Visual Analytics for Spatial Clustering: Using a Heuristic Approach for Guided Exploration

Eli Packer, Peter Bak, *Member, IEEE*, Mikko Nikkilä, Valentin Polishchuk, and Harold J. Ship

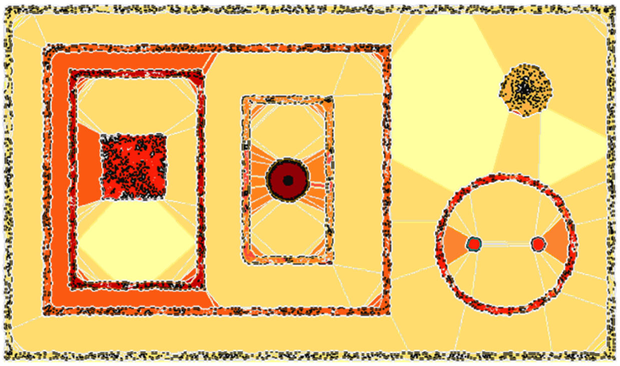


Fig. 1. Hierarchical clustering results on a synthetic point dataset (the black dots) are shown as a heatmap. Colored levels of the hierarchy (with darker red colors corresponding to higher levels) highlight those cluster constellations that were found interesting by our heuristic, aiding the user in selecting appropriate algorithmic settings.

Abstract—We propose a novel approach of distance-based spatial clustering and contribute a heuristic computation of input parameters for guiding users in the search of interesting cluster constellations. We thereby combine computational geometry with interactive visualization into one coherent framework. Our approach entails displaying the results of the heuristics to users, as shown in Figure 1, providing a setting from which to start the exploration and data analysis. Addition interaction capabilities are available containing visual feedback for exploring further clustering options and is able to cope with noise in the data. We evaluate, and show the benefits of our approach on a sophisticated artificial dataset and demonstrate its usefulness on real-world data.

Index Terms—Heuristic-based spatial clustering, interactive visual clustering, k-order α-(alpha)-shapes

1 Introduction

As the availability of data with spatial attributes grows due to the abundance of location-enabled devices—such as GPS or smart mobile devices—the need for analysis of such data grows as well. Spatial data analysis can be applied in such varied domains as tourism, municipal service, safety and security force planning, emergency management, and fighting disease spread in epidemiology. Spatial data analysis entails a great challenge, as it needs to cope with arbitrary distribution, noise, and a large quantity of events buried in data. Analysts, facing such large quantities of spatially-distributed event datasets, need a way to efficiently carry out tasks such as trend or outlier detection, prediction of future events, and resource allocation planning.

As an early step in the knowledge discovery process, analysts and domain experts would like to gain overviews about their data at hand. Such overviews are required to enhance understanding of the spatial distribution and grouping of data. One often-suggested technique for this is clustering the data over space.

The currently available algorithms for clustering mostly require users to define input parameters before running the algorithm, and rarely make recommendations about interesting settings of these parameters. Depending on the type of algorithm, such parameters could be the number of target clusters or the density or distance threshold for partitioning the space between clusters. Such algorithms rarely provide visual feedback about the suitability of a particular setting with respect to the data or the users' purpose for clustering. Most critically, the number of clustering possibilities resulting from changing a parameter can be tremendous. It became inevitable to introduce the user as a core part of the clustering process, as they need to be able to steer and guide the process and to express their interests in the findings, which need to be expressed in the input parameters required by algorithms.

We suggest a novel visual analytics solution for hierarchical spatial clustering that enables (1) changing input parameters to receive immediate visual feedback for the algorithmic performance and to reduce noise, (2) using heuristics to suggest interesting algorithmic settings for exploration, (3) give to the resulting clusters a shape by displaying their borders, and (4) viewing the hierarchy levels during exploration. Given these capabilities, our visual analytics approach provides a comprehensive solution to hierarchical spatial clustering. We thereby provide algorithmic computational capabilities that can be steered by the user through interactive visualization.

Our heuristic is based on a set of topological and geometric features of the obtained clusters, described in the examples throughout this paper, which can be redefined as required by the domain or the user. The visualization itself is based on a well-known heatmap view of cluster-hierarchies. The shaping of the clusters is conducted using the α-shape data structure - a well known shape approximating method in computational geometry. The noise reduction is interactively performed with the k-order α-shape technique, which is a novel generalization of the α-shape for outliers filtering.

