A Parallel Algorithm for Computing the Flow Complex

Theory and Applications

Dissertation

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

vorgelegt dem Rat der Fakultät für
Mathematik und Informatik

der Friedrich-Schiller-Universität Jena

von Dipl.-Inf. Lars Kühne
geboren am 15.06.1987 in Altenburg

Gutachter

1. Prof. Dr. Joachim Giesen
   Friedrich-Schiller-Universität Jena
2. Prof. Klaus Mueller
   Stony Brook University New York
3. Prof. Dr. Artur Andrzejak
   Universität Heidelberg
4. Dr. Sören Laue
   Friedrich-Schiller-Universität Jena

Tag der öffentlichen Verteidigung: 04. Mai 2018

Ehrenwörtliche Erklärung

Hiermit erkläre ich,

- dass mir die Promotionsordnung der Fakultät bekannt ist,
- dass ich die Dissertation selbst angefertigt habe, keine Textabschnitte oder Ergebnisse eines Dritten oder eigenen Prüfungsarbeiten ohne Kennzeichnung übernommen und alle von mir benutzten Hilfsmittel, persönliche Mitteilungen und Quellen in meiner Arbeit angegeben habe,
- dass ich die Hilfe eines Promotionsberaters nicht in Anspruch genommen habe und dass Dritte weder unmittelbar noch mittelbar geldwerte Leistungen von mir für Arbeiten erhalten haben, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen,
- dass ich die Dissertation noch nicht als Prüfungsarbeit für eine staatliche oder andere wissenschaftliche Prüfung eingereicht habe.

Bei der Auswahl und Auswertung des Materials, sowie bei der Herstellung des Manuskripts hat mich mein Betreuer, *Prof. Dr. Joachim Giesen*, unterstützt.

Ich habe weder die gleiche, noch eine in wesentlichen Teilen ähnliche bzw. eine andere Abhandlung bereits bei einer anderen Hochschule als Dissertation eingereicht.

Jena, den 04.05.2018, Lars Kühne

Abstract

We present a concurrent algorithm and its implementation for computing the entire Hasse diagram of the flow complex of a point cloud in Euclidean space. Known algorithms for computing the flow complex in two and three dimensions compute the geometric realization of the flow complex and need to compute the Delaunay triangulation of the point cloud first. Our algorithm computes less information, namely only the Hasse diagram of the flow complex that is augmented with enough geometric information to allow the same topological multiscale analysis of point cloud data as the alpha shape filtration without computing the Delaunay triangulation explicitly. We show experimental results for medium dimensions that demonstrate that our algorithm scales well with the number of available cores on a multicore architecture.

We apply our algorithm for sketching the support of a probability measure on Euclidean space from samples that have been drawn from the measure. This problem is closely related to certain manifold learning problems, where one assumes that the sample points are drawn from a manifold that is embedded in Euclidean space. We prove that a flow complex is homotopy equivalent to the support of the measure for sufficiently dense samplings, and demonstrate the feasibility of our approach on real world data sets.

We apply part of our algorithm to scatter plots, which are mostly used for correlation analysis, but are also a useful tool for understanding the distribution of high-dimensional point cloud data. An important characteristic of point cloud data, that has received little attention so far, are regions that contain no or only few data points. We show, that augmenting scatter plots by flow lines along the gradient vector field of the distance function to the point cloud reveals such empty regions or voids. The augmented scatter plots, that we call sclow plots, enable a much better understanding of the geometry underlying the point cloud than traditional scatter plots, and by that support tasks like dimension inference, detecting outliers, or identifying data points at the interface between clusters.

Zusammenfassung

Wir präsentieren einen parallelen Algorithmus zur Berechnung des Hasse-Diagramms des Flow-Komplexes einer Punktwolke im euklidischen Raum. Bekannte Algorithmen in zwei und drei Dimensionen berechnen zunächst dessen geometrische Realisierung und müssen vorher die Delaunay-Triangulierung berechnen. Unser Algorithmus berechnet nur das Hasse-Diagramm des Flow-Komplexes, welches, mit ausreichend geometrischen Informationen versehen, die selbe topologische Multiskalenanalyse ermöglicht wie die Alpha-Shape Filtration. Wir zeigen mit experimentelle Ergebnissen für mittlere Dimensionen, dass unser Algorithmus gut mit der Anzahl der verfügbaren Kerne auf einer Mehrkern-Architektur skaliert.

Wir wenden unseren Algorithmus an, um den Träger eines Wahrscheinlichkeitsmaßes auf Basis von Punkten zu skizzieren, welche aus dem euklidischen Raum gezogen wurden. Dieses Problem ist eng mit dem des *manifold learning* verwandt, bei welchem man annimmt, dass die Datenpunkte von einer Mannigfaltigkeit gezogen werden, welche im euklidischen Raum eingebettet ist. Wir zeigen, ein hinreichend dichtes Sample vorausgesetzt, dass der Flow-Komplex Homotopie-äquivalent zum Träger des Wahrscheinlichkeitsmaßes ist, und demonstrieren die Machbarkeit unseres Ansatzes auf realen Datensätzen.

Wir wenden unseren Algorithmus auf *Streudiagramme* an, welche zur Korrelationsanalyse verwendet werden, aber auch ein nützliches Werkzeug sind, um die Verteilung hochdimensionaler Punktwolken zu verstehen. Bisher wurden Regionen, welche nur wenige Datenpunkte enthalten wenig beachtet. Wir zeigen, dass durch das Augmentieren von Streudiagrammen durch *Fluss-Linien* entlang des Gradientenvektorfelds der Abstandsfunktion zur Punktwolke derartig leere Gebiete oder Hohlräume aufgezeigt werden. Die so augmentierten Streudiagramme ermöglichen ein viel besseres Verständnis der Geometrie der Punktwolke als zuvor. Dadurch unterstützen sie Aufgaben wie Dimensionsinferenz, Ausreißertests, oder das Identifizieren von Datenpunkten an der Schnittstelle zwischen Clustern.

Acknowledgments

I want to thank my advisor Prof. Dr. Joachim Giesen for giving me the opportunity to work in his group; for introducing me to the field of computational geometry and machine learning, in particular as a part of the *Computational Geometric Learning* project within the 7th Framework Programme of the EC; for allowing me to organize a lecture by myself; for knowing when to help with comments, ideas and discussions; and for sharing his knowledge and experience with me. Your influence on me has shaped my life for the better, and therefore I cannot thank you enough.

Many thanks to Annemarie Kunze for taking care of all the administrative work during my time as a PhD student. I also want to thank Jens K. Müller, Christopher Schneider and Julien Klaus for sharing an office with me and providing me with valuable discussions over all the years. Special thanks to Sören Laue and Philipp Lucas for their valuable collaboration, and Sören Laue in particular for introducing me to the field of convex optimization, and giving me the opportunity to be part of the *Scaling Up Generic Optimization* project funded by DFG.

Furthermore I want to thank Prof. Dr. Joachim Giesen, Prof. Klaus Müller, Prof. Dr. Artur Andrzejak and Dr. Sören Laue for their willingsness and efforts as thesis examiners.

Finally I would like to thank my wife for giving me the most precious gifts I could ask for; love, time and our son Hendrik. This thesis is dedicated to both of you.

Lars Kühne
Jena, May 4th, 2018

Contents

Abstract i
Zusammenfassung ii
Acknowledgments iii

1 Computing the Flow Complex 1
1.1 Introduction 1
1.2 Basic Definitions 3
1.3 The Algorithm 9
1.3.1 Downflow operation 11
1.3.2 Upflow operation 14
1.3.3 Putting things together 15
1.3.4 Computing in parallel 17
1.3.5 Low level predicates 17
1.3.6 Correctness 20
1.3.7 Topological simplification of the Hasse diagram 24
1.4 Results and Discussion 25
1.4.1 Scalability of the Algorithm 25
1.4.2 Simplifying the Hasse diagram 28

2 Applications 31
2.1 Sketching the Support of a Measure 31
2.1.1 Homotopy equivalent sketch 33
2.1.2 Choosing a good value for α 37
2.1.3 Distributing the algorithm 41
2.1.4 Computing α-flow-complexes 42
2.2 Visualizing Empty Space: Sclow Plots 50
2.2.1 Context 52
2.2.2 Distance function and flow lines 56
2.2.3 Sclow plots 60
2.2.4 Synthetic data sets 66
2.2.5 Real data sets 76
2.2.6 User interaction 84
2.2.7 Scalability 87

3 Conclusions 91

Appendix 99
List of Figures 101
List of Tables 103

Chapter 1

Computing the Flow Complex

1.1 Introduction

The flow complex is a data structure on a finite set of points in Euclidean space $\mathbb{R}^d$. A flow complex has been first introduced by Edelsbrunner [18, 19] in three dimensions for surface reconstruction with the WRAP algorithm and for applications in structural computational biology, or more specifically for finding pockets in proteins. The flow complex as defined by Edelsbrunner is always a sub-complex of the Delaunay triangulation of the point set. Here we study a variant of the flow complex that has been introduced by Giesen and John [24, 25], and generalized later by Buchin et al. [7]. Notably, this variant is not a subcomplex of the Delaunay triangulation anymore. In the following we will always refer to the latter variant just as the flow complex. The flow complex has been used for provably correct surface reconstruction [16] and for medial axis approximation with geometric and topological guarantees [30].

It is known that the flow complex can be used for a topological multiscale analysis of point cloud data, in fact the flow complex can be seen as the topologically sparsest encoding of the alpha shape filtration of the Delaunay triangulation of point cloud data [7, 15, 17]. Here topologically sparsest means that any combinatorial change during the filtration of the complex also corresponds to a topological change. This is not true for alpha shapes that can change combinatorially while keeping the homotopy type. All known algorithms for computing the flow complex [24, 10, 26] compute the Delaunay triangulation of the point set first. So far only implementations in two and three dimensions exist. In fact, all these implementations compute the geometric realization of the flow complex which is a polyhedral complex by definition. Cazals and Cohen-Steiner [9] have pointed out that the most important information, i.e., all the information that is needed for a topological multiscale analysis of the point cloud data, can be encoded in an augmented Hasse diagram of the flow complex. The Hasse diagram encodes the incidence structure between the cells of the flow complex and its vertices represent *critical points* of the distance function to the point cloud. In an augmented Hasse diagram also the value of the distance function at the critical points is stored. Here we build on the observation by Cazals and Cohen-Steiner and devise an algorithm for computing the augmented Hasse diagram of the flow complex. Our algorithm avoids the explicit, global computation of the Delaunay triangulation or Voronoi diagram of the point set. The algorithm is essentially a graph exploration algorithm (breadth-first search) that computes implicit information about the Delaunay triangulation only locally. The algorithm borrows ideas from an algorithm by Fischer et al. [22] for computing the smallest enclosing ball of a point cloud in high dimensions. The latter algorithm uses fairly different primitives than the ones that are usually employed for computing Delaunay triangulations and Voronoi diagrams, i.e., in-circle and left-of predicates. In fact the primitives used in the algorithm are not predicates but constructions, namely finding the “nearest point” along a ray, and projecting a point onto the affine hull of a point set. It turns out that these constructions can be implemented with sufficient numerical accuracy using standard floating point arithmetic. Previous versions of this work have appeared in [28] and [29].

1.2 Basic Definitions

Let $P \subset \mathbb{R}^d$ always be a finite point set. In order to simplify the exposition we assume that P is in general position and that P has at least $d+1$ points.

Distance function The distance function

$$h : \mathbb{R}^d \to [0, \infty)$$

induced by P is given as

$$h : \mathbb{R}^d \ni x \mapsto \min_{p \in P} \|x - p\|.$$

The distance function value at x is realized by the neighbors $N(x) = \{p \in P \,:\, \|x - p\| = h(x)\}$. The *driver* $d(x)$ of x is the center of the smallest enclosing ball of $N(x)$, and the gradient of the distance function at x is given as

$$\partial_h(x) = \frac{x - d(x)}{h(x)}, \qquad \text{if} \qquad x \neq d(x),$$

and 0 otherwise.

Critical points The points $x \in \mathbb{R}^d$ with $\partial_h(x) = 0$, i.e., the points for which $x = d(x)$, are called the critical points of the distance function. Critical points can be defined equivalently by the condition $x \in \text{conv}(N(x))$, i.e., critical points are contained in the convex hull of their neighbors in P. The latter characterization can be used to assign an index $i(x)$ to a critical point x, namely the dimension of the affine hull of $N(x)$. Critical points with index 0 are the points in P, i.e., the minima of the distance function. Critical points with index d are maxima of the distance function, and all other critical points are saddle points of the distance function. With this definition the following index theorem holds, see [46],

$$\sum_{i=0}^{d} (-1)^i \, n_i \; = \; 1,$$

where n_i is the number of critical points of index i.

Maximum at infinity We add a symbolic maximum at infinity, i.e., a symbolic critical point of index d. The alternating sum from above now always gives 2 in even dimensions and 0 in odd dimensions, if the symbolic maximum at infinity has been added.

Flow complex The flow complex is a cell complex that consists of the *stable manifolds* of the flow induced by the gradient vector field ∂_h. The flow is a mapping

$$\phi : [0, \infty) \times \mathbb{R}^d \to \mathbb{R}^d$$

defined by the equations $\phi(0, x) = x$ and

$$\lim_{t \downarrow t_0} \frac{\phi(t, x) - \phi(t_0, x)}{t - t_0} = \partial_h(\phi(t_0, x)).$$

The set $\phi(x) = \{\phi(t, x) \,|\, t \geq 0\}$ is called the *orbit* or *flow line* of the point x. The stable manifold $S(x)$ of a critical point x is the set of all points in $\mathbb{R}^d$ that flow into x, i.e.,

$$S(x) = \{y \in \mathbb{R}^d \; : \; \lim_{t \to \infty} \phi(t, y) = x\}.$$

The flow complex is given by the stable manifolds of all critical points together with the following *incidence information* that is defined via the *unstable manifolds* of critical points. Given a neighborhood U of a critical point x and setting

$$V(U) = \{y \in \mathbb{R}^d \; : \; \exists z \in U, t \geq 0 \text{ s.t. } \phi(t, z) = y\},$$

the unstable manifold of x is the set

$$U(x) = \bigcap_{\text{Neighborhood } U \text{ of } x} V(U).$$

The stable manifold of a critical point y is incident to the stable manifold of a critical point x if $S(x) \cap U(y) \neq \emptyset$, i.e., if there is a point in the unstable manifold of y that flows into x. The unstable manifold of the symbolic maximum at infinity is incident to any unbounded stable manifold, i.e., where the set $U(x)$ is unbounded.

Hasse diagram The incidence structure on the stable manifolds of the critical points is a binary relation that is

1. *reflexive*,
   because $S(x) \cap U(x) = \{x\}$ for any critical point x.

2. *antisymmetric*,
   because $S(x) \cap U(y) \neq \emptyset$ and $S(y) \cap U(x) \neq \emptyset$ implies $x = y$.

3. *transitive*,
   because $S(x) \cap U(y) \neq \emptyset$ implies $U(x) \subseteq U(y)$, and hence if x is incident to z, i.e., $S(z) \cap U(x) \neq \emptyset$ then also y is incident to z.

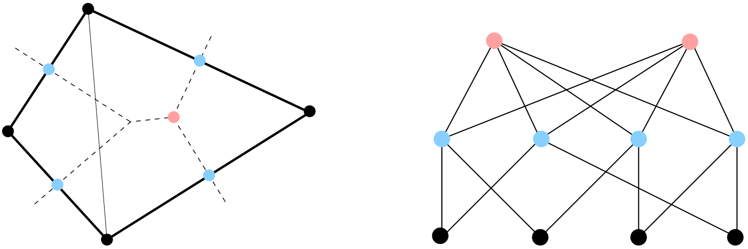


Figure 1.1: On the left: An example of a flow complex in two dimensions. Input are four points. Shown is the Delaunay triangulation of the points, their Voronoi diagram and its critical points. The flow complex has four index-0 critical points, four index-1 critical points, and two index-2 critical points (one of them is the maximum at infinity). On the right: The Hasse diagram of the flow complex.

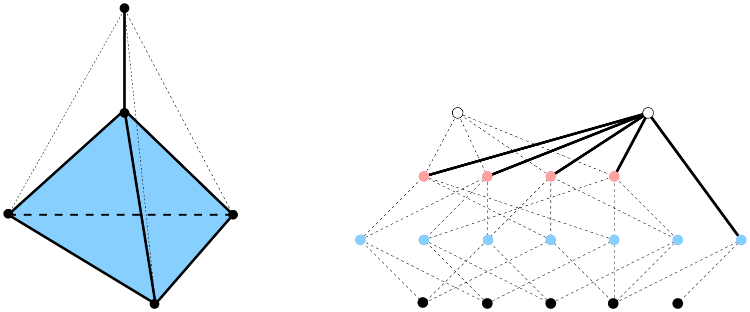


Figure 1.2: On the left: An example of a flow complex in three dimensions. Input are five points. Shown is the Delaunay triangulation of the points and its critical facets (bold). The flow complex has seven critical Delaunay 1-facets (one of them is dangling), four critical Delaunay 2-facets, and two critical Delaunay 3-facets (one finite maximum and the maximum at infinity). On the right: The direct predecessors of the maximum at infinity.

Hence, the combinatorial structure of the flow complex induces a partial order on the set of stable manifolds which can be encoded in a Hasse diagram see Figure 1.1.

Any chain of this partial order is also ordered by the indices of the critical points. Note though that there may exist chains that do not correspond to consecutive indices and have no super chain with consecutive indices. Geometrically, this is caused by “dangling” stable manifolds for critical points of index k that are not in the boundary of any stable manifold of index $(k+1)$, see Figure 1.2.

For $\alpha \geq 0$, the *α-flow-complex* is the Hasse diagram of the flow complex restricted to the critical points x of h for which $h(x) \leq \alpha$. Our goal is to compute only the Hasse diagram of the flow complex for a given point set P and not the geometrically realized complex. But additionally, we also compute the index and the distance function value for every critical point. Since the flow complex is intimately linked to the Voronoi diagram and the Delaunay triangulation of the point set we also briefly restate their definitions here.

Voronoi diagram and Delaunay triangulation

The Voronoi cell $V(p)$ of a point $p \in P$ is defined as

$$V(p) = \{x \in \mathbb{R}^d \ : \ \forall q \in P \ \|x - p\| \leq \|x - q\|\}.$$

The non-empty intersection of k Voronoi cells is called a Voronoi $(d+1-k)$-facet, i.e., the Voronoi cells themselves are d-facets. The 0-facets are also called Voronoi vertices. The set of all Voronoi facets together with the incidence structure given by inclusion is called the Voronoi diagram of P. Since any Voronoi cell corresponds to exactly one point in P any Voronoi $(d+1-k)$-facet corresponds to k points in P. The convex hull of these k points is called the Delaunay $(k-1)$-facet dual to the Voronoi $(d+1-k)$-facet. The set of all Delaunay facets together with the incidence structure given by inclusion is called the Delaunay triangulation of P.

The connection with the flow complex is that any critical point of index k is the unique intersection point of a Voronoi $(d+1-k)$-facet and its dual Delaunay $(k-1)$-facet, if it exists. If a Voronoi $(d+1-k)$-facet and its dual Delaunay $(k-1)$-facet intersect, then we call both facets critical.

1.3 The Algorithm

The basic idea behind our algorithm is to explore (the Hasse diagram of) the Delaunay triangulation of the finite point set P in a breadth-first manner, since this allows us to discover the incidence structure of the flow complex for the same point set. The exploration of the Delaunay triangulation is done on the fly, i.e., the triangulation has not been computed a priori.

