Visualization of High-Dimensional Point Clouds Using Their Density Distribution's Topology

Patrick Oesterling, Christian Heine, Heike Jänicke, *Member*, *IEEE*,
Gerik Scheuermann, *Member*, *IEEE*, and Gerhard Heyer, *Member*, *IEEE Computer Society*

Abstract—We present a novel method to visualize multidimensional point clouds. While conventional visualization techniques, like scatterplot matrices or parallel coordinates, have issues with either overplotting of entities or handling many dimensions, we abstract the data using topological methods before presenting it. We assume the input points to be samples of a random variable with a high-dimensional probability distribution which we approximate using kernel density estimates on a suitably reconstructed mesh. From the resulting scalar field we extract the join tree and present it as a *topological landscape*, a visualization metaphor that utilizes the human capability of understanding natural terrains. In this landscape, dense clusters of points show up as hills. The nesting of hills indicates the nesting of clusters. We augment the landscape with the data points to allow selection and inspection of single points and point sets. We also present optimizations to make our algorithm applicable to large data sets and to allow interactive adaption of our visualization to the kernel window width used in the density estimation.

Index Terms—Clustering, pattern analysis, point clouds, graphs, topology.

1 Introduction

Point clouds, or more precisely sets of higher dimensional vectors, are a standard format to represent and model a variety of problems from different application domains. Especially for lower dimensional data, simple visualizations became widely accepted to study the relationships between single entities or to reveal a data set's global structure. Although these methods support the analysis with many helpful features like coloring, compositing, and transparency, they are, in the end, not practical for very large or high-dimensional data sets. Most of these common approaches simply do not scale for higher dimensions and the visualizations finally lead to either more confusion or suffer from huge overplotting problems. Therefore, the goal is to break the habit of applying nonscalable approaches to large scale data sets by using more intuitive metaphors that make complex information easier to understand. For example, *Topological Landscapes* [1] show the topology of a scalar function, e.g., a density function, as a 3D landscape which is easily understood by a user through his familiarity with terrain structure.

For point clouds, this indirect visualization has several advantages. Compared to common direct visualizations in 2D or 3D, the topology of the points' density distribution can be computed in all dimensions (Fig. 1). Because the topology is an abstraction, finding a data set's structure does not necessarily require each single data item. By sampling only a part of the input data we are still able to estimate the point cloud's structure.

In this paper, we propose to treat the input data as samples of a random variable having a certain probability density function which we estimate in the first step. This estimate resembles the density distribution of points in the high-dimensional space. We approximate the estimate's topology by its join tree as this tree captures the nesting of dense regions and ignores the, for our purposes uninteresting, topological relation between sparse regions. This join tree topology is then represented in the Topological Landscape, where hills correspond to clusters in the data. We also place the original data into suitable locations in this landscape so that they reflect their membership with the clusters as well as their density. The latter allows to express for each data point how close it is to its cluster center.

A limitation of our approach is the inability to preserve absolute distances and geometric cluster properties like the shape, size, and local point distribution. However, the density-based notion of clusterings, i.e., clusters are dense regions that are separated by regions of low density, allows us to find clusters during our analysis without constraining the input data with respect to these properties. Still, some clusterings cannot be captured by this notion, e.g., a dense cluster inside a sparse cluster usually leads to only one density maximum and also close clusters might be combined if their point distribution is very anisotropic. Also, our topological analysis of the density function is not capable to provide information about empty regions in the data set, and, as every density-based approach, the result depends heavily on one parameter: the kernel window width.

2 Related Work

There are several ways to obtain structural information of a high-dimensional data set: Either the data are drawn using

- *P. Oesterling, C. Heine, G. Scheuermann, and G. Heyer are with the Institut für Informatik, Universität Leipzig, PF 100920, Leipzig 04009, Germany.*
  E-mail: {oesterling, heine, scheuermann, heyer}@informatik.uni-leipzig.de.
- *H. Jänicke is with the Interdisciplinary Center for Scientific Computing (IWR), Heidelberg University, Im Neuenheimer Feld 368, Heidelberg 69120, Germany. E-mail: heike.jaenicke@iwr.uni-heidelberg.de.*

Manuscript received 31 May 2010; revised 16 Dec. 2010; accepted 3 Jan. 2011; published online 26 Jan. 2011.
Recommended for acceptance by H.-W. Shen, J.J. van Wijk, and S. North.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCGSI-2010-05-0110.
Digital Object Identifier no. 10.1109/TVCG.2011.27.

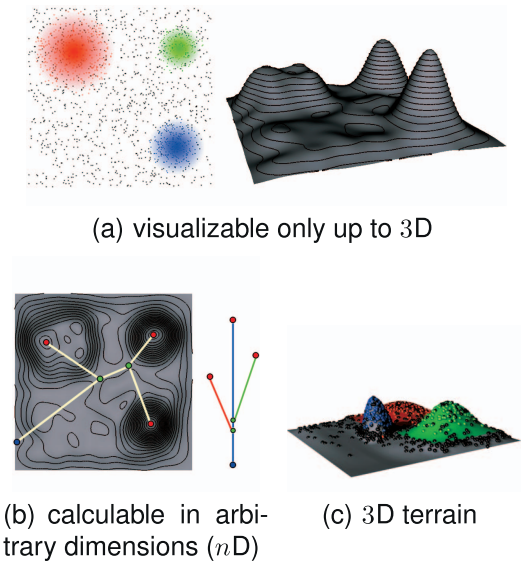


Fig. 1. (a) Because direct visualizations via scatterplots are limited to three dimensions, we cannot *look* into a high-dimensional data set by simply visualizing its point cloud or, e.g., its density distribution. (b) Instead, we calculate the density distribution's topology, described by a join tree, in arbitrary dimension. (c) Finally, the join tree is turned directly into a *Topological Landscape* [1] which has the same topology as the given input data set.

visualizations that take all dimensions into account, or the data points are projected directly down to 2D/3D and visualized via scatterplots. For the former, star plots [2], parallel coordinate plots (PCP) [3], and scatterplot matrices (SPLOM) [4] are the most common techniques. These visualizations, however, have drawbacks and limitations like the crucial order of axes, the overplotting of points, the pixel-costs per entity, or the fact that some of them only face a subset of all dimensions at a time. To compensate the drawbacks of scatterplots and parallel coordinates for very high-dimensional data, Tatu et al. [5] proposed an automatic analysis to determine the most relevant visual structures from a set of candidate visualizations. Unlike PCP or SPLOM, our visualization's complexity does not increase much with the data's dimensionality and we do not overplot structural features. We can always encode them in a 3D terrain and the user can rotate, zoom, and move the landscape to extract all the important features from the visualization. Another advantage over these conventional methods is that we do not just display all the data and let the user try to identify structures. We instead determine and visualize both the structure and the data. In contrast, the crucial order of axes in PCP or star plots heavily influences the ability to identify any global structure at all.

Common ways to reasonably project the data into lower dimensions are PCA [6], LDA [7], MDS [8], or Kohonen Maps [9]. For example, Choo et al. [10] analyze the combination of several projections for 2D visualization of clustered high-dimensional data. ClusterSculptor [11] is a visual analytics tool for high-dimensional data that uses many of the techniques mentioned above.

Another reduction approach is to remove irrelevant or redundant dimensions, or to search for structure and correlations only in subspaces [12], [13]. By default, our approach considers *all* dimensions and therefore special structures in subspaces might not be found. In principle, we could also apply our method only to particular subspaces, thus accelerating our overall runtime and finding similar features.

If direct visualizations are not feasible, other visualization paradigms might be useful. For example, a dendrogram, as the result of hierarchical clustering [14], encodes a data set's clustering in a tree structure, thus indirectly visualizing the data. Here, instead of the arrangement of points their connection with edges leads to the final structural insights. Compared to a tree layout, the terrain metaphor is much more intuitive and allows for an easier description of several properties for each hill. The idea of defining clusters as regions of dense point arrangements surrounded by regions containing few points is also present in DENCLUE [15] or DBSCAN [16] and a widely accepted model.

Another approach that also utilizes topology was introduced by Takahashi et al. [17]. Their mapping from nD to 3D results in a point arrangement that reflects the approximated topology of the input scalar function. In contrast to our approach their input already is a scalar function, and its domain manifold is learned. We consider the full high-dimensional space and reconstruct the scalar function from the points ourselves using kernel density estimates.

3 Background

3.1 Neighborhood Graphs

In order to extract a meaningful density function in a high-dimensional space, we need methods to fill the void between data samples and methods to describe closeness between points.