- *Eli Packer is with IBM Research Haifa Lab.*
- *Peter Bak is with IBM Research Haifa Lab. E-mail: peter.bak@il.ibm.com*
- *Mikko Nikkilä is with University of Helsinki, Finland.*
- *Valentin Polishchuk is with University of Helsinki, Finland.*
- *Harold Ship is with IBM Research Haifa Lab.*

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 4 October 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

2 Related Work

The main idea behind clustering is to partition a space into homogeneous – meaning consistent, close, or dense, depending on the type of the algorithm used – regions to reveal interesting spatial patterns and phenomena. Resulting clusters can then be investigated to present regular or irregular behavior and interaction between spatial and non-spatial data. Clustering in data mining falls into the category of unsupervised learning, finding the structure of clusters directly from the data. As such, the primary purpose in data mining is to gain insights into the distribution of the data and to understand the characteristics of each cluster. Clustering in data mining can be achieved in several ways. Literature [34, 23, 26, 22, 29] suggests distinguishing among the different types of methods:

Partitioning methods aim to organize data into a number of distinct clusters, such that the total deviation of the data points from the cluster centers is minimized. In general, methods based on partitioning are sensitive to noise and clusters are regularly-shaped and requiring the number of resulting clusters as an input parameter, even though this type of algorithm is scalable to the number of objects. Examples for such methods can be found in [16]. Probably the most popular method of this type is the K-Means algorithm for clustering, which requires the number of clusters (K) as an input parameter [27]. A more sophisticated and improved version is described by Arthur & Vassilvitskii [3], referring to the K-Means++ algorithm that overcomes the difficulties of its predecessor. *Hierarchical* methods are based on a recursive split of the data into a tree structure. The resulting smaller subsets form groups that can be based on e.g., the Euclidean distance between the points. In general, methods of this type are scalable with the number of objects and the resulting clusters can have arbitrary shapes. Examples for such methods can be found in [18, 28]. One method somewhat related to ours is *Single-link Clustering (SLC)* [36], in which the edges of cluster shapes that we output with the α-shape are a subset of edges in the SLC with the threshold of 2α). SLC, however, does not output any shapes (in fact, edges in the SLC may cross one another). *Density-based* clustering methods regard clusters as regions with high densities of objects, separated by sparse areas. Such methods can be used to efficiently filter noise and can discover arbitrary-shaped clusters. These clusters may have an arbitrary shape and the points inside a region may be arbitrarily distributed, but knowing the distribution and density might be a prerequisite to achieve meaningful clustering results. For further information, refer to [12, 35, 24]. Alternatively to the above methods, *grid-based* approaches were introduced. This method type uses a grid data structure to enhance the efficiency of the clustering. It quantizes the space into cells to perform the clustering, making the methods independent of the number of data objects and yielding high efficiency. As an example, a combination of grid- and density-based clustering was introduced by the CLIQUE algorithm [1] showing high scalability and performance.

α-shapes were earlier used for clustering and classification in [31, 33, 37]; however, the problems considered in these papers, as well as their solutions, are completely different from ours. Mu and Liu [33] assumed that a "rank" is given for every data point; then for every rank r they compute the α-shape for points of rank $\leq r$, obtaining a sequence of α-shapes of increasing α (we do not have any ranks in the input). Lucieer and Kraak [31] used α-shapes to define shapes in the feature space of remotely sensed images; they then classified new images using the distances to the α-shapes. van de Weygaert *et al.* [37] explored topology of the cosmic web by analyzing α-shapes of galaxies distribution. Similarly to us, Alper *et al.* [2] and Collins *et al.* [10] present techniques to visualize shapes on top of geospatial data; however, geometric techniques used in [2, 10] are different from ours. Also, we do clustering while in [2, 10] the partition of the points into sets is *given in the input*. In [2] the shape of the set is a curve through *all* elements of a set; on the contrary, our crucial idea is to have the shape go only through *boundary* points of the set.