Conceptually, the exploration works as follows: Starting from the maximum at infinity, all maxima of the flow complex are enumerated. This is done be exploring the *maxima graph*, whose vertices are the maxima of the flow complex and whose edges correspond to the index-$(d-1)$ critical points of the flow complex. The unstable manifolds of the index-$(d-1)$ critical points connect these critical points to maxima. Since the latter unstable manifolds are one-dimensional they can be tracked algorithmically. The remaining critical points can be discovered by recursively computing the predecessors of the predecessors of a critical point. We refer to computing predecessors of a critical point as a *downflow* operation and to tracking the unstable manifold of an index-$(d-1)$ critical point as an *upflow operation*, see Figure 1.3 for a visualization of one step in the downflow and upflow operation, respectively.

Representation of the maximum at infinity To obtain this representation the point cloud P is augmented by $d+1$ points that are the vertices of a bounding simplex for P. Let $\hat{P}$ be the augmented point set. The additional $d+1$ points are by construction exactly the vertices of the convex hull of $\hat{P}$. The maximum at infinity is represented by a point for each Voronoi edge that is dual to the $d+1$ $(d-1)$-facets on the boundary

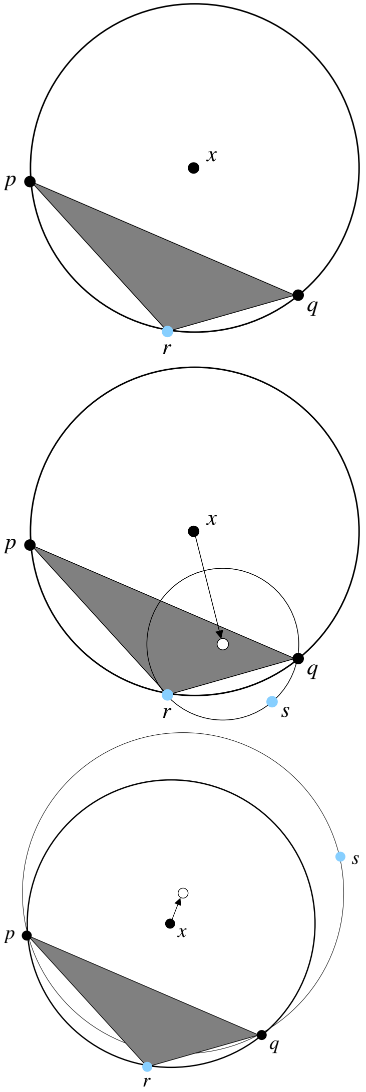


Figure 1.3: Top: The point x can be expressed as an affine combination of the points p, q and r. The coefficient λ_r of r in this combination is negative. The other two coefficients λ_p and λ_q are positive. Middle: The point p (with positive coefficient λ_p) has been dropped, and we have been walking from x towards the center of the smallest enclosing ball of $\{r, q\}$ until s also becomes a nearest neighbor of x. This operation is employed in the downflow operation in Section 1.3.1. Bottom: The point r (with negative coefficient λ_r) has been dropped, and we have been walking starting at x away from the center of the smallest enclosing ball of $\{p, q\}$ until s also becomes a nearest neighbor of x. This operation is employed in the upflow operation in Section 1.3.2 for finding the maxima of the flow complex.

of the convex hull of $\hat{P}$, i.e., there are $d+1$ representatives for the maximum at infinity. The representatives are chosen such that the ball centered at the representatives that has the vertices from $\hat{P} \setminus P$ of the corresponding facet on its boundary does not contain any point from P. Let x be a representative of the maximum at infinity and σ_x be the corresponding facet on the boundary of the convex hull of $\hat{P}$. By construction σ_x is a Delaunay $(d-1)$-facet in the Delaunay triangulation of $\hat{P}$ whose vertices are in $\hat{P} \setminus P$.

1.3.1 Downflow operation

In a downflow operation we compute predecessors of a critical point x in the Hasse diagram of the flow complex. To do so, we are employing a breadth-first strategy, and thus our central data structure is a queue $\mathcal{Q}$.

Initialization Let x be an index-k critical point with $k > 0$. We distinguish two cases, either (a) the point x is the critical point at infinity, or (b) it is not.

(a) The maximum at infinity has, by our construction, $d+1$ representatives $x_1, \ldots, x_{d+1}$ that correspond to Delaunay $(d-1)$-facets $\sigma_1, \ldots, \sigma_{d+1}$ on the boundary of the convex hull of $\hat{P}$. We initialize the queue $\mathcal{Q}$ with the $d+1$ tuples $(\sigma_j, x_j), j = 1, \ldots, d+1$.

(b) Since x is a finite critical point, it is the center of the smallest enclosing ball of a Delaunay k-facet σ. The facet σ is the convex hull of $k+1$ points, namely the vertices of σ. The $(k-1)$-facets $\sigma_j, j = 1, \ldots, k+1$, incident to σ are the convex hulls of the k element subsets of the vertex set of σ, i.e., each σ_j can be obtained by "dropping" one vertex from the vertex set of σ. We initialize the queue $\mathcal{Q}$ with the $k+1$ tuples $(\sigma_j, x), j = 1, \ldots, k+1$.

Exploration Assume that the queue $\mathcal{Q}$ is not empty, otherwise the exploration has been completed. Dequeue the tuple (σ, x) from $\mathcal{Q}$. Our exploration strategy ensures that we never enqueue Delaunay 0-facets (vertices), i.e., the points of P, and thus σ is a Delaunay k-facet with $k > 0$. Let c be the circumcenter of σ, i.e., the center of the smallest ball that has the vertex set V of σ on its boundary. Note that c is contained in the affine hull of V. The exploration strategy is such that it ensures that the $k+1$ points in V are among the nearest points to x in P. We walk from x towards c until one the following two events occurs,

1. a point in $P \setminus V$ also becomes a nearest point to x in P (but not at the very beginning of the walk), or
2. we reach c.

In case of Event 1 let v be the additional nearest point. The convex hull of $V \cup \{v\}$ is a Delaunay $(k+1)$-facet τ, and at the end of the walk x is the circumcenter of $V \cup \{v\}$. As the circumcenter, the point x is contained in the affine hull of $V \cup \{v\}$, and thus x can be written as an affine combination of the points in $V \cup \{v\}$, i.e.,

$$x = \sum_{p \in V} \lambda_p p \, + \, \lambda_v v, \qquad \text{with} \qquad \sum_{p \in V} \lambda_p \, + \, \lambda_v = 1,$$

where the index λ_v is negative because x and v are by construction on opposite sides of the affine hull of V. Let $Q \subset (V \cup \{v\})$

be such that $\lambda_q < 0$ for all $q \in Q$, and let $V' = V \setminus Q$. We consider Delaunay k-facets $\sigma_j, j = 1, \ldots, k+2-|Q|$, incident to τ. These Delaunay facets are the convex hulls of the $k+1-|Q|$ element subsets of V' together with the points in Q, i.e., each σ_j can be obtained by "dropping" one vertex from the vertex set of τ that is not contained Q. We enqueue the tuples $(\sigma_j, x), j = 1, \ldots, k+2-|Q|$, to $\mathcal{Q}$.

In case of Event 2 we distinguish two sub-cases, namely (a) either the circumcenter c of σ coincides with the center of the smallest enclosing ball of σ in which case it is contained in the convex hull of the vertex set V, or (b) it is not.

(a) Since c is contained in the convex hull of V we have found another critical point.

(b) Although c is not contained in the convex hull of V it is still contained in the affine hull of V. Hence, c can be written as an affine combination of the points in V, i.e., $c = \sum_{p \in V} \lambda_p p$ with $\sum_{p \in V} \lambda_p = 1$, where not all coefficients λ_p are non-negative. Let $Q \subset V$ be such that $\lambda_q < 0$ for all $q \in Q$, and let $V' = V \setminus Q$. We consider Delaunay $(k-1)$-facets $\sigma_j, j = 1, \ldots, k+1-|Q|$, incident to σ. These Delaunay facets are the convex hulls of the $k-|Q|$ element subsets of V' together with the points in Q, i.e., each σ_j can be obtained by "dropping" one vertex from the vertex set of σ that is not contained Q. We enqueue the tuples $(\sigma_j, c), j = 1, \ldots, k+1-|Q|$, to $\mathcal{Q}$.

Note that it is exactly this case that allows us to find *dangling* critical points, i.e., critical points that are not incident to any critical point whose index is just one greater. The index gap is reflected in a dimension gap in the exploration as we move from enqueueing k-facets to enqueueing $(k-1)$-facets.

1.3.2 Upflow operation

The implementation of the upflow operation is similar to the implementation of the downflow operation. A crucial difference though is that it only applies to index-$(d-1)$ critical points since they have one-dimensional unstable manifolds that can be tracked. The upflow operation tracks exactly one branch of the unstable manifold of a given index-$(d-1)$ critical point, see [31]. We use the upflow operation to find a maximum that is incident to the given index-$(d-1)$ critical point.

We initialize the upflow operation with a tuple (σ, x), where σ is a Delaunay $(d-1)$-critical facet and x is a non-critical point in the Voronoi edge dual to σ, i.e., x is not the intersection of σ and its dual Voronoi edge.

Assume that we are processing the tuple (σ, x). Let V be the vertex set of σ, and let c be its circumcenter. The exploration strategy is such that it ensures that the points in V are among the nearest points to x in P. We walk from x in the direction $x - c$ until a point $v \in P \setminus V$ also becomes a nearest point to x in P. Let $\hat{V} = V \cup \{v\}$ and τ be the convex hull of $\hat{V}$, i.e., τ is a Delaunay facet. We distinguish two cases, namely (a) either x is contained in τ, or (b) it is not.

(a) In this case x is a critical point, more specifically a maximum (see [31]), and the upflow operation is finished.

(b) In this case x is contained in the affine hull but not in the convex hull of $\hat{V}$. Hence, x can be written as an affine combination of the points in $\hat{V}$, i.e., $x = \sum_{p\in\hat{V}} \lambda_p p$ with $\sum_{p\in\hat{V}} \lambda_p = 1$, where not all coefficients λ_p are non-negative. Let $Q \subset \hat{V}$ be such that $\lambda_q < 0$ for all $q \in Q$. We distinguish two subcases , namely (i) either the convex hull of $\hat{V} \setminus Q$ is a Delaunay $(d-1)$-facet on the boundary of the convex hull of $\hat{P}$, or (ii) it is not.

(i) In this case we have reached a representative of the maximum at infinity, and the upflow operation has finished.

(ii) Let σ' be the convex hull of $\hat{V} \setminus Q$, i.e., σ' is a Delaunay facet incident to τ. We continue the upflow operation with the tuple (σ', x).

1.3.3 Putting things together

With the downflow and upflow operations we have everything at hand to describe our algorithm. The algorithm maintains two data structures, a task queue $\mathcal{Q}$ and a list $\mathcal{L}$ of critical points that have been found already.

A task is a triple $(\sigma, x, label)$, where σ is a Delaunay facet from the Delaunay triangulation of $\hat{P}$, x is a point from the Voronoi facet that is dual to σ and is not contained in the convex hull of σ, and *label* is either *down* if it is downflow task, or *up* it is an upflow task. The task queue $\mathcal{Q}$ generalizes and replaces the queues used in the downflow operations.

Initialization The task queue $\mathcal{Q}$ is initialized with a downflow task at the maximum at infinity. (see Section 1.3.1, Case

(b) of the initialization), i.e., it is initialized with

$$(\sigma_1, x_1, down), \ldots, (\sigma_{d+1}, x_{d+1}, down)$$

where $x_1, \ldots, x_{d+1}$ are the $d+1$ representatives of the maximum at infinity. that correspond to Delaunay $(d-1)$-facets $\sigma_1, \ldots, \sigma_{d+1}$ on the boundary of the convex hull of $\hat{P}$.

Exploration Assume that the queue $\mathcal{Q}$ is not empty, otherwise the computation of the flow complex has been completed. Dequeue the triple $(\sigma, x, label)$ from $\mathcal{Q}$. Depending on *label* either start a downflow or an upflow task. The tasks are basically handled as described in the downflow (Section 1.3.1) and upflow (Section 1.3.2) operations with some small modifications that we describe here. The modifications in the exploration step of the downflow operation are:

1. Whenever a tuple (σ, x) gets enqueued, then the task $(\sigma, x, down)$ is enqueued to the task queue $\mathcal{Q}$ instead.

2. Any critical point x that is discovered during this step is added to the list $\mathcal{L}$ if it is not already stored there. Let σ be the critical Delaunay facet associated with x. If x has not been stored in $\mathcal{L}$ already, then all the tasks $(\sigma_j, x, down)$ are enqueued to the task queue $\mathcal{Q}$, where σ_j are the faces of σ.

The only modification in the exploration step of the upflow operation is: whenever a new tuple (σ, x) has to be processed, then the task (σ, x, up) is enqueued to the task queue $\mathcal{Q}$ instead. If x is a critical point then we have to augment the enqueued upflow tuple with the direction of the flow.

Cleansing A downflow operation may also discover critical points that are predecessors of the initializer x of the operation but not direct predecessors, i.e., it can happen that a discovered predecessor y is also a predecessor of another predecessor of x, see Figure 1.4. Hence, the edge computed by the downflow operation that connects y to x needs to be removed. We refer to removing these spurious edges as cleansing of the *computed* Hasse diagram.

1.3.4 Computing in parallel

Since the tasks in Section 1.3.3 are independent of each other they can be scheduled in parallel. The only shared resources are reading access to the point set P, writing access to the task queue $\mathcal{Q}$, and read/write access to the list $\mathcal{L}$ of critical points that have been found already. It may happen that two downflow tasks that run in parallel reach the same critical point x. In that case, of course, we have to coordinate the update of the successors of x to not override each other. In our implementation we delegate the necessary synchronization to concurrency-aware containers provided by some library, see Section 1.4.1.

1.3.5 Low level predicates

It remains to describe the implementation of the necessary low level predicates. The algorithm makes use of only two primitive operations/constructions, namely finding a nearest point along a ray and projecting a point orthogonally onto an affine hull.

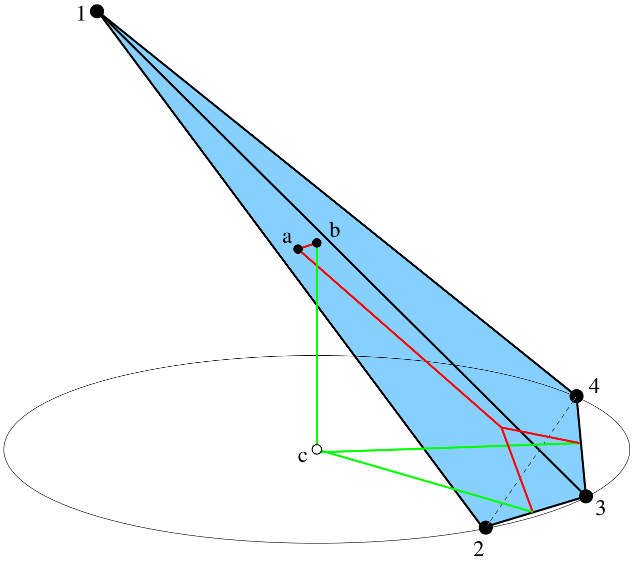


Figure 1.4: Shown is a point set in $\mathbb{R}^3$ with four points. The critical Delaunay facets (in terms of their vertices) are $1, 2, 3, 4, 12, 13, 14, 23, 34, 123, 124, 134, 1234$. The critical point corresponding to 124 is a, and the critical point corresponding to 1234 is b which is the circumcenter of $\{1, 2, 3, 4\}$. Note that a is a predecessor of b. The downflow operation initialized with a finds the critical points corresponding to 23 and 34 (red segments). These points are also found when the downflow operation is initialized with the maximum b (green segments) although they are not direct predecessors of b in the Hasse diagram of the flow. Note that c is the circumcenter of $\{2, 3, 4\}$ and the projection of b onto the affine hull of this set.

Nearest point along a ray The walks that have been described in Sections 1.3.1 and 1.3.2 are piecewise linear and every linear piece is of the following form: given a point x whose nearest neighbors in P include the set $V \subset P$, where x is not contained in the affine hull of V. Let c be the projection of x onto the affine hull of V. Then we either walk from x towards c (as in Section 1.3.1, where the distance function value is decreasing along the walk) or away from c (as in Section 1.3.2, where the distance function value is increasing along the walk), i.e., we either walk from x into the direction $c - x$, or into the direction $x - c$. We stop walking when we reach c in the case that walk towards c, and in both cases we stop once another point from $P \setminus V$ becomes an additional nearest neighbor of x. Assume now that we are walking from x in direction v, where v is either $c - x$ or $x - c$. Let p be some point from V. If we compute t_q for any $q \in P \setminus V$ as the solution of the following equation,

$$\|(x + t_q v) - q\|^2 = \|(x + t_q v) - p\|^2$$

which solves to

$$t_q = \frac{\|p\|^2 - \|q\|^2 - 2\langle x, p - q\rangle}{2\langle v, p - q\rangle},$$

then the nearest point q in $P \setminus V$ along the ray given by x and v is the one for which $t_q > 0$ is minimal. This nearest point stops our walk from x into direction v, if we have not reached c before.

Projection onto an affine hull The walks from Sections 1.3.1 and 1.3.2 repeatedly need to project a point x onto the affine hull of a point set $V \subset P$. For computing such a projection we follow the strategy that has been introduced by Fischer et al. [22] for computing the smallest enclosing ball of a point set in high dimensions. At the heart of this strategy is a dynamic QR-decomposition that allows insertion into and deletion from V and supports computing the affine coefficients with respect to V of the projection of x in the affine hull of V. In [22] it has been shown that the orthogonal projections and affine coefficients can be computed in quadratic time in the dimension d by employing a QR-decomposition.

The nice thing about the strategy in [22] is that the QR-decomposition does not have to be computed from scratch every time along a piecewise linear path (like in the walks in Sections 1.3.1 and 1.3.2) but can be updated incrementally along the path. Setting up the initial QR-decomposition takes time cubic in the dimension d. But the incremental updates can be implemented using Givens rotations such that they need only quadratic time for every update. Note, that the updates that we consider here are either adding a point to or removing a point from V, see Sections 1.3.1 and 1.3.2.

Fischer et al. [22] have already observed that the QR-decomposition behaves nicely with respect to numerical stability using standard floating point arithmetic when all updates on the factors Q and R are implemented with orthogonal Givens rotations.

1.3.6 Correctness

The key property that we need to prove is that (a) all predecessors of a critical point x that are computed in a downflow operation are actually predecessors of x in the Hasse diagram of the flow complex, and (b) that all direct predecessors of x in the Hasse diagram are found by the downflow operation. With this property all critical points and their incidences are found by our algorithm by starting the exploration from the maxima of the flow complex. The maxima are enumerated using upflow operations that track the unstable manifolds of index-$(d-1)$ critical points that connect these points to maxima of the flow (and hence connect the maxima of flow indirectly through index-$(d-1)$ critical points).

In the following we are going to analyze the downflow operation which traverses a tree whose root is the index-k critical point that initializes the operation. By definition of the downflow operation the leaves of the tree are critical points that have a strictly smaller index than the initializer. The edges of the tree are line segments of the form xx', where x' is enqueued to $\mathcal{Q}$ while processing x that has been dequeued before. Actually, x and x' are enqueued together with Delaunay simplices σ and σ', respectively, and we have the following observation.

Observation 1. *The vertices V_x of σ are among the nearest neighbors of x, and the union $V_x \cup V_{x'}$, where $V_{x'}$ is the vertex set of σ', contains only nearest neighbors of x'. Hence, x and x' are both contained in the Voronoi facet that is dual to σ.*

Let $x_n, \ldots, x_1$ be a path in the tree from the root node x_n to a leaf x_1, i.e., in the downflow operation we are always walking from x_{i+1} towards x_i. We need to show that x_1 is a predecessor of x_n in the Hasse diagram of the flow complex. Let $\sigma_i = \sigma_{x_i}$ and $V_i = V_{x_i}$. Consider an edge $x_{i+1}x_i$. There are two cases:

1. The projection of x_{i+1} onto the affine hull of V_i is center c of smallest enclosing ball of V_i, i.e., we are walking from x_{i+1} towards c. The center c is contained in σ_i and is the driver of x_i, i.e., x_i flows into x_{i+1} under the gradient flow.