Let $\delta(x, y)$ denote the distance of two points x, y in a d-dimensional euclidean space $\mathbb{R}^d$, the *Voronoi diagram* [18] of a point set $P = \{p_1, p_2, \ldots, p_n\}$ is a partition of $\mathbb{R}^d$ into cells $c_i = \{x \in \mathbb{R}^d | \forall j \neq i : \delta(x, p_i) \leq \delta(x, p_j)\}$. Each cell's points have the same closest point of P.

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

The *Delaunay triangulation* [18] of a point set P in $\mathbb{R}^d$ in so-called general position is a simplicial complex such that for each simplex there exists an empty circum-hypersphere. There is a well known duality: two cells c_i, c_j are neighbors in the Voronoi diagram if and only if the Delaunay triangulation contains an edge between p_i and p_j. There are algorithms [19] that can compute the Delaunay triangulation directly and they perform very well in two and three dimensions. The number of simplices is in $\Omega(n^{d/2})$ in the worst case [18], giving an exponential lower bound on the runtime with respect to the dimension. We will refer to the vertices and edges of the Delaunay triangulation, omitting all other simplices, as the *Delaunay graph (DG)*.

A *neighborhood graph* [20], also called a *proximity graph*, is a graph in which *neighbored* vertices are connected by an edge. The Delaunay graph is a suitable neighborhood graph as it maps Voronoi cell neighbors to edges.

The *Gabriel graph (GG)* [21] of a point set P in $\mathbb{R}^d$ is a graph (P, E) that contains an edge between two vertices u, v if and

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

The *relative neighborhood graph (RNG)* [20] of a point set P in $\mathbb{R}^d$ is a graph (P, E) that contains an edge between two vertices u, v if and only if their *RNG-lune* is devoid of points from P. This lune is the union of two hyperspheres being centered around u and v and having a radius equal to the distance of u and v. The construction algorithms of RNG and GG differ only in the test for each edge.

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

The *nearest neighbor graph (NNG)* [20] of a point set P in $\mathbb{R}^d$ is the smallest graph (P, E) where each vertex has an edge to its nearest neighbor.

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

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

Because the euclidean minimum spanning tree is a connected graph, all its supergraphs are connected as well. Being a supergraph of the NNG also means that the connection of each vertex to its nearest neighbor is also contained. An illustration of the neighborhood graphs that we use in our work is given in Fig. 7.

3.2 Scalar Field Topology

Scientific data are often provided as a discrete scalar function (or scalar field), describing, e.g., pressure, density or temperature. The task for the visualization is to illustrate the data's structure and complexity and to examine whether the data contain unexpected features that require a closer investigation.

In addition to common rendering approaches, like, e.g., volume rendering [22] or isosurfaces [23], computing the topology of a scalar function is another approach to investigate the data's structure. This aims at a complete study of the variation of the function, described by critical points or entire regions [24].

Let the data be given as a point set $P = \{p_1, p_2, \ldots, p_n\}$ in $\mathbb{R}^d$, with corresponding scalar measurements $H = \{h_1, h_2, \ldots, h_n\}$. To interpolate values for points not in P, the data are extended to the entire space by means of a mesh with vertex set P, and a continuous function f that satisfies $f(p_i) = h_i$. A *level set of f at some function value h* is the set $\{x \in \mathbb{R}^d | f(x) = h\}$ and may consist of zero, one, or more connected components. For lower dimensions, these sets of connected components are known as *isolines* (2D) and *isosurfaces* (3D). In general, they are referred to as *contours*. If the function value $f(x)$ is thought of as time, we can watch the evolution of contours of f over time, seeing them appear, join, split and disappear. The *contour tree* [25] is a graph in which each contour is contracted to a single point and that tracks these topological changes. Fig. 1b shows a density scalar function. The rings represent some contours and the overlayed tree shows the topological changes at critical points.

If a simplicial mesh and piecewise linear interpolation is used, the contour tree can be determined by computing and merging two trees: the join tree and the split tree [26]. The join tree of a scalar function contains the global minimum and all local maxima as well as all saddles that join contours. As we are only interested in areas of high density and their nesting, the join tree is sufficient for our purposes. Due to the limited space, we refer the reader to Carr et al. [26] who present a simple and fast graph algorithm to compute the join tree based on the vertices and the edges of the mesh. We would also like to remark that if the split tree is a single path of vertices, the contour tree is equal to the join tree. We can therefore always use an algorithm that is applicable to a contour tree on a join tree as well.

The contour tree is often partitioned into *branches* which are paths that are monotonic with respect to the function values. We treat a branch's *persistence* [27] as the difference between its highest and its lowest function value. The branches can be organized in a *branch decomposition* [28], which is a tree. Each of this tree's nodes is a branch; the root is the branch of highest persistence. One node is a child of another node if the branch it represents has lower persistence than the other node's branch and both branches are connected by a saddle.

3.3 Topological Landscape

Due to the page limitation, we cannot give an appropriate account of the algorithm to create the Topological Landscape, because this would require us to introduce a number of concepts unrelated to the rest of our work. We refer the reader to Weber et al. [1] for the construction details. For our purposes, we view the algorithm as a black box to which a branch decomposition is given and that results in a list of triangles with endpoints in $\mathbb{R}^3$. The main properties of the landscape are that hills correspond to branches of the input branch decomposition, the nesting of hills equals that of the branches, and hills can be metrically distorted to match another property of each branch, e.g., its volume. The generation of the Topological Landscape is fast and the landscapes are easily understood by humans. Each triangle is annotated with the branch it originates from. This information is necessary to map the original data points to the correct region of the landscape. Fig. 3 shows a contour tree and the corresponding landscape which has the same topology.

4 Algorithm

Our algorithm can be split into mainly five parts, whereas two of them are optional runtime optimizations. An overview of the algorithm is given in Fig. 2. In this section, we first describe the mandatory phases of the algorithm; the optional phases, namely the sampling and reinsertion, are explained in Section 5.

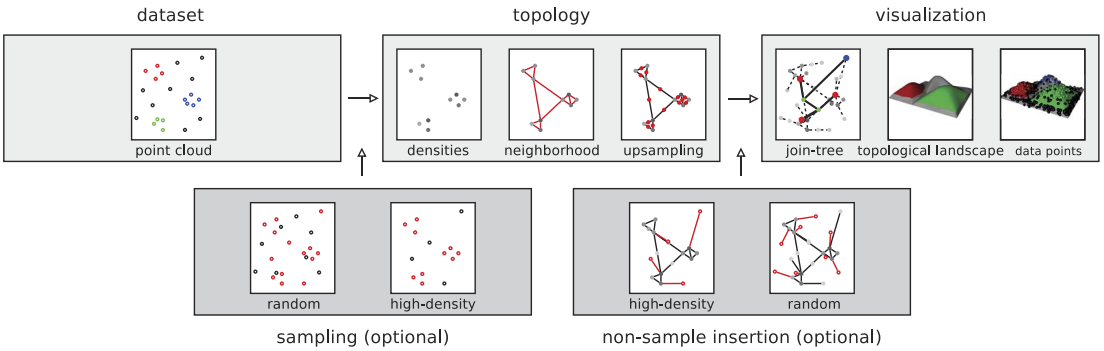


Fig. 2. Overview: Optional sampling is applied to the input data to represent it by a smaller set of points. We determine that point set's neighborhood relationships and approximate the density function. After an optional reinsertion of nonsamples, we compute the join tree and visualize the tree structure and the data points using a Topological Landscape.

Given the high-dimensional point cloud as a set $P = \{p_1, p_2, \ldots, p_n\}$ of points in a fixed-dimensional euclidean space $\mathbb{R}^d$, we start with sampling the implicit density function at these positions, using a Gaussian kernel density estimate using window width σ. Afterwards, we connect these points with a particular neighborhood graph on whose edges we search for additional saddles of the density function. We compute the function's join tree and use it as the input to generate the Topological Landscape. Finally, we add the original data points as small spheres into this landscape to improve the perception of the density distribution on the hills. This arrangement of data points gives us a tool to find a σ iteratively that is suitable to identify structures in the data. Moreover, we can color these points according to a possibly existent classification, or we can annotate them with labels to facilitate interaction or semantic zoom.

4.1 Approximating the Density Function

We approximate the density function using a Gaussian kernel density estimate [29] of the given $p_i \in P$:

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

with $\delta(x, p_i)$ being the euclidean distance between point x and sample p_i. Because the computation of this function's topology is very hard, we approximate f by a simpler function f' of very similar topology for which we can use the join tree computation. We aim for an f' that is a piecewise linear interpolation on a simplicial complex M. Using the Delaunay triangulation on P with $f'(p_i) = f(p_i), \forall p_i \in P$ for this purpose has two disadvantages: a prohibitive runtime of $\Omega(n^{d/2})$ and the approximation is rather coarse since the density between two points can be lower than their density, as is illustrated in Fig. 4a. In this case, f' lacks a corresponding saddle value which indicates that the two points are separated by a region of low density.

