015), pp. 261–270. ISSN: 1467-8659.
  - [94] S. Kaski, T. Honkela, K. Lagus, and T. Kohonen. **WEBSOM—Self-Organizing Maps of Document Collections**. *Neurocomputing* 21.1 (1998), pp. 101–117.
  - [95] L. Kaufman and P. J. Rousseeuw. **Finding Groups in Data: An Introduction to Cluster Analysis**. Wiley Series in Probability and Statistics. Wiley-Interscience, 2005. ISBN: 0471735787.

- [96] D. A. Keim. **Information Visualization and Visual Data Mining.** *IEEE Transactions on Visualization and Computer Graphics* 8.1 (2002), pp. 1–8.
- [97] D. A. Keim, M. Ankerst, and H.-P. Kriegel. **Recursive Pattern: A Technique for Visualizing Very Large Amounts of Data.** *Visualization, 1995. Visualization '95. Proceedings., IEEE Conference on.* IEEE Computer Society, 1995, pp. 279–286. ISBN: 0-8186-7187-4.
- [98] D. A. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann, eds. **Mastering The Information Age-Solving Problems with Visual Analytics.** Eurographics Association, 2010.
- [99] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. **Visual Analytics: Definition, Process, and Challenges.** *Information Visualization.* Ed. by A. Kerren, J. T. Stasko, J.-D. Fekete, and C. North. Vol. 4950. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 154–175. ISBN: 978-3-540-70955-8.
- [100] D. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler. **Visual Analytics: Scope and Challenges.** English. *Visual Data Mining.* Ed. by S. J. Simoff, M. Böhlen, and A. Mazeika. Vol. 4404. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 76–90. ISBN: 978-3-540-71079-0.
- [101] P. Keller and M. Bertram. **Modeling and Visualization of Time-varying Topology Transitions guided by Hyper Reeb Graph Structures.** *Computer Graphics and Imaging.* Citeseer. 2007, pp. 15–20.
- [102] L. Kettner, J. Rossignac, and J. Snoeyink. **The Safari Interface for Visualizing Time-dependent Volume Data using Iso-surfaces and Contour Spectra.** *Computational Geometry* 25.1 (2003), pp. 97–116.
- [103] S. Koch, H. Bosch, M. Giereth, and T. Ertl. **Iterative Integration of Visual Insights during Patent Search and Analysis.** *IEEE Symposium on Visual Analytics Science and Technology (VAST).* 2009, pp. 203–210.
- [104] T. Kohonen. **Self-Organizing Maps.** 3rd. Springer, 2001.
- [105] N. Kong, J. Heer, and M. Agrawala. **Perceptual Guidelines for Creating Rectangular Treemaps.** *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 990–998.
- [106] J. B. Kruskal and J. M. Landwehr. **Icicle Plots: Better Displays for Hierarchical Clustering.** *The American Statistician* 37.2 (1983), pp. 162–168. ISSN: 00031305.

- [107] J. B. Kruskal and M. Wish. **Multidimensional Scaling**. SAGE Publications, 1978.
- [108] T. von Landesberger, S. Bremm, N. Andrienko, G. Andrienko, and M. Tekusova. **Visual Analytics Methods for Categorical Spatio-temporal Data**. *IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE, 2012, pp. 183–192.
- [109] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J.-D. Fekete, and D. W. Fellner. **Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges**. *Computer Graphics Forum* 30.6 (2011), pp. 1719–1749.
- [110] R. Laramée, H. Hauser, L. Zhao, and F. Post. **Topology-Based Flow Visualization, The State of the Art**. *Topology-Based Methods in Visualization (TopoInVis)*. Springer Berlin Heidelberg, 2007, pp. 1–19.
- [111] M. Levoy. **Display of Surfaces From Volume Data**. *IEEE Computer Graphics and Applications* 8.3 (1988), pp. 29–37.
- [112] T. Liebmann, P. Oesterling, S. Jänicke, and G. Scheuermann. **A Geological Metaphor for Geospatial-temporal Data Analysis**. *IVAPP '14: Proceedings of the 5th International Conference on Information Visualization Theory and Application*. SciTePress, 2014.
- [113] S. Liu, B. Wang, J. J. Thiagarajan, P.-T. Bremer, and V. Pascucci. **Visual Exploration of High-Dimensional Data through Subspace Analysis and Dynamic Projections**. *Computer Graphics Forum* 34.3 (2015), pp. 271–280. ISSN: 1467-8659.
- [114] Y. Livnat, H.-W. Shen, and C. R. Johnson. **A Near Optimal Isosurface Extraction Algorithm Using the Span Space**. *IEEE Transactions on Visualization and Computer Graphics* 2.1 (1996), pp. 73–84. ISSN: 1077-2626.
- [115] W. E. Lorensen and H. E. Cline. **Marching Cubes: A High Resolution 3D Surface Construction Algorithm**. *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '87. New York, NY, USA: ACM, 1987, pp. 163–169. ISBN: 0-89791-227-6.
- [116] J. MacQueen. **Some Methods for Classification and Analysis of Multivariate Observations**. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 1. Berkeley, California: University of California Press, 1967, pp. 281–297.
- [117] A. Mascarenhas. private communication. 2013.