2. The projection of x_{i+1} onto the affine hull of V_i is not the center of smallest enclosing ball of V_i. Hence, x_i does not flow into x_{i+1} under the gradient flow.

If there is no edge that falls under Case 2, then we are done since x_1 flows through all the x_i into x_n under the gradient flow. Assume now that there is an edge $x_{i+1}x_i$ that falls under Case 2. Our goal now is to eliminate x_i from the path. Note that the edge x_2x_1 does by the definition of the downflow operation always fall under Case 1, and thus it holds $i > 1$. Assume that i is the smallest index such that the edge $x_{i+1}x_i$ falls under Case 2, and consider the three points x_{i+1}, x_i and x_{i-1}. By the minimality assumption x_{i-1} flows into x_i under the gradient flow, i.e., the edge x_ix_{i-1} falls under Case 1. Note that all points in the interior of a Voronoi facet have the same driver, see [24]. That is, the driver of x_i is the driver for all points in the Voronoi facet that is dual to σ_i. Since this Voronoi facet contains the edge $x_{i+1}x_i$, see Observation 1, there are flow lines connecting x_{i-1} to all points on this edge. Thus there is a flow line that connects x_{i-1} to x_{i+1}. It follows that we can eliminate x_i from the path. If we do this iteratively for the edge that falls under Case 2 with smallest index among these edges, then we end up with a sequence of points that connect x_1 to x_n and are traversed under the gradient flow. Hence, x_1 is a predecessor of x_n in the Hasse diagram of the flow complex. We summarize this in the following lemma.

Lemma 1. *All predecessors of a critical point x that are computed in a downflow operation that has been initialized with x are predecessors of x in the Hasse diagram of the flow complex.* □

It remains to show that if a critical point y is a direct predecessor of the critical point x in the Hasse diagram of the flow complex, then the downflow operation initialized with x will discover y. A direct predecessor y of x in the Hasse diagram is connected to x by a flow line. Note that not every critical point that is connected to x by a flow line is a direct predecessor of x, namely if this point is also connected to critical point of higher index that is also connected to x by a flow line. Still, this does not imply that the direct predecessors of x have an index one less than the index of x. It also does not imply that all predecessors that are discovered by the downflow operation are direct predecessors in the Hasse diagram though they are, as we have seen in Lemma 1, connected to x by a flow line.

Assume now that y is a direct predecessor of x, i.e., we have $S(x) \cap U(y) \neq \emptyset$. Let $y = x_1, \ldots, x_n = x$ be the vertex sequence of a flow line in $S(x) \cap U(y)$ that connects y to x. Remember that flow lines are always piecewise linear and every line segment $x_i x_{i+1}$ is contained in a Voronoi facet V_i, see [24]. Let σ_i be the dual Delaunay facet of V_i. If the driver of the Voronoi facet V_i is contained in the line through the segment $x_i x_{i+1}$ for all segments $i = 1, \ldots, n-1$, then the segments are traversed (in the opposite direction) by the downflow operation that has been initialized with $x = x_n$. Otherwise there exists a line segment $x_i x_{i+1}$ such that the driver of V_i is not contained in the line through $x_i x_{i+1}$. Let $x_i x_{i+1}$ be the line segment with this property that has the smallest index i. By construction it must hold $i > 1$ since the line segment that connects x_1 to x_2 is driven by the critical point $y = x_1$ itself. Now, let c be the projection of x_i onto the affine hull of the vertex set of σ_i, and let $\hat{x}$ be the first point on the ray shooting from c to x_i that is contained in V_i. Note that it is possible that $\hat{x} = c$, for example when y is a *dangling* critical point. Replace the segments $x_{i-1}x_i$ and x_ix_{i+1} by the three segments $x_{i-1}\hat{x}$, $\hat{x}x_i$ and x_ix_{i+1}. The latter segments can be traversed (in the opposite direction) by the downflow operation. By repeating this construction as long as there are segments left that cannot be traversed by the downflow operation we can transform any flow line that connects y to x into a sequence of line segments that can be traversed by the downflow operation. Hence, y will be discovered by the downflow operation initialized with x. We summarize this in the following lemma.

Lemma 2. *All direct predecessors of a critical point x in the Hasse diagram of the flow complex are found by the downflow operation that has been initialized with x.* □

The correctness of the algorithm in Section 1.3.3 follows from Lemmas 1 and 2.

1.3.7 Topological simplification of the Hasse diagram

The flow complex as we compute it here, i.e., the the augmented Hasse diagram, can be topologically simplified by a simple combinatorial scheme that has been devised by Cazals and Cohen-Steiner [9]. The scheme works iteratively. In each iteration two incident critical points cancel each other. Given a threshold value t, the scheme works as follows:

1. Determine the pair (x, y) of critical points that are incident in the Hasse diagram such that $i(y) = i(x) + 1$, i.e., the index of y is one greater than the index of x, and the ratio $r_{(x,y)} = \frac{h(y)}{h(x)}$ is minimal among all such pairs.

2. The iterative scheme stops, if $r_{(x,y)} > t$.

3. x and y are removed from the Hasse diagram and the edges incident to x and y are either removed from the Hasse diagram or redistributed as follows: let $In(y)$ be the set of all direct predecessors of y, and let $Out(x)$ be the set of all direct successors of x. If a tuple $(x', y') \in In(y) \times Out(x)$ is not an edge in the Hasse diagram, then it is added to it. All other edges incident to either x or y are removed from the Hasse diagram.

Note the the alternating sum $\sum_{i=0}^{d}(-1)^i \, n_i$, where n_i is the number of critical points of index i, is not affected by the cancellation scheme. Cazals and Cohen-Steiner prove that their simplification scheme cancels pairs of critical points in order of increasing topological persistence.

1.4 Results and Discussion

1.4.1 Scalability of the Algorithm

We have implemented our algorithm for computing the flow complex in C++ using the parallelization library Intel TBB [37]. We ran our experiments in a shared-memory environment on a 32-core AMD Opteron 6128 system with 256 GB of memory. Our implementation turned out to be memory efficient, i.e., in

data set	no. samples	dimension	distribution	time
Random within sphere	50	3	50-137-122-34	0.03s
Random within sphere	50	4	50-199-284-178-44	2.66s
Spiral 3d	30	3	30-41-12-0	0.03s
Spiral 3d	30	4	30-41-12-0-0	0.97s
Spiral 3d	30	5	30-41-12-0-0-0	182.4s
Beethoven Model	500	3	500-1076-649-72	11.81s
Ecoli	336	5	336-1705-2632-1669-436-29	14min
MovieLens	1612	6	1612-9135-16379-13059-5116-975-63	48min

Table 1.1: This table shows the distribution of the number of critical points over their index for various data sets. The notation $a - b - c$ in the distribution column means a critical points of index 0, b critical points of index 1, and c critical points of index 2.

our experiments we never used more than 1% of the available memory.

We examined both the scalability of our algorithm with respect to the number of available cores as well as the correctness of the result in terms of the alternating sum formula (whose fulfillment, of course, is only a necessary condition). Table 1.1 shows the distribution of critical points with respect to their indices as well as the time taken to compute the flow complex for some selected data sets. One can verify the correctness of the alternating sum formula that as it is required always gives 1.

In Figure 1.5 we show the speedup of the computation with an increasing number of utilized cores. This experiment was run on a data set of randomly chosen points in a sphere in

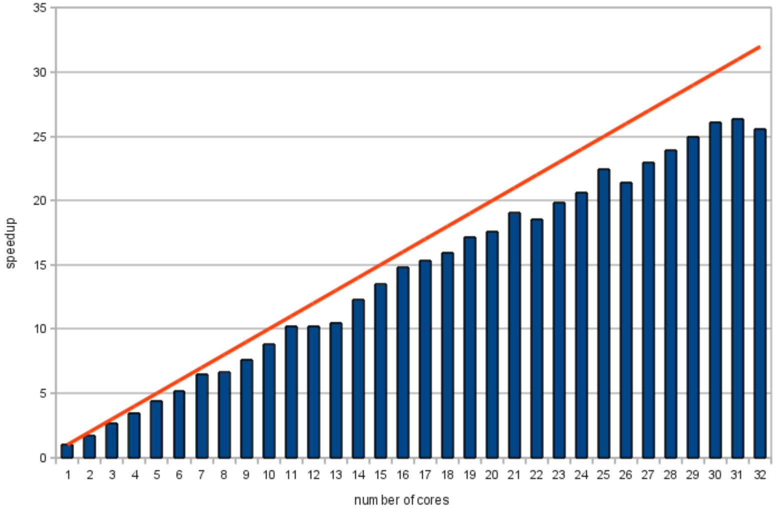


Figure 1.5: A plot of the speedup achieved using a growing number of cores computed on 200 random points in five dimensions. The red line illustrates the ideal linear speedup.

five dimensions. As can be seen, the algorithm scales well in this shared-memory scenario. With an increasing number of cores the communication on the memory bus, that is necessary for insertion of critical points as well as for checking duplicate computation, becomes the bottleneck. The overall parallel overhead reaches its maximum of about 10% when all 32 cores are used.

1.4.2 Simplifying the Hasse diagram

We have also implemented the simplification scheme of Cazals and Cohen-Steiner, see Section 1.3.7, and applied this scheme to the data sets from Section 2.1.4. Table 1.2 shows the index distribution of the critical points before and after the simplification of the Hasse diagram of the flow complex for a given threshold value.

data set	before simplification	after simplification	t
Body Dimensions	507-3807-8306-7966-3551-590	507-1234-830-108-6-0	1.056
Ecoli	336-1705-2632-1669-436-29	336-563-238-10-0-0	1.080
MovieLens 4D	1612-6506-8485-4282-692	1612-3222-1771-160-0	1.036
MovieLens 5D	1612-8060-12964-8917-2698-296	1612-2812-1252-51-0-0	1.063
MovieLens 6D	1612-9135-16379-13059-5116-975-63	1612-3299-1789-104-3-0-0	1.048

Table 1.2: Critical point distribution of flow complexes computed for various data sets before and after the simplification of the Hasse diagram. The dashes separate the number of critical points of increasing index from left to right. Note that the alternating sum formula still holds as it should be the case.

The distribution of critical points of each index in Table 1.2 shows a rapid decline of the number of high index critical points already for small values of the threshold parameter t. The number of critical points can be orders of magnitude smaller after the simplification, and some indices vanish completely, e.g., critical points of index four and five in the case of the Ecoli data set. This indicates that many of the high dimensional stable manifolds (or high index critical points) in the flow complex of these point clouds are not topologically persistent, e.g., correspond to slivers in the Delaunay triangulation of the point clouds.

The simplified flow complex for a threshold value t at which the number of critical points stabilizes can be considered as a topologically persistent simplification of the flow complex or its Hasse diagram. Such a threshold parameter can be determined visually by plotting the number of critical points as a function of the threshold parameter t. See Figure 1.6 for a logarithmically scaled plot for the MovieLens data set. For this data set the number of critical points stabilizes at some threshold value slightly larger than 1.

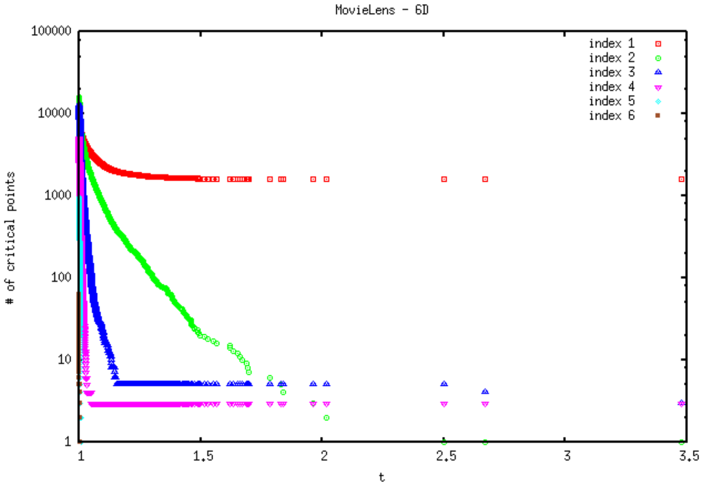


Figure 1.6: A plot of the number of critical points of each available index for the MovieLens data set in six dimensions after each cancellation as a function of the threshold value t.

Chapter 2

Applications

2.1 Sketching the Support of a Probability Measure

We use our algorithm described in the previous chapter for sketching the support of a probability measure on Euclidean space from a sample drawn from the measure. We sketch the support of the measure by a flow complex that has the same homotopy type as the support. We show that the critical points of the distance function are either close to the support or close to a dual structure that is called the medial axis of the support. If the support does not exhibit several geometric scales, then the critical points that belong to the support and the critical points that belong to the medial axis can be separated by simple thresholding, i.e., all critical points at which the distance function takes values less than the threshold value belong to the support and the remaining critical points belong to the medial axis. Restricting the flow complex to the critical points with distance function values less than α constitutes the α-flow-complex. If α is a threshold value at which the two types of critical points can be separated, then the α-flow-complex is homotopy equivalent to the support for sufficiently dense samplings.

We also provide a simple method for choosing a good value for α in practice. The work presented in this chapter is related to certain manifold learning problems that we briefly summarize here. In machine learning, manifold learning is often used synonymously with non-linear dimensionality reduction, but there is also quite some work (mostly in computational geometry) that aims at learning a manifold from samples (that need to satisfy certain conditions), where learning a manifold refers to computing an approximation from a finite sampling that is guaranteed to be topologically equivalent and geometrically close to the manifold. Exemplary for this line of work is the technique by Boissonnat and Ghosh [5]. The body of work in computational geometry does not consider the probabilistic setting where the sample points are drawn at random from the manifold. The probabilistic setting was first considered by Niyogi et al. [41] who show how to compute the homology of a randomly sampled manifold with high confidence. Later Niyogi et al. [42] have extended this approach for recovering the geometric core of Gaussian noise concentrated around a low dimensional manifold, i.e., to the case where the samples are not necessarily drawn from the manifold itself. This can be seen as a topological approach to unsupervised learning.

2.1.1 Homotopy equivalent sketch

We specify conditions under which an α-flow-complex of sample points in $X = \{x1, \ldots, x_n\}$ drawn from a probability measure μ on $\mathbb{R}^d$ is homotopy equivalent to $\mathrm{supp}(\mu)$. But first, we give some basic definitions and provide some necessary background.

Probability measure A non-negative measure μ on $\mathbb{R}^d$ is an additive function that maps every Borel subset $B \subseteq \mathbb{R}^d$ to $\mathbb{R}_{\geq 0}$. Additivity means that

$$\mu\left(\bigcup_{i\in\mathbb{N}} B_i\right) = \sum_{i\in\mathbb{N}} \mu(B_i),$$

where (B_i) is a countable family of disjoint Borel subsets. The measure μ is finite if $\mu(\mathbb{R}^d) < \infty$ and it is a probability measure if $\mu(\mathbb{R}^d) = 1$.

Support The support of a probability measure μ is the set

$$\mathrm{supp}(\mu) = \{x \in \mathbb{R}^d \,|\, \mu(B(x, r)) > 0 \text{ for all } r > 0\},$$

where $B(x, r)$ is the closed ball with radius r that is centered at x. Note that $\mathrm{supp}(\mu)$ is always closed.

Our reconstruction result builds on a theorem proved by Chazal et al. [12]. We need the following technical definitions to state their result. In the following let K always denote a compact subset of $\mathbb{R}^d$.

Reach The reach of K is defined as

$$\inf_{x\in K,\, y\in \mathrm{ma}(K)} \|x-y\|.$$

If the reach of K is positive, then we say that K has finite reach.

α-offset For any set $K \subset \mathbb{R}^d$ and $\alpha > 0$ let K_α be the Minkowski sum of K and $B(0,\alpha)$, i.e.,

$$K_\alpha = \left\{ x \in \mathbb{R}^d \,|\, x \in B(x',\alpha),\, x' \in K \right\}.$$

Theorem 1. [Chazal et al. [12]] *Let $\rho > 0$ be the reach of K and let $K' \subset \mathbb{R}^d$ be a compact set such that the Hausdorff distance between K and K' is less than $\frac{\rho}{17}$, i.e., $d_H(K,K') < \frac{\rho}{17}$, then the complement $\mathbb{R}^d \setminus K'_\alpha$ of K'_α is homotopy equivalent to the complement $\mathbb{R}^d \setminus K$ of K, and K'_α is homotopy equivalent to K_η for all sufficiently small $\eta > 0$, provided that*

$$4 \cdot d_H(K,K') \le \alpha \le \rho - 3 \cdot d_H(K,K').$$

□

Let $X = \{x1,\ldots,x_n\}$ be the sample points drawn from μ. In [15] it has been shown that the union of balls X_α and the α-flow-complex of the finite point set X are homotopy equivalent. Hence, also the α-flow-complex of X is homotopy equivalent to K_η for small $\eta > 0$, if the Hausdorff distance between X_α and K is small, and α is in the range given by Theorem 1.

Finally, we also need a corollary of the following lemma, see [14] (Lemma 5.1).

Lemma 3. *Given a sequence of sample points $x_1, \dots, x_n$ drawn independently from a probability measure μ on $\mathbb{R}^d$. Then, for every $\varepsilon > 0$ and any $x \in supp(\mu)$,*

$$\lim_{n\to\infty} P\big[\|x_1(x) - x\| > \varepsilon\big] = 0,$$

where $x_1(x)$ is the nearest neighbor of x in $\{x_1, \dots, x_n\}$. □

An immediate consequence of this lemma is the following corollary.

Corollary 2. *Given a sequence of sample points $x_1, \dots, x_n$ drawn independently from a probability measure μ with compact support on $\mathbb{R}^d$. Then, for every $\varepsilon > 0$,*

$$\lim_{n\to\infty} P\big[d_H\big(supp(\mu),\, X\big) > \varepsilon\big] = 0,$$

where $d_H\big(supp(\mu),\, X\big)$ is the Hausdorff distance between $supp(\mu)$ and $X = \{x_1, \dots, x_n\}$.

Proof. Since $\mathrm{supp}(\mu)$ is compact there is a finite set of points

$$y_1, \dots, y_m \in \mathrm{supp}(\mu)$$

such that the union of balls $\bigcup_{i=1}^m B(y_i, \varepsilon/2)$ covers $\mathrm{supp}(\mu)$. Assume there exists $y \in \mathrm{supp}(\mu)$ such that $\min_{x\in X} \|x - y\| > \varepsilon$. By construction there exists y_i such that $\|y - y_i\| \le \varepsilon/2$, and thus $\min_{x\in X} \|x - y_i\| > \varepsilon/2$. It follows that

$$\begin{aligned} P\left[\sup_{y\in\mathrm{supp}(\mu)} \min_{x\in X} \|x-y\| > \varepsilon\right] &\le P\left[\max_{y\in\{y_1,\dots,y_m\}} \min_{x\in X} \|x-y\| > \frac{\varepsilon}{2}\right] \\ &\le \sum_{i=1}^m P\left[\min_{x\in X} \|x-y_i\| > \frac{\varepsilon}{2}\right] \\ &= \sum_{i=1}^m P\left[\min_{x\in X} \|x_1(y_i)-y_i\| > \frac{\varepsilon}{2}\right], \end{aligned}$$

where the second inequality follows from a simple union bound. From Lemma 3 we have

$$\lim_{n\to\infty} P[\|x_1(y_i) - y_i\| > \varepsilon/2] = 0,$$

for all y_i, and thus

$$\lim_{n\to\infty} P\left[\sup_{y\in\operatorname{supp}(\mu)} \min_{x\in X} \|x-y\| > \varepsilon\right] \;=\; 0,$$

which implies the claim on the Hausdorff distance since we also have $x_i \in \operatorname{supp}(\mu)$ for all sample points and thus

$$\max_{x\in\{x_1,\ldots,x_n\}} \inf_{y\in\operatorname{supp}(\mu)} \|x-y\| \;=\; 0.$$

$\square$

Now we are prepared to state and prove our topological approximation guarantees.