The quality of the approximation could be improved by augmenting the Delaunay triangulation by all critical points of f, which are very hard to compute exactly. As the topology of the density distribution is merely a tool to find the nesting of dense regions, we only require a heuristic to find the saddles between them. For this purpose, we add a further sample on the midpoint m of each mesh edge and require $f'(m) = f(m)$. We refer to this process as *upsampling*. To speed up this process, we use a special property of euclidean spaces: for any midpoint c between x and y, the distance $\delta(c, z)$ can be computed in $O(1)$ instead of $O(d)$ purely from the distances $\delta(x, y)$, $\delta(x, z)$, and $\delta(y, z)$ observing that x, y, z span a planar subspace that includes c and in which the law of cosines holds: $4\delta(c,z)^2 = 2\delta(x,z)^2 + 2\delta(y,z)^2 - \delta(x,y)^2$ (cf. Fig. 4b). Furthermore, we only add a midpoint sample if its density is lower than its two surrounding vertices', as only then it can affect the topology.

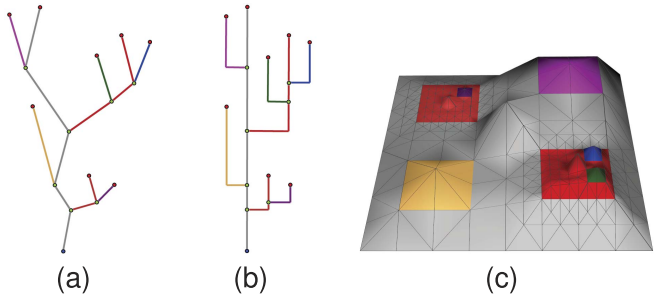


Fig. 3. (a) A contour tree, (b) its branch decomposition, and (c) the Topological Landscape having the same contour tree. Colors indicate which parts of the landscape correspond to which branches of the contour tree.

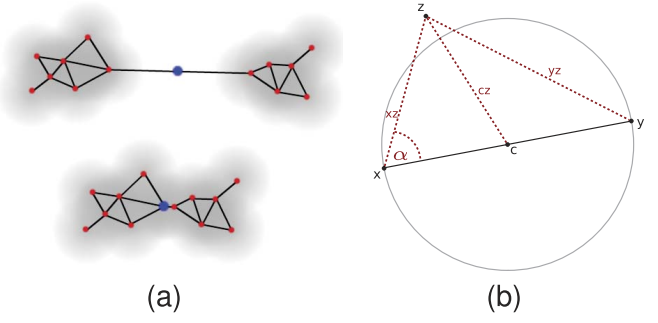


Fig. 4. (a) Without searching for an extra low density saddle between two clusters, the saddle would lie on one of the clusters' borders and we would mistakenly overlook the separation. The topological analysis would "see" the lower rather than the upper cluster configuration. (b) Because the triangles xyz and xcz share the angle α, the distance $\delta(c, z)$ can be computed directly from the distances between x, y, z using the law of cosines in $O(1)$.

The problem of high runtime can be solved by considering only a subset of the Delaunay triangulation. In particular, we considered the Gabriel graph, the relative neighborhood graph, and the euclidean minimum spanning tree. An informal evaluation of the choice of these neighborhood graphs is given in Section 5.1 after we finish the general description of the algorithm.

4.2 Computing the Topology

This phase computes the join tree as described by Carr et al. [26] on our approximation of f, which is given by the neighborhood graph along with the values of f at each vertex and at the midpoints of edges. The join tree contains all the information about maxima and the densities at which contours join. It thus preserves the number and nesting of dense regions. The algorithm itself is performed on a graph for which we trivially bound the number of edges by $n(n-1)/2$, which in turn bounds the runtime of the join tree computation by $O(n^2 \log n)$, n being the number of graph vertices. Using a trivial modification of the join tree algorithm, we can remember for each graph vertex that represents a data point, on which join tree edge it lies. Since join tree edges are reflected by regions in the Topological Landscape, this mapping allows for a proper placement of data points in the landscape.

4.3 Generating the Landscape

In order to visualize the join tree, we essentially make use of the landscape metaphor as introduced by Weber et al. [1]. Prior to the landscape calculation, we remove topological noise by pruning branches from the branch decomposition that exhibit a persistence or an accumulated volume smaller than a given value.

Weber et al. present a way of distorting the landscape in order to better reflect the approximated contour volumes in the original space. This distortion is achieved by resizing all the triangles of a hill according to the hill's corresponding branch volume. Computing the volume in the high-dimensional space is computationally difficult, and for pragmatic reasons, we simply use the number of a branch's vertices as its volume. Because this volume is distributed equally to all the hill's triangles, the centered hill of nested hierarchies usually becomes heavily distorted. This primarily destroys the visual expressiveness of that hill's corresponding cluster in the landscape. Instead of dividing a branch's volume by the number of corresponding hill-triangles, we assign the volume above the first saddle to the eight triangles of the centered hill and the rest equally to all the remaining triangles. Although this could be done more accurately by considering the volume between each saddle pair, this strategy sufficiently highlights center clusters' volumes.

We also augment the landscape with the data points by placing them as small spheres at their appropriate positions. Because each data point lies on exactly one branch, we can place it randomly along the correct contour of the branch's hill. Each triangle may have a different amount of intersection with the contour and thus we select a random triangle using probabilities proportional to the amount of triangle-contour intersection. If each triangle had the same chance of being selected to host a point, points would accumulate at sharp triangles and triangle corners, resulting in a distorted impression of point distribution (cf. Fig. 5a). To quickly exclude triangles that do not intersect a certain contour and therefore have a probability of zero, we use the *span space* [30] data structure for each branch (cf. Fig. 5b).

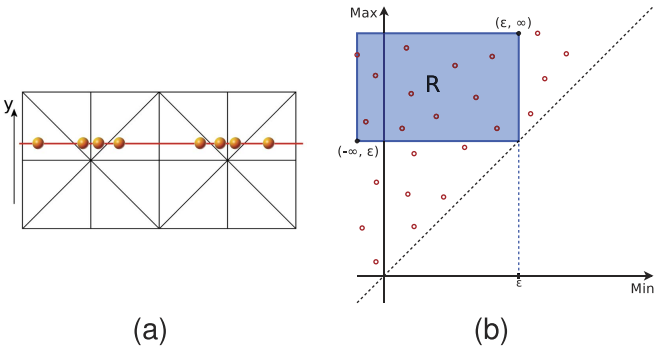


Fig. 5. (a) Assuming y equals isovalue, the red line shows a contour. When randomly selecting a triangle to place a point, we use probabilities proportional to the amount of contour-triangle intersection to avoid accumulation at sharp triangles and triangle edges. (b) In *span space* each triangle is represented by a point (min, max) of its isovalue range and a query for the triangles that contain the isovalue ϵ is done by a window query of the points.

4.4 Interpreting the Landscape

The only parameter for our density estimation, the kernel window width σ, has a crucial influence on the quality of the generated landscape. If it is too large, the result is a rather blurry density distribution and merges dense regions. Too small, and it creates peaks at small noise accumulations and inside dense regions. Conceptually, we want to find a σ that minimizes the number of hills and maximizes their height at the same time. This reflects that each dense region has its own maximum, but noise points do not. We start with $\sigma_{opt} = 4/(n(d+2))^{1/(d+4)}$ which minimizes the approximate mean integrated square error, if the original data points' distribution followed a d-variate normal density [29]. As this assumption is usually not true, the user can then modify the value σ based on the current sample locations in the landscape:

1. *The number of hills.* Unless the data set actually contains only one dense region, the presence of one single hill (or only a few hills) is an indicator of the σ being too large. Many small hills with only a few data points on it indicate that σ is too small.
2. *The data point distribution on a hill.* In case of separated dense regions, the cluster's points' densities are significantly higher than the saddle value outside the cluster. Therefore, these points are arranged on the upper part of the hill. If many points are arranged at the saddle height, i.e., at the foot of the hill, they are either noise or σ is too small and splits a single dense region into several ones.
3. *The data point separation on a hill.* If the data points on a hill constitute rings of different height, then σ is too large and combines dense regions of different densities.

An example for this process is given in Fig. 12.

