

Hierarchical Correlation Clustering in Multiple 2D Scalar Fields

Tom Liebmann¹, Gunther H. Weber^{2,3}, and Gerik Scheuermann¹

¹Department of Computer Science, Leipzig University, Germany

²Lawrence Berkeley National Laboratory

³University of California, Davis

Abstract

Sets of multiple scalar fields can be used to model different types of varying data, such as uncertainty in measurements and simulations or time-dependent behavior of scalar quantities. Many structural properties of such fields can be explained by dependencies between different points in the scalar field. Although these dependencies can be of arbitrary complexity, correlation, i.e., the linear dependency, already provides significant structural information. Existing methods for correlation analysis are usually limited to positive correlation, handle only local dependencies, or use combinatorial approximations to this continuous problem. We present a new approach for computing and visualizing correlated regions in sets of 2-dimensional scalar fields. This work consists of three main parts: (i) An algorithm for hierarchical correlation clustering resulting in a dendrogram, (ii) adaption of topological landscapes for dendrogram visualization, and (iii) an optional extension of our clustering and visualization to handle negative correlation values. All steps are designed to incorporate the special properties of correlation consistently. The results are visualized in two linked views, one showing the cluster hierarchy as 2D landscape and the other one providing spatial context in the scalar field's domain. Different color and texturing schemes coupled with interactive selection support an exploratory data analysis.

CCS Concepts

•**Mathematics of computing** → Cluster analysis; Point-set topology; •**Human-centered computing** → Dendrograms;

1. Introduction

Many domains require analyzing sets of multiple scalar fields defined over a common grid. In climate research, seismology, or medicine, e.g., measurements introduce errors, which can be modeled with scalar field *ensembles*, i.e., multiple observations of the same phenomenon. Another important application is identifying structures that persist over longer time spans in time-varying data sets. Across multiple scalar fields, the values at different points of the domain usually have dependencies. In temperature measures taken over the course of a year, e.g., close locations are unlikely to exhibit large value differences. While these dependencies can be of arbitrary complexity, the linear dependency, described by correlation coefficients, already carries valuable information while being simple enough for an efficient analysis.

Since dependency information is given between point pairs, visualizing correlation directly is unfeasible. Instead, one has to reduce correlation information to a manageable amount of features that can be presented to the user. A common approach organizes points into clusters with high linear dependency. This, however, raises a series of new questions, such as: What is the suitable similarity measure to decide whether to group two points into the same cluster? How does one select a suitable similarity threshold that

avoids grouping together too many or too few points? What is the appropriate way of dealing with negative correlation, i.e., points that have high dependency but inverse behavior?

Our work addresses these questions by presenting a new method for hierarchical point clustering and subsequently using it to extract correlated regions in multiple scalar fields. The overall goal is to cover all aspects of correlation, including local and global dependencies as well as positive and negative correlation coefficients. We accomplish this goal by first transforming the scalar fields into a space that better reflects the properties of correlation—the surface of a high-dimensional hypersphere. In this representation, correlation corresponds to angular distance, making a geometric interpretation possible. We then use a new variant of agglomerative clustering, which first reduces the number of considered point pairs through the use of a neighborhood graph before successively merging points depending on an edge measure. We introduce and compare different edge measures as they exhibit different benefits and drawbacks. Furthermore, we propose a new way for consistently handling negative correlation. The result is a hierarchical clustering that contains information about correlated regions, their relation and nesting, as well as inverse dependencies. We use two linked views to visualize the clustering: a 2D landscape and a clus-

ter visualization in the original domain. The landscape provides the cluster hierarchy as well as simple selection and filter mechanics. The domain view shows the correlated regions in a spatial context. This combination enables interactive exploration of correlated structures.

Our main contributions are:

- a hierarchical and flexible correlation clustering algorithm that can easily be generalized to arbitrary point clouds;
- a mathematically consistent approach to handling negative correlation coefficients;
- generalizing topological landscape visualization to arbitrary hierarchical clusterings.

2. Related Work

One of the main instances of multiple scalar fields is ensemble data, which is commonly used to model uncertainty in scalar fields. Several taxonomies [THM*05, PRJ12] and surveys [BOL12, BHJ*14, HLH*16] cover the representation and visualization of uncertainty. Pfaffelmoser and Westermann [PW13] investigated the influence of correlation on the structure of uncertain fields. They showed that correlation is responsible for a significant portion of a field's topological features and used glyphs to encode local correlation and anisotropy information. Sauber et al. [STS06] extracted correlated regions in multi-fields, summarizing pair-wise dependencies in a graph structure. The graph then aids the selection of subsets of correlated scalar fields for which the correlated areas are visualized using standard volume rendering techniques. Chen et al. [CWMW11] reduced the amount of correlation values that have to be considered in time-dependent multi-field data by sampling regions of the correlation matrix. Zhang et al. [ZMZM15] represented correlation as labeled graph and augmented a classical graph visualization with scatterplots for additional dependency information.

Many methods for correlation clustering are based on the work of Bansal et al. [BBC04]. Using a weighted undirected graph with correlation-based similarity measures as edge weights, clustering becomes a problem of maximizing similarity within and minimizing similarity between clusters. There are only few methods that do not use the graph model. Zhang et al. [ZHQL16] defined a correlation-based distance metric for time-varying multi-variate data and used k -means clustering to extract correlated regions. Sukharev et al. [SWMW09] also used k -means clustering but enhanced it with a local search to minimize intra-cluster correlation. Pfaffelmoser and Westermann [PW12] developed an algorithm for correlation clustering in uncertain 2D scalar fields that also deals with negative correlation and is thereby closely related to our work. They used positive and negative correlation thresholds to find regions that have a certain dependency to centroids in the domain. Centroids are those points with the highest cardinality, i.e., the number of points in the domain, whose correlation to the given point is above the threshold. Their method, however, is sensitive to small value perturbations and also tends to be ambiguous by not assigning points to their most correlated cluster.

3. Background

In the following, we describe the data representation before reviewing the two main clustering approaches our method is related to: hierarchical clustering and topological density-based clustering. Furthermore, we briefly introduce topological landscapes as well as methods from density estimation that are used later in this paper.

3.1. Correlation in Multiple Scalar Fields

A deterministic scalar field is a mapping $s : D \rightarrow \mathbb{R}$ over some domain $D \subseteq \mathbb{R}^m$. Values usually are given only at certain grid points $D_S = \{x_1, \dots, x_n\} \subseteq D$ which are connected to form simplicial cells. Linear interpolation for values within cells provides a dense approximation of the function. Multiple scalar fields can be provided as data matrix $X \in \mathbb{R}^{n \times d}$, which contains d scalar fields as columns, each storing values for n grid points. Given d scalar fields over the same domain, the *Pearson correlation coefficient* [BCHC09] $\rho_{x_i, x_j} \in [-1, 1]$ describes the linear dependency between two sample points $x_i, x_j \in D_S$. While $\rho_{x_i, x_j} = 0$ indicates independence, positive/negative correlation signifies that the two variables behave proportional/anti-proportional to each other. All correlation coefficients form the correlation matrix $\text{Corr} \in [-1, 1]^{n \times n}$ with entries $\text{Corr}_{ij} = \rho_{x_i, x_j}$, which is symmetric and positive semidefinite.

3.2. Hierarchical Clustering

Hierarchical clustering is a very common technique with the majority of applications in machine-learning and computational linguistics. Berkhin [Ber06] gives an overview over common clustering methods, including *agglomerative clustering*, which is the category our method falls into. A variant of agglomerative clustering that is related to our method is *single-linkage clustering* [ELLS11], in which points with highest similarity are successively merged until only one cluster is left. The results of a hierarchical clustering are commonly presented as a *dendrogram* [SR62], for which Figure 2 shows a basic example.

3.3. Topological Density-based Point Clustering

Our clustering method is also based on the point cloud clustering developed by Oesterling et al. [OHJ*11]. It operates on a point cloud given in a high-dimensional Euclidean space and consists of two main parts: (i) approximation of the point cloud's density function and (ii) topological analysis using the join tree.