Theorem 3. *Given a sequence of sample points $x_1,\ldots,x_n$ drawn independently from a probability measure μ with compact support on $\mathbb{R}^d$ whose reach ρ is positive. Then, for every $0<\alpha<\rho$ and sufficiently small $\eta>0$,*

$$\lim_{n\to\infty} P\big[\text{ the } \alpha\text{-flow complex of } \{x_1,\ldots,x_n\} \text{ is not homotopy equivalent to } \operatorname{supp}_\eta(\mu)\,\big] = 0.$$

Proof. Since the α-flow-complex of $X=\{x_1,\ldots,x_n\}$ is homotopy equivalent to the union of balls $B(x_i,\alpha), i=1,\ldots,n$ it suffices to show that

$$\lim_{n\to\infty} P\left[\bigcup_{i=1}^{n} B(x_i,\alpha) \text{ is not homotopy equivalent to } \operatorname{supp}_\eta(\mu)\right] = 0$$

. For that we check that α satisfies the conditions of the reconstruction theorem (Theorem 1). By Corollary 2,

$$\lim_{n\to\infty} P\big[d_H\big(\mathrm{supp}(\mu),\, X\big) > \varepsilon\big] = 0$$

for every $\varepsilon > 0$. Hence,

$$\lim_{n\to\infty} P\big[4 \cdot d_H\big(\mathrm{supp}(\mu), X\big) > \alpha\big] = 0,$$

and

$$\lim_{n\to\infty} P\big[\rho - 3 \cdot d_H\big(\mathrm{supp}(\mu), X\big) < \alpha\big] = 0,$$

which implies the claim about the homotopy equivalence of $\bigcup_{i=1}^{n} B(x_i, \alpha)$ and $\mathrm{supp}_\eta(\mu)$ and hence the claim of the theorem.

$\square$

2.1.2 Choosing a good value for α

At last we prove a theorem that allows us to chose a good value for α in practice. The theorem states that the critical points of the distance function to the set $X = \{x_1, \ldots, x_n\}$ of sample points can be partitioned into two subsets. The first set contains the critical points that are close to $\mathrm{supp}(\mu)$, and the second set contains the critical points that are close to the medial axis $\mathrm{ma}(\mu)$ of $\mathrm{supp}(\mu)$, i.e., there are no critical points in the complement of $\mathrm{supp}(\mu) \cup \mathrm{ma}(\mu)$ or more precisely in

$$\mathrm{compl}_\varepsilon(\mu) \;=\; \mathrm{closure}\big(\mathrm{conv}(\mathrm{supp}(\mu)) \setminus \big(\mathrm{supp}_\varepsilon(\mu) \cup \mathrm{ma}_\varepsilon(\mu)\big)\big)$$

for any small enough $\varepsilon > 0$. Hence, for large samplings only the critical points with small distance values are relevant for sketching $\mathrm{supp}(\mu)$.

Theorem 4. *Given a sequence of sample points* $x_1, \ldots, x_n$ *drawn independently from a probability measure* μ *with compact support on* $\mathbb{R}^d$*. If the reach* ρ *of the support is positive, then, for every* $0 < \varepsilon < \rho/2$,

$$\lim_{n\to\infty} P\big[\mathit{compl}_\varepsilon(\mu) \textit{ contains a critical point of } d_n\big] = 0,$$

where $d_n : \mathbb{R}^d \to \mathbb{R}$ *is the distance function to the set* $X = \{x_1, \ldots, x_n\}$.

Proof. Let (x_n) be a sequence of points in $\mathrm{supp}(\mu)$ such that $c_n \in \mathrm{compl}_\varepsilon(\mu)$ is a critical point of d_n, i.e., the distance function to the first n points of the sequence. Since the closure of $\mathrm{compl}_\varepsilon(\mu)$ is compact we can assume by turning to an appropriate subsequence that the sequence (c_n) converges to $c \in \mathrm{compl}_\varepsilon(\mu)$. By the same argument we can even assume that all the c_n have the same index $i \in \{1, \ldots, d\}$. Let $y_{0n}, \ldots, y_{in}$ be the points in $N(c_n) \subset X$ such that c_n is the center of the smallest enclosing ball of $\{y_{0n}, \ldots, y_{in}\}$, i.e., this ball is given as $B(c_n, \|c_n - y_0\|)$ and does not contain any point from X in its interior. By the compactness of $\mathrm{supp}(\mu)$ we can assume that the sequence (y_{jn}) converges to $y_j \in \mathrm{supp}(\mu)$. Since c_n is the center of the smallest enclosing ball of the points $y_{0n}, \ldots, y_{in}$ it can be written as a convex combination of these points, i.e.,

$$c_n = \sum_{j=0}^{i} \lambda_{jn} y_{jn}$$

with

$$\sum_{j=0}^{i} \lambda_{jn} = 1 \text{ and } \lambda_{jn} \geq 0,\ j = 0, \ldots, i.$$

That is, the vector $\lambda_n = (\lambda_{0n}, \ldots, \lambda_{in})$ is from the i-dimensional standard simplex which is compact. Hence by turning to yet another subsequence we can assume that λ_n converges in the standard simplex. Let $\lambda = (\lambda_0, \ldots, \lambda_i)$ be the limit of (λ_n), then we have $c = \sum_{j=0}^{i} \lambda_j y_j$ and thus c is the center of the smallest enclosing ball $B(c, \|c - y_0\|)$ of the points $y_0, \ldots, y_i$. If $B(c, \|c - y_0\|)$ does not contain any point from $\mathrm{supp}(\mu)$ in its interior, then c must be a point of the medial axis $\mathrm{ma}(\mu)$ which is impossible since the points $c_n \in \mathrm{compl}_\varepsilon(\mu)$ are at distance at least ε from the medial axis, and hence (c_n) can not converge to c. Thus, $B(c, \|c - y_0\|)$ must contain a point $z \in \mathrm{supp}(\mu)$ in its interior, i.e., there exists $\delta > 0$ such that $B(z, \delta) \subset B(c, \|c - y_0\|)$. Since c_n converges to c and the radii $\|c_n - y_{0n}\|$ converge to the radius $\|c_n - y_0\|$ we also have $B(z, \delta) \subset B(c_n, \|c_n - y_{0n}\|)$ for n large enough, and thus $\lim_{n\to\infty} \|x_1(z) - z\| \geq \delta$. That is, for n large enough the event

$$\big[\mathrm{compl}_\varepsilon(\mu) \text{ contains a critical point of } d_n\big]$$

implies the event $\big[\|x_1(z) - z\| \geq \delta\big]$. Hence,

$$\lim_{n\to\infty} P\big[\mathrm{compl}_\varepsilon(\mu) \text{ contains a critical point of } d_n\big] > 0.$$

implies that

$$\lim_{n\to\infty} P\big[\|x_1(z) - z\| \geq \delta\big] > 0 \quad \text{for} \quad z \in \mathrm{supp}(\mu),$$

which contradicts Lemma 3. Thus we have

$$\lim_{n\to\infty} P\big[\mathrm{compl}_\varepsilon(\mu) \text{ contains a critical point of } d_n\big] = 0.$$

$\square$

In practice we expect that the number of critical points whose distance value is at most $\alpha \geq 0$ is increasing fast with growing α for small values of α. Once α is large enough such that all the critical points that belong to supp(μ) have been found, the number of critical points remains constant for growing α, and is only increasing again once the critical points that belong to ma(μ) are being discovered. There are two things one should bear in mind though. First, this behavior is only expected if supp(μ) does not exhibit geometric features on different scales, because otherwise critical points that belong to the medial axis can be discovered before critical points that belong to the support, and second, by construction the medial axis ma(μ) is sampled much more sparsely by critical points than supp(μ). Hence, if supp(μ) does not exhibit geometric features on different scales, then we expect the number of critical points to grow at first with growing α and to remain almost constant once all the critical points that belong to supp(μ) have been discovered. A good value for α should be the point at which the number of critical points stays almost constant. We have computed α-flow-complexes for real data sets and all values for α in the interval $[0, \infty)$, see Section 2.1.4. On these data sets we have not observed geometric multiscale behavior. Hence, in these situations the simple thresholding was enough to compute a complex that is homotopy equivalent to supp(μ) for sufficiently dense samplings.

There is also a straightforward way to distribute the algorithm if we are only interested in computing an α-flow-complex for a small value of $\alpha > 0$. The idea for distributing the algorithm is based on the following simple observation which is implied by the triangle inequality.

Observation 2. *For any* $\alpha > 0$*, if* x *is a critical point of the distance function* $h : \mathbb{R}^d \to \mathbb{R}$ *to the set* $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^d$ *whose distance value* $h(x)$ *is at most* α *and whose nearest neighbor set* $N(x)$ *contains* p_i*, then* $N(x)$ *is contained in the ball* $B(p_i, 2\alpha)$*, i.e.,* $N(x) \subset B(p_i, 2\alpha)$*.* □

2.1.3 Distributing the algorithm

The distributed algorithm can now be implemented through the following map- and reduce steps.

Map For every $p_i \in P = \{p_1, \ldots, p_n\}$ let $P_i = B(p_i, 2\alpha) \cap P$. For $i = 1, \ldots, n$, compute the α-flow-complex for P_i. This can be done by computing the whole flow complex, i.e., the ∞-flow complex, for P_i and removing all critical points with distance value larger than α.

Reduce Let $G = (V, E)$ be the graph whose vertex set is $V = [n] = \{1, \ldots, n\}$ and whose edge set is $E = \big\{(i, j) \in [n] \times [n] \,|\, P_i \cap P_j \neq \emptyset\big\}$. Combine the α-flow-complexes for the sets P_i by traversing the connected components of the graph G in a breadth-first manner. Note that the α-flow-complex is itself a graph, namely a Hasse diagram. The combination of two α-flow-complexes is achieved by identifying all common vertices in the respective Hasse diagrams.

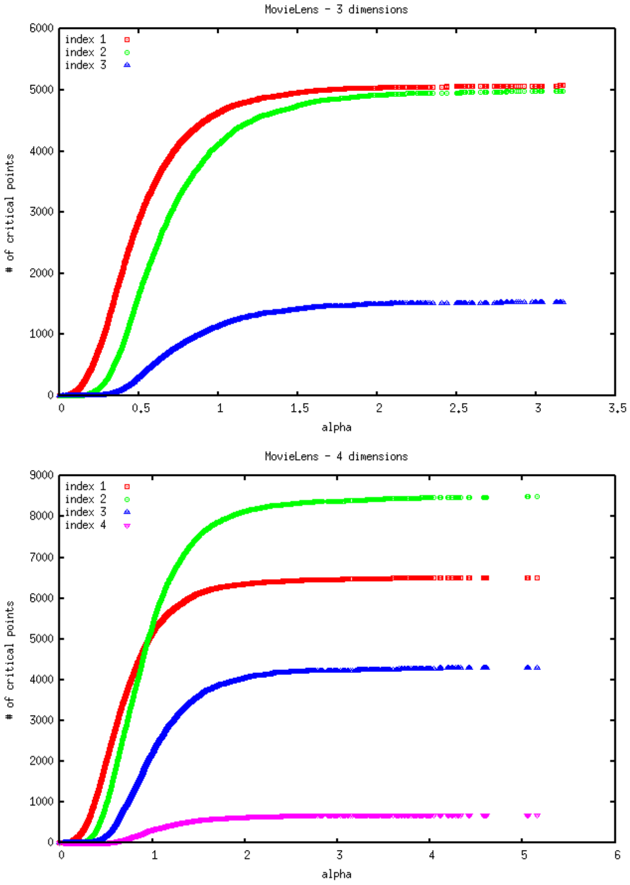
Theorem 5. *The distributed algorithm that comprises the map- and reduce step computes the* α*-flow-complex of* $P = \{p_1, \ldots, p_n\}$*.*

Proof. We need to argue that the algorithm finds all critical points of the distance function h and connects them in the right way. By Observation 2 the α-flow-complex of P_i does contain any critical point x of the distance function h with $p_i \in N(x)$ whose distance function value is at most α. Hence, any critical point of h with distance value at most α is contained in the union of the α-flow-complexes of the sets P_i, where they are also connected in the right way. □

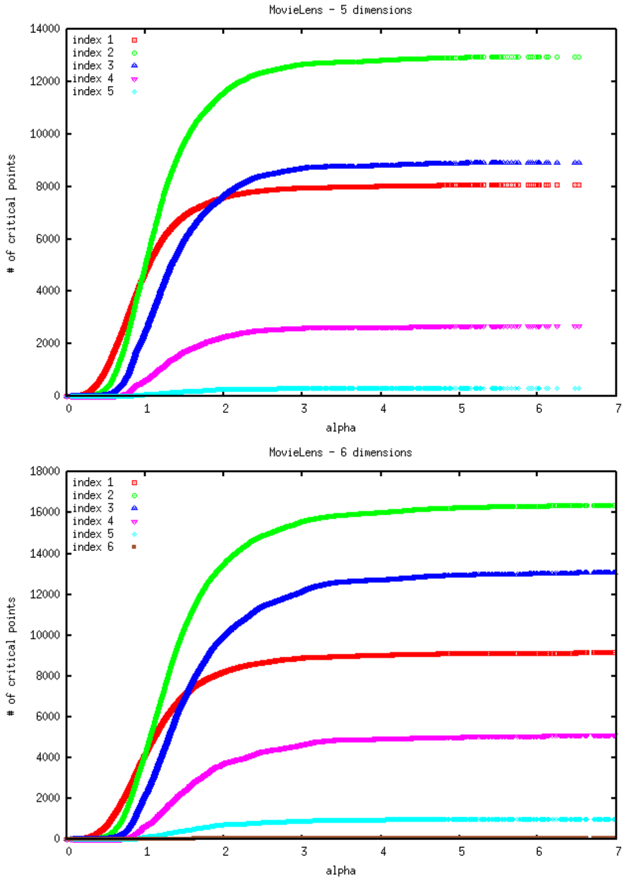
2.1.4 Computing α-flow-complexes

We have also computed the whole α-flow-complex filtration of the data sets MOVIELENS, BODY DIMENSIONS and ECOLI that we describe in the following. These data sets either reside or can be embedded, respectively, in medium dimensional Euclidean space. The α-flow-complex filtration provides us with a multiscale analysis of these data sets that is summarized in Figures 2.1 and 2.4, where the cumulative number of critical points is shown for each possible index.

MovieLens The MOVIELENS 100k data set [33] was collected by the GroupLens Research Project at the University of Minnesota. It consists of 100,000 ratings from 943 users on 1,682 movies. The data set can be viewed as an incomplete matrix that is indexed by the users and the movies, respectively, where the matrix entries are the ratings. Therefore, the MOVIELENS data set is not a point cloud data set itself, but it is straightforward to derive point clouds for the movies and for the users, respectively, from the completed ratings matrix using principal component analysis. We used a technique by Bell et al. [2] (called ComputeNextFactor) for completing the ratings matrix that at the same time computes a low dimensional spectral embedding for the movies. Using this technique we created a



(a) Dimension three (top) and four (bottom)



(b) Dimension five (top) and six (bottom)

Figure 2.1: The number of critical points of the α-flow-complex as a function of α for the first three to six dimensions for the MovieLens data set. Note that in dimension d we can only have critical points of index up to d.

14-dimensional embedding of the 1682 movies. When visualized using scatter plots, see Figure 2.2, it can be seen, that the trailing dimensions are correlated with the leading three dimensions and thus contribute less geometric information than the leading dimensions.

We therefore computed α-flow-complexes only for the data sets in three to six dimensions. Figure 2.1 shows the number of critical points as a function of the value α. The functions look like expected, namely we observe a fast increase in the number of critical points up to a threshold value for α. Beyond the threshold value the number of critical points stays almost constant. Note that the threshold value increases with the dimension from ≈ 2 in three dimensions to ≈ 3 in six dimensions. This increase is expected since the distances between the points that represent the movies also increase with the dimension. Another interesting observation is the following: the plots in Figure 2.1 for five and six dimensions indicate that the intrinsic dimension of the data set is four since almost no critical points of index six and only very few critical points of index five can be found.

Simplification We can apply apply the same simplification scheme as described in section 1.4.2 to α-flow-complexes. Hence in Figure 2.3 we also show a plot of the cumulative number of critical points of simplified α-flow-complexes for increasing values of α and a fixed value for the threshold parameter t. Note that the α-flow-complexes are much sparser than their non-simplified counterparts. Also, the critical points that appear for larger values of α are better separated from the critical points that appear already for small values of α, i.e., the α gap has widened, which is important for distinguishing between the support and the medial axis of a probability measure, see Section 2.1.

Body Dimensions The Body Dimensions data set [35] contains 507 points with 21 attributes (excluding four nominal attributes) that represent measurements of the human body. The first nine attributes are skeletal measurements, whereas the latter 12 are girth measurements. The first five skeletal measurements regard the body's torso. Here we restrict ourselves to these first five dimensions. Figure 2.4 (on the left) shows the number of critical points as a function of the value α for this data set. The function again looks like expected. We observe a fast increase in the number of critical points up to a threshold value for α which is ≈ 2.

Ecoli The Ecoli data set [40] contains 336 points in eight dimensions. From these dimensions we removed two binary attributes and the sequence number and considered only the remaining five metric (Euclidean) dimensions. Figure 2.4 (on the right) shows the number of critical points as a function of the value α for this data set. Again, this function looks like expected. The threshold value for α here is ≈ 1.75.

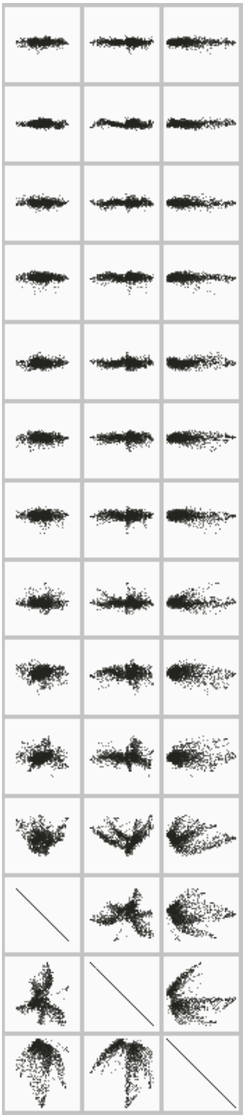


Figure 2.2: First three rows of the scatter plot matrix of the 14-dimensional embedding of the MovieLens data set.

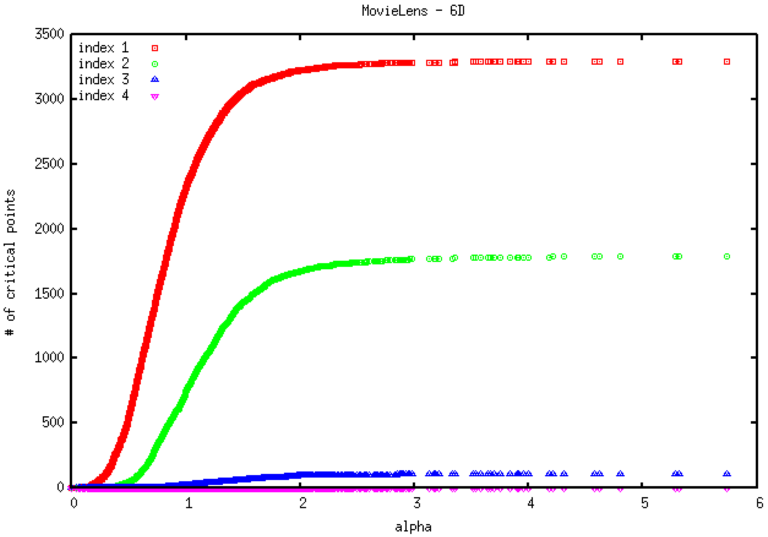


Figure 2.3: A plot of α-flow-complexes for increasing values of α of the MovieLens data set after simplification using the threshold parameter value $t = 1.048$.