5 Optimizations

In this section, we explain what effect the choice of neighborhood graphs has on the algorithm and we present

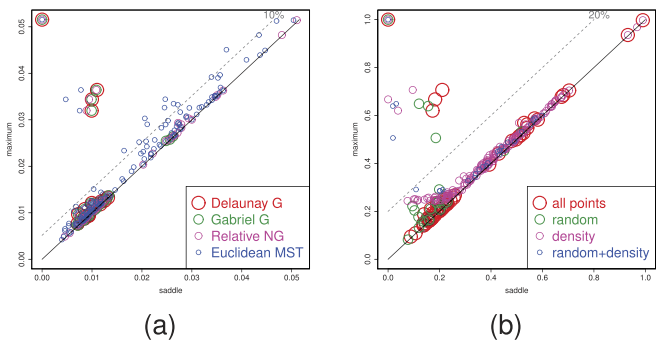


Fig. 6. Persistence diagrams of the join trees for (a) different neighborhood graphs (Figs. 7a, 7b, 7c, and 7d) and (b) different sampling strategies (Figs. 8a, 8b, and 8c). The circles correspond to the persistence intervals of all single branches.

an optional phase that reduces the overall runtime by reducing the number of input points by random and density-based sampling.

5.1 Approximating the Delaunay Triangulation

The exponential increase in runtime when computing the Delaunay triangulation in arbitrary dimensions led us to the consideration of other neighborhood graphs as surrogates. However, these surrogates are not simplicial, therefore a precondition of the join tree algorithm is not met. Yet, this condition is only sufficient and not necessary, as it is trivial to construct simplicial complexes for which the deletion of some edges does not alter the output of the join tree algorithm. Also, high-dimensional simplices force each vertex to have an increasingly high edge degree, resulting in very dense neighborhood graphs that can have trivial topology, i.e., only one branch. Using sparser alternatives for the Delaunay graph avoids this and significantly decreases our algorithm's runtime due to the smaller number of edges that have to be upsampled.

In comparison to the Delaunay graph, the Gabriel graph, the relative neighborhood graph, and the euclidean minimum spanning tree increasingly omit the longer edges of simplices. The density along these omitted edges is usually lower than the densities along the preserved edges. We therefore make only a small mistake because the saddles of the join tree are also likely to be preserved. However, the chance increases to remove an edge that hosts the correct saddle.

Regarding the runtimes, the GG and the RNG can be computed in $O(n^3)$. As this can be still a very long time, we present a faster computation of these graphs in the appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.27. The EMST is our smallest possible Delaunay graph approximation. Having the fewest edges and the asymptotic runtime $O(n^2)$, it is the fastest connected neighborhood graph we can use.

Fig. 7 shows the DG, the GG, the RNG, and the EMST for a noisy point set, along with the respective join trees that result from using less complex neighborhood descriptions. Fig. 6a shows the effect of using different neighborhood graphs on the topology using a persistence diagram. A persistence diagram [31] is a 2D scatterplot where each branch's isovalue range is mapped to a point (min, max). Branches of low persistence show up as points near the main diagonal. In the top-left corner of Fig. 6a it is clearly visible that using the four graphs lead to exactly four branches of high persistence that correspond to the four dense regions in the data set. We can see that, as the density has not changed for the data points, the maximum isovalue of the four features does not change. However, the branches corresponding to the other three dense regions are increasingly moving to the left. This results from a decreased connectivity of the graphs and the join tree algorithm having less freedom (edges) to connect contours.

We also see an increasing amount of *topological noise*. This is reflected by the low-persistent intervals near the diagonal. Two main effects are caused by reducing the number of edges: First, while the DG and the GG tend to produce topological noise only in low-density regions (bottom-left quadrant), the RNG and the EMST also produce noise inside the clusters (along the whole diagonal). Second, the persistence, i.e., the gap to the diagonal, of the topological noise is also increasing, especially for the EMST.

The reason for these effects is that the graph is increasingly degenerating to a tree. That is, the (at first good) neighborhood description inside a cluster is more and more breaking into several separated parts, and whole clusters or even complete regions are eventually represented

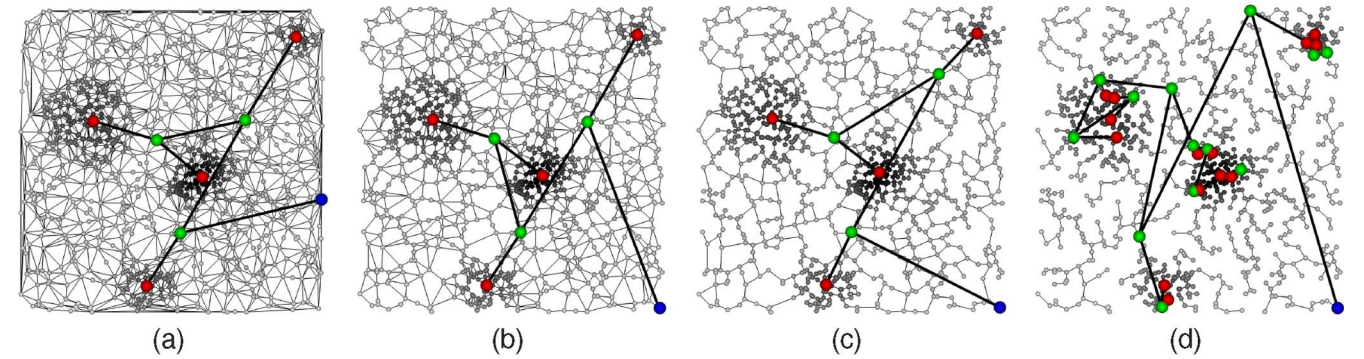


Fig. 7. Comparison of (a) the Delaunay graph, (b) the Gabriel graph, (c) the relative neighborhood graph, and (d) the minimum spanning tree. The overlayed join trees ($\mathrm{red} = \mathrm{maximum}$, $\mathrm{green} = \mathrm{saddle}$, and $\mathrm{blue} = \mathrm{minimum}$), that have been pruned using a 10 percent threshold, illustrate the change in the density distribution's ($\mathrm{dark} = \mathrm{dense}$) topology when using less complex neighborhood graphs. Note that the upsampled vertices have been inserted after the graph construction.

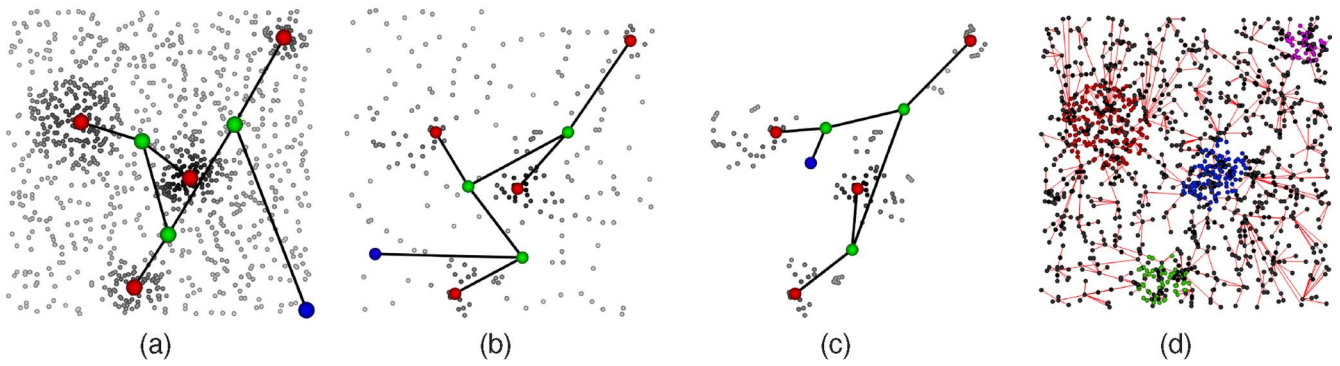


Fig. 8. Comparison of different sampling strategies (a) using all points, (b) after randomly sampling 20 percent of the points, (c) after additionally sampling only those points that have a density greater than 20 percent of the maximum density. (d) The RNG after reinserting all nonsamples.

only as a subtree, being connected by only one edge to the remaining graph. In the end, the evolution of the contours becomes more and more dominated and restricted by the graph structure itself. As a result, small contours appear and produce topological noise.

However, we can eliminate this noise by pruning all the branches from the branch decomposition that have a persistence smaller than a given threshold. As mentioned, this threshold increases with using smaller neighborhood graphs. Using the 10 percent threshold as indicated by the dotted line in the persistence diagram, we obtain the pruned join trees shown in Fig. 7.