- [118] A. Mascarenhas and J. Snoeyink. **Implementing Time-varying Contour Trees**. *Proceedings of the Twenty-first Annual Symposium on Computational Geometry*. ACM. 2005, pp. 370–371.
- [119] A. Mascarenhas and J. Snoeyink. **Isocontour-Based Visualization of Time-varying Scalar Fields**. *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*. Springer Berlin Heidelberg, 2009, pp. 41–68.
- [120] K. T. McDonnell and K. Mueller. **Illustrative Parallel Coordinates**. *Computer Graphics Forum* 27.3 (2008), pp. 1031–1038.
- [121] N. E. Miller, P. Chung Wong, M. Brewster, and H. Foote. **Topic Islands - A Wavelet-Based Text Visualization System**. *Proceedings of the Conference on Visualization '98*. VIS '98. Research Triangle Park, North Carolina, USA: IEEE Computer Society Press, 1998, pp. 189–196. ISBN: 1-58113-106-2.
- [122] E. J. Nam, Y. Han, K. Mueller, A. Zelenyuk, and D. Imre. **ClusterSculptor: A Visual Analytics Tool for High-Dimensional Data**. *Proceedings of the 2007 IEEE Symposium on Visual Analytics Science and Technology*. IEEE Computer Society, 2007, pp. 75–82.
- [123] R. Ng and J. Han. **CLARANS: A Method for Clustering Objects for Spatial Data Mining**. *IEEE Transactions on Knowledge and Data Engineering* 14.5 (2002), pp. 1003–1016. ISSN: 1041-4347.
- [124] M. Novotny and H. Hauser. **Outlier-Preserving Focus+Context Visualization in Parallel Coordinates**. *IEEE Transactions on Visualization and Computer Graphics* 12 (2006), pp. 893–900. ISSN: 1077-2626.
- [125] P. Oesterling, C. Heine, H. Jänicke, and G. Scheuermann. **Visual Analysis of High-Dimensional Point Clouds using Topological Landscapes**. *IEEE Pacific Visualization Symposium (PacificVis 2010)*. Ed. by S. North, H.-W. Shen, and J. van Wijk. 2010, pp. 113–120.
- [126] P. Oesterling, C. Heine, H. Jänicke, G. Scheuermann, and G. Heyer. **Visualization of High-Dimensional Point Clouds Using Their Density Distribution's Topology**. *IEEE Transactions on Visualization and Computer Graphics* 17.11 (2011), pp. 1547–1559. ISSN: 1077-2626.
- [127] P. Oesterling, C. Heine, G. H. Weber, D. Morozov, and G. Scheuermann. **Computing and Visualizing Time-Varying Merge Trees for High-Dimensional Data**. *Topology-Based Methods in Visualization (TopoInVis)*. (to appear, received “best paper award”). Springer, 2015.