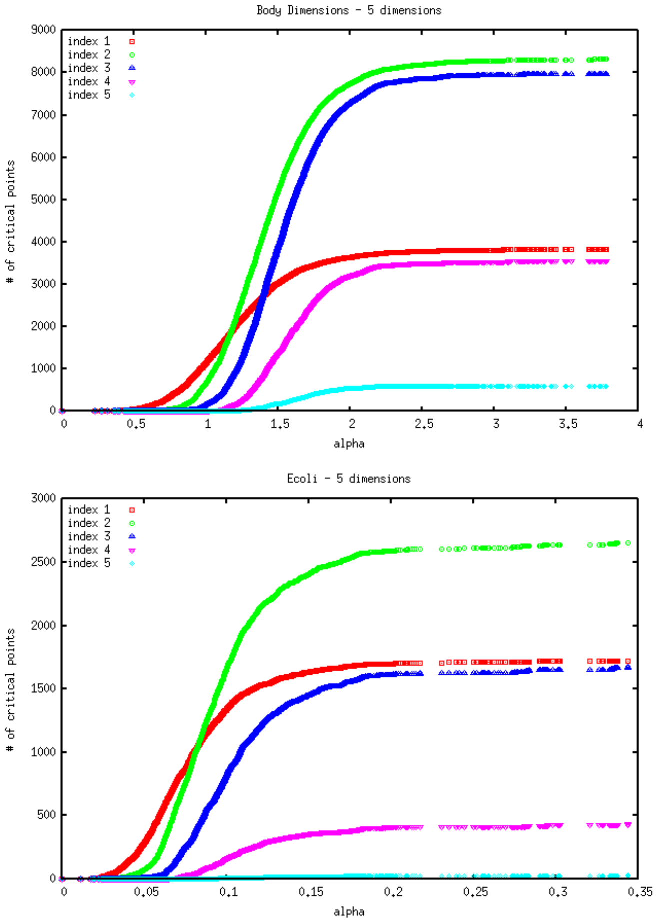


Figure 2.4: The number of critical points of the α-flow-complex as a function of α for the Body Dimensions data set restricted to the first five dimensions (at the top) and the ECOLI data set (at the bottom).

2.2 Visualizing Empty Space: Sclow Plots

Finally, we would like to introduce a novel approach of visualizing high-dimensional Euclidean data by focusing on areas within the embedding space with few or no data points at all. As we will show, the Up-flow operation from section 1.3.2 will provide us with sufficient flexibility to implement a visual analytics tool that is both expressive and computational efficient.

Understanding the distribution of a data source is at the heart of most data analysis methods. Typically, the data source is only accessible through a finite sampling and many techniques have been designed to understand the distribution from the sampling. An important characteristic of a sampling are clusters. The meaning of the term cluster is fuzzy, but intuitively refers to sets of sample points that are close to each other while being well separated from the rest of the sample points. The importance of clustering is reflected in the plenitude of different clustering algorithms that have been developed over the years. Another characteristic of the sampling that has received much less attention are regions in the domain of the data generating process that contain no, or only very few data points. We want to refer to such regions as empty space or voids. Abstractly, one can view clusters as separated regions in the domain where sample points are generated with high probability, whereas voids are regions where sample points are generated with low probability or even probability zero. From a practical point of view, knowing where the data generating mechanism does not generate any points can be very valuable information, think for example of patients with certain characteristics that do not develop a condition.

Here we are still considering the special case of distributions on Euclidean space. A sampling from such a distribution is a point cloud. Visualizing Euclidean point cloud data is a special case of visualizing high dimensional data. The two most prominent classes of techniques for high-dimensional data visualization are parallel coordinates and scatter plot matrices. Both techniques are primarily used for detecting clusters in the data. The following technique focuses on augmenting scatter plots for also detecting voids. We augment the scatter plots by the flow lines of the gradient vector field of the distance function to the point cloud. If $P = \{p_1, \ldots, p_m\} \subset \mathbb{R}^n$ is the point cloud, then the distance function(see Section 1.2) is given as

$$d_P : \mathbb{R}^n \to [0, \infty),\; x \mapsto \min_{p \in P} \|x - p\|.$$

A point on a flow line is characterized by following the direction of steepest ascent of the distance function d_P. The point either flows to ‘infinity’ or ends in a local maximum of the distance function. Our approach builds on the observation that the distance function attains local maxima deep within voids. But as we will describe later, the flow lines that continue to infinity can provide even more useful information about the empty space, i.e., regions with no sample points, and the interface between the support of the distribution and the empty space. This interface is the boundary of the distribution.

This approach is inherently interactive. Since there can be many flow lines that correspond to different aspects of empty space, showing all of them just leads to non intelligible visual clutter. Thus the user has to filter the flow lines. Flow lines can be distinguished by being either finite or infinite. The existence of many finite flow lines is a good indicator for the full dimensionality of the point cloud. It turns out that filtering infinite flow lines by their combinatorial length, i.e., number of segments, separates different features very well without requiring much effort from the user.

2.2.1 Context

A comprehensive overview of high-dimensional visualizations has been given by Grinstein et al. [32], who highlight the important distinction between a visualization of high-dimensional data and visualizations that are high-dimensional themselves. Among the most popular visualizations for high-dimensional data are parallel coordinates and scatter plot matrices. Other multi-dimensional visualization techniques can be more expressive, however also more complex and difficult to use correctly. This is especially true if these techniques involve preprocessing of the data, for example by multi-dimensional scaling, principal component analysis or its modern variants like t-SNE [48]. Still, it could be an interesting avenue for future research to combine the aforementioned projection techniques with our flow line approach.

Friendly and Denis [23] have documented the origins and interesting history of scatter plots. Their highly recommended historical account traces scatter plots back to Francis Galton (1822-1911). A 2D scatter plot projects pairs $(x_i, y_i) \in X \times Y$ as dots on an axis-aligned grid, where the values of X and Y are identified along the axes, respectively. Scatter plot projections, both 2D and 3D, are predominantly used for correlation/association analysis since they allow to identify the direction of correlation (which might be positive or negative), its intensity, as well as the existence of outliers.

The importance of scatter plots for cluster analysis has been demonstrated impressively by Hertzsprung and Russel in 1910, who used scatter plots to group stars into categories like White Dwarfs, Giants, and Supergiants [36, 45].

Adding additional information to scatter plots also has a long history. Already in 1984 Cleveland and McGill [13] have discussed how to enhance scatter plots by adding further graphical information. Chan et al. [11] augmented scatter plots with flow information representing an approximation of the gradient around each data point. Their technique supports the perception of local variations in the underlying distribution and aids the user in analyzing clusters of the data set. A related technique, called continuous scatter plots, has been introduced by Bachthaler and Weiskopf[1] for the case when the input data is not a collection of discrete points, but a continuous field on a continuous domain.

The arrangement of scatter plots into a matrix (SPLOMs) was first published by J. Hartigan [34]. The rows and columns of a scatter plot matrix are indexed by the dimensions of the data set, and each entry of the matrix shows the scatter plot of the corresponding dimensions. Thus, in general, for a n-dimensional data set, there will be n^2 individual scatter plots.

Elmqvist et al. [21] describe interactive methods to explore multi-dimensional data using scatter plots, or more precisely scatter plot matrices. The methods include navigating within a scatter plot matrix by moving from a scatter plot to an adjacent scatter plot, where scatter plots are adjacent when they share a common dimension. Moving to an adjacent scatter plot can be animated through a rotation in three dimensional space. Other interaction techniques include reordering the dimensions within the scatter plot matrix, and querying the multi-dimensional data set by bounding volume sculpting.

Nan Cao et al. [8] have developed DICON for interactive visual analysis of multi-dimensional clusters. They also provide a thorough discussion of the value of scatter plots for visual cluster analysis. DICON contains scatter plots among other visualizations.

A very successful technique for the analysis of higher dimensional data are parallel coordinates [38] that, just like SPLOMs and most other techniques for visualizing high dimensional data, map the points into 2D space. Parallel coordinates achieve this mapping by serializing the dimensions. They are often used in addition to scatter plot matrices for example in the popular GGobi tool [47].

Finally, we also want to mention topological techniques for the analysis of point cloud data, e.g. the computation of the simplified Hasse-diagram of the flow-complex in section 1.3.7 or persistent homology [20], but also want to point out already here that it is not suited for detecting prominent empty space features like cavities. In a nutshell, persistent topology identifies topological features that are persistent over a filtration of some cell complex. The filtration is controlled by some scale parameter and incrementally adds the cells of the complex to the initially empty complex. Persistent features then need to survive over a long enough scale interval. For point cloud data analysis one needs to define first a good cell complex on the point cloud, for instance the Delaunay triangulation or the flow complex of the point cloud, and second a filtration of the complex, e.g. the alpha shape filtration of the Delaunay

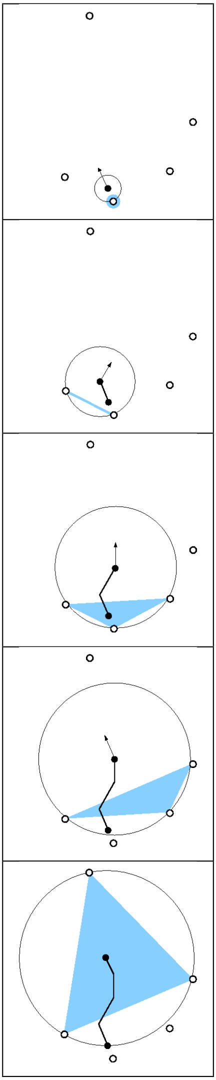


Figure 2.5: Example flow line in two dimensions. Shown on the left is the starting point x of the flow line together with the starting flow direction that is determined by the smallest enclosing ball of $N(x)$, which is here just a single data point. Then, shown from top to bottom are the points where the set $N(x)$, and thus the direction of flow, changes. The flow line ends (shown on the right) in a local maximum of the distance function, i.e., x is contained in the convex hull of $N(x)$. Note that smallest enclosing balls of more than one point in n dimensions are always n-dimensional. Here only the boundaries (circles in two dimensions) of the balls (disks in two dimensions) are shown.

triangulation [17], where alpha is the scale parameter. Topological features that can be detected by persistent homology are for instance connected components (zeroth homology) and topological voids (n-th homology). The point clouds that we consider in this chapter do not exhibit persistent voids. This should come as no surprise, because almost all natural data sets have low intrinsic dimension, i.e. dimension much smaller than n, but persistent voids (generators of the n-th persistent homolgy group) require a hypersurface (dimension $n-1$) that encloses the void. Actually, for similar reasons only the low dimensional persistent homology groups of the point cloud data that we consider here should be non-trivial. In that sense persistent homology as a technique is closer to clustering than to identifying empty space.

As already hinted at the beginning of this section, we want to note that although the technique that we introduce here is inspired by the algorithm to compute the flow complex 1.3, we do not need to compute the whole flow complex, but only individual flow lines which can be done much more efficiently.

2.2.2 Distance function and flow lines

In this section we briefly review basic concepts for computing flow lines based on the definitions introduced in section 1.2.

The flow induced by the gradient vector field ∂_{d_P} is a mapping

$$\phi : [0, \infty) \times \mathbb{R}^n \to \mathbb{R}^n$$

defined by the equations $\phi(0, x) = x$ and

$$\lim_{t \downarrow t_0} \frac{\phi(t, x) - \phi(t_0, x)}{t - t_0} = \partial_{d_P}(\phi(t_0, x)).$$

The set $\phi(x) = \{\phi(t, x) \,|\, t \geq 0\}$ is called the flow line of the point x. Flow lines are piece-wise linear curves.

The flow line starting at $x \in \mathbb{R}^n$ can be computed as follows, see also [24, 27]: compute the set of nearest neighbors $N(x)$ and the center $c(x)$ of the smallest enclosing ball of $N(x)$. Move x along the ray from $c(x)$ in the direction towards x until a new point from P enters $N(x)$. Now, either x is contained in the convex hull of $N(x)$, in which case x is a critical point and the flow line ends here, or the direction of the flow needs to be updated. It is possible that no new point enters $N(x)$ while moving along the ray from $c(x)$ to x. In this case the flow line is infinite. Hence, a flow line either ends in a local maximum of the distance function, or continues to infinity. For completeness, we restate that a flow line might also end in saddle point of the distance function, but that occurs only with probability 0, i.e., when choosing x uniformly at random from some open subset of $\mathbb{R}^n$ with compact closure. The computation of a flow line is shown conceptually in Figure 2.5.

For a better understanding of why flow lines help to detect voids in point cloud data, it is helpful to note that the concepts of the distance function and its gradient vector field can be generalized straightforwardly to arbitrary compact subsets of $\mathbb{R}^n$. The critical points of the distance function to some compact subset $C \subset \mathbb{R}^n$ are the center points of maximal empty balls that are contained in convex hull of their contact points on the boundary of C, or formally, let $x \in \mathrm{conv}(C) \setminus C$ and

$$N(x) = \{p \in C \;:\; \|x - p\| = d(x, C)\},$$

then x is critical, if $x \in \mathrm{conv}(N(x))$. The set of centers of maximal empty balls, or equivalently the set of points in $\mathrm{conv}(C) \setminus C$

that have least two closest points in the boundary of C, is the medial axis of $\mathrm{conv}(C)\setminus C$, see [4]. The medial axis lies centrally within $\mathrm{conv}(C) \setminus C$ and hence is often used as a skeleton for this set, or at least as a starting point to define and compute a skeleton. Thus, by definition, critical points of the distance function are contained in the medial axis which itself lies centrally in the complement of the compact set C, i.e., in the empty space.

Assume that the point cloud P is drawn from some finite probability distribution μ on $\mathbb{R}^n$. The support of μ, i.e., the set

$$\mathrm{supp}(\mu) \;=\; \{x \in \mathbb{R}^n \,|\, \mu(B(x,r)) > 0 \text{ for all } r > 0\},$$

is a compact subset of $\mathbb{R}^n$. It can be observed that the critical points of the distance function d_P are either close to $\mathrm{supp}(\mu)$ or close to the medial axis of $\mathrm{conv}(\mathrm{supp}(\mu)) \setminus \mathrm{supp}(\mu)$. The two types of critical points can be distinguished essentially by their distance values, i.e., the value of d_P is small at critical points that correspond to $\mathrm{supp}(\mu)$ compared to the value of d_P at critical points close to the medial axis. The latter type of critical points represents the voids and some parts of the empty space of the distribution. Note that $\mathrm{supp}(\mu)$ does not need to be full dimensional. The existence of local maxima of d_P indicates that the distribution is at least locally full dimensional, and the absence of any local maxima with small distance function values indicates that the support of the distribution is lower dimensional. The collection of all flow lines that end in the same local maximum of the distance function illustrates the extent of the void that is represented by the local maximum.

So far we argued that the medial axis of

$$\mathrm{conv}(\mathrm{supp}(\mu)) \setminus \mathrm{supp}(\mu)$$

is a good representation for the voids of $\mathrm{supp}(\mu)$, but that is not necessarily true for the local maxima of the distance function close to the medial axis, simply because they might not exist, see Figure 2.6. That is where infinite flow lines become important. An infinite flow line in the continuous case,

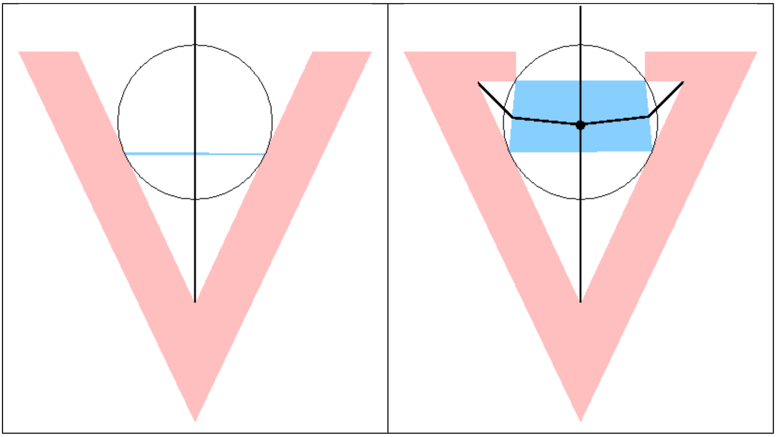


Figure 2.6: Voids do not need to be represented by critical points of the distance function. Shown are two similar compact sets C that both have a significant void, i.e., empty space in $\mathrm{conv}(C) \setminus C$. For both sets a maximal empty ball is shown together with the convex hull of its closest points in the boundary of C. The center of the ball is a local maximum of the distance function only for the set shown on the right, but it is contained in the medial axis for both sets.

i.e., considering not the sample points but the support of the distribution, hits the medial axis either at some finite point and then stays on the medial axis, see [30], or it hits the medial axis at infinity. Infinite flow lines that hit the medial axis at some finite point typically have many segments whereas infinite flow lines that hit the medial axis only at infinity have just one segment. Sample points from which flow lines originate that have only one segment are on (or close to the) boundary of the convex hull conv(supp(μ)). See also Figure 2.7.

2.2.3 Sclow plots

Sclow plots are augmented scatter plot matrices. Projections of flow lines are shown along with the projection of the data points. Note that the projection of a flow line is straightforward since flow lines are by definition piecewise linear, and hence also the projection is piecewise linear. The flow lines for our sclow plots originate on small spheres around the data points from which starting points are sampled uniformly at random. The number of starting points determines the resolution of the sclow plots. If the resolution is low, i.e., only a few starting points are sampled from the spheres, then it is possible to miss some of the local maxima. Increasing the resolution is computationally more expensive, though the running time only scales linearly with the number of starting points. A high resolution can also lead to visual clutter.

Different types of flow lines. To avoid visual clutter we distinguish three types of flow lines whose rendering can be individually switched on and off:

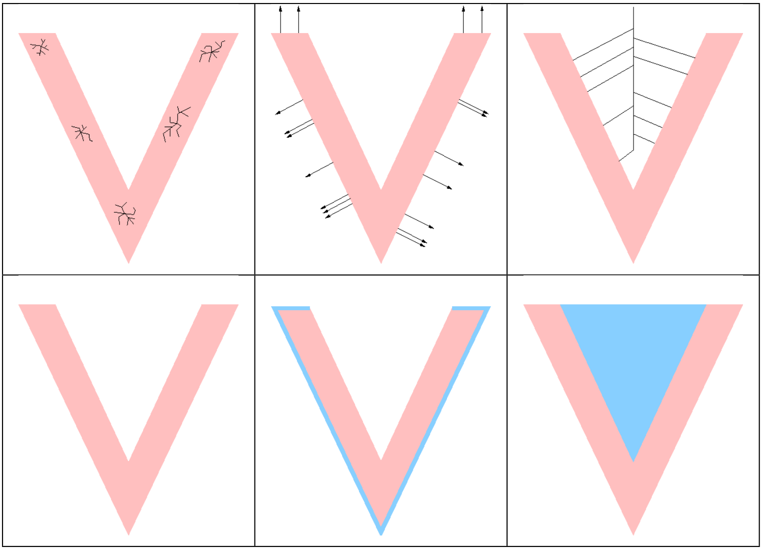


Figure 2.7: Three types of flow lines from left to right, finite flow lines that end in a local maximum of the distance function and represent the interior of the full dimensional shape shown below, infinite flow lines with only one segment that originate on the boundary of the convex hull conv(supp(μ)) shown below in light blue, and infinite flow lines with many segments that hit the medial axis at some finite point and thus represent the cavity shown below in light blue.