5.2 Sampling

Sampling is an optional phase of our algorithm to reduce the number of input points and therefore its overall runtime. As we assume the given points to be samples using an unknown probability density function $\hat{f}$, which we approximated with f, we can consider the effect of a further reduction in samples. The mean integrated square error between $\hat{f}$ and f can be approximated [29]:

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

where the first term can be thought of as a systematic error resulting from kernel density estimation in general and the second term as a random error based on sampling. There is an inverse linear relation between the random error and the number of samples and for high dimensions the systematic error dominates. As $\hat{f}$ is unknown, the error cannot be computed, but it can still be minimized with respect to σ [29]:

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

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

We use both random sampling as it is very fast and indiscriminate as well as density-based sampling that removes samples with a density lower than a certain threshold and thus increases the "signal-noise" ratio of the data. The threshold might be a fixed percentage of the maximum density or a value determined automatically. Fig. 8b shows the result of randomly sampling 20 percent of the input, and Fig. 8c further removes samples of a density lower than 20 percent of the maximum.

Fig. 6b shows the persistence diagram for sampling the 2D data set used in Fig. 8. Unlike the persistence diagram for the different neighborhood graphs, we normalized it with respect to the main branch's persistence, as this value can vary. Especially in the density-based sampling, the density "moves" from low-density to high-density regions, often resulting in a main branch of higher persistence than in the fully sampled data. For this data set it is easy to identify the four prominent features and it is interesting to note that density-based sampling and subsequent normalization did not change their maxima, only their saddles, regardless whether applied to the data directly or on the randomly sampled data. On the other hand, the random sampling decreases the signal-noise ratio.

When all data points should be represented in the final landscape as small spheres, we add the points that were removed in the sampling phases back into the neighborhood graph before the join tree algorithm is run. The points are inserted in the inverse order of their removal. Let S denote the set of sampled points and $\bar{S}$ the set of nonsampled points. For each point $\bar{p} \in \bar{S}$, we approximate its neighborhood by an edge to its nearest sampled neighbor $NN(\bar{p}) \in S$. $\bar{p}$ may have several valid neighbors, dependent on the chosen type of neighborhood graph, but if $\bar{p}$ were a sampled point, the edge to its nearest neighbor would have been a graph edge, as all neighborhood graphs contain the NNG. Since the nearest neighbor's distance can vary, we also test whether it is necessary to upsample the edge to avoid noise points to fall into neighboring clusters. Because each $\bar{p}_i$ is connected with only one edge to the neighborhood graph of S, *stars* start to appear at points $p_i \in S$ (see Fig. 8d). We, again, want to clarify that the insertion is only for the purpose of data point illustration in the final landscape and not to additionally approximate the point cloud's topology. Therefore, if the user is primarily interested in the coarse data structure rather than a complete visualization, skipping the optional reinsertion can provide a landscape preview.

6 Results

In this section, we demonstrate the application of our approach to several data sets. We compare the runtime of

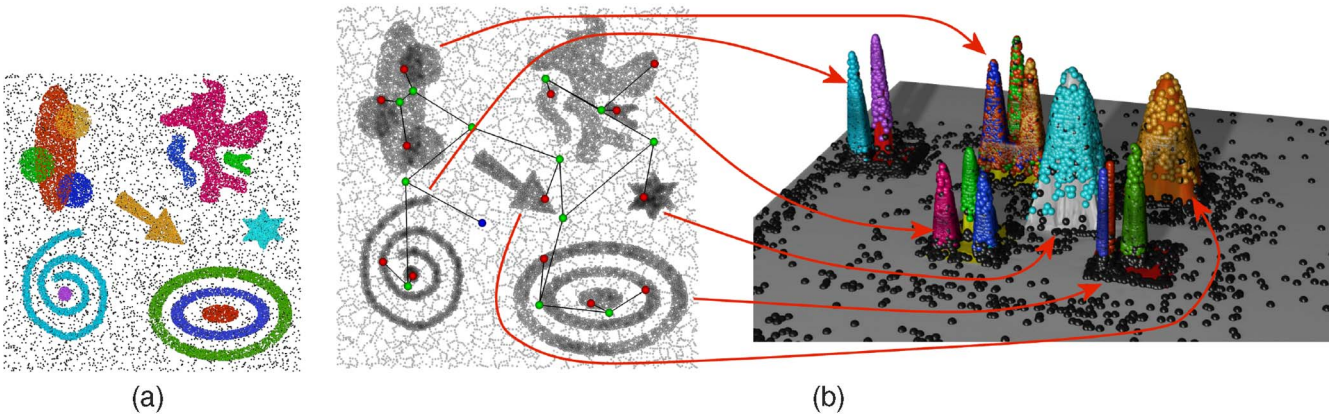


Fig. 9. (a) Synthetic, noisy 2D clustering (colors represent cluster association). (b) Density distribution (dark = dense) and overlayed join tree (red = maximum, green = saddle, and blue = minimum). The red arrows indicate the relation between the parts of the data set and their place in the Topological Landscape.

several settings for different graphs and sampling strategies, and we show the final landscapes including their respective interpretation. Throughout the examples, we use given classification information to color the data points in the landscape. This is just for illustration purposes; no part of our algorithm uses the classification. We implemented our method in C++ and with the exception of the join tree and the Topological Landscape computation, each phase of the algorithm is executed in parallel. We tested our algorithm on a computational platform with two 2.6 GHz Quad-Cores.

Before we use real-world data sets, we demonstrate our method on a manually created 2D clustering. This primarily aims to achieve a better understanding of the whole process because we can give 2D illustrations of some relationships and intermediate results. In order to describe as many features as possible, we generated the scenario shown in Fig. 9a. It has 31,834 points and contains several clusters of different size and shape. Some of the clusters are partially intertwined, others contain each other. Furthermore, the whole domain is afflicted with background noise. The colors represent the cluster association. We applied our method using different neighborhood graphs, with and without sampling. The runtimes for these settings are summarized in Table 1. Fig. 9b shows the density distribution and the overlayed join tree which describes the evolution of contours. No sampling was performed and the relative neighborhood graph was used. As can be seen, each dense region contains a density maximum and is separated from other dense regions by a saddle. Saddles occur in less dense regions between two dense regions, i.e., either on noise points, cluster borders, or on upsampled points. The representation of this join-tree by the final Topological Landscape is indicated by red arrows in the picture. Starting with the global density maximum (in the cyan star) on the root branch, all child branches are positioned in the order of descending saddle values on a spiral around the center hill. The same scheme is applied to each child branch until each branch is represented by a hill at its appropriate position in the landscape. Nested hills represent nested contours which, in turn, reflect either nested clusters or clusters that are locally close (justified by a neighborhood edge and a saddle value that is higher than surrounding saddles). Moreover, the noise points are spread over the whole landscape, as they are also spread over the whole data set. In order to distinguish between small and big clusters, or to separate nearby clusters, a small window width σ is needed. But using a small σ results in many local maxima if the cluster is bigger than σ itself. These close saddle-maximum pairs then evolve inside a cluster until its border (i.e., a low-density region) is reached (cf. Fig. 10). While this is the key to find arbitrarily shaped clusters, this behavior also results in more

TABLE 1
Runtimes for Synthetic 2D Example (Times in Seconds)

graph	number of edges	time for graph	time for upsampling	time for reinsertion	total time
	with sampling (20% random, 20% density)				
EMST	4,315	< 1	< 1	14	18
RNG	3,395	< 1	< 1	14	19
GG	6,876	< 1	< 1	26	31
	without sampling				
EMST	31,833	19	2	-	28
RNG	44,942	76	3	-	91
GG	77,077	79	6	-	95

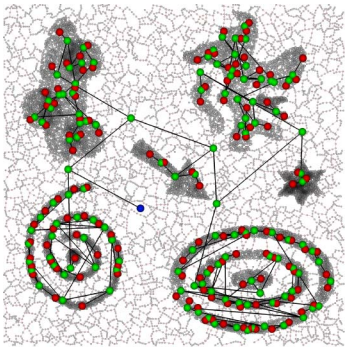


Fig. 10. Depending on their window width, the application of Gaussian-like kernels to arbitrarily shaped clusters, generally, leads to many maximum-saddle pairs that take the form of the cluster (until they reach regions of low density). A subsequent pruning of this *topological noise* finally leads to fewer maximum-saddle pairs per cluster.