Oesterling et al. approximated the density function by computing its values only at points of the point cloud using Gaussian kernel density estimation. While this gives a smooth approximation, they used a single estimated kernel bandwidth for all points, which leads to granularity problems (cf. Section 6.5). With the representation of the density function, the final clustering is computed through investigation of the density's topology. Cluster candidates are dense regions which can be extracted by looking at superlevel sets of the density function for certain thresholds. Connected components in a superlevel set represent dense regions that are separated by areas with low density. The *join tree* [CSA03] is computed to capture all

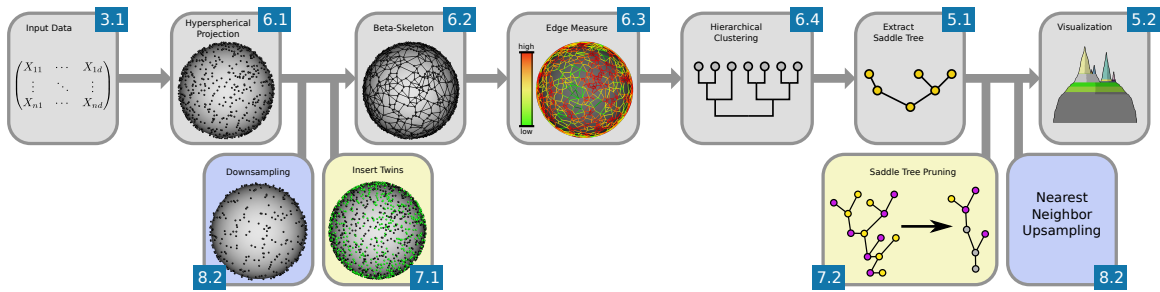


Figure 1: Overview over clustering pipeline and the corresponding paper sections (blue boxes). Yellow and blue parts mark optional steps for handling negative correlation and point count reduction.

connected components of all thresholds at once. A problem with using the point cloud itself to sample the density function is that valleys separating clusters are often not represented in the sampling. This problem is bypassed by an *edge sampling* step, introducing additional sample points on edges with low density values.

3.4. Topological Landscapes

Since join trees can be computed in spaces of arbitrary dimension, they are powerful tools for capturing topological information. One way of visualizing this information is by generating *topological landscapes* [WBP07]. The core idea is to generate a low-dimensional function with the same topological features as the high-dimensional one. Due to the simple structure of the join tree, it can be represented as a landscape profile in two dimensions [OHWS13]. Topological landscape profiles represent the join tree as nested hills with heights indicating the scalar values of the underlying function. Further details on the construction of landscape profiles are given in Section 5.2.

3.5. Density Estimation

When estimating the density function of a point cloud in a metric space, the points are assumed to follow a common density distribution. One of the most basic methods for approximating the density function is *Parzen-window estimation*. It estimates the density by counting all points in a certain area around a point x :

$$p(x) \approx \frac{k}{n \cdot V}, \quad (1)$$

where n is the total number of points, k is the number of points within the region and V is the window's volume.

Kernel Density Estimation (KDE) – For a smoother approximation, the fixed window can be replaced with a weighted *kernel function* c [WJ94]:

$$p(x) \approx \sum_{i=1}^n \frac{c(d(x, x_i), b)}{n \cdot V(b)}. \quad (2)$$

The parameter b is used to alter the shape of the kernel thus influencing the volume V . In case of the commonly used Gaussian kernel, b is called the *bandwidth parameter* and determines the width of the bell-shaped kernel. The choice of b is very critical and challenging as it has a strong impact on the shape and accuracy of the resulting density approximation.

k-nearest neighbor density estimation (kNN) – Instead of counting points within a certain window size, one can fix the number of points k and estimate the volume that encloses exactly k points. This approach is proven to converge towards the real distribution with increasing number of sample points n [FH73]. However, the resulting density tends to be very spiky, which makes this method unsuitable for practical application.

dynamic bandwidth KDE – The kNN can be combined with KDE to solve two problems at once: the choice of the kernel bandwidth and the spikiness of the kNN density. Instead of taking a uniform bandwidth, one can use a dynamic bandwidth that depends on the distance to the k -nearest-neighbor. This way, regions of low density receive a larger kernel that preserves global structure while dense areas retain details with smaller kernel sizes.

4. Overview

Figure 1 gives an overview over all steps of our clustering pipeline together with the corresponding sections. The description of these steps is divided into three main parts. We first describe how topological landscapes can be used to visualize arbitrary dendrograms (Section 5). This is used in the second part, where we describe the main clustering algorithm and its variations (Section 6). To handle correlation consistently, we transform the data into a point cloud embedded on the surface of a hypersphere (Section 6.1). Using beta-skeletons as neighborhood approximation (Section 6.2) makes it possible to merge adjacent regions based on a precomputed similarity measure (Section 6.3). In the third part we describe a novel approach for incorporating negative correlation (Section 7). It uses the surface of the hypersphere to introduce twin points, which leads to cluster symmetries that provide information about negatively correlated regions. We also give a brief performance analysis together with some optimizations (Section 8.1). At the end of the paper, we demonstrate the applicability and typical use cases for our pipeline on three different data sets (Section 9).

5. Visualization of Hierarchical Clusterings

Throughout the literature, hierarchical clusterings are almost exclusively visualized by directly plotting the tree-like structure of the dendrogram (cf. top-left of Figure 2). This method, however, uses space very poorly, as there is significant overplotting at the leaves and only a single line representing the main cluster at the

root. Augmenting the lines with color (e.g. [KH03]) often requires visual parsing and the overwhelming background reduces the number of distinguishable colors [War12].

The 2D topological landscapes by Oesterling et al. [OHWS13] provide much better properties, but are defined only for join trees. In the following, we will describe how dendrograms can be translated into a join-tree-like structure, the *saddle tree*, which we then use to build topological landscapes for arbitrary hierarchical clusterings. This not only makes it possible to add additional information through texture and color (see Section 7.3), but also enables topological simplification to reduce the complexity of the clustering through the saddle tree.

5.1. Saddle Tree

The dendrogram and the join tree of the point-cloud’s density function have some key similarities. Both are defined in terms of a varying threshold that represents similarity between clusters. Furthermore, tree branches represent sets of points with *saddle points* connecting branches at certain similarity thresholds to build a cluster hierarchy. Note that throughout the paper, we use the term *saddle point* for all merge-points in hierarchical clusterings, which is common terminology in Morse theory and scalar field topology.

Despite the similarities, join trees and dendrograms can not be directly identified with each other. All nodes in the join tree, e.g., have a direct representative in the point cloud or scalar field. In the dendrogram, however, only leaves correspond to original data points and all saddles are purely virtual. To bridge the gap between the different meanings of tree-nodes, we first introduce the notion of *trivial clusters* in dendrograms as clusters consisting only of a single point. Using this concept, we can identify points in the join tree and the dendrogram that serve the same purpose. *Regular points* in the join tree represent events of a single point attaching to an already existing cluster. In the dendrogram, these are saddles that merge a trivial with a non-trivial cluster. *Saddle points* in the join tree are points where two existing clusters merge. The counterpart in the dendrogram are saddles merging two non-trivial clusters. Similarly, maxima in the join tree correspond to saddles merging two trivial clusters, i.e., the birth of a non-trivial cluster. The last point left is the global minimum, which has no real representation in the dendrogram as the lowest point always is a saddle.

Since the number of points in the dendrogram and the join tree is usually different, the above identification of the point types is not a one-to-one mapping. However, it shows that the dendrogram can be translated into a join-tree-like structure that only considers the saddle points and thus will be called the *saddle tree* for the rest of the paper.

5.2. Landscape Construction

The construction of the landscape for a given dendrogram can be reduced to a small and simple set of steps, which can be seen in Figure 2. First, we translate the dendrogram into the aforementioned saddle tree. Second, we apply different construction rules depending on the type of the saddle tree node, as shown at the bottom of Figure 2. While these steps only slightly differ from the construction of topological landscapes, there are some important details to

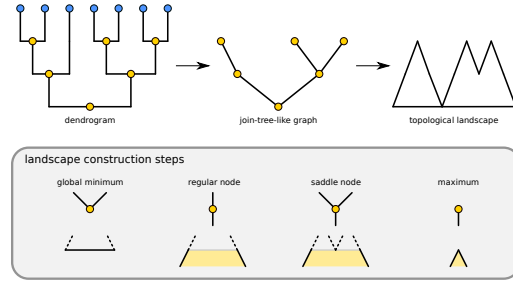


Figure 2: The process of converting a dendrogram into a topological landscape (top) and the construction steps for individual node types (bottom).

take care of. For the landscape to produce peaks at the leaf nodes of the saddle tree, one has to start with a total landscape width of $n - 2$ at the global minimum, with n being the size of the clustered point cloud. At every node v of the saddle tree, the width of the landscape corresponds to the *upper node count*, which is the number of nodes in the upper subtree with root v , excluding v itself. This results in the width of the landscape being two less than the number of points in the corresponding cluster. While this seems undesirable, this difference becomes negligible for bigger point clouds and is required for a consistent construction.

6. Agglomerative Correlation Clustering

This section covers the main clustering algorithm and its variations. The proposed method introduces the concept of *edge measures*, which heavily influence the final clustering and therefore require further investigation. After describing the central approach and the clustering algorithm, we therefore compare different edge measures, discuss advantages and disadvantages, and give an overview over noise stability. Finally, we put the algorithm into perspective between existing methods to highlight major similarities and differences.