1. finite flow lines that end in a local maximum of the distance function,
2. infinite flow lines that have only a few segments, i.e., the number of segments is below some user defined threshold,

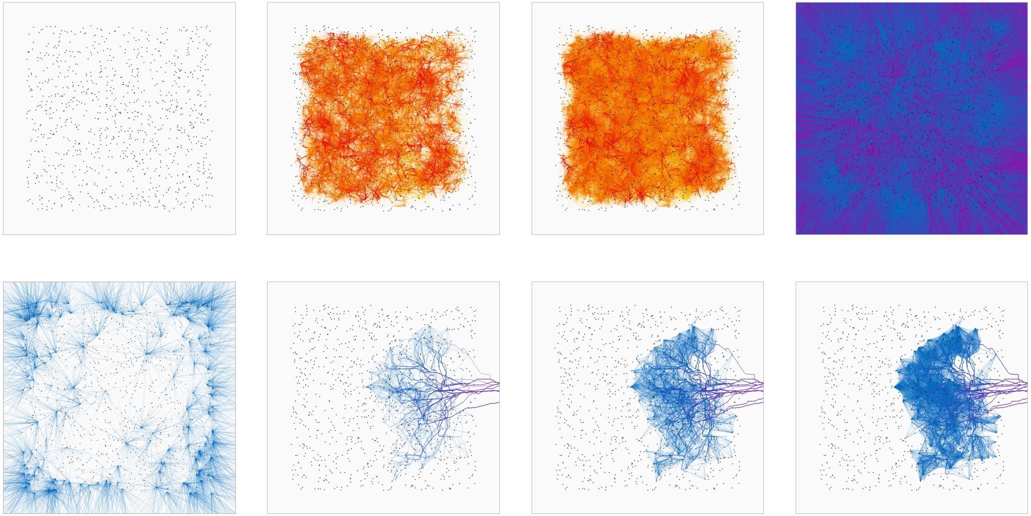


Figure 2.8: Different types of flow lines. Top, from left to right: scatter plot of the sampled cube data set, flow lines that end in finite maxima [Resolution 250], flow lines that end in finite maxima [Resolution 750], and all infinite flow lines [Resolution 750]. Bottom, from left to right: all infinite flow lines with at most three segments [Resolution 750], all infinite flow lines with at least 38 segments [Resolution 50], all infinite flow lines with at least 38 segments [Resolution 250], and all infinite flow lines with at least 38 segments [Resolution 750].

and

3. infinite flow lines that have at least as many segments as the user defined threshold.

Each type of flow line can be used to detect a different type of feature in the point cloud. The existence of finite flow lines witnesses the full dimensionality of the point cloud. The technical reason for this is that local maxima of the distance function correspond to full dimensional cells in the flow complex. Please refer to [24] for details. As explained before the conceptual difference between the two types of infinite flow lines is the following: infinite flow lines with only a few segments tend to flow to infinity without hitting the medial axis of $\mathrm{conv}(\mathrm{supp}(\mu)) \setminus \mathrm{supp}(\mu)$—they hit the medial axis at infinity, whereas the infinite flow lines with many segments hit the medial axis earlier. The latter flow lines do not in general hit the medial axis immediately, but only after passing through some small segments. Typically, they originate on or close to the boundary of $\mathrm{supp}(\mu)$, i.e., on the boundary of a cavity—a void in $\mathrm{conv}(\mathrm{supp}(\mu)) \setminus \mathrm{supp}(\mu)$ that is connected to infinity.

We illustrate the difference between the different types of flow lines in Figure 2.8. In this figure one can see that all three types of flow lines carry important information about the data set, i.e., the sampled cube in four dimensions, where a cone opening to one side has been cut out. The existence of finite flow lines shows that the data set is full dimensional and furthermore, that the data points are distributed rather uniformly, because otherwise we would see more variation in the distance function values at the finite maxima as encoded in the color scheme. The infinite flow lines with only a few segments allow to detect extreme points of the data set, i.e., data points on or close to the convex hull of the set, since many of the infinite flow lines with only a few segments originate from the extreme points. From the flow lines, one even gets an impression in which directions the points are extreme. Finally, the infinite flow lines with many segments allow to detect the cavity that results from the cone that has been cut out from the cube.

Types of flow lines and tasks. Tasks related to the analysis of high dimensional data can be characterized as (1) cluster oriented, (2) dimension oriented, or (3) novelty oriented. The first two categories have been identified by Brehmer et al. [6] who conducted interviews with ten data analysts from six application domains. Here we want to add the third category. Tasks in the first category include detecting clusters, verifying clusters, and naming clusters, see [6]. Tasks in the second category include identifying the intrinsic dimension of data, and naming synthesized dimensions, see again [6]. We want to summarize the following tasks in the third category: Identifying outliers, finding data points in between clusters, and locating data points at the boundary of clusters.

Scatter plots augmented by flow lines support the following tasks: As mentioned before finite flow lines can be used to decide the full-dimensionality of the data. Infinite flow lines that have only a few segments can be used to identify outliers, while infinite flow lines with many segments help to locate data points at cluster boundaries, especially for complicatedly shaped, non-round clusters, and locating data points at the interface between clusters. Finite flow lines also support the latter task in the case of full dimensional point clouds. We make the link between tasks and the different types of lines more explicit in Section 2.2.4 on synthetic data sets and in Section 2.2.5 on real data sets.

Interaction. As can be seen in Figure 2.8 (top-right) it makes no sense to show all the flow lines at once, because this just leads to visual clutter. Thus the user needs to filter the flow lines depending on the task at hand. If the task is dimension inference, then only finite flow lines should be selected. For outlier detection one should show the infinite flow lines with only a few segments, and for exploring the boundaries of clusters only infinite flow lines with many segments should be displayed. We discuss the possible interactions that we implemented in more detail in Section 2.2.6.

Color scheme. Flow lines can be colored to convey distance function value information. By definition, the distance function value, i.e., the value of d_P, is strictly increasing along a flow line. We use a color gradient along the hue axis to encode the change of d_P along a flow line. We provide the user with a set of distance metrics that determine the maximum value for the data set at hand, and therefore define a scale. The most intuitive measures, among others, are the actual maximum distance as given by the distance function, or the maximum number of segments among all the flow lines that have been computed. We assign the minimum value (usually zero) to one hue, and the so determined maximum value to another hue in such a way, that both hues are neighbors on the color circle. The colors for all distance values in between are determined by linear interpolation along the hue axis. By doing so, we avoid what is known as *rainbow effects*, that—although visually appealing in an artistic way, can be misleading and thus result in a false understanding of the data set, because the human visual system perceives hue information as categorical information rather than continuous function values. By using neighboring hues we express the increase in distance in the fashion of a heat map. We allow to choose two sets of hue-pairs, one for the finite flow lines and the other one for the infinite flow lines. That allows to distinguish these two types of flow lines visually even when they are displayed at the same time.

2.2.4 Synthetic data sets

In this section we use synthetic data sets for highlighting different features that sclow plots can reveal.

Parabola

The purpose of this data set is to show how sclow plots represent a cavity, i.e., a connected component in $\mathrm{conv}(\mathrm{supp}(\mu))\setminus\mathrm{supp}(\mu)$ that is connected to infinity. Cavities can be considered also as boundaries of non-round, complicatedly shaped clusters. Since flow lines as we compute them here originate close to data points, they can be used to locate data points on the boundary of such clusters. Hence detecting cavities can be linked to the task of detecting cluster boundaries.

The points of this data set in four dimensions have been generated as follows: the first coordinate x_1 has been sampled uniformly at random from the interval $[-1, 1]$, the second coordinate has been computed as $x_2 = x_1^2$ and then been slightly perturbed, the coordinates x_3 and x_4 have been sampled again

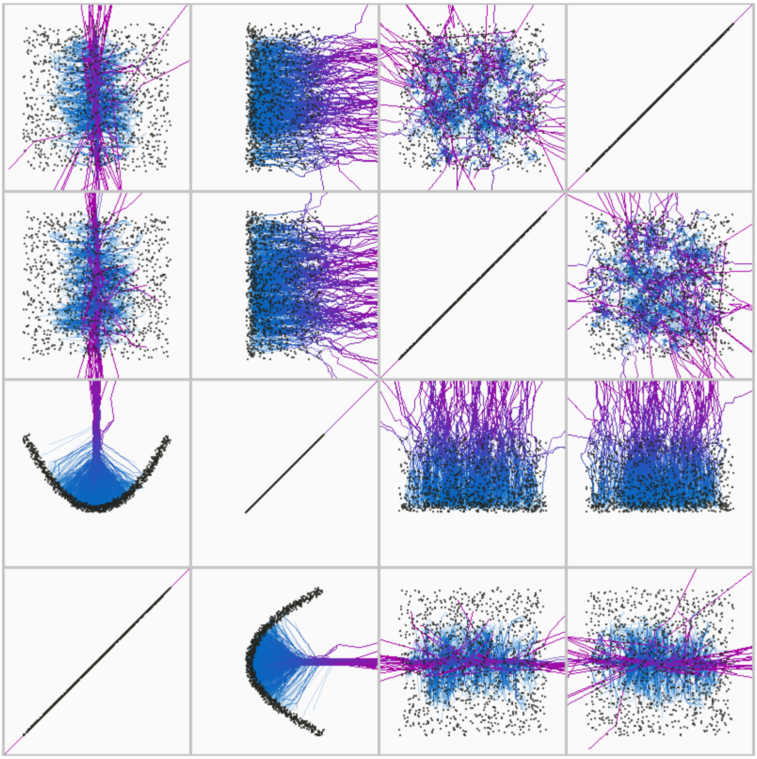


Figure 2.9: A scatter plot matrix for the sampled shape S together with infinite flow lines that have many segments. [Resolution 50]

uniformly at random from $[-1, 1]$. The unique cavity of the three dimensional shape

$$S = \{x \in \mathbb{R}^4 \,:\, x_2 = x_1^2,\, x_1, x_3, x_4 \in [-1, 1]\}$$

from which the points are sampled with some perturbation corresponds to a three dimensional sheet of the medial axis, but not to finite critical points. The sheet of the medial axis can be easily recognized in the sclow plots, see Figure 2.9. The cavity itself is also clearly visible in the pure scatter plot matrix. This changes when we transform the original data set, e. g., by just rotating it in space such that the medial axis sheet of the cavity is no longer aligned with a coordinate plane. Still, the cavity can be easily read off from the slow plot, see Figure 2.10.

The problem becomes even more pronounced when we transform the data set non-linearly, e. g., by the transformation

$$(x_1, x_2, x_3, x_4) \mapsto (x_1x_2, x_2x_3, x_3, x_4).$$

The void remains visible in the sclow plot whereas it is hard to detect from the scatter plot matrix, see Figure 2.11.

Two Gaussians

On this data set we want to demonstrate how sclow plots represent a cavity that separates two clusters. That is, this data set serves as an example for using infinite flow lines with many segments to aid the task of locating data points at the interface of two clusters. The clusters in this case are given by two Gaussians, i.e., the points are drawn from a normal distribution $N(\mu, \Sigma)$, whose centers μ lie on diagonally opposite vertices of a four dimensional cube with side length 1. For the covariance matrix Σ we chose $\Sigma = \sigma\mathbf{1}$, i.e., the variance is equal for all four dimensions. We varied the variance σ from 1.4 to 1.8 with step size 0.2. The void that is present in this data set is schematically shown in Figure 2.12 (on the left).

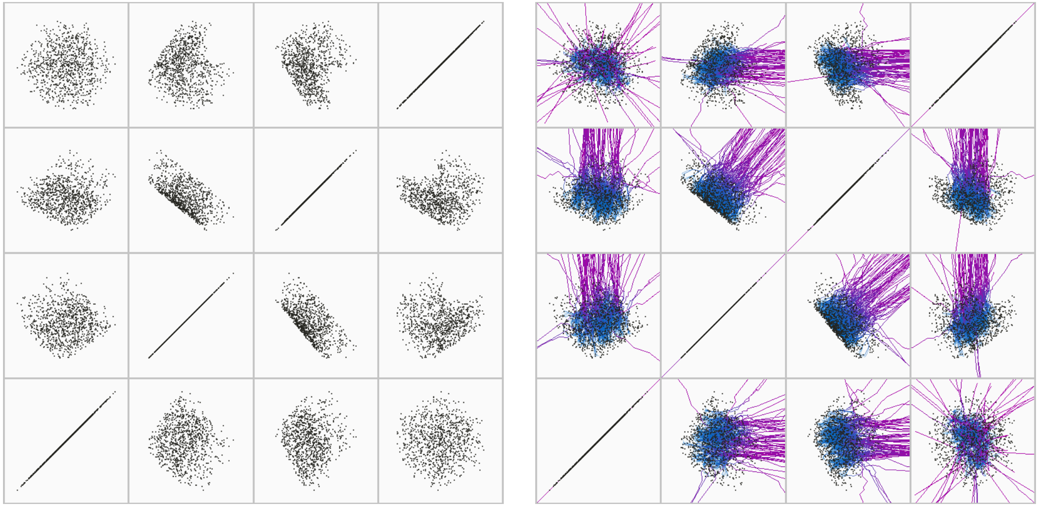


Figure 2.10: Left: A scatter plot matrix for the sampled shape S after some space rotation. Right: The same scatter plot matrix together with infinite flow lines that have many segments. [Resolution 50]

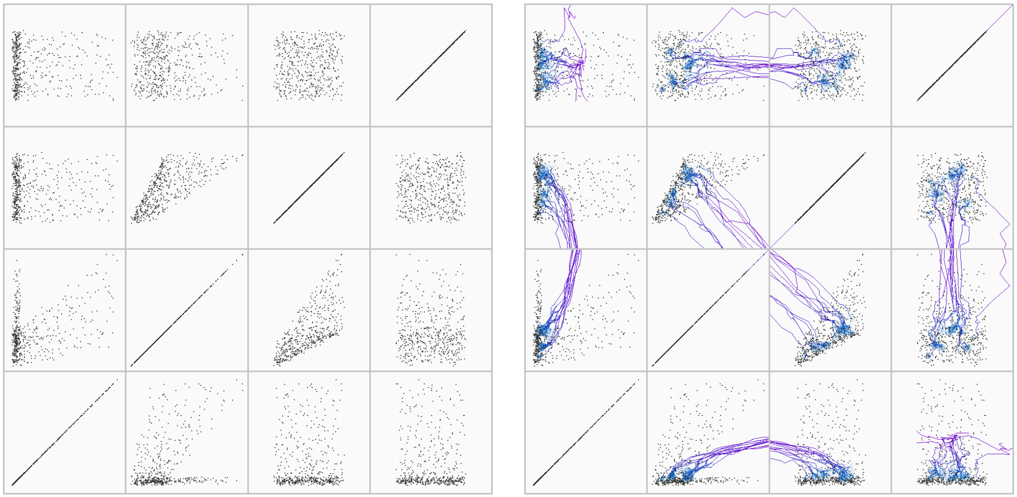


Figure 2.11: Left: A scatter plot matrix for the sampled shape S after a non-linear transformation. Right: The same scatter plot matrix together with infinite flow lines that have many segments. [Resolution 50]

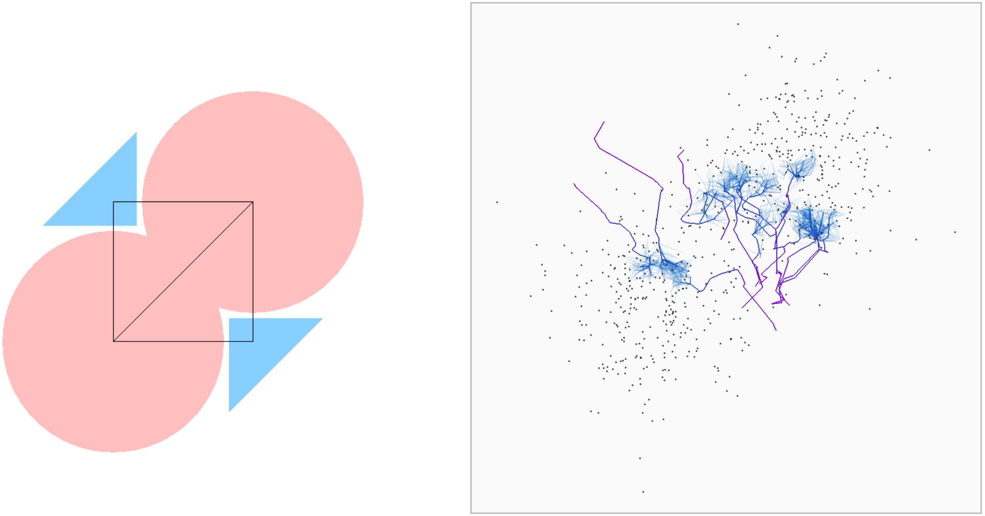


Figure 2.12: Left: The essential support of two Gaussians centered on diagonally opposite vertices of a unit cube is shown shown in pink, and the void that separates the support of the two Gaussians is shown in blue. In two dimensions as shown here that void has two components. In three dimensions it is a connected ring (torus). Right: An example in four dimensions where 300 points each have been sampled from two Gaussians with variance 1.4. [Resolution 250]

In Figure 2.13 we show sclow plots for two Gaussians with different variance. The circular void that separates the two Gaussians is visible in both plots.

One Gaussian

As an example for dimension inference with sclow plots we generated data from a single Gaussian in three dimensions that we lifted non-linearly into six dimensional space, i.e.,

$$(x_1, x_2, x_3) \mapsto \big(f_1(x_1, x_2, x_3), \ldots, f_6(x_1, x_2, x_3)\big)$$

where the f_i involve low degree polynomials and trigonometric functions, i.e., the data set is intrinsically three dimensional. In Figure 2.14 we show two sclow plots, one for the data set after two dimensions have been dropped and another one after three dimensions have been dropped. Finite flow lines only show up once three dimensions have been dropped, i.e., once we have reached the intrinsic dimension of the data set.

One Gaussian with outliers

Finally, to show how outliers, i.e., points on or close to the convex hull of a data set, can be detected we have sampled points from a standard Gaussian in four dimensions and additionally a few points on a large sphere with radius 2.5 that is centered at the mean of the Gaussian. Figure 2.15 shows a sclow plot for this data set where only finite and infinite flow lines with one segment have been selected.