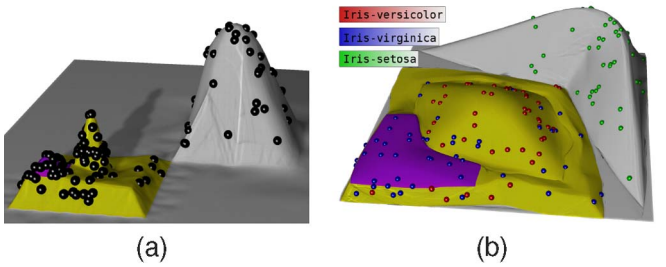


Fig. 11. Iris data set: (a) The topology of the landscape indicates a single cluster and two other overlapping clusters and (b) the colors confirm that the Iris flowers cluster based on their type; the size of a distorted hill reflects the number of flowers in this cluster.

complex topological descriptions that disturb our final perception and insights. Therefore, we prune all branches with only a small saddle-maximum difference to get the clearer result in Fig. 9b.

The next example is the *Iris* plant data set [32]. It is one of the most popular data sets in pattern recognition and contains 150 instances of three types of iris flowers. Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. It is known that one class is linearly separable from the other two; while the latter are not linearly separable from each other. We try to detect this with our technique: In Fig. 11a we see two hilly structures. This means that we have at least two density maxima that are separated by a saddle of low density, i.e., two point accumulations are separated in space. While one is a single hill, the other hill actually consists of two hills. We, thus, identify two density maxima that are separated by a saddle value that is significantly higher than the saddle value of their parent hill. This means that the corresponding clusters are overlapping at their borders (but without constituting an own density maximum). Fig. 11b shows the volumetric distorted version of this landscape. Now, the hills' sizes reflect the clusters' numbers of points. By additionally using the classification information to color the data points, we now see that the landscape reflects the already known clustering situation; with the advantage that we can easily identify the overlapping region: the points at the height of the saddle value (being mixed on both hills). The computation time was less than one second.

While we took the optimal window width σ as granted in the last examples, we now demonstrate how we find the most suitable value to identify relevant structures in the data. As described in Section 4.4, the user obtains the final landscape by changing σ iteratively; each time interpreting the landscape's visual feedback. We want to demonstrate this on the *Image Segmentation* data set from the UCI machine learning repository [32]. This data set has 2,310 instances from seven outdoor images. Each instance is a 3×3 pixel region and 19 attributes describe various features like position, line densities, edges, and color values. A manual segmentation classifies each instance into one of the following types: brickface, sky, foliage, cement, window, path, and grass. We start our analysis with choosing an obviously too big window width, like, e.g., a multiple of the maximum distance. This aims at creating just one single hill whose distribution of data points can lead to first insights.

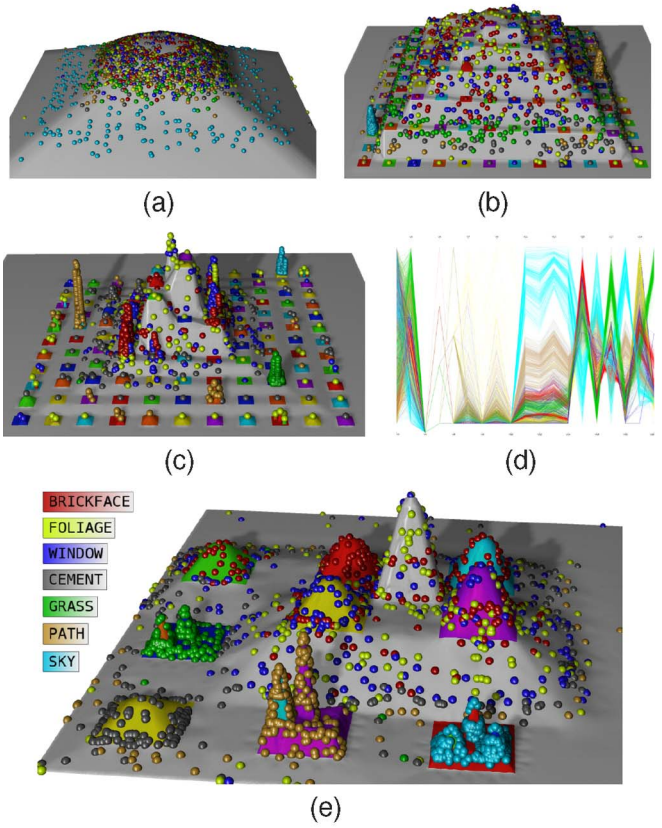


Fig. 12. Segmentation data set: (a)-(c), (e) The process of interpreting the landscape. In every step, a smaller kernel window width σ reveals more structure in the data. If σ becomes too small, many single peaks indicate that small point accumulations start to form clusters. (d) PCP (using colors and transparency) for comparison.

While a positioning of all points at the top would just indicate a σ too big, rings of points indicate clusters in the data. As shown in Fig. 12a, this is the case in our scenario. We therefore reduce σ, e.g., by dividing it by two. The resulting pyramid-like structure in Fig. 12b still suggests a big *blob* because the saddles between the maxima are still high and not low. However, the current σ is yet small enough to detect the first clusters; and there are still some equally colored points separated at different heights. A further reduction of σ, in fact, reveals more separated hills, and the pyramid has become smaller (Fig. 12c). We gain two important insights from this landscape: First, the structures with a significant height that host many points represent separated clusters in the data. Second, these structures are rather fine grained, and we see many other hills with only a few points. This indicates that the current kernel window starts to cluster local point accumulations. Because a further reduction just results in significantly more single hills we stop reducing σ. Instead, we eliminate the hills that feature only a few points by pruning the corresponding branches from the branch decomposition. As a consequence, the final landscape looks much clearer now (Fig. 12e). We conclude that the 3×3 instances really cluster in the high-dimensional feature space. However, because many data points are at saddle heights, and because some clusters are separated by high saddle values, the clustering is not really well separated. We point out that although colored data

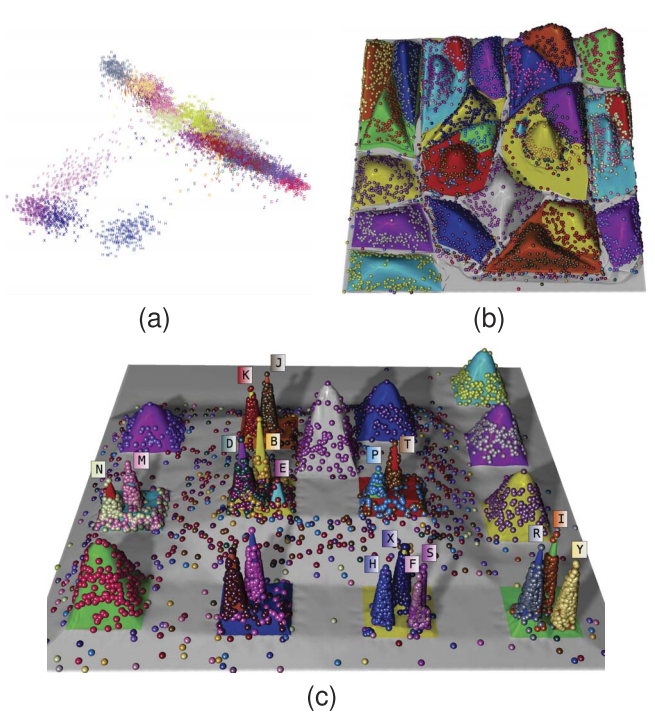


Fig. 13. ISOLET data set: (a) Due to overplotting in the 2D LDA-projection, the clustering situation is still unclear. (b) Distorted landscape that better reflects the contours' sizes. (c) Topological landscape of the data set. Hills correspond to letters, and letters that sound similar are also nearby in the original domain.

points make this interpretation of the landscape easier, it is generally the same without classification information. Fig. 12d shows a PCP of this data set. Although colors are used, overplotting and the order of axes make it hard to reveal the global separation of clusters; without coloring it would be almost impossible. Also, the breaking of clusters into subclusters (as can be seen on the brown, green, and cyan hills in the landscape) is, generally, much harder to recognize due to these limitations. However, a shortcoming of our method is that we only give global insights, i.e., the correlations between single dimensions are better visualized via PCPs. Thus, a linking to PCPs when clicking on a hill might overcome this drawback.