- [128] P. Oesterling, C. Heine, G. H. Weber, and G. Scheuermann. **A Topology-Based Approach to Visualize the Thematic Composition of Document Collections.** *Text Mining: From Ontology Learning to Automated Text Processing Applications*. Ed. by C. Biemann and A. Mehler. Theory and Applications of Natural Language Processing. Springer International Publishing, 2014, pp. 63–85. ISBN: 978-3-319-12654-8.
- [129] P. Oesterling, C. Heine, G. H. Weber, and G. Scheuermann. **Visualizing nD Point Clouds as Topological Landscape Profiles to Guide Local Data Analysis.** *IEEE Transactions on Visualization and Computer Graphics* 19.3 (2013), pp. 514–526. ISSN: 1077-2626.
- [130] P. Oesterling, P. Jähnichen, G. Heyer, and G. Scheuermann. **Topological Visual Analysis of Clusterings in High-Dimensional Information Spaces.** *it - Information Technology* 57.1 (2015). Special Issue: Visual Analytics, pp. 3–10. ISSN: 1611-2776.
- [131] P. Oesterling, G. Scheuermann, S. Teresniak, G. Heyer, S. Koch, T. Ertl, and G. H. Weber. **Two-stage Framework for a Topology-Based Projection and Visualization of Classified Document Collections.** *IEEE Symposium on Visual Analytics Science and Technology (VAST)*. IEEE Computer Society, 2010, pp. 91–98.
- [132] S. Pantazi, Y. Kagolovsky, and J. Moehr. **Cluster Analysis of Wisconsin Breast Cancer Dataset using Self-organizing Maps.** *Studies in Health Technology and Informatics* 90 (2002), pp. 431–6.
- [133] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. **Multi-resolution Computation and Presentation of Contour Trees.** *Lawrence Livermore National Laboratory, Tech. Rep. UCRL-PROC-208680*. 2005.
- [134] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. **The Toporrery: Computation and Presentation of Multi-resolution Topology.** *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*. Ed. by T. Möller, B. Hamann, and R. Russell. Mathematics and Visualization. Springer Berlin Heidelberg, 2009, pp. 19–40. ISBN: 978-3-540-25076-0.
- [135] F. V. Paulovich and R. Minghim. **HiPP: A Novel Hierarchical Point Placement Strategy and its Application to the Exploration of Document Collections.** *IEEE Transactions on Visualization and Computer Graphics* 14 (2008), pp. 1229–1236. ISSN: 1077-2626.

- 
- [136] F. V. Paulovich, L. G. Nonato, R. Minghim, and H. Levkowitz. **Least Square Projection: A Fast High-Precision Multidimensional Projection Technique and Its Application to Document Mapping.** *IEEE Transactions on Visualization and Computer Graphics* 14 (2008), pp. 564–575. ISSN: 1077-2626.
- [137] F. V. Paulovich, M. C. F. Oliveira, and R. Minghim. **The Projection Explorer: A Flexible Tool for Projection-Based Multidimensional Visualization.** *Proceedings of the XX Brazilian Symposium on Computer Graphics and Image Processing*. SIBGRAPI '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 27–36. ISBN: 0-7695-2996-8.
- [138] F. V. Paulovich and R. Minghim. **Text Map Explorer: A Tool to Create and Explore Document Maps.** *International Conference on Information Visualisation*. IEEE Computer Society, 2006, pp. 245–251.
- [139] H. Purchase, C. Pilcher, and B. Plimmer. **Graph Drawing Aesthetics - Created by Users, Not Algorithms.** *IEEE Transactions on Visualization and Computer Graphics* 18.1 (2012), pp. 81–92. ISSN: 1077-2626.
- [140] F. Reinders, F. H. Post, and H. J. Spoelder. **Visualization of Time-Dependent Data with Feature Tracking and Event Detection.** *The Visual Computer* 17.1 (2001), pp. 55–71.
- [141] J. C. Roberts. **State of the Art: Coordinated Multiple Views in Exploratory Visualization.** *Proceedings of the 5th International Conference on Coordinated Multiple Views in Exploratory Visualization (CMV2007)*. IEEE Computer Society Press, 2007, pp. 61–71.
- [142] E. Rosch. **Principles of Categorization.** *Concepts: Core Readings* (1999), pp. 189–206.
- [143] G. Salton, A. Wong, and C. S. Yang. **A Vector Space Model for Automatic Indexing.** *Communications of the ACM* 18.11 (1975), pp. 613–620. ISSN: 0001-0782.
- [144] G. Salton and C. Buckley. **Term Weighting Approaches in Automatic Text Retrieval.** Tech. rep. Ithaca, NY, USA, 1987.
- [145] J. W. Sammon. **A Nonlinear Mapping for Data Structure Analysis.** *IEEE Transactions on Computers* 18.5 (1969), pp. 401–409. ISSN: 0018-9340.
- [146] R. Samtaney, D. Silver, N. Zabusky, and J. Cao. **Visualizing Features and Tracking their Evolution.** *Computer* 27.7 (1994), pp. 20–27.