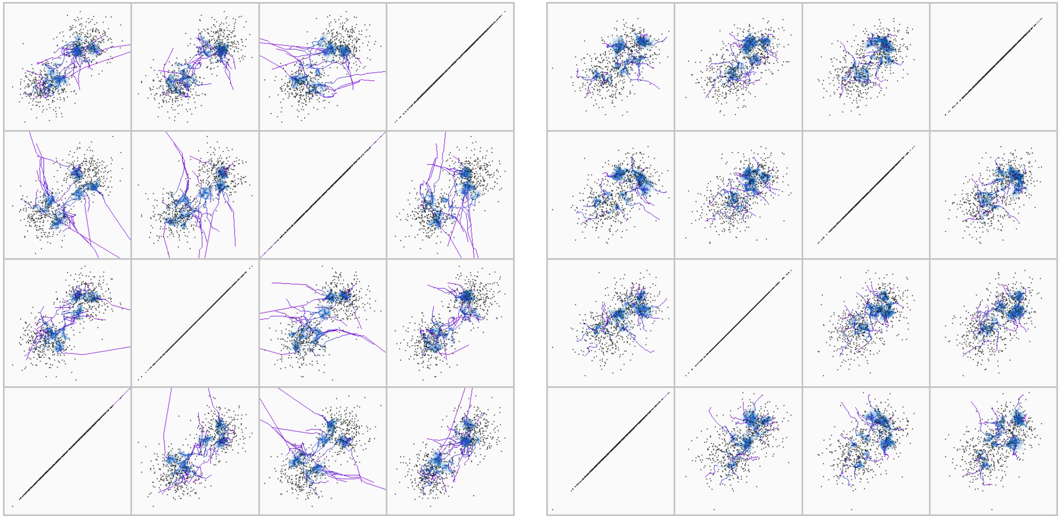


Figure 2.13: Sclow plots for two Gaussians (300 sample points each) in four dimensions. The variance of the Gaussians on the left is 1.2, whereas it is 1.8 for the Gaussians on the right. [Resolution 250]

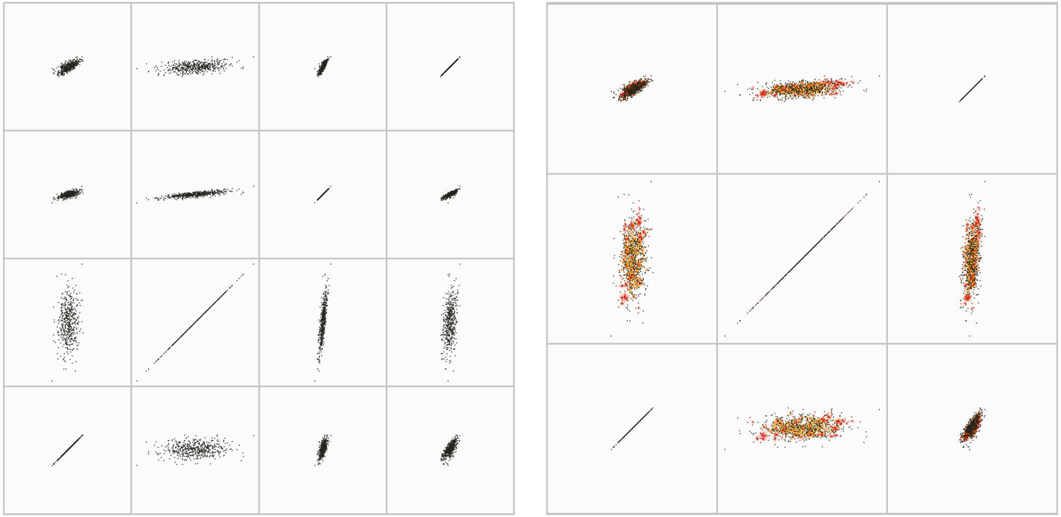


Figure 2.14: Sclow plots for 500 points sampled from a Gaussian in three dimensions that has been lifted non-linearly into six dimensions. Here only finite flow lines are shown. Shown on the left is this data set from which the Dimensions 6 and 5 have been dropped and shown on the right is this data set from which the Dimensions 6, 5, and 4 have been dropped. Note that finite flow lines only appear after three dimensions have been dropped. We should note that dropping any other three dimensions shows the same effect. [Resolution 250]

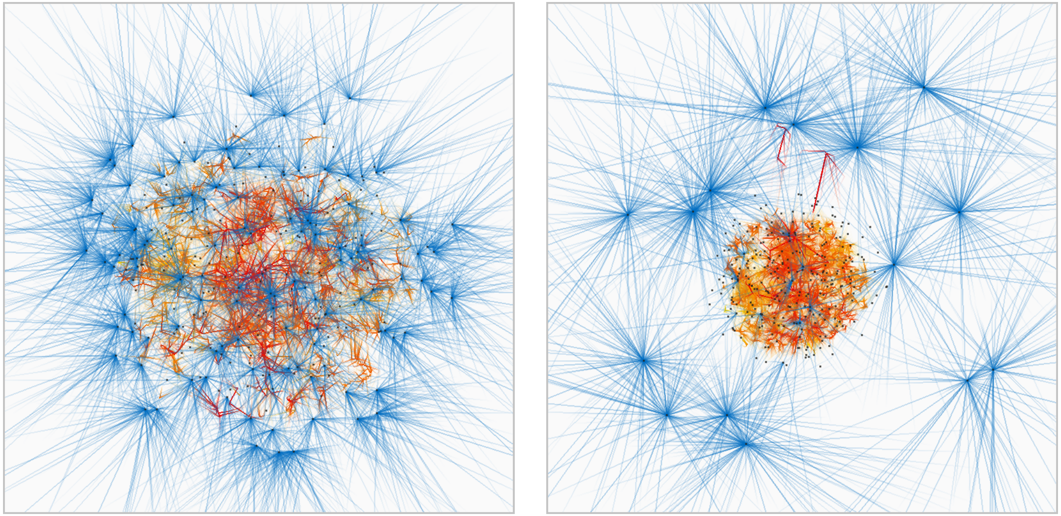


Figure 2.15: Left: A sclow plot of 250 samples from a Gaussian in four dimensions. Finite flow lines allow to sketch the essential support of the Gaussian. Infinite flow lines with few segments indicate the boundary points. Right: The same Gaussian together with 15 outliers on a large sphere. The infinite flow lines with few segments now highlight the outliers. [Resolution 350]

2.2.5 Real data sets

In order to show that the aforementioned discoveries can also be made on non-synthetic data sets, we explored two freely available, real data sets, namely the Body Dimensions and MovieLens data sets, by using sclow plots. Both data sets are amenable to an analysis through sclow plots, because they can reasonably well be interpreted as Euclidean point clouds, i.e., the different dimensions are continuous and comparable to each other. Note that using sclow plots successfully hinges on knowing the meaning of the different types of flow lines. Inference with sclow plots does not work by loading some data set together with some flow lines and starting the reasoning process from there, but it starts with a task like dimension inference, detecting outliers, or identifying data points at the interface between clusters.

Body Dimensions

Heinz et al. have published a data set [35] of body girth and skeletal diameter measurements from 507 physically active individuals, 247 men and 260 women. Here we are working with the skeletal measurements, where the first five measurements regard the torso, and the last four measurements regard the extremities.

A scatter plot of this nine dimensional data set shows the last four measurements are quite correlated. Note that a strong correlation of two dimensions essentially decreases the effective dimensionality by one. The intuition that the data set is not full dimensional is confirmed in the sense that we could not find any finite maxima even at high resolutions. The first five

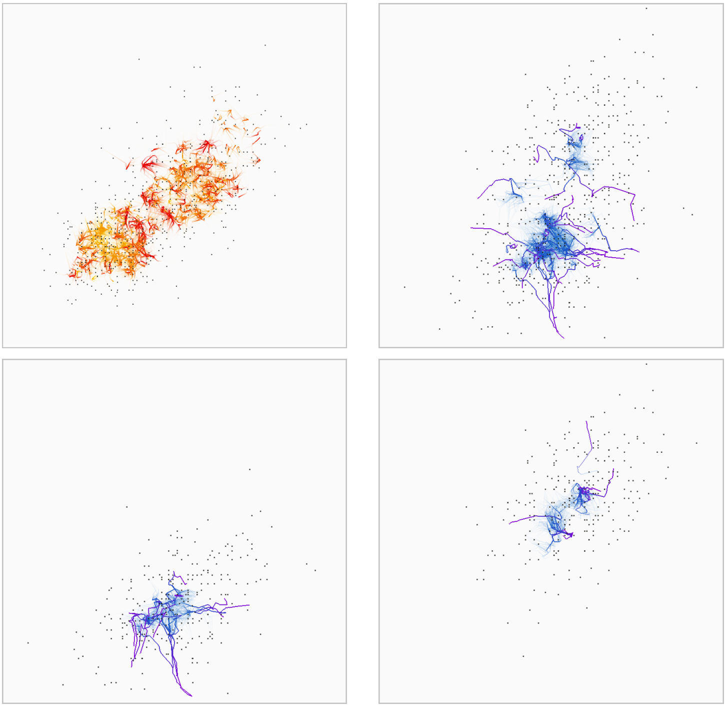


Figure 2.16: Body Dimensions data set. Top left: A sclow plot of the finite maxima in five dimensions, showing both the support (yellow-orange) and deep finite maxima (dark red) separating the male from the female cluster. Top right: A sclow plot showing infinite flow lines with many segments of the torso dimensions, computed on both the samples of the male and female cluster. Bottom left: The same sclow plot, this time computed only on the samples of the female cluster. Bottom right: The same sclow plot, this time computed only on the samples of the male cluster. [Resolution 250]

measurements that regard the torso are less correlated, and indeed the data set exhibits several finite maxima after the last four dimensions have been discarded, see Figure 2.17.

Looking at the infinite flow lines with many segments reveals two fairly large empty regions in the five dimensional data set. It is not clear, however, if these voids are contained within the male and female clusters, respectively, or if they separate the two clusters from each other. By computing sclow plots on just the male and female subset, respectively, the same deep voids become visible. This supports the assumption, that these voids are contained within the respective clusters. There are a few more voids, that are less deep and that only appear when the flow lines are computed on the combined male and female data points. Thus, these voids indicate the boundary between the two clusters as in the synthetic example of the two Gaussians, see Figure 2.16.

By selecting the infinite flow lines with only very few segments, it is possible to highlight the extreme points of the data set. This allows us to get a better understanding of the local shape in high dimensions, and to relate the voids to the boundary of the shape. A zoom into one of the sclow plots allows to detect outliers, i.e., extreme points with a relatively large spread in the flow line angles. See Figure 2.18.

MovieLens

The MovieLens 100k data set was collected by the GroupLens Research Project at the University of Minnesota [33]. It consists of 100,000 one to five stars ratings from 943 users on 1,682 movies. The MovieLens data sets are popular test sets for recommender systems. The data sets can be viewed as incom-

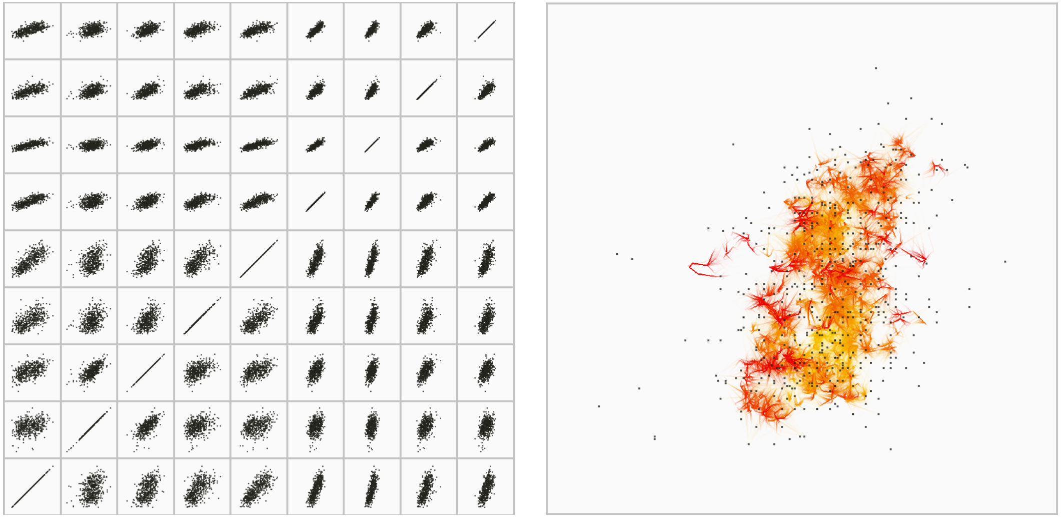


Figure 2.17: Dimension inference. Left: A scatter plot matrix of all nine dimensions showing high correlation between the last four dimensions and no visible finite maxima. Right: The same scatter plot matrix after the last four dimension have been removed—note that the data set now is full dimensional which is witnessed by an abundance of finite maxima. [Resolution 250]

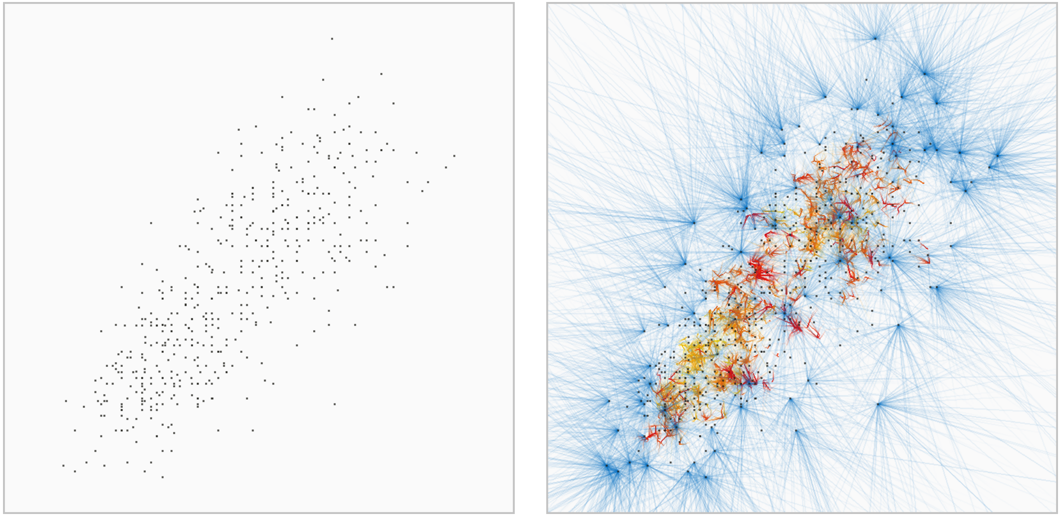
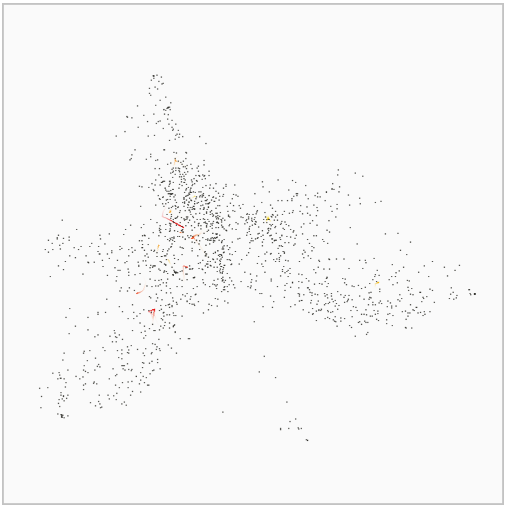


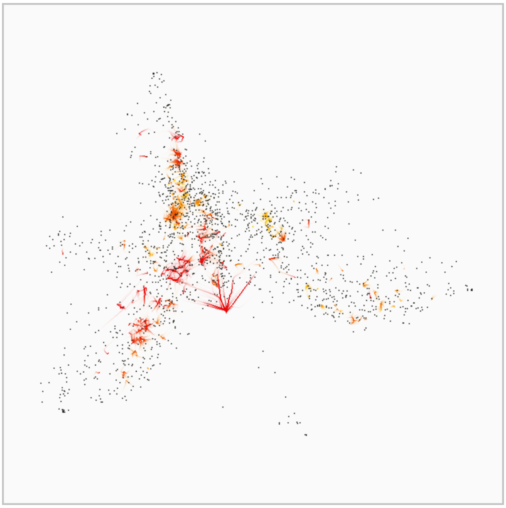
Figure 2.18: Left: A scatter plot of the torso dimensions. Right: The same scatter plot together with finite flow lines and with infinite flow lines with only very few segments. The finite flow lines allow to sketch the essential support of the data set, and the infinite flow lines highlight the extreme points. [Resolution 350]

plete matrices that are indexed by the users and the movies, respectively. The matrix entries are the ratings. A straightforward idea for a recommendation system is to complete the incomplete matrix, and to recommend movies to users whose predicted ratings are high. The MovieLens data set is not an Euclidean point cloud itself, but it is straightforward to derive point clouds for the movies and for the users, respectively, from the completed ratings matrix by using principal component analysis (PCA). We used a technique by Bell et al. [3] (called ComputeNextFactor) for completing the ratings matrix that at the same time computes a low dimensional spectral embedding for the movies, i.e., every point in the low dimensional point cloud corresponds to a movie. The Euclidean distances between the points can be considered as a dissimilarity measure for the movies.

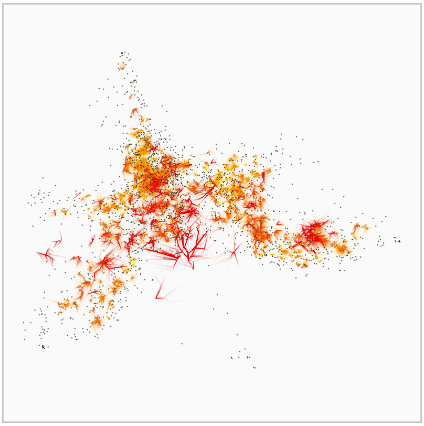
We used the technique by Bell et al. to create a 14 dimensional Euclidean point cloud from the original MovieLens 100k data set. A first visual inspection of this point cloud using scatter plots suggests that most of the similarity information is encoded in the first few dimensions. We checked this intuition in two ways, both times using sclow plots. First, we looked at sclow plots featuring finite flow lines for projections of the data set onto the first few dimensions. Figure 2.19 shows such sclow plots for the first six, five, four and three dimensions. Note that only in dimensions four and lower we can observe a significant number of finite flow lines. Which leads us to the conclusion that the essential dimension of the data set is around four. In a second analysis we discarded the first few dimensions. The resulting projections looked similar to points sampled from a single Gaussian, compare Figures 2.20 and 2.15,



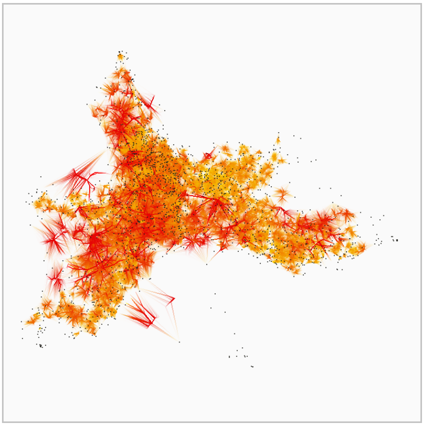
(a) projection onto the first six dimensions



(b) projection onto the first five dimensions



(c) projection onto the first four dimensions



(d) projection onto the first three dimensions

Figure 2.19: Sclow plots featuring finite flow lines for projections of the 14 dimensional MovieLens data set. Finite flow lines stop to appear in large numbers in five dimensions. [Resolution 300]

whereas projections on the first few dimensions exhibit a much richer geometric structure, compare for example Figures 2.20 and 2.21. The sclow plots like in Figure 2.20 suggest that the intrinsic dimension of the “Gaussian noise” is around six.

We also explored the geometry of the four dimensional projection of the data set. Figure 2.21 shows a sclow plot of the now four dimensional data set together with infinite flow lines with many segments. This flow lines aid a better understanding of the shape of the data set which bears some similarity to the parabola data set from Section 2.2.4. Especially, a well pronounced cavity becomes clearly visible. Looking at sclow plots that exhibit infinite flow lines with only a few segments confirms this interpretation of the geometry of this data set.

Further analysis revealed that the data points located at the tip of the cavity correspond to well known movies (blockbusters), whereas points in the extremities correspond to niche movies that are not very popular among the mainstream of users. Also, the extremities seem to constitute clusters of movies from similar genres.