In our next example, the (Isolated Letter Speech Recognition) ISOLET data set [32], 150 subjects spoke the name of each letter of the alphabet twice. From 7,797 recordings of the speakers, a total of 617 attributes were extracted from the pronunciation of each letter. These features include spectral coefficients, contour features, and several sonorant features. The features are described in the paper by Fanty and Cole [33]. Regarding this scenario, we have to admit that we cannot use this data set in its original state. The two reasons for this are its high dimensionality and the fact that our approach is purely distance based. *Curse of dimensionality* results in all distances being almost the same, which, in turn, negatively affects the construction of a meaningful density function and a good neighborhood graph. This constitutes a clear limitation, as only a *better* point-dimensionality ratio reduces the influence of this phenomenon. We therefore use a prior dimension reduction to project the data into an intermediate space that is manageable. The linear discriminant analysis (LDA) [7], [10]

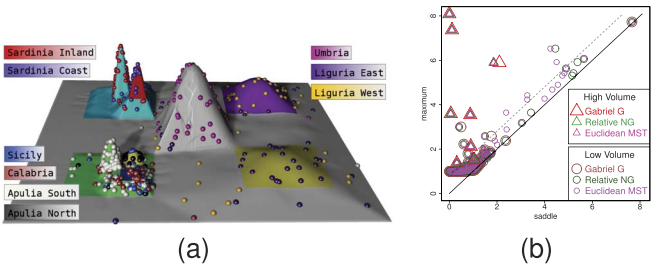


Fig. 14. Italian olive oils: (a) The Topological Landscape tells us that oils form clusters. Moreover, oils from a given growing region share the concentration of fatty acids; but the three regions differ in their acid composition. (b) Persistence diagram highlighting branches of high volume.

is a supervised projection that uses additional classification information to project the data into a lower dimensional space. The LDA optimization criterion for this intermediate space is to maximize the distance between the centroids of the classes, and to minimize the distances from points of one class to their centroid. The reduced dimensionality is $(k-1)$, where k is the number of classes. Applying the LDA to the ISOLET data, we end up with 7,797 points in the $(k-1)=25$D space. This is still a rather high dimensionality for data, and, as we see in Fig. 13a, the common approach to project all points down to 2D still does not answer the question whether clusters are separated or not. Using the same procedure as in the previous example, we get the Topological Landscape shown in Fig. 13c. The structure of the hills tells us that the points really form clusters. The interesting aspect of this example is the question which points form clusters and which clusters are nearby. Semantically, in this feature space points accumulate if they share similar vocal aspects. As we see from the color distribution of the data points, this applies to the letters themselves because they are pronounced similarly by all the 150 subjects. Even more interesting is that clusters, i.e., letters that sound similar, are also near to each other in the 25D space. This is evidenced by the structures whose hills are separated by a rather big saddle value, like the hills of "m/n," "k/j," "b/e/d," "t/p," or "f/s." The computation time for this example was about three seconds (the prior LDA took around 22 seconds). Finally, the distorted landscape in Fig. 13b illustrates the approximated contour sizes.

In the Italian olive oils data set [34], 572 olive oil samples from nine different regions in Italy have been analyzed chemically. The measured features describe the percentage of eight fatty acids: palmitic, palmitoleic, stearic, oleic, linoleic, arachidic (eicosanoic), linolenic, and eicosenoic fatty acid. The nine growing regions are: Sardinia (Inland + Coast), Umbria, Liguria (East + West), Sicily, Calabria, and Apulia (North + South). We will now try to find out whether these oils can be distinguished based on their combinations of the fatty acids. Fig. 14a shows the Topological Landscape that we obtained from our visual analysis process. First of all, the olive oils clearly form clusters and we can distinguish mainly three structured regions in the landscape: the hilly structure in the top-left corner corresponding to the two growing regions in Sardinia, and the bottom-left hierarchical structure corresponding to the growing regions in southern Italy. The third component is a bit harder to identify because the clustering situation is less clear in this area: it is the

Fig. 15. Topological landscape showing all 127,995 points. Hilly structures represent clusters and the surrounding ring of red spheres are low-dense noise points.

combination of the center hill and the subsequent two hills on the spiral (which begins with the Liguria East hill and continues counter clockwise). Although these three hills are not hierarchical, they are not completely separated though. Two essential hints are given by the descending spiral: the contours (i.e., the hills) are not separated by low density, and the contours are neighbored (otherwise they could not have joined). This makes these three growing areas in northern Italy a slight compound, but less significant compared to the other two structures. In summary, we conclude that oils from Sardinia, North Italy, and South Italy are different to some extent. In fact, a deeper statistical analysis of the features reveals that southern oils have much higher eicosenoic acid on average and slightly higher palmitic and palmitoleic acid content. The north and Sardinian oils have some difference in the average oleic, linoleic, and arachidic acids.

The persistence diagram in Fig. 14b shows all branches for different neighborhood graphs. According to Section 5.1, the results are similar if topological noise is ignored. However, instead of pruning branches of low persistence we prune branches of low volume. The reason for this is that sometimes a few points can accumulate to denser regions (of high persistence) than more voluminous groups of points would do. In this case, a persistence-based pruning would eliminate spacious, but less dense clusters and it would preserve small groups of high density (like, e.g., the noise accumulations in the bottom-left-hand corner when using the dotted line). By pruning branches of low volume we eliminate those hills in the landscape that feature only very few points. The volume threshold can be obtained from a histogram.

In our last example, we want to demonstrate both the strength and limitations of our approach. We therefore manually created a sufficiently large and high-dimensional data set with a generator described by Handl and Knowles [35]. The generator uniformly arranges multinomial distributions in a high-dimensional space, allowing different standard deviations for each dimension. After randomly inserting 50 percent extra noise points, our scenario has a total of 127,995 points in 100 dimensions. Regarding conventional visualization alternatives like PCA+2D scatterplot, SPLOMs, or PCPs, such a high-dimensional data set usually leads to huge overplotting of entities and confusing visualizations. In Fig. 15, we see a Topological Landscape of this point cloud. The 20 hilly structures indicate 20 clusters in the guise of separated density maxima. Regarding the computational limitations of our algorithm, some aspects are of importance: At first, the relationship between the number of input points and their dimensionality. In higher dimensions, the number of neighborhood graph edges generally increases quickly, and so do the computational costs. In lower dimensions, where fewer neighborhood graph edges exist, more input data points are manageable. To reduce the overall costs, either the input points or the resulting edges can be reduced. This can be achieved by a prior sampling or by choosing sparser neighborhood graphs. In our given example, we randomly sampled only 10 percent of the data and neglected those points that have less than 10 percent of the maximum density. Furthermore, we used the RNG to describe the neighborhood relationship. The quickest possible result can be achieved by not reinserting the nonsampled points. This process took around 55 seconds. If the user is not only interested in the structure but also wants to visualize the nonsampled data entities, with re-inserting only the skipped low-density points the process takes around 75 seconds, and reinserting all the remaining 90 percent of the original input data (as we did in Fig. 15), the result was available after around 12 minutes. The process gets more expensive if denser neighborhood graphs are chosen. While the RNG computation produced around 25,000 edges in a few seconds, the GG computation resulted in around 15 million edges and took around 4 minutes. While the subsequent upsampling on the RNG took only some seconds, it takes around 32 minutes on all GG edges. Applying the algorithm to the whole input data, i.e., without any sampling, while using the Gabriel graph, certainly would take several hours or even days. Therefore, the basic idea of the optimization steps is to reduce the input data to some 10,000 points for around 50D, or more points for fewer dimensions.

7 Conclusion and Future Work

We presented a multiple stage process to visualize the approximation of a high-dimensional point cloud's topology. This approximation aims at the study of the data's structure in terms of dense regions, including their size and nesting structure. We also presented methods to reduce the runtime of our algorithm. We propose to use moderate random sampling, because it does not change the overall shape of the estimated density function, and therefore preserves its topology. However, since this also increases the impact of noise especially in low-density regions, we also propose to use a sampling that favors regions of high density. Furthermore, we showed that instead of using the Delaunay graph, which is inefficient to compute in high-dimensional spaces, we can use sparser graphs that represent neighborhood relations between points. The join tree's complexity increases with the "sparseness" of the neighborhood graph, but this can be countered with topological simplification that reduces the differences in the final landscapes. However, we cannot attest one neighborhood graph for all cases, as we have observed that the sparseness may only be increased along with the dimension: while for lower dimensions the Gabriel graph should be used, for medium-dimensional data, it should be sufficient to use the sparser RNG or even the EMST.

In the future, we want to investigate automatic approaches to find the most appropriate kernel window width σ. We think that the persistence diagram might be

useful to present a sequence of precalculated join trees for different window widths. Furthermore, we want to study whether other metrics can be used to build meaningful neighborhood graphs. We also consider a substitution of the Gaussian kernel estimation with other estimators that incorporate the local point distribution.

ACKNOWLEDGMENTS

The authors thank Hamish Carr for valuable discussions regarding our method and Gunther H. Weber for assisting our Topological Landscape implementation. This work was supported by a grant from the German Science Foundation (DFG), number SCHE663/4-1 within the strategic research initiative on Scalable Visual Analytics (SPP 1335).

REFERENCES

[1] G. Weber, P.-T. Bremer, and V. Pascucci, "Topological Landscapes: A Terrain Metaphor for Scientific Data," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 6, pp. 1416-1423, Nov./Dec. 2007.