6.1. Approach

The main goal of the proposed clustering method is to preserve all information that is contained in correlation coefficients. Because correlation is a cosine-measure, it is directly related to *angular distance* between points, which is given by

$$d_{\text{ang}}(x_i, x_j) = \arccos(\rho_{x_i, x_j}) \in [0, \pi]. \quad (3)$$

This leads to the following geometric interpretation: Starting with two points x and y , their angular distance geometrically can be seen as the angle between two points on the unit circle. A third point then can be added by moving from the unit circle to the unit sphere. Continuing, the angular distances and thus the correlations between n points can be geometrically represented by embedding n points on the surface of a n -dimensional unit hypersphere S^{n-1} . This embedding is what we call *hyperspherical projection* as every grid point has exactly one representative on the unit hypersphere. These points can be extracted by decomposing the positive semi-definite correlation matrix $\text{Corr} \in \mathbb{R}^{n \times n}$ into $\text{Corr} = AA^T$ with $A \in \mathbb{R}^{n \times d}$. The inner products of all pairs of row vectors $a_i \in \mathbb{R}^d$ then correspond to the correlation values $\rho_{x_i, x_j} = \langle a_i, a_j \rangle$ between two points

x_i and x_j . From the diagonal entries $\text{Corr}_{ii} = \langle a_i, a_i \rangle = 1$ it follows that the row vectors a_i are the aforementioned hyperspherical unit vectors. Note that the dimension d often is much smaller than n , which is caused by dependencies in the data that reduce the rank d of the correlation matrix. This can also be used to deliberately reduce the dimension by computing A using *Principle Component Analysis* [Hot33].

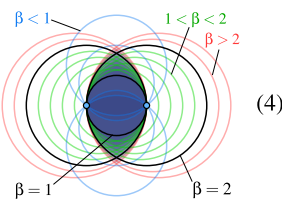
While the interpretation of correlation as angular distance is not a new concept, hardly any clustering methods explicitly make use of it. To make use of the spatial relation of hyperspherical points, our algorithm first approximates the hyperspherical surface with a neighborhood graph. We then merge adjacent points into clusters using an *edge measure*, which leads to a hierarchical representation in the form of a dendrogram.

6.2. Neighborhood Approximation

We investigated several different options for approximating the neighborhood of the hyperspherical point cloud. This search was driven by three major requirements: the neighborhood structure has to *approximate the space* with a suitable density, it has to be *computational feasible* for real-world data sets, and there has to be a way to estimate the *stability* regarding small changes in the input data. First, we will give a brief overview over possible options before focusing on *beta-skeletons* and how we translate them to the hyperspherical surface.

Oesterling et al. [OHJS10] use either the Gabriel graph (GG) or the Relative Neighborhood graph (RNG) in a high-dimensional Euclidean space. Since the GG is a much denser supergraph of the RNG, using both graphs makes it possible to adjust the tradeoff between accuracy and performance. Unfortunately they still differ too much to serve as estimates for output stability, which is why a more fine-grained control is preferable. Jaromczyk and Toussaint [JT92] provide an overview of graphs related to the RNG varying in dimensionality and complexity. Furthermore, stochastic graphs, like the relaxed Gabriel graph introduced by Correa and Lindstrom [CL11], can act as stability indicators by looking at variations over multiple runs.

We selected *beta-skeletons*, which are a generalization of the GG and RNG that was first introduced by Kirkpatrick and Radke [KR85]. Not only do they provide a continuous parameter for trading performance for accuracy, but their deterministic construction makes it possible to estimate stability reliably. A beta-skeleton over a set of points is defined in terms of a single parameter $\beta \in [0, \infty)$. Two points $x_i, x_j \in D_s$ are connected, if and only if their beta-lune L_{ij}^β does not contain a third point $x_k \in D_s \setminus \{x_i, x_j\}$. For $\beta \geq 1$ the beta-lune is the intersection of two d -balls defined by

$$\begin{aligned}
 L_{ij}^\beta &= B_{r,\beta}((1-\alpha)x_i + \alpha x_j) \\
 &\cap B_{r,\beta}(\alpha x_i + (1-\alpha)x_j), \\
 r^\beta &= \alpha \|x_i - x_j\|,
 \end{aligned} \tag{4}$$


where $\alpha = \beta/2$ and $B_r(c) = \{x \in \mathbb{R}^d \mid \|x - c\| \leq r\}$ is the ball with

radius r centered around the point c . On the right, some 2D balls and their lunes (intersections) for different ranges of β are shown.

Using angular distances, the same ball definition applies to the hyperspherical surface, which implies a valid beta-lune definition. Because the norm of every point on the unit-hypersphere is equal to 1, the beta-lune test simplifies to

$$\begin{aligned}
 &(2 - \beta)(1 - \langle x_k, x_i \rangle) + \beta(\langle x_i, x_j \rangle - \langle x_k, x_j \rangle) < 0 \\
 \wedge &(2 - \beta)(1 - \langle x_k, x_j \rangle) + \beta(\langle x_i, x_j \rangle - \langle x_k, x_i \rangle) < 0.
 \end{aligned} \tag{5}$$

All inner products $\langle x_i, x_j \rangle = \rho_{x_i, x_j}$ can be either retrieved by a single lookup in the correlation matrix or computed on-the-fly with d multiplications. The beta-skeleton is equivalent to the Gabriel graph for $\beta = 1$ and to the relative neighborhood graph for $\beta = 2$. We only use values $\beta \in [1, 2]$ since $\beta < 1$ increases computation times dramatically and $\beta > 2$ results in a disconnected graph. An example for a spherical beta-skeleton with $\beta = 2$ can be seen in Figure 1.

While a naive implementation of the beta-skeleton has complexity of $O(n^3)$, there are some tricks that bring computation times closer to $O(n^2 \log(n))$. Due to the direct relationship to the Gabriel graph, we can use the improvements proposed by Oesterling et al. [OHJS10]. When testing whether a pair of points x_i, x_j is connected, all points x_k are traversed with increasing distance to x_i . This way it is very likely that the beta-lune-test fails early since points causing that failure are usually close to the testing pair.

6.3. Edge Measure

The clustering method uses the beta-skeleton to successively merge adjacent regions. The merge order is determined by a graph *edge measure*, which is the likelihood of merging the two regions that are connected by the corresponding edge. Per definition, the edges of a beta-skeleton represent a small portion of the domain due to the beta-lune not containing any points from the point cloud. Although these lunes may overlap, they give every edge a representation in the domain itself, making the question for a suitable edge measure more meaningful: How likely is it for the two by the edge connected regions to merge over the portion of the domain this edge represents?

We investigated multiple different measures, each having advantages and disadvantages. For simplicity, we will first use the simplest measure, the edge correlation, and discuss more advanced ones in Section 6.5. The *edge correlation* for an edge $e = (x_i, x_j)$ is the correlation $\mu_{\text{corr}}(e) = \rho_{x_i, x_j}$ between the two end points. As it is a cosine-measure, it fulfills the requirements for a merging threshold, as points with higher correlation should be more likely to merge into a single cluster.

6.4. Clustering Algorithm

After computing the neighborhood graph and assigning merging thresholds to every edge, the final clustering algorithm is quite simple. Appendix 2 gives a detailed description of the algorithm.

Initially, there are n clusters, each consisting of a single point. The main loop iterates over all edges of the neighborhood graph in descending order of their edge measures. For every edge, there are two cases: (a) The hyperspherical points connected by the edge

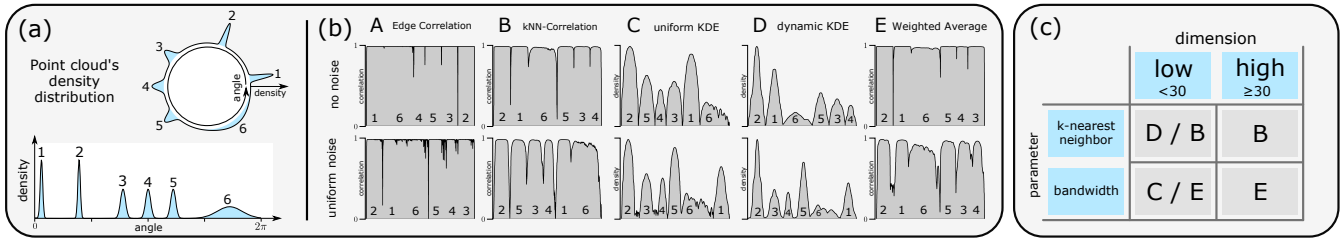


Figure 3: Agglomerative correlation clustering on a synthetic 2D point cloud. 5000 points were generated following a Gaussian mixture density distribution (a)(bottom), which is defined on the circle (a)(top). Different edge measures (b) lead to different cluster separation and noise stability. Note that in (A)-(E), the vertical axis represents a correlation or density measure while the plot width equals the number of points. As there is no single best measure for all data types, we give a basic recommendation for choosing an edge measure (c).

are in different clusters; (b) The connected points already share the same cluster. In case (a), the two clusters the points belong to are merged into a single cluster. In case (b), the edge is simply skipped as it is just an intra-cluster connection. Throughout the process, we have to keep track of the existing clusters as well as the merging process to build the dendrogram. Clusters can be tracked efficiently using the UnionFind [Tar75] data structure. The dendrogram is built incrementally using an additional array of cluster representative nodes. After initializing the dendrogram with n nodes, a new one is added every time two clusters are merged. It becomes the representative of the merged cluster and gets connected to the representatives of the old clusters. After traversing all edges, the dendrogram contains a complete representation of the cluster hierarchy with $2n - 1$ nodes.