Recent works also addressed the interactivity with the data visualization as integral part of visual analytics systems. Wang *et al.* [38] investigate visual techniques for high-dimensional data for interactive exploration. In particular, they offer a novel method of hierarchical ordering, spacing and filtering based on similarities of dimensions. They generate default settings for ordering, spacing and filtering parameters which can then be fine-tuned by users. A hierarchy is assigned to the dimensions, in order to allow this interactive control. Visual analytics tools exist that aid sense making by providing various interaction techniques that allow users to change the representation through altering model parameters. However, users need to express their reasoning on the data and within their domain language, instead of directly on the statistical or algorithmic model. Endert *et al.* [15] explore two possible observation-level interactions, namely exploratory and expressive, within the context of PCA and MDS for dimension reduction, thus illustrating how interaction can be designed into visual analytics tools. Choo *et al.* [8] investigated the supervised dimension reduction and designed an interactive system that was proved to lead to better understanding and insights. Similarly, Brown *et al.* [5] present a system that allows experts to interact directly with a visualization by dening an appropriate distance function, thus avoiding direct manipulation of model parameters. Based on user input, the system can perform optimization in the background to learn a new distance function and redraw the visualization. Maciejewski *et al.* [32] provide a predictive visual analytics toolkit whose purpose is to predict hot-spots in spatio-temporal data. The goal of this search is optimal resource allocation. They combine kernel density estimation with seasonal trend decomposition and provide error estimation, spatio-temporal alerts, and hot-spot identification. They claim their prediction model can be used for hypothesis testing, intervention planning and resource allocation.

All these approaches highlight the advantages of interactive visualization and visual analytics. In the tighter scope of this paper, interactive visualization has been investigated within the spatial clustering field. To make spatial clustering more useful, the user must be provided with interaction techniques to steer performance of the algorithm and obtain interesting results. This was leveraged by Chen & Liu [6, 7] who provided a visual framework for algorithmic approaches that allow the users to refine the clustering process. Another framework, *IMAGE*, in which the user can interactively control parameters of the clustering methods and see the immediate result, is presented by Guo *et al.* [20, 21, 19]. As such, the framework includes two techniques—a hierarchical clustering and a density/grid-based approach—giving the interaction needed to effectively support a human-led exploratory analysis. Estivill-Castro & Lee [17] detected different levels of cluster partitions by analyzing the edge lengths of the Delaunay triangulation. They introduced a mainly automatic way of generating cluster hierarchies as point clouds.

In our approach, we suggest giving the clusters explicit shapes and borders and allowing the user to change these shapes by altering the input parameters of the algorithm, interactively guided by a heuristic. We thereby extend on the earlier works by introducing a heuristic computation of interesting algorithmic parameters and presenting them in a systematic manner to the user. Our current research efforts focus on a distance-based, hierarchical clustering algorithm. This type of algorithm gives a hierarchical view of the clusters in the output. Our solution applies a heuristic for interactively selecting interesting values for the input parameter during clustering. Specifically, it provides many potential starting points for users to explore possible clustering results.

Our Contributions. We propose a novel method to address the following tasks: (I) splitting data into clusters, (II) visualizing a geometric shape for each cluster, and (III) doing I and II interactively. Much prior work has been done separately on each of the above problems – literature on clustering is vast, shape reconstruction is classic in several fields, and interactive visualization is a recurring topic at IEEE VAST and other forums. But our technique is **unique** in that it addresses all three of the above problems **simultaneously**. Our solution also includes the following features: *(A) Computational efficiency:* Precomputing and storing the boundary edges of clusters takes near-linear time, which is comparable to classical efficient computational-geometry techniques, such as plane sweeps, convex hull computations, and others. This gives us the interactivity for free, as cluster boundaries only have to be rendered (and not computed) when required.

(B) Perception quality: Presenting shapes to the user makes the picture more clear than, e.g., assigning color to points according to which cluster they belong. In future visual analytics tools, this may help to address the issue that, in general, humans may have difficulty to consistently clustering spatial data based only on visual perception of the point set alone. In addition, visualizing overlapping shapes may help to reveal the structure of nesting clusters. *(C) Clustering filtration:* We developed heuristics to select only the visually appealing clusters in the same dataset. This aids in filtering out irrelevant information, while keeping the user in the visualization loop. *(D) Visualizing holes:* Holes in shapes often occur naturally in datasets sampled in real-world situations. While holes in shapes are "obvious" to the human eye, designing an automated procedure for detecting and visualizing the holes is a challenging task that we successfully address with α-shapes and with our heuristics for boundary detection. *(E) Query support:* The points of $\mathbb{R}^2$ enclosed by a cluster shape are assigned to the cluster. This is crucial, e.g., in query operations in which the user would like to know to which cluster a new point (not from the original dataset) belongs. *(F) Subsequent applications:* Giving shapes to clusters in a dataset can be motivated also from a wider point of view.For example, it is often the case that two different datasets are available for the same geographic area. Using the shapes from clusters of the two sets, users can perform logical operations on the shapes (taking their union, intersection, etc.), which may reveal valuable information about the area, such as correlation between the two datasets.