[2] *Graphical Methods for Data Analysis,* J.M. Chambers, W.S. Cleveland, B. Kleiner, and P.A. Tukey, eds. Wadsworth Int'l Group, 1983.

[3] A. Inselberg and B. Dimsdale, "Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry," *VIS '90: Proc. First Conf. Visualization,* pp. 361-378, 1990.

[4] R.A. Becker and W.S. Cleveland, "Brushing Scatterplots," *Technometrics,* vol. 29, no. 2, pp. 127-142, 1987.

[5] A. Tatu, G. Albuquerque, M. Eisemann, J. Schneidewind, H. Theisel, M. Magnor, and D. Keim, "Combining Automated Analysis and Visualization Techniques for Effective Exploration of High-Dimensional Data," *Proc. IEEE Symp. Visual Analytics Science and Technology (VAST),* pp. 59-66, 2009.

[6] I.T. Jolliffe, *Principal Component Analysis.* Springer, 2002.

[7] K. Fukunaga, *Introduction to Statistical Pattern Recognition,* second ed. Academic Press Professional, Inc., 1990.

[8] J.B. Kruskal and M. Wish, *Multidimensional Scaling (Quantitative Applications in the Social Sciences).* SAGE Publications, 1978.

[9] T. Kohonen, *Self-Organizing Maps,* third ed. Springer-Verlag, 2001.

[10] J. Choo, S. Bohn, and H. Park, "Two-Stage Framework for Visualization of Clustered High Dimensional Data," *Proc. IEEE Symp. Visual Analytics Science and Technology (VAST),* pp. 67-74, 2009.

[11] E.J. Nam, Y. Han, K. Mueller, A. Zelenyuk, and D. Imre, "Clustersculptor: A Visual Analytics Tool for High-Dimensional Data," *Proc. IEEE Symp. Visual Analytics Science and Technology (VAST),* pp. 75-82, 2007.

[12] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data," *Data Mining Knowledge Discovery,* vol. 11, no. 1, pp. 5-33, 2005.

[13] C.C. Aggarwal, J.L. Wolf, P.S. Yu, C. Procopiuc, and J.S. Park, "Fast Algorithms for Projected Clustering," *Proc. ACM SIGMOD Conf. Management of Data,* 1999.

[14] L. Kaufman and P. Rousseeuw, *Finding Groups in Data An Introduction to Cluster Analysis.* Wiley Interscience, 1990.

[15] A. Hinneburg and D.A. Keim, "An Efficient Approach to Clustering in Large Multimedia Databases with Noise," *Proc. Knowledge Discovery and Data Mining (KDD),* pp. 58-65, 1998.

[16] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Proc. Second Int'l Conf. Knowledge Discovery and Data Mining (KDD),* pp. 226-231, 1996.

[17] S. Takahashi, I. Fujishiro, and M. Okada, "Applying Manifold Learning to Plotting Approximate Contour Trees," *IEEE Trans. Visualization and Computer Graphics (TVCG),* vol. 15, no. 6, pp. 1185-1192, Nov./Dec. 2009.

[18] S. Fortune, *Voronoi Diagrams and Delaunay Triangulations.* World Scientific Press, pp. 193-233, 1992.

[19] M. De Berg, O. Cheong, and M. van Kreveld, *Computational Geometry: Algorithms and Applications.* Springer, 2008.

[20] G. Jaromczyk and J.W. Toussaint, "Relative Neighborhood Graphs and Their Relatives," *Proc. IEEE,* vol. 80, no. 9, pp. 1502-1517, Sept. 1992.

[21] R.K. Gabriel and R.R. Sokal, "A New Statistical Approach to Geographic Variation Analysis," *Systematic Zoology,* vol. 18, no. 3, pp. 259-270, 1969.

[22] M. Levoy, "Display of Surfaces from Volume Data," *IEEE Computer Graphics and Applications,* vol. 8, no. 3, pp. 29-37, May 1988.

[23] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *SIGGRAPH Computer Graphics,* vol. 21, no. 4, pp. 163-169, 1987.

[24] G.H. Weber, G. Scheuermann, and B. Hamann, "Detecting Critical Regions in Scalar Fields," *Proc. IEEE TCVG Symp. Visualization,* pp. 1-11, 2003.

[25] R.L. Boyell and H. Ruston, "Hybrid Techniques for Real-Time Radar Simulation," *AFIPS '63 (Fall): Proc. Nov. 12-14, 1963, Fall Joint Computer Conf.,* pp. 445-458, 1963.

[26] H. Carr, J. Snoeyink, and U. Axen, "Computing Contour Trees in All Dimensions," *Computational Geometry,* vol. 24, no. 2, pp. 75-94, 2003.

[27] H. Edelsbrunner, D. Letscher, and A. Zomorodian, "Topological Persistence and Simplification," *Discrete and Computational Geometry,* vol. 28, no. 4, pp. 511-533, 2002.

[28] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli, "The Toporrery: Computation and Presentation of Multi-Resolution Topology," *Proc. Math. Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration,* pp. 19-40, 2009.

[29] B.W. Silverman, *Density Estimation for Statistics and Data Analysis,* no. 26. Chapman and Hall, 1986.

[30] Y. Livnat, H.-W. Shen, and C.R. Johnson, "A Near Optimal Isosurface Extraction Algorithm Using the Span Space," *IEEE Trans. Visualization and Computer Graphics,* vol. 2, no. 1, pp. 73-84, Mar. 1996.

[31] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, "Stability of Persistence Diagrams," *Discrete and Computational Geometry,* vol. 37, pp. 103-120, 2007.

[32] A. Frank and A. Asuncion, "UCI Machine Learning Repository," http://archive.ics.uci.edu/ml, 2010.

[33] M.A. Fanty and R. Cole, "Spoken Letter Recognition," *Proc. Conf. Advances in Neural Information Processing Systems (NIPS),* 1990.

[34] M. Forina, C. Armanino, S. Lanteri, and E. Tiscornia, "Classification of Olive Oils from Their Fatty Acid Composition," *Proc. Food Research and Data Analysis,* pp. 189-214, 1983.

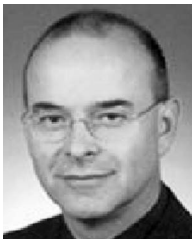
[35] J. Handl and J. Knowles, "Cluster Generators for Large High-Dimensional Data Sets with Large Numbers of Clusters," http://dbkgroup.org/handl/generators, 2005.

Patrick Oesterling received the MS degree (Diplom) in computer science in 2009 from the University of Leipzig, Germany. He is currently working toward the PhD degree at the Department of Computer Science at the University of Leipzig, where his research focuses on computer graphics, information visualization, and visual analytics.

Christian Heine received the MS degree (Diplom) degree in computer science in 2006 from the University of Leipzig. Currently, he is working as a research assistant at the Department of Computer Science at the University of Leipzig. His research interests include scientific and graph visualization.

Heike Jänicke received the MS degree (Diplom) in computer science with a special focus on medical computer science in 2006 from Leipzig University. For her research of information-theoretic methods in visualization, she was awarded the PhD degree in 2009 from the same university. During 2009 and 2010, she worked as a postdoctoral researcher at Swansea University. Since March 2010, she is working as junior professor for computer graphics and visualization at Heidelberg University. Her research interests include data analysis methods from statistics and information theory, analysis of structure and self-organization, visualization of unsteady, multivariate data with applications in biology, medicine, astronomy, and climate research. She is a member of the IEEE.

Gerik Scheuermann received the BS and MS degrees in mathematics in 1995 from the University of Kaiserslautern. In 1999, he received the PhD degree in computer science, from the University of Kaiserslautern. During 1995-1997, he conducted research at Arizona State University for about a year. He worked as postdoctoral researcher at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis, in 1999 and 2000. Between 2001 and 2004, he was an assistant professor for computer science at the University of Kaiserslautern. Currently, he is a full professor in the Computer Science Department of the University of Leipzig. His research topics include algebraic geometry, topology, Clifford algebra, image processing, graphics, and scientific visualization. He is a member of the ACM, IEEE, and GI.



Gerhard Heyer studied Mathematical Logic and Philosophy at Cambridge University (Philosophy Tripos, Christs College 1973-1976, Robert-Birley Scholarship), and General Linguistics at the University of the Ruhr, where he received the PhD degree in 1983. Since April 1994, Gerhard Heyer holds the chair on Automatic Language Processing at the computer science department of the University of Leipzig. His field of research is focussed on automatic semantic processing of natural language text with applications in the area of information retrieval and search as well as knowledge management. He is a member of the IEEE Computer Society.