6.5. Alternative Edge Measures

While taking the end point's correlation as edge measure produces valid hierarchical clusterings, the result is not as novel as it might seem. In fact, it produces the same result as single-linkage clustering, for which we provide a proof in Appendix 1. Since the results are the same, our clustering also inherits all of the flaws like high noise sensitivity and the tendency to produce long thin clusters [Fos98] [NJB06]. The benefit of our algorithm lies in its flexibility as one can easily replace the edge correlation measure with one providing better properties. The choice of the correct measure highly depends on the data complexity and what features the user is interested in.

Figure 3 gives an overview over the measures that are described in the following section. To compare measures, we generated a 2-dimensional data set with 5000 points following the density distribution shown in (a). At the bottom, the plain density distribution as a mixture from six Gaussians can be seen while at the top we plotted the same density onto a circle to highlight the periodicity of the circular data. In (b), figures (A)-(E) show the data clustered with the different edge measures presented in this work. While the top row uses the normal point cloud, the bottom row contains additional 1000 points of uniformly distributed noise. This way, we can evaluate the stability of the proposed clustering algorithm to small value perturbations depending on the choice of the edge measure. Finally, Figure (c) provides a very simplified recommendation for the measure choice depending on data dimension and the user's parameter preference.

In (A), the high noise sensitivity of the edge correlation measure can be seen. Not only does the bottom row show large grooves splitting the clusters, but also the top lacks a clean separation between clusters (1) and (6), which is caused by single outlier points being enough to merge both clusters.

Edge sample point – For a given edge between two points x_i, x_j , we introduce the *edge sample point* x_e as the edge's center point $x_e = \frac{x_i + x_j}{\|x_i + x_j\|} \in S^{d-1}$. Note that the normalization is required due to x_e being the geodetic center point. It is used by all edge measures below and makes it possible to compute correlation values and angular distances efficiently as the correlation between x_e and a point $x_k \in S^{d-1}$ is given by $\rho_{x_e, x_k} = (\rho_{x_i, x_k} + \rho_{x_j, x_k}) / \sqrt{2 + 2 \cdot \rho_{x_i, x_j}}$.

6.5.1. k-Nearest-Neighbor Correlation

The main problem with the edge correlation measure is that it only considers a single edge and completely ignores the global point cloud structure. To solve this, we first rephrase the measure for a given edge $e = (x_i, x_j)$:

$$\begin{aligned} \mu_{\text{corr}}(e) &= \rho_{x_i, x_j} = \cos(d_{\text{ang}}(x_i, x_j)) \\ &= \cos(2 \cdot d_{\text{ang}}(x_e, \text{NN}(x_e))), \end{aligned} \quad (6)$$

where $\text{NN}(x_e)$ is the nearest-neighbor of the edge sample point x_e and d_{ang} is the angular distance as defined in Equation 3. This way, the measure can easily be generalized by replacing the nearest-neighbor with the k -nearest-neighbor:

$$\mu_{\text{kNN}}(e) = \cos(2 \cdot d_{\text{ang}}(x_e, \text{kNN}_k(x_e))). \quad (7)$$

With $k \in \{1, 2\}$ we get the single-linkage clustering as the k -nearest-neighbor would be x_i or x_j . As shown in Figure 3(a)(B), $k > 2$ leads to a smoothing effect, which increases cluster separation and noise stability. The strength of the effect, however, depends on the parameter k . A general rule of thumb that was also used throughout Figure 3 is $k = \sqrt{n}$.

6.5.2. Density Estimation

The point cloud's density function can also be used to get an estimate on the merging threshold for a specific edge. By definition, the density function has small values in regions of high point separation and high values in areas containing dense point clusters. Using the edge sample point, we evaluate the density value for every edge using the different estimators introduced in Section 3.5.

To compute the cluster hierarchy, our algorithm only considers the order of edges with respect to their measure and the actual measure values only influence the values of the final dendrogram. Therefore, it can be shown that the result of ***k*-nearest-neighbor density estimation** is just a distorted version of *k*-nearest-neighbor correlation from the previous section. Because correlation values are much more expressive and the volumes of windows on the hypersphere are numerically hard to compute, using *kNN*-correlation is more suitable in this context.

Kernel density estimation usually gives more accurate results when it comes to approximating the distribution of points. To apply it to the hypersphere, two components have to be specified: the *kernel function* and the *kernel bandwidth*. On the hypersphere, the von-Mises-Fisher-distribution serves as a counterpart to the common Gaussian distribution in Euclidean spaces. When it comes to bandwidth selection, the scale of the point cloud in Euclidean spaces usually is unknown, which requires advanced techniques for bandwidth estimation. On the hypersphere, however, we benefit from the limited value range, making 0.2 a good entry point for further manual bandwidth refinement. A major problem on the hypersphere is the normalization of the kernel function, as it is highly unstable for smaller bandwidths and higher dimensions. With a uniform bandwidth, this normalization can be avoided as it only introduces a constant factor to the density and does not affect the value order. Unfortunately, a single bandwidth focuses on a specific feature granularity but is unable to capture fine detail and global structure simultaneously, which can be seen in Figure 3(b)(C). A good cluster separation of the smaller clusters requires a small bandwidth, which causes cluster (6) to start splitting up.

To overcome this, we also implemented the **dynamic bandwidth selection**, which uses the distance to the *k*-nearest-neighbor to assign individual bandwidths to all points. This smoothes areas with low density for the benefit of global connection while smaller bandwidths preserve details in dense areas. In Figure 3(b)(D), cluster (6) is much more stable and the overall noise stability increased as well. To weigh points equally, kernel normalization is crucial which makes this method applicable only for low dimensional data.

Although density estimation as edge measure produces valid clusterings, there are still two major problems left. First, the density values do not transport much useful information about correlation. Since the density function is a random distribution function, its values represent likelihoods rather than separation thresholds. The second problem is that in the final 2D landscape, cluster size is represented in two ways. By construction, the width of a hill directly reflects the number of points inside a cluster. In the density function, however, hills with more points also have higher density values. A clear separation between point count and merging measure would not only make the visualization more readable but also prevent big clusters from suppressing smaller ones solely because of cluster size.

6.5.3. Weighted Average Correlation

To resolve double representation of the point count, we developed the *weighted average correlation* measure, which can be seen in

Figure 3(E). It is defined by

$$\mu_{\text{wavg}}(e) = \cos \left(\frac{\sum_{k=1}^n c(d_{\text{ang}}(x_e, x_k), b) \cdot d_{\text{ang}}(x_e, x_k)}{\sum_{k=1}^n c(d_{\text{ang}}(x_e, x_k), b)} \right), \quad (8)$$

where c is a 1-dimensional Gaussian kernel function with bandwidth b . The idea is to use the average correlation in a local neighborhood around the sample point as indicator for cluster coherence. If there are only few points close by, the average angular distance increases and thus the cosine as correlation measure decreases. The kernel function therefore lowers the influence of distant clusters thus reducing the problem of point count influencing correlation strength. As with all highly localized measures, smaller bandwidths can in some instances fail to capture global cluster relations. When testing weighted average correlation on the temperature data in Section 9.2, e.g., the expected negative correlation between northern and southern hemisphere got lost with smaller bandwidths. Weighted average correlation should therefore not be used to investigate global dependencies in high dimensions.

6.6. Relation to Existing Methods

In this section we put our clustering method into context by discussing the main relations to existing methods.