3 Method

Given a set of points in $\mathbb{R}^2$, we construct a hierarchy of cluster partitions. Each partition consist of a set of clusters that collectively contain the input points. We further associate geometric shapes to the clusters for visualization purposes. These shapes correctly enclose the input points to illustrate the subdivision of the points into clusters.

In this framework, two main questions are raised: how to find suitable different partitions of points into clusters and how to shape the clusters. We use a powerful computational-geometry tool—the *α-shapes*—to answer both questions. One of the main applications of the α-shape (if not the main one) is to give an approximating shape to a collection of points. It is defined in any dimension, but in $\mathbb{R}^2$ the α-shape comes to associate a collection of (one or more) polygons to a set of points, such that the polygons enclose the points. By computing the polygons, we answer the above two questions: the points enclosed within each polygon correspond to a single cluster and the polygon constitutes the shape of the cluster. As we describe below, any set of points can be associated with many α-shapes, only some of which are of interest to us – those that form both an "interesting" partition into clusters and enclose the points "tightly" enough.

Real-world geographic data are often noisy; the noise is a result of both measure errors (e.g., GPS errors) and true outlier data. It appears that the α-shape is very sensitive to outliers – even a small amount of outliers may result in both clusters merging and big shape distortions. In order to overcome these difficulties, we use a new data structure that generalizes the α-shape. Introduced by Krasnoshchekov and Polishchuk [30], it was termed *k-order α-shape*. Its main motivation is to exclude outliers from the shape. Our experiments have shown that using the k-order α-shape instead of the ordinary α-shape improves the results significantly.

In the following three subsections we describe how we use the α-shape to detect hierarchies of clusters. In subsection 3.4 we demonstrate how we integrate the k-order α-shape to filter noise. In the remaining subsections we discuss the flow and the visualization of our method.

3.1 Alpha-Shapes

Let $P \in \mathbb{R}^2$ be the input point set. Given a parameter α, let $B(p,\alpha)$ be the (open) disk of radius α centered at point $p \in \mathbb{R}^2$. The α-hull of P is defined as $H_\alpha(P) = (\bigcup_{B(p,\alpha)\cap P=\emptyset, p\in\mathbb{R}^2} B(p,\alpha))^c$. In words, the α-hull is the set of points that do not lie in an empty disk of radius α. The boundary of α-hull consists of circular arcs; if we substitute the arcs by segments, we get the α-shape. Figure 2 illustrates several α-shapes on the same input points. It is known that the edges of the α-shape is a subset of the Delaunay triangulation of P and that each edge e is associated with a numeric interval $[e_1, e_2]$ such that the edge belongs to the α-shape if only if $\alpha \in [e_1, e_2]$ [14]. Note that the α-shape is a generalization of the convex hull; when α approaches ∞, the α-shape converges to the convex hull. As α decreases, the shape shrinks and may generate tunnels and voids, and may also be split into several disjoint components; when $\alpha = 0$, the shape contains nothing but the input points.

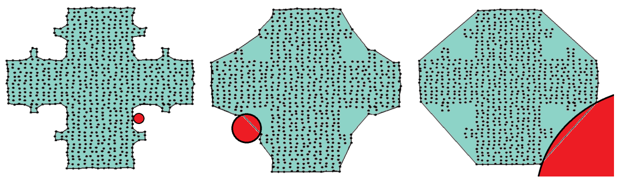


Fig. 2. α-shapes (shaded) for increasing α values, from left to right. In each shape we show one possible position of the α-ball (in red) where it touches two points on the boundary of the shape.

The α-hulls and α-shapes were introduced in [14]. Edelsbruner [13] presented an excellent survey of the work devoted to α-shapes over the years. A central objective of the α-shape was to reconstruct the shape of a set of points. As far as we know, we are the first to use the α-shape for hierarchical cluster analysis.

3.2 Generating α-shape Breakpoints

We start from computing the Delaunay triangulation $T(P)$ of P. Next, for each edge $e \in T(P)$, we find the corresponding interval where the edge belongs to the α-shape. This is done locally by analyzing the adjacent triangles, as described in [14]. The endpoints of the intervals define a set S of α-shape *steps*; each step $s \in S$ corresponds to a change in the α-shape – either an addition or a removal of an edge. We sort the elements of S in increasing order. Simulating the steps in this order, we get an iterative procedure that traverses all possible α-shapes for P. Note that the first α-shape includes no edges and the last one coincides with the convex hull.