2.2.6 User interaction

As we have pointed out before, user interaction is indispensible when working with sclow plots. Hence, the tool that we have implemented offers the following functionality: (1) Selecting the flow lines by type (finite or infinite), and specifying lower and upper bounds for the number of segments for the infinite flow lines. (2) Coloring the flow lines by type. (3) Selecting single sclow plots from a slow plot matrix to analyze them in detail. Here we follow the *focus+context* technique that is used in a many visualization systems [39, 44, 43]. This technique allows

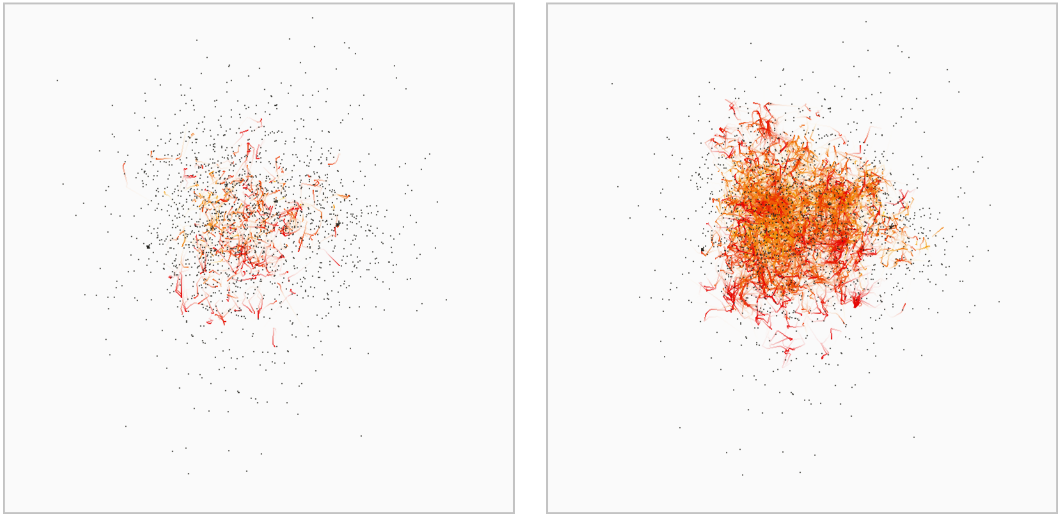


Figure 2.20: Left: A sclow plot showing finite flow lines of MovieLens data set that has been projected onto the last seven dimensions. Right: The same sclow plot for the MovieLens data set projected onto the last six dimensions shows many more finite flow lines. [Resolution 250]

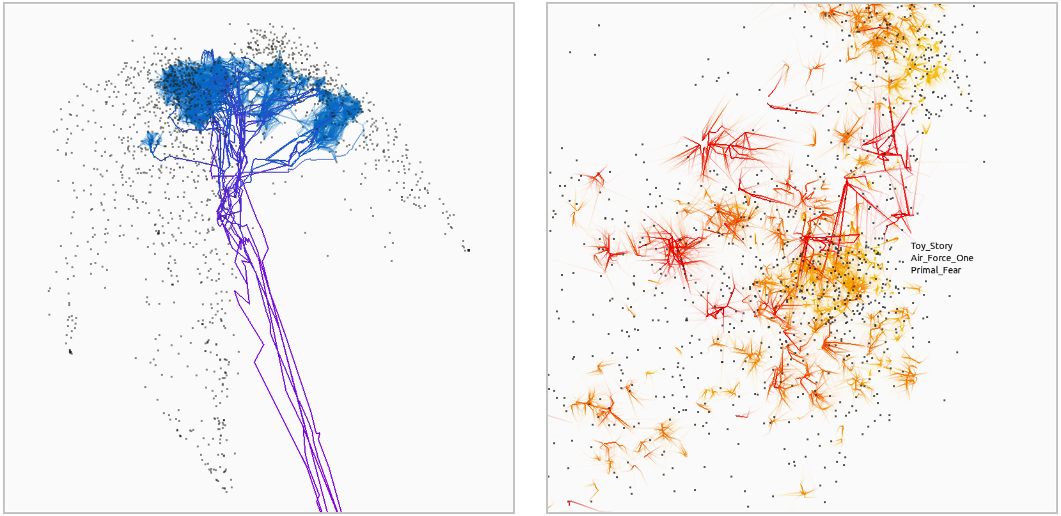


Figure 2.21: Left: A cavity in the four dimensional projection of the MovieLens data set. Right: Finite flow lines at the tip of the cavity together with movie label information. [Resolution 250]

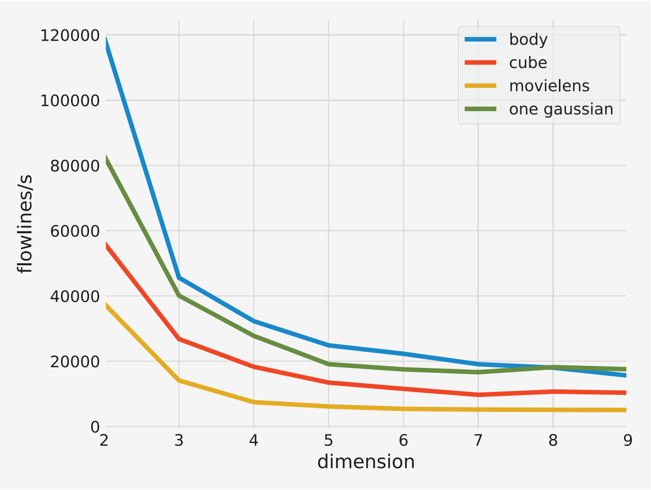
to display details where needed, while the environment remains visible. (4) Zooming into sclow plots and rotating sclow within an axis aligned two-dimensional plane, i.e., rotating a given sclow plot. (5) Rendering label information for selected data points in the sclow plots. (6) Deleting dimensions.

Interactive techniques, as they have been implemented by Elmqvist et al. [21] in their 'Rolling the Dice' approach, would be also very helpful when working with sclow plots, especially animated transitions between adjacent scatter plots.

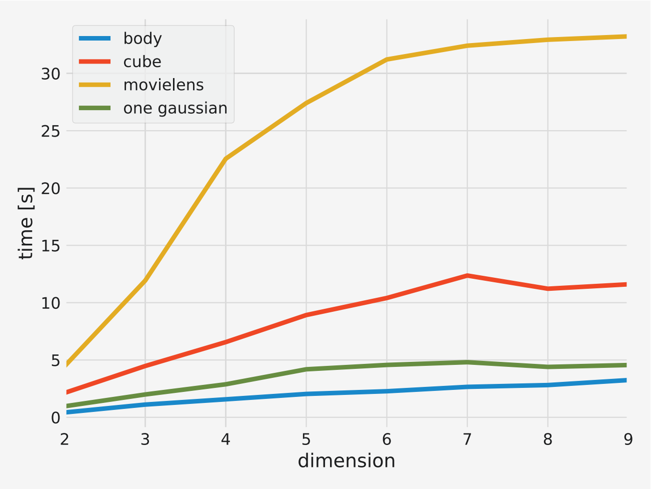
2.2.7 Scalability

In Figure 2.22 we summarize some performance data that we have collected for the data sets that have been discussed before. The results have been obtained using an Intel i7-4770 CPU with 3.40GHz on four cores.

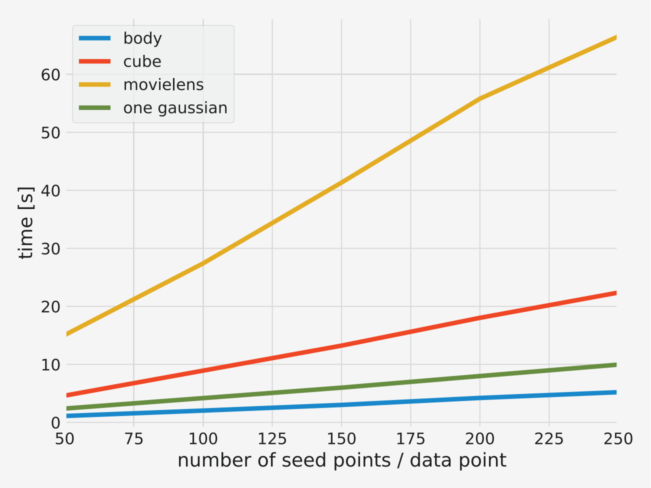
Our tool always remains interactive because flow lines at a given resolution have to be computed only once before user interaction can start, e.g., selecting flow lines of different types or zooming into sclow plots. Note that 100 flow lines per data point is a very reasonable starting point for a visual exploration. Computing these flow lines in nine dimensions for the MovieLens data set takes less than 35 seconds, see Figure 2.22b. We consider this an acceptable upstart time for our system. If a higher resolution is required, then more flow lines can always be computed progressively in the background at rate of a few thousand flow lines per second, see Figure 2.22a. The running time for increasing the resolution scales linearly with the resolution as can be seen in Figure 2.22c.



(a) Flow lines per second as a function of the dimension



(b) Time for computing 100 flow lines per data point again as a function of the dimension



(c) Compute time as a function of the number of seed points per data point

Figure 2.22: Performance data for computing flow lines

Chapter 3

Conclusions

We have presented a new algorithm for computing the flow complex. The algorithm differs from known algorithms by avoiding an explicit computation of the Delaunay triangulation of the input point set. It is also inherently parallel. We implemented the algorithm that scales well with the number of available cores on a multicore platform. Experimental results that have been obtained with the implementation show that numerical robustness is not a critical issue although the algorithm makes use of explicit constructions and does not evaluate predicates like most algorithms for computing Delaunay triangulations.

In addition and as an application, we presented an approach to sketch the support of a probability measure on $\mathbb{R}^d$ by an α-flow-complex. With high probability, the α-flow-complex is homotopy equivalent to the support of the measure for large enough samplings and good values for α. We have shown how to choose a good value for α in theory and in practice (on some real data sets).

Further we report our research on exploring empty space in medium dimensional point cloud data. For that purpose we introduced sclow plots that are scatter plots augmented by flow lines, i.e., integral curves for the gradient vector field of the distance function to the point cloud. Here we were concerned with an evaluation of the analytical power of flowlines. For this evaluation we chose scatter plot matrices, because they provide a flexible interface that allows to tweak many aspects of computation and visualization that boost our understanding of what can be done with flow lines and on how to use them. Our rationale for visualizing flowlines together with the data points as 2D projections is along the same lines, as the arguments for using 2D projections for analyzing medium dimensional Euclidean point cloud data, namely 2D projections are natural and easy to comprehend and as such impose little mental overhead for the user. We have discussed tasks on medium dimensional point clouds that can be supported by our flow line approach such as dimension inference, detecting outliers, locating data points at the boundaries of clusters and at the interface between clusters. Experiments on synthetic and real data sets indicate that sclow plots can indeed be helpful for these tasks. Future work should also explore different approaches for visualizing flow lines, e.g., by showing only aggregations of flow lines or even more abstract, inferred features like voids, dents or outliers that probably can be represented by abstract symbols. Also, our flow line technique is not bound to be used in conjunction with scatter plot matrices. Another avenue for future research is combining projection techniques like multi-dimensional scaling, principal component analysis with the projection of flow lines.

Bibliography

[1] Sven Bachthaler and Daniel Weiskopf. Continuous Scatterplots. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1428–1435, 2008.

[2] Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 95–104. ACM, 2007.

[3] Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 95–104. ACM, 2007.

[4] Harry Blum. A Transformation for Extracting New Descriptors of Shape. In W. Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, 1967.

[5] Jean-Daniel Boissonnat and Arijit Ghosh. Manifold reconstruction using tangential Delaunay complexes. In

Proceedings of the ACM Symposium on Computational Geometry (SOCG), pages 324–333, 2010.

[6] Matthew Brehmer, Michael Sedlmair, Stephen Ingram, and Tamara Munzner. Visualizing dimensionally-reduced data: interviews with analysts and a characterization of task sequences. In *Proceedings of the Workshop on Beyond Time and Errors: Novel Evaluation Methods for Visualization (BELIV)*, pages 1–8. ACM Press, 2014.

[7] Kevin Buchin, Tamal K. Dey, Joachim Giesen, and Matthias John. Recursive geometry of the flow complex and topology of the flow complex filtration. *Computational Geometry: Theory and Applications*, 40(2):115–137, 2008.

[8] Nan Cao, David Gotz, Jimeng Sun, and Huamin Qu. DICON: Interactive Visual Analysis of Multidimensional Clusters. *IEEE Trans. Vis. Comput. Graph.*, 17(12):2581–2590, 2011.

[9] Frédéric Cazals and David Cohen-Steiner. Reconstructing 3D compact sets. *Computational Geometry: Theory and Applications*, 45(1-2):1–13, 2012.

[10] Frédéric Cazals, Aditya G. Parameswaran, and Sylvain Pion. Robust construction of the three-dimensional flow complex. In *Symposium on Computational Geometry (SOCG)*, pages 182–191, 2008.

[11] Yu-Hsuan Chan, Carlos D. Correa, and Kwan-Liu Ma. Flow-based scatterplots for sensitivity analysis. In *IEEE onference on Visual Analytics Science and Technology (VAST)*, pages 43–50, 2010.

[12] Frédéric Chazal, David Cohen-Steiner, and André Lieutier. A sampling theory for compact sets in Euclidean space. In *Symposium on Computational Geometry (SOCG)*, pages 319–326, 2006.

[13] W. S. Cleveland and R. McGill. The many faces of a scatterplot. *Journal of the American Statistical Association*, 79:807–822, 1984.

[14] Luc Devroye, Laszlo Györfi, and Gabor Lugosi. *A Probabilistic Theory of Pattern Recognition.* Springer, 1991.

[15] Tamal K. Dey, Joachim Giesen, and Matthias John. Alpha-shapes and flow shapes are homotopy equivalent. In *Annual ACM Symposium on Theory of Computing (STOC)*, pages 493–502, 2003.

[16] Tamal K. Dey, Joachim Giesen, Edgar A. Ramos, and Bardia Sadri. Critical points of the distance to an epsilon-sampling of a surface and flow-complex-based surface reconstruction. In *Symposium on Computational Geometry (SOCG)*, pages 218–227, 2005.

[17] Herbert Edelsbrunner. The union of balls and its dual shape. *Discrete & Computational Geometry*, 13:415–440, 1995.

[18] Herbert Edelsbrunner. Surface Reconstruction by Wrapping Finite Sets in Space. In *The Goodman-Pollack Festschrift (Algorithms and Combinatorics)*, pages 379–404. Springer, 2003.

[19] Herbert Edelsbrunner, Michael Facello, and Jie Liang. On the Definition and the Construction of Pockets in

Macromolecules. *Discrete Applied Mathematics*, 88:83–102, 1998.

[20] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002.

[21] Niklas Elmqvist, Pierre Dragicevic, and Jean-Daniel Fekete. Rolling the Dice: Multidimensional Visual Exploration using Scatterplot Matrix Navigation. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1539–1148, 2008.

[22] Kaspar Fischer, Bernd Gärtner, and Martin Kutz. Fast Smallest-Enclosing-Ball Computation in High Dimensions. In *Annual European Symposium on Algorithms (ESA)*, pages 630–641, 2003.

[23] Michael Friendly and Daniel Denis. The early origins and development of the scatterplot. *Journal of the History of the Behavioral Sciences*, 41(2):103–130, 2005.

[24] Joachim Giesen and Matthias John. Computing the Weighted Flow Complex. In *Vision, Modeling, and Visualization (VMV)*, pages 235–243, 2003.

[25] Joachim Giesen and Matthias John. The flow complex: a data structure for geometric modeling. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 285–294, 2003.

[26] Joachim Giesen and Matthias John. The flow complex: A data structure for geometric modeling. *Computational Geometry: Theory and Applications*, 39(3):178–190, 2008.

[27] Joachim Giesen and Lars Kühne. A Parallel Algorithm for computing the Flow Complex. In *ACM Symposium on Computational Geometry (SOCG)*, 2013.

[28] Joachim Giesen and Lars Kühne. A parallel algorithm for computing the flow complex. In *Proceedings of the 29th annual symposium on Symposuim on computational geometry*, pages 57–66. ACM, 2013.

[29] Joachim Giesen, Lars Kühne, and Sören Laue. Sketching the support of a probability measure. *Journal of Machine Learning Research*, Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics(33):257–265, apr 2014.

[30] Joachim Giesen, Edgar A. Ramos, and Bardia Sadri. Medial axis approximation and unstable flow complex. In *Symposium on Computational Geometry*, pages 327–336, 2006.

[31] Joachim Giesen, Edgar A. Ramos, and Bardia Sadri. Medial Axis Approximation and Unstable Flow Complex. *International Journal on Computational Geometry and Applications*, 18(6):533–565, 2008.

[32] G. Grinstein, M. Trutschl, and U. Cvek. High-dimensional visualizations. In *Proceedings of the Visual Data Mining workshop at the ACM/SIGKDD Data mining Conference*, 2001.

[33] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):19:1–19:19, 2016.

[34] J. Hartigan. Printer graphics for clustering. *Journal of Statistical Computation and Simulation*, 4(3):187–213, 1975.

[35] Grete Heinz, Louis J. Peterson, Roger W. Johnson, and Carter J. Kerk. Exploring relationships in body dimensions. *Journal of Statistics Education*, 11(2), 2003.

[36] Ejnar Hertzprung. Über die Verwendung Photographischer Effektiver Wellenlängen zur Bestimmung von Farbenaequivalenten. *Publikationen des Astrophysikalischen Observatoriums zu Potsdam*, 22(1), 1911.

[37] http://threadingbuildingblocks.org.

[38] Alfred Inselberg and B. Dimsdale. Parallel Coordinates: A Tool for Visualizing Multi-dimensional Geometry. In *IEEE Visualization*, pages 361–378, 1990.

[39] Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The perspective wall: detail and context smoothly integrated. In *Conference on Human Factors in Computing Systems (CHI)*, pages 173–176, 1991.

[40] Kenat Nakai. Ecoli data set, 1996. Data set available at `http://archive.ics.uci.edu/ml/datasets/Ecoli`.

[41] Partha Niyogi, Stephen Smale, and Shmuel Weinberger. Finding the Homology of Submanifolds with High Confidence from Random Samples. *Discrete & Computational Geometry*, 39(1-3):419–441, 2008.

[42] Partha Niyogi, Stephen Smale, and Shmuel Weinberger. A Topological View of Unsupervised Learning from Noisy Data. *SIAM Journal on Computing*, 40(3):646–663, 2011.

[43] Ramana Rao and Stuart K. Card. The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *Conference on Human Factors in Computing Systems (CHI)*, pages 318–322, 1994.

[44] George G. Robertson and Jock D. Mackinlay. The Document Lens. In *ACM Symposium on User Interface Software and Technology (UIST)*, pages 101–108, 1993.

[45] Henry Norris Russell. Relations between the spectra and other characteristics of the stars. *Popular Astronomy*, 22:275–294, 1913.

[46] Dirk Siersma. Voronoi diagrams and morse theory of the distance function. In *Geometry in Present Day Science*, pages 187–208. World Scientific, 1999.

[47] Deborah F. Swayne, Duncan Temple Lang, Andreas Buja, and Dianne Cook. GGobi: evolving from XGobi into an extensible framework for interactive data visualization. *Computational Statistics & Data Analysis*, 43:423–444, 2003.

[48] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

List of Figures

1.1 2D example a flow complex and its Hasse diagram 6
1.2 3D Flow-Complex with dangling edge 7
1.3 Down- and Upflow operations 10
1.4 Example for Cleansing the Flow-Complex in $\mathbb{R}^3$ 18
1.5 Speedup of Flow-Complex computation 27
1.6 Simplification of the flow-complex 29

2.1 α-flow-complex of MovieLens data set 44
2.2 Scatter plot matrix of MovieLens data set . . . 47
2.3 Simplification of the α-flow-complex 48
2.4 α-flow-complex of the BODY DIMENSIONS data set 49
2.5 Computation of a flow line 55
2.6 Relationship between voids and critical points . 59
2.7 Types of flow lines - schematic 61
2.8 Types of flow lines - cube data set 62
2.9 Scatter plot matrix of PARABOLA data set . . . 67
2.10 Scatter plot matrix of PARABOLA data set . . . 69
2.11 Scatter plot matrix of transformed PARABOLA data set 70
2.12 Flow lines induced by two Gaussians 71

2.13 Flow lines induced by two Gaussians with differing variance 73
2.14 Finite flow lines on a single Gaussian 74
2.15 Finite flow lines on a signle Gaussian including outlier detection 75
2.16 Finite and infinite flow lines of the body dimensions data set 77
2.17 Dimension inference of the BODY DIMENSIONS data set using sclow plots 79
2.18 Scatter and sclow plot of the BODY DIMENSIONS data set 80
2.19 Sclow plots for projections of the 14 dimensional MovieLens data set 83
2.20 Finite flow lines of the MOVIELENS data set . . 85
2.21 Finite and infinite flow lines of MOVIELENS data set 86
2.22 Performance data for computing flow lines . . . 89

List of Tables

1.1 Computation time in various dimensions 26
1.2 Critical point distribution after Simplification . 28

Curriculum Vitae

	Lars Kühne born June, 15, 1987 in Altenburg, Germany married, one child
June, 27, 2016 — September, 30, 2016	Software Engineering Intern Google LLC Mountain View, CA, USA
January, 15, 2012 — today	PhD Student and Scientific Assistant Friedrich-Schiller-University Jena Jena, Germany
January, 02, 2012	Diploma in Computer Science (1,0) Friedrich-Schiller-University Jena Jena, Germany
June, 21, 2005	Abitur (1,4) Veit-Ludwig-von-Seckendorff-Gymnasium Meuselwitz, Germany