With our method successively merging points into a hierarchy of clusters, it falls into the category of **agglomerative clustering**. In most methods, the next cluster merge is decided based on some similarity measure which in theory is given between every pair of clusters. Because the number of all possible pairs grows quadratically with the number of points, the algorithmic challenge lies in the efficient tracking of only the relevant values. Our method does so by explicitly making use of the point cloud's structure by first computing the beta-skeleton. This reduces the set of relevant edges to those that claim a small portion (beta-lune) of the underlying domain thus express adjacency of regions. Furthermore, we avoid updating similarity measures by precomputing the edge-measures based on local or global point set properties. By considering the point cloud's density, e.g., we have a measure of cluster similarity beforehand without actually merging points into clusters.

This also shows the similarity to **density-based clustering**. Oesterling et al. [OHWS13] perform clustering by using the points themselves as samples for the common density function. While this makes sense for capturing more dense areas with more points, it actually misses a key aspect of clustering—the separating valleys. Because the areas separating the clusters are by definition poorly sampled regions, the main focus should not be the points themselves but the regions between them. Oesterling et al. try to solve this problem by performing edge-sampling, i.e. introducing new sample points on neighborhood graph edges to find valleys. This, however, introduces inconsistency as the point cloud now consists of two different types of points that are semantically different. Instead of computing density on the core point set, our method focuses on the edges, which comes with some major benefits. First, the method becomes more efficient as we do not have to consider measures at the point cloud elements themselves. Second, not having two types of sample points also leads to greater consistency and flexibility in the choice of the edge measure, like the *k*-nearest-neighbor or weighted average measure. Furthermore, the dendrogram is guaranteed to contain

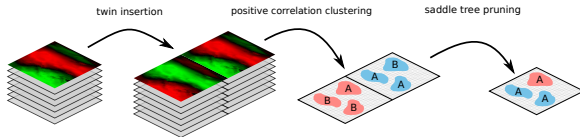


Figure 4: Process of handling negative correlation for an exemplary set of scalar fields. By first inserting twin points with inverse behavior, positive correlation clustering yields correlated regions in both parts of the domain. A final merging step produces a consistent clustering that incorporates positive and negative correlation.

exactly $n - 1$ saddle points while edge-sampling introduces a previously unknown amount of saddles.

7. Negative Correlation

In this section, we describe how to incorporate negative correlation in our clustering method as well as the landscape visualization. While positive correlation can be found in almost all data sets, inverse linear dependency is usually less common. The main reason is the strong connection between point proximity and positive correlation, i.e., points with a small distance in the domain usually have similar values across different scalar field instances. This makes finding inverse correlation using localized methods impossible, which is why many methods do not handle it at all. Due to it being a global feature, negative correlation also can not easily be explained by point proximity, which makes it even more interesting for an exploratory analysis. Methods that handle negative correlation usually just omit the sign and take absolute correlation values instead. While this is possible in terms of correlation as similarity measure, it also comes with some drawbacks. First, this strategy is not applicable to our method, as it is not clear how to embed the points on the hypersphere when only some correlation values are changed. While one could argue that this is a problem that our method introduced, it actually unveils the mathematical inconsistency of this approach. Just omitting the sign destroys the positive semi-definiteness of the correlation matrix, which also breaks the metric properties of angular distances.

In the following, we introduce a different approach to dealing with negative correlation that is consistent with our method and preserves the properties of correlation coefficients. While this approach was mainly developed to support our clustering, it is general and simple enough to be applied to other clustering methods as well, making them mathematically more consistent.

7.1. Approach

Our method handles correlation by mapping all points of the domain to the hypersphere through the use of hyperspherical projection (cf. Section 6.1). Points with high positive correlation have projections in close proximity while negatively correlated points will be located on opposite sides. Instead of changing the correlation values, we add a twin point for every hyperspherical point that lies exactly on the opposite side of the sphere's surface. This maintains a consistent hyperspherical point cloud while transforming negative correlation coefficients into positive ones by essentially doubling the number of points. The clustering pipeline as well as

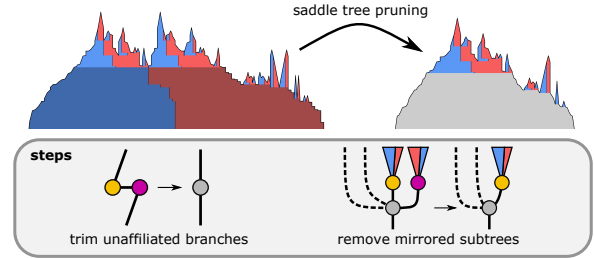


Figure 5: Reduction of the clustering using the symmetry introduced by twin insertion. Only two rules (bottom) are required in order to modify the clustering (top) to only contain one representative for every grid point. Yellow and purple nodes represent mirrored neighborhood graph edges while red/blue coloring marks the ratio between original and twin points in a cluster.

the final visualization can be applied without any changes, although we introduce some modifications in order to improve performance and provide additional information about inverse correlation.

A different way to describe the twin insertion process and its effects it outlined in Figure 4. First, a twin with exact inverse scalar values is inserted for every grid point, which essentially duplicates the domain. After performing positive correlation clustering, we get clustered regions that can span the extended part (red) as well as the original part (blue). In this example we have two clusters A and B that are represented on both sides and also have a very prominent symmetry. This is used in our *saddle tree pruning* step to not only reduce the domain to its original size, but also extract information about negative cluster correlation.

7.2. Saddle Tree Pruning

Since the twin insertion mirrors all points to the other side of the hypersphere, the point cloud becomes symmetrical. This symmetry also extends to the neighborhood graph creating pairs of edges on opposite sides. In the following, for illustration purposes we color original points red and their twins blue. Furthermore, we randomly assign yellow and purple to all mirrored edge pairs.

Figure 5 (top) shows an exemplary cluster landscape for a point set with twins. At every height, we mark the ratio between original points and twins in the corresponding cluster with color and a vertical separation line (cf. Section 7.3). There are two major observations: (i) Red/blue hills always have a mirrored counterpart and (ii) there is a dark branch from which all mirrored hills originate. The light colored hills mark clusters that are on one side of the hypersphere and, due to symmetry, have a mirrored counterpart on the other side. The dark branch, on the other hand, marks clusters that span both sides of the hypersphere, which is why we refer to them as *unaffiliated branches*.

As shown in the overview (Figure 4), the goal is to merge the clustering such that there is only one representative for every domain point left. This is achieved by modifying the saddle tree with two simple simplification rules: (a) remove one of the mirrored edges in unaffiliated branches and (b) remove one of each pair of mirrored subtrees. This can be achieved by first making sure that during edge measure ordering, pairs of mirrored edges are right

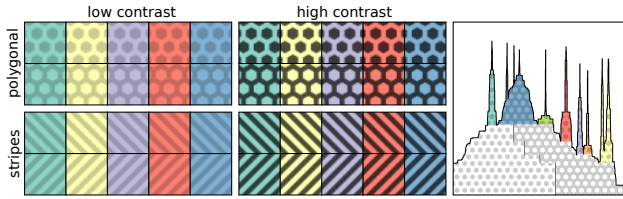


Figure 6: Different texture patterns used to highlight the type of correlation (left) and an exemplary topological landscape with polygonal texturing (right). Color marks different selected clusters and the user can freely specify contrast to either put focus on color or texture.

next to each other, which usually is implicit since they share the same measure value. This reduces the number of cases to consider to only the two shown in Figure 5 (bottom). Mirrored edge pairs on unaffiliated branches (left) can now be merged and labeled *unaffiliated* (gray). If an edge pair is spread across different subtrees (b), there has to be another node in the saddle tree connecting those trees together. We track down this node and remove one subtree to get the final reduced saddle tree.

7.3. Visualization

We modify the dendrogram landscape in order to incorporate negative correlation into the visualization. After saddle tree pruning, every point is represented only once in the clustering. Clusters, however, contain a mixture of original and twin points, which we visualize in two ways. First, we draw a vertical separation line as shown in Figure 6 to mark the ratio of the two point types. Second, we differentiate original from twin points using texture patterns. Note that due to unaffiliated branches lacking information about negative correlation, they neither are textured nor have a separation line. To build the connection between landscape and domain view, the same texture pattern is used. For different types of domains we use the two different texturing options shown in Figure 6. Differently oriented stripe patterns can differentiate between negatively correlated clusters as introduced by Pfaffelmoser et al. [PW12]. While this works for flat 2D domains, it is impossible to find two distinct directions for arbitrary 2D manifolds, e.g. a sphere's surface. MacEachren [Mac04] lists a range of visual variables that can be varied to distinguish between different map regions, including grain, arrangement, and also orientation. Besides orientation, however, none of these variables seem fit to encode equal-rank features in a distinctive manner. We therefore chose a texturing option that is mainly based on color and also directly reflects the underlying domain—the polygonal pattern. The polygonal pattern uses the surface triangulation and swapped foreground and background colors to separate negatively correlated clusters. However, this texture does not support large domains as polygons get too small. Furthermore, we provide an option to vary the contrast between foreground and background color. This allows to either put the focus on cluster affiliation (color) or negative correlation (texture).