- [147] E. Sandhaus. **The New York Times Annotated Corpus**. *Linguistic Data Consortium* 6.12 (2008).
- [148] H. Sanftmann and D. Weiskopf. **Illuminated 3D Scatterplots**. *Computer Graphics Forum* 28.3 (2009), pp. 751–758.
- [149] F. Sebastiani. **Machine Learning in Automated Text Categorization**. *ACM Computing Surveys* 34.1 (2002), pp. 1–47. ISSN: 0360-0300.
- [150] G. Sheikholeslami, S. Chatterjee, and A. Zhang. **WaveCluster: A Wavelet-Based Clustering Approach for Spatial Data in Very Large Databases**. *The International Journal on Very Large Data Bases* 8.3-4 (2000), pp. 289–304. ISSN: 1066-8888.
- [151] M. Sherman. **K-means Clustering of Wine Chemistry Data: Creating and Evaluating a Predictive Model and an Analysis of the Data**. Report, <http://www.michaelwsherman.com/projects/winecluster/report.pdf>.
- [152] B. Shneiderman. **The Eyes Have it: A Task by Data Type Taxonomy for Information Visualizations**. *IEEE Symposium on Visual Languages*. 1996, pp. 336–343.
- [153] B. Shneiderman. **Tree Visualization with Tree-Maps: 2-d Space-filling Approach**. *ACM Transactions on Graphics* 11 (1 1992), pp. 92–99. ISSN: 0730-0301.
- [154] B. W. Silverman. **Density Estimation for Statistics and Data Analysis**. Monographs on Statistics and Applied Probability 26. Chapman & Hall, 1986.
- [155] B.-S. Sohn and C. Bajaj. **Time-varying Contour Topology**. *IEEE Transactions on Visualization and Computer Graphics* 12.1 (2006), pp. 14–25.
- [156] H. Späth. **Cluster Analysis Algorithms**. Chichester, U.K.: Ellis Horwood, 1980.
- [157] M. Steinbach, L. Ertöz, and V. Kumar. **The Challenges of Clustering High-Dimensional Data**. *New Directions in Statistical Physics: Applications in Econophysics, Bioinformatics, and Pattern Recognition*. Springer Berlin Heidelberg, 2004, pp. 273–309.
- [158] G. Sun, Y. Wu, S. Liu, T.-Q. Peng, J. J. H. Zhu, and R. Liang. **EvoRiver: Visual Analysis of Topic Coopetition on Social Media**. *IEEE Transactions on Visualization and Computer Graphics* 99.PrePrints (2015), pp. 1753–1762. ISSN: 1077-2626.

- [159] A. Szymczak. **Subdomain-aware Contour Trees and Contour Evolution in Time-dependent Scalar Fields**. *Shape Modeling and Applications, 2005 International Conference*. IEEE, 2005, pp. 136–144.
- [160] S. Takahashi, I. Fujishiro, and M. Okada. **Applying Manifold Learning to Plotting Approximate Contour Trees**. *IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009), pp. 1185–1192. ISSN: 1077-2626.
- [161] A. Tatu, G. Albuquerque, M. Eisemann, J. Schneidewind, H. Theisel, M. Magnor, and D. Keim. **Combining Automated Analysis and Visualization Techniques for Effective Exploration of High-Dimensional Data**. *IEEE Symposium on Visual Analytics Science and Technology (VAST)*. Atlantic City, New Jersey, USA, 2009, pp. 59–66.
- [162] J. B. Tenenbaum, V. de Silva, and J. C. Langford. **A Global Geometric Framework for Nonlinear Dimensionality Reduction**. *Science* 290.5500 (2000), pp. 2319–2323. ISSN: 00368075.
- [163] J. J. Thomas and K. A. Cook. **Illuminating the Path: The Research and Development Agenda for Visual Analytics**. IEEE Computer Society Press, 2005.
- [164] C. Turkay, J. Parulek, N. Reuter, and H. Hauser. **Interactive Visual Analysis of Temporal Cluster Structures**. *Computer Graphics Forum* 30.3 (2011), pp. 711–720.
- [165] W. Wang, J. Yang, and R. R. Muntz. **STING: A Statistical Information Grid Approach to Spatial Data Mining**. *Proceedings of the 23rd International Conference on Very Large Data Bases*. VLDB '97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 186–195. ISBN: 1-55860-470-7.
- [166] G. H. Weber, G. Scheuermann, and B. Hamann. **Detecting Critical Regions in Scalar Fields**. *Proceedings of the Symposium on Data Visualisation 2003*. Eurographics Association, 2003, pp. 85–94.
- [167] G. H. Weber, P.-T. Bremer, M. Day, J. Bell, and V. Pascucci. **Feature Tracking using Reeb Graphs**. *Topological Methods in Data Analysis and Visualization*. Springer Berlin Heidelberg, 2011, pp. 241–253.
- [168] G. H. Weber, D. Morozov, K. Beketayev, J. Bell, P.-T. Bremer, M. Day, B. Hamann, C. Heine, M. Haranczyk, M. Hlawitschka, V. Pascucci, P. Oesterling, and G. Scheuermann. **Topology-Based Visualization and Analysis of High-Dimensional Data and Time-varying Data at the Extreme**