Some of the steps constitute candidates for interesting cluster partitions. We denote these candidates by $B \subseteq S$ and refer to them as *breakpoints*. Each breakpoint $b \in B$ will encode a partition of points into clusters. It will also induce shapes for the clusters; those will be the α-shapes themselves, potentially post-processed to give more topological and geometrical meaning. Using our technique, the partition and the shapes associated with the breakpoints will make up the results.

We emphasize that the process of identifying the breakpoints and constructing the shapes is semi-automatic. This is in contrary to most previous work with α-shapes, in which the authors usually arbitrarily determined α for subjective purposes, often for good presentation effects.

3.3 A Heuristic for Generating Cluster Partition

We devised and tried several heuristics to find breakpoints and corresponding shaping. In this work, we describe the heuristic we chose, which was a clear winner in terms of generating expected cluster partitions. As described in the previous section, we proceed by processing S in order. At the beginning, each point is associated with a cluster containing just the point. For each edge being added, we iteratively merge the clusters of its endpoints. No splits are performed when removing edges; this means that once an edge was processed, its endpoints will belong to the same cluster until the process terminates. In particular, when processing the steps, the number of clusters decreases, while the size of the clusters increases. Upon terminating, we are left with one cluster that contains all points. During this process, we determine the breakpoints and the interesting partitions with their shapes, as described next.

- Breakpoints are considered only after each $p \in P$ is a part of an edge that was added. It follows that each point will belong to a cluster of at least two points when starting to consider breakpoints. The goal is to avoid dealing with partitions in which isolated points make up one-point clusters.

- Breakpoints are considered only if the corresponding geometric graph that represents the α-shape contains only simple polygonal cycles. In other words, the degree of all vertices of the α-shape is either 0 or 2. Note that since the geometric graph is planar, no cycle intersection will be introduced. From those polygonal cycles and the characteristics of the α-shape structure, we can easily detect associated polygons as follows: Polygonal cycles that are not contained within any other polygonal cycles are the boundaries of polygons; the cycles that are immediately inside them will be the holes of these polygons. Cycles inside holes will be associated with other polygons, and so on. It follows that each polygon will have 0 or more holes, and polygons may surround other polygons. The reason for considering such breakpoints is that their geometric and topological interpretations, as described above, cover the entire set of points, nicely and meaningfully partitioning them and giving them suitable shapes. Figure 3 illustrates the shapes obtained with various breakpoints on a single dataset.

- From the previous items, it follows that any $p \in P$ will be located in one of the polygons; the inclusion of points inside polygons will constitute the partition of the points into clusters.

- We partition the breakpoints into *topologically equivalent classes*. In each class, the partition of the points into clusters is identical. The breakpoints of the same class differ by their corresponding shapes. To find representative shapes for each class, we use the following three options (Fig. 4):

  - The minimum α with which the cluster partition is obtained: The rationale behind this option is to find a shape that tightly fits the data, avoiding voids as much as possible.

  - The minimum α with which the minimum number of polygonal cycles was obtained (in other words, the minimum number of holes): The rationale behind this option is to simplify the shape as much as possible while avoiding voids.

  - The maximum α with which the cluster partition is obtained. The rationale behind this option is to obtain simplified shapes with small sizes. In many cases, taking the maximum possible α leads to convex or almost convex shapes, which are easier to manipulate and work with.

  It follows that each topological class will have at most three associated shapes (less when one or more of the shapes coincide), providing the analyst with several possible shapes to choose.

3.4 Handling Outliers using k-order α-shape

Since real data is usually noisy, we employ a recently developed generalization of the α-shape, called *k-order α-shape* [30]. The k-order α-shape is a generalization of both the α-shape and of the *k-hull* [9] – a statistical data depth measuring tool. In their turns, both the α-shape and the k-hull are generalizations of the convex hull. The idea behind the k-order α-shape is that the disk defining the shape may contain up to k data points; thus the ordinary α-hull is obtained by setting $k = 0$. Using $k > 0$ allows one to remove the noise locally by digging deeper into the data; the amount digging is controlled by k. Formal definitions of the k-order α-shape and α-hull and their combinatorial and algorithmic properties are given in [30]; here we illustrate the concept in Figure 5 (in the full version of the paper we will discuss noise reduction in much greater length). The left column in the figure shows the