8. Usability and Performance

This section focuses on our analysis tool for correlation-based data exploration. First, we discuss two important optimizations that ex-

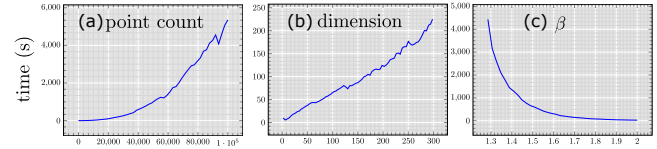


Figure 7: Computation times with varying parameters. The method depends polynomially on point count (a), almost linearly on dimension (b), and inverse exponentially on neighborhood graph complexity β .

tend the applicability of the proposed clustering pipeline. After that, the basic analysis setup is described before we give a brief analysis on the remaining parameters and performance.

8.1. Optimizations

To extend the applicability of our method, we use several optimizations that deal with the main performance bottlenecks. The most influencing factor throughout the whole pipeline is the **number of grid points** n . Since correlation often has a strong connection to point proximity, sampling can be used to reduce the point count significantly. First, a subset of the point set is chosen randomly, on which the whole clustering pipeline including the final topological landscape is computed. At the end the missing points are assigned to the same cluster as their nearest hyperspherical neighbor, i.e., the point they have the highest dependency to.

Even with sampled data, the quadratic **memory consumption** of the correlation matrix can exceed the available main memory. Instead of storing correlation values explicitly, one can compute them on-the-fly with an inner product of the row vectors of the data matrix (see 3.1). Since every correlation value has to be computed with an d -dimensional inner product, on-the-fly computation introduces a runtime penalty, which we investigate in Section 8.3.

The introduction of twin points for dealing with **negative correlation** values doubles the point count. However, many of the clustering steps can be modified to make use of the point cloud's symmetry, which leads to an overall linear slowdown. Edge measures, e.g., only have to be computed for half the edges since the other half is completely symmetrical thus sharing the same measures. The edge symmetry can also be used to speed up the beta-lune check and edge sorting.

8.2. Analysis Setup

Our tool consists of two linked views: one with the 2D landscape and one showing the grid of the scalar fields. The landscape view provides basic exploratory tools for simplification and selection. Three sliders allow for *topological simplification* filtering branches by point count or feature significance [CSvdP04]. Through the selection of hills, the user then can focus on single features, for which the corresponding regions in the domain view are highlighted to provide spatial context. With the distinct color schemes provided by ColorBrewer [HB03], the user can choose to assign a unique color to each selected cluster or to each individual superarc, i.e., each region between two nodes in the dendrogram. Although caus-

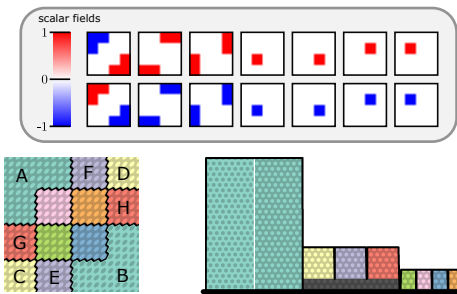


Figure 8: A set of synthetic scalar fields (top) and the resulting hierarchical clustering (bottom). The domain view (left) exactly resembles the modeled structures and the landscape (right) shows cluster nesting information.

ing a lot of visual clutter, the second method emphasizes cluster nesting and global structure, which provides a good overview.

8.3. Performance Measures

We applied our pipeline to a synthetic data set in order to measure the influence of the three main parameters: point count n , dimension d , and the neighborhood graph parameter β . With default values of $n = 10,000$, $d = 30$ and $\beta = 2$, points were generated such that their hyperspherical projections follow a uniform distribution. Figure 7 shows the results when varying one parameter at a time. The runtime dependency for the point count (a) is approximately the expected $O(n^2 \log(n))$ that comes from the neighborhood graph computation. In (b), the almost linear dependency on the data dimensionality shows that our method is fairly robust to complex data. While a precomputed correlation matrix can make the actual clustering method independent from dimension, we specifically performed on-the-fly computation of correlation values to show worst-case performance. Finally, Figure (c) shows that for smaller β , the number of edges and also the number of lune-tests grows exponentially. In practice, however, the impact of the chosen β is rather small thus allowing for selecting $\beta = 2$ and only using smaller values for stability estimation.

9. Results

In this section, we will demonstrate the applicability using three examples that cover different types of data. First, we present a synthetic data set to show the correctness of our approach. The other two data sets are real world data sets that highlight different aspects, i.e., a large domain and high dimensionality. For spatial context, we enhanced some figures with borders provided by the OpenStreetMap project [linc].

9.1. Synthetic Data Set

Our first example is an ensemble of 14 scalar fields shown in Figure 8 (top) defined on a regular 20×20 grid. The columns contain pairs of inverted realizations, which ensures an overall mean of zero at every grid point and makes it possible to model correlated regions explicitly. In the right-most column, e.g., the non-zero

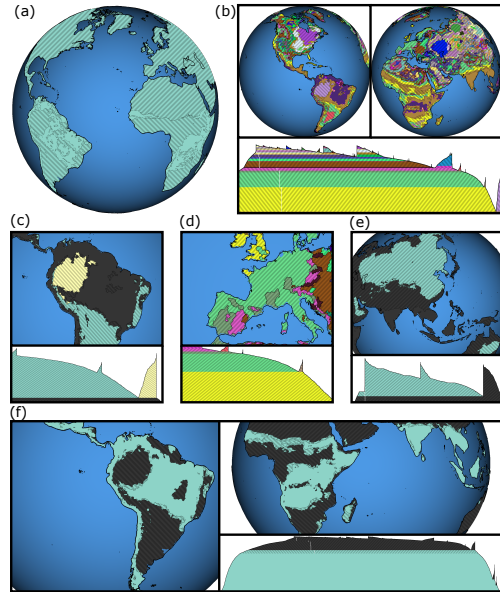


Figure 9: Hierarchical correlation clustering for temperature data. While the overview (a,b) shows global structure and cluster shapes, a closer investigation through selection and highlighting mechanics (c-f) unveils different correlation structures, such as prominent clusters, negative correlation, or independent regions.

valued points are zero in all other fields and therefore create an independent region with positive correlation.

The clustering shown in Figure 8 (bottom) was computed with uniform kernel density estimation as edge measure with a bandwidth of 0.2, as suggested in Section 6.5. We chose the polygonal pattern to highlight inverse correlation and selected the most significant hills for color highlighting. The domain segmentation into correlated clusters exactly resembles the regions modeled with the set of realizations. While there are four independent regions in the center, region A and B are negatively correlated, indicated by having the same color but inverted brightness. Furthermore, there is a positive correlation between the region pairs (C,D), (E,F), and (G,H), which is also reflected in the topological landscape. Since regions (C,D), (E,F), and (G,H) are subbranches of the same parent and have a dark background, they are correlated positively.

9.2. Temperature Measures

The second data set consists of 12 scalar fields that resemble the monthly average temperatures of the year 2016. It is provided by the European Center of Medium-Range Weather Forecast [linb] and is originally defined on a WGS84-projected regular grid. We applied two pre-processing steps: (i) we limited the data to land-masses and (ii) we resampled the scalar fields on a subdivided icosahedron in order to improve correspondence between point count and surface area. The resulting data set consists of 190,000 points, of which we take a 10% subset for cluster computation (cf. Section 8.1). Following the recommendation in Figure 3, the clustering in Figure 9 was computed with k -nearest-neighbor correlation measure ($k = \sqrt{10\% \cdot n} = 139$) and $\beta = 2$.

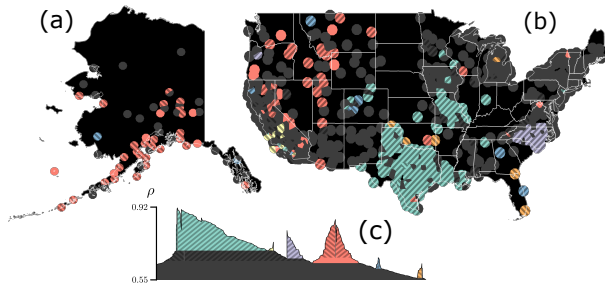


Figure 10: Hierarchical correlation clustering for the CORS data set (c) and the spatial context in the domain view (a,b). Close-ups on Alaska (a) and the contiguous US (b) reveal strong spatial dependencies, which can also be seen in the clustering landscape (c).