- Scale**. *DOE Exascale Research Conference*. LBNL-5691E-Poster. Portland, OR, 2012.
- [169] G. Weber, P.-T. Bremer, and V. Pascucci. **Topological Landscapes: A Terrain Metaphor for Scientific Data**. *IEEE Transactions on Visualization and Computer Graphics* 13.6 (2007), pp. 1416–1423. ISSN: 1077-2626.
- [170] W. Widanagamaachchi, C. Christensen, V. Pascucci, and P.-T. Bremer. **Interactive Exploration of Large-scale Time-varying Data using Dynamic Tracking Graphs**. *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. IEEE. 2012, pp. 9–17.
- [171] W. Willett, J. Heer, and M. Agrawala. **Scented Widgets: Improving Navigation Cues with Embedded Visualizations**. *IEEE Transactions on Visualization and Computer Graphics* 13 (2007), pp. 1129–1136. ISSN: 1077-2626.
- [172] J. A. Wise, J. J. Thomas, K. Pennock, D. Lantrip, M. Pottier, A. Schur, and V. Crow. **Visualizing the Non-visual: Spatial Analysis and Interaction with Information from Text Documents**. *Proceedings of the 1995 IEEE Symposium on Information Visualization*. Ed. by N. D. Gershon and S. G. Eick. IEEE Computer Society, 1995, pp. 51–58. ISBN: 0-8186-7201-3.
- [173] P. Xu, Y. Wu, E. Wei, T.-Q. Peng, S. Liu, J. Zhu, and H. Qu. **Visual Analysis of Topic Competition on Social Media**. *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), pp. 2012–2021.
- [174] R. Xu, S. Damelin, B. Nadler, and D. C. Wunsch II. **Clustering of High-Dimensional Gene Expression Data with Feature Filtering Methods and Diffusion Maps**. *Artificial Intelligence in Medicine* 48.2-3 (2010), pp. 91–98. ISSN: 0933-3657.
- [175] R. Xu and I. Wunsch D. **Survey of Clustering Algorithms**. *IEEE Transactions on Neural Networks* 16.3 (2005), pp. 645–678. ISSN: 1045-9227.
- [176] J. Yang, M. O. Ward, and E. A. Rundensteiner. **Interactive Hierarchical Displays: A General Framework for Visualization and Exploration of Large Multivariate Data Sets**. *Computers & Graphics* 27.2 (2003), pp. 265–283.
- [177] T. Zhang, R. Ramakrishnan, and M. Livny. **BIRCH: An Efficient Data Clustering Method for Very Large Databases**. *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*. SIGMOD '96. Montreal, Quebec, Canada: ACM, 1996, pp. 103–114. ISBN: 0-89791-794-4.

- [178] H. Zhou, W. Cui, H. Qu, Y. Wu, X. Yuan, and W. Zhuo. **Splatting the Lines in Parallel Coordinates**. *Computer Graphics Forum* 28 (2009), pp. 759–766.