Figures (a) to (f) show different observations that can be made using the selection and highlighting mechanics of our analysis tool. In (a), the full clustering was given the same color, revealing the very prominent negative correlation between the northern and southern hemisphere through different stripe orientations. To get an overview over the general clustering structure, we use topological simplification to remove small clusters and color remaining parts by landscape segment (cf. Section 8.2), which can be seen in (b). Although producing an overwhelming amount of colors, this view gives a good impression on cluster shapes and aids further investigation. On the very right of the landscape, e.g., one can see a clearly separated purple cluster, for which a closeup with corresponding selection is shown in (c). Because of the significant height of the hill, this cluster, that is bordered by the Andes to the north-west, can be seen as a well-separated, stable feature with common temperature behavior. The opposite can be seen in (d), where the whole area of Europe is just represented by regular points in the landscape that attach to an already existing hill rather than creating its own. Therefore, Europe does not exhibit large topographical dependency when it comes to temperature variation, which is further emphasized by it being located rather low in the landscape. The highest parts of the landscape, on the other hand, are shown in (e) and, due to the high simplification, span huge parts of North-East Asia. Also note the small part in the lower left of the selected landscape, which represents the part of Australia, that has a high negative correlation indicated by the different stripe orientation. Finally, we selected the very bottom of the landscape in (f), which unveils the parts of the world that are mainly independent. The lack of a stripe pattern also shows that these parts belong to an *unaffiliated branch* in the landscape (cf. Section 7.2), for which no clear statement about the type of the correlation can be made. As expected, these parts are mainly located in the equatorial regions with some exceptions being coastlines on the east of the pacific ocean.

9.3. GPS Reference Stations

The last data set consists of GPS altitude measures from Continuously Operation Reference Stations (CORS) provided by the National Oceanic and Atmospheric Administration (NOAA) [lina]. 366 scalar fields, one for every day of the year 2016, contain 815 altitude measures of reference stations that are distributed over the US. GPS altitude estimation is known to be very inaccurate, which

causes the standard deviations of some stations to be as high as 40m over the course of the year. Sources of this uncertainty are of widely different nature, ranging from atmospheric effects affecting the signal speed, the local weather, sensor quality, and several terrain-based effects, such as multipath issues and signal reflection.

Since this is a high-dimensional data set with a small number of sample points, we used the k -nearest-neighbor edge measure (cf. Section 6.5.1) with a small $k = 5$. Figure 10 shows the clustering landscape (c) as well as the spatial context in Alaska (a) and the contiguous US (b). Note that even though the data consists of sample points with scalar values, there is no grid connecting the points to a dense domain. We therefore draw a circle around every point to serve as canvas for texturing and cluster colors. Occlusion is resolved by actually drawing cones in a top-down view, which ensures that every point is visible even in dense areas.

Since influencing factors for GPS data are mainly of topographical nature, we expected correlation to follow terrain properties. However, to our surprise the significant clusters mainly resemble state borders. The turquoise cluster, e.g., directly follows the borders of Texas, Louisiana, and Missouri, while the violet cluster only is present in North Carolina. Furthermore, one can observe a strong negative correlation between Alaska and the western contiguous part of the US, which is indicated by different stripe orientations.

Explaining the causes for these correlations is not trivial and would require further investigation. However, because they resemble the state borders, a first guess is that similar value behavior comes from different agencies managing the stations, maybe using different sensors or acquisition strategies. In additional analysis steps, the clustering can be used to further investigate the data and locate the source of these possibly unwanted dependencies. Understanding these effects can help improving the accuracy of the positioning system.

10. Conclusion and Future Work

We presented a method for computing hierarchical correlation-based clusterings for multiple scalar fields. A new clustering method was proposed that preserves the metric properties of correlation and generates a cluster hierarchy through agglomeration. Furthermore, we incorporated negative correlation coefficients consistently via twin insertion and visualized the results with two linked views. Besides showing the clustering inside the domain, we translated 2D topological landscapes to work with arbitrary dendrogram to then build an interactive tool for data exploration. We evaluated the method on multiple synthetic and two real world data sets to capture different data aspects and common applications.

Since correlation alone is not enough for a comprehensive data analysis, one of the main future goals is to couple our method with additional analysis tools, e.g. investigating deviations and higher-order dependencies. We would also like to extend our clustering method to applications different than correlation and work with domain experts to answer more domain-specific questions.

11. Acknowledgements

We want to thank the Deutsche Forschungsgemeinschaft for funding the project SCHE 663/11-1. Furthermore, we would like to ac-

knowledge Martin Reckziegel for his valuable feedback and constructive discussions.

References

- [BBC04] BANSAL N., BLUM A., CHAWLA S.: Correlation clustering. *Machine Learning* 56, 1-3 (Nov. 2004), 89–113. 2
- [BCHC09] BENESTY J., CHEN J., HUANG Y., COHEN I.: Pearson correlation coefficient. In *Noise reduction in speech processing*. Springer, 2009, pp. 1–4. 2
- [Ber06] BERKHIN P.: A survey of clustering data mining techniques. *Grouping multidimensional data* 25 (2006), 71. 2
- [BHJ*14] BONNEAU G.-P., HEGE H.-C., JOHNSON C. R., OLIVEIRA M. M., POTTER K., RHEINGANS P., SCHULTZ T.: Overview and state-of-the-art of uncertainty visualization. In *Scientific Visualization*. Springer, 2014, pp. 3–27. 2
- [BOL12] BRODLIE K., OSORIO R. A., LOPES A.: A review of uncertainty in data visualization. In *Expanding the frontiers of visual analytics and visualization*. Springer, 2012, pp. 81–109. 2
- [CL11] CORREA C., LINDSTROM P.: Towards robust topology of sparsely sampled data. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (dec 2011), 1852–1861. URL: <http://dx.doi.org/10.1109/TVCG.2011.245>, doi: 10.1109/TVCG.2011.245. 5
- [CSA03] CARR H., SNOEYINK J., AXEN U.: Computing contour trees in all dimensions. *Computational Geometry* 24, 2 (Feb. 2003), 75–94. 2
- [CSvdP04] CARR H., SNOEYINK J., VAN DE PANNE M.: Simplifying flexible isosurfaces using local geometric measures. In *Visualization* (Piscataway, NJ, USA, 2004), IEEE, pp. 497–504. 9
- [CWMW11] CHEN C.-K., WANG C., MA K.-L., WITTENBERG A. T.: Static correlation visualization for large time-varying volume data. In *Pacific Visualization Symposium* (Piscataway, NJ, USA, 2011), IEEE, pp. 27–34. 2
- [ELLS11] EVERITT B. S., LANDAU S., LEESE M., STAHL D.: Hierarchical clustering. *Cluster Analysis, 5th Edition* (2011), 71–110. 2
- [FH73] FUKUNAGA K., HOSTETLER L.: Optimization of k nearest neighbor density estimates. *IEEE Transactions on Information Theory* 19, 3 (1973), 320–326. 3
- [Fos98] FOSTER P.: Exploring multivariate data using directions of high density. *Statistics and Computing* 8, 4 (1998), 347–355. 6
- [HB03] HARROWER M., BREWER C. A.: Colorbrewer.org: an online tool for selecting colour schemes for maps. *The Cartographic Journal* 40, 1 (2003), 27–37. 9
- [HLH*16] HEINE C., LEITTE H., HLAWITSCHKA M., IURICICH F., DE FLORIANI L., SCHEUERMANN G., HAGEN H., GARTH C.: A survey of topology-based methods in visualization. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 643–667. 2
- [Hot33] HOTELLING H.: Analysis of a complex of statistical variables into principal components. *Journal of educational psychology* 24, 6 (Sept. 1933), 417–441. 5
- [JT92] JAROMCZYK J. W., TOUSSAINT G. T.: Relative neighborhood graphs and their relatives. *Proceedings of the IEEE* 80, 9 (1992), 1502–1517. 5
- [KH03] KOREN Y., HAREL D.: A two-way visualization method for clustered data. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (2003), ACM, pp. 589–594. 4
- [KR85] KIRKPATRICK D. G., RADKE J. D.: A framework for computational morphology. In *Computational Geometry*, TOUSSAINT G. T., (Ed.), vol. 2 of *Machine Intelligence and Pattern Recognition*. North-Holland, 1985, pp. 217–248. 5
- [lina] Continuously Operating Reference Station (CORS). <https://www.ngs.noaa.gov/CORS/>. Accessed: 12/11/2017. 11
- [linb] ECMWF services for the WIS. <https://www.ecmwf.int/en/forecasts/datasets/public-wmo-and-acmad-datasets>. Accessed: 12/11/2017. 10
- [linc] OpenStreetMap Project. <http://openstreetmapdata.com/data/coastlines>. Accessed: 03/29/2017. 10
- [Mac04] MACEACHREN A. M.: *How maps work: representation, visualization, and design*. Guilford Press, 2004. 9
- [NJB06] NARASIMHAN M., JOJIC N., BILMES J. A.: Q-clustering. In *Advances in Neural Information Processing Systems* (2006), pp. 979–986. 6
- [OHJ*11] OESTERLING P., HEINE C., JANICKE H., SCHEUERMANN G., HEYER G.: Visualization of high-dimensional point clouds using their density distribution’s topology. *IEEE Transactions on Visualization and Computer Graphics* 17, 11 (Feb. 2011), 1547–1559. 2
- [OHJS10] OESTERLING P., HEINE C., JÄNICKE H., SCHEUERMANN G.: Visual analysis of high dimensional point clouds using topological landscapes. In *Pacific Visualization Symposium* (Piscataway, NJ, USA, 2010), IEEE, pp. 113–120. 5
- [OHWS13] OESTERLING P., HEINE C., WEBER G. H., SCHEUERMANN G.: Visualizing nd point clouds as topological landscape profiles to guide local data analysis. *IEEE Transactions on Visualization and Computer Graphics* 19, 3 (May 2013), 514–526. 3, 4, 7
- [PRJ12] POTTER K., ROSEN P., JOHNSON C. R.: From quantification to visualization: A taxonomy of uncertainty visualization approaches. In *Uncertainty Quantification in Scientific Computing*. Springer, 2012, pp. 226–249. 2
- [PW12] PFAFFELMOSE T., WESTERMANN R.: Visualization of global correlation structures in uncertain 2D scalar fields. In *Computer Graphics Forum (Proceedings of Eurographics Conference on Visualization)* (Piscataway, NJ, USA, 2012), IEEE, pp. 1025–1034. 2, 9
- [PW13] PFAFFELMOSE T., WESTERMANN R.: Correlation visualization for structural uncertainty analysis. *International Journal for Uncertainty Quantification* 3, 2 (2013), 171–186. 2
- [SR62] SOKAL R. R., ROHLF F. J.: The comparison of dendrograms by objective methods. *Taxon* 11, 2 (1962), 33–40. 2
- [STS06] SAUBER N., THEISEL H., SEIDEL H.-P.: Multifield-graphs: An approach to visualizing correlations in multifield scalar data. *IEEE Transactions on Visualization and Computer Graphics* 12, 5 (Nov. 2006), 917–924. 2
- [SWMW09] SUKHAREV J., WANG C., MA K.-L., WITTENBERG A. T.: Correlation study of time-varying multivariate climate data sets. In *Pacific Visualization Symposium* (Piscataway, NJ, USA, 2009), IEEE, pp. 161–168. 2
- [Tar75] TARJAN R. E.: Efficiency of a good but not linear set union algorithm. *Journal of the ACM (JACM)* 22, 2 (Apr. 1975), 215–225. 6
- [THM*05] THOMSON J., HETZLER E., MACEACHREN A., GAHEGAN M., PAVEL M.: A typology for visualizing uncertainty. In *Electronic Imaging 2005* (2005), pp. 146–157. 2
- [War12] WARE C.: *Information visualization: perception for design*, 3 ed. Elsevier, 2012. 4
- [WBP07] WEBER G., BREMER P.-T., PASCUCCI V.: Topological landscapes: A terrain metaphor for scientific data. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (Nov. 2007), 1416–1423. 3
- [WJ94] WAND P. M., JONES M. C.: *Kernel smoothing*. Crc Press, 1994. 3
- [ZHQL16] ZHANG H., HOU Y., QU D., LIU Q.: Correlation visualization of time-varying patterns for multi-variable data. *IEEE Access* 4 (Aug. 2016), 4669–4677. 2
- [ZMZM15] ZHANG Z., MCDONNELL K. T., ZADOK E., MUELLER K.: Visual correlation analysis of numerical and categorical data on the correlation map. *IEEE Transactions on Visualization and Computer Graphics* 21, 2 (Aug. 2015), 289–303. 2

1. Proof A

This section provides the proof that single-linkage clustering (SL) with correlation coefficients as similarity measure produces the same results as our clustering algorithm with the correlation edge measure (CEM). In the following, $X = \{x_1, \dots, x_n\}$ denotes the set of points getting clustered and ρ_{ij} is the correlation between two points $x_i, x_j \in X$.

In every iteration step, SL merges the two clusters A and B with the highest similarity, which is defined as the highest similarity between two points $x_i \in A, x_j \in B$. Therefore, it finds the pair of yet unmerged points that have the highest correlation. CEM does exactly the same by first sorting all edges by similarity and then merging them successively. The only difference is that instead of considering all possible point pairs, CEM only merges along edges of the beta-skeleton with $1 \leq \beta \leq 2$. Since graphs resulting from smaller β are always subgraphs of ones with higher β , we only have to prove similarity for the neighborhood graph with the fewest edges, i.e., the relative neighborhood graph (RNG) with $\beta = 2$.

Both clustering algorithms start with each point in its own separate cluster. Assume that up to a certain iteration, both clustering hierarchies are equivalent and $C(x) \subseteq X$ denotes the cluster containing point $x \in X$. Let $A \subset X$ and $B \subset X$ be two different clusters with points $x_i \in A$ and $x_j \in B$. To prove equality of the clusterings, let us compare the two following statements:

- I the next two cluster getting merged by SL are A and B due to the highest similarity between x_i and x_j
- II the next two clusters getting merged by CEM are A and B due to the edge between x_i and x_j corresponding to the highest correlation

I \rightarrow II

Let us assume statement (I) is true. It follows that

$$\rho_{ij} = \max\{\rho_{ab} \mid C(x_a) \neq C(x_b)\}. \quad (9)$$

To show that these points have to be connected in the RNG, we have to look at the beta-lune. Two points are connected in the RNG if and only if their beta-lune does not contain a third point x_k . As of Formula 5, the test for containment in the hyperspherical lune reduces to

$$\begin{aligned} (2 - \beta)(1 - \rho_{ki}) + \beta(\rho_{ij} - \rho_{kj}) &< 0 \\ \wedge (2 - \beta)(1 - \rho_{kj}) + \beta(\rho_{ij} - \rho_{ki}) &< 0. \end{aligned}$$

For $\beta = 2$, this simplifies to

$$\begin{aligned} 2 \cdot (\rho_{ij} - \rho_{kj}) < 0 \quad \wedge \quad 2 \cdot (\rho_{ij} - \rho_{ki}) < 0 \\ \Leftrightarrow \quad \rho_{ij} < \rho_{kj} \quad \wedge \quad \rho_{ij} < \rho_{ki}. \end{aligned}$$

If there would be a point x_k fulfilling this criterion, thus causing the edge between x_i and x_j to not be part of the neighborhood graph, either one of the two conditions would contradict Implication 9. There also can be no edge with higher correlation, as this would also violate Implication 9.

\neg I \rightarrow \neg II

To complete the proof, let us now assume that statement (I) is false. That means there has to be a pair of points x_k, x_l with $C(x_k) \neq C(x_l)$ and

$$\rho_{kl} = \max\{\rho_{ab} \mid C(x_a) \neq C(x_b)\} < \rho_{ij}.$$

With the same implications as above, there has to be an edge between x_k and x_l in the neighborhood graph, which causes statement (II) to be false as well.

This proves equality of the two clusterings. \square

2. Agglomerative Correlation Clustering

Algorithm 1: Agglomerative Correlation Clustering

input : $G=(V,E)$ - Neighborhood Graph with vertices $V = \{v_1, \dots, v_n\}$ and edges $E \subseteq V \times V$
measures - Array of edge measures

output: Dendrogram

```
// Initialize data structures
d  $\leftarrow$  Dendrogram with  $n$  nodes;
uf  $\leftarrow$  UnionFind of size  $n$ ;
repr  $\leftarrow$  Array of size  $n$ ;
for  $i \leftarrow 1$  to  $n$  do
    | repr[i] =  $i$ ;

// Main loop
sort edges  $E$  by measures;
foreach edge  $(v_1, v_2)$  in  $E$  do
    |  $r_1 \leftarrow$  uf.root( $v_1$ );
    |  $r_2 \leftarrow$  uf.root( $v_2$ );
    | if  $r_1 \neq r_2$  then
        | newRoot  $\leftarrow$  uf.merge( $r_1, r_2$ );
        | newVertex  $\leftarrow$  d.addVertex();
        | d.connect(newVertex, repr[ $r_1$ ]);
        | d.connect(newVertex, repr[ $r_2$ ]);
        | repr[newRoot]  $\leftarrow$  newVertex;

return  $d$ ;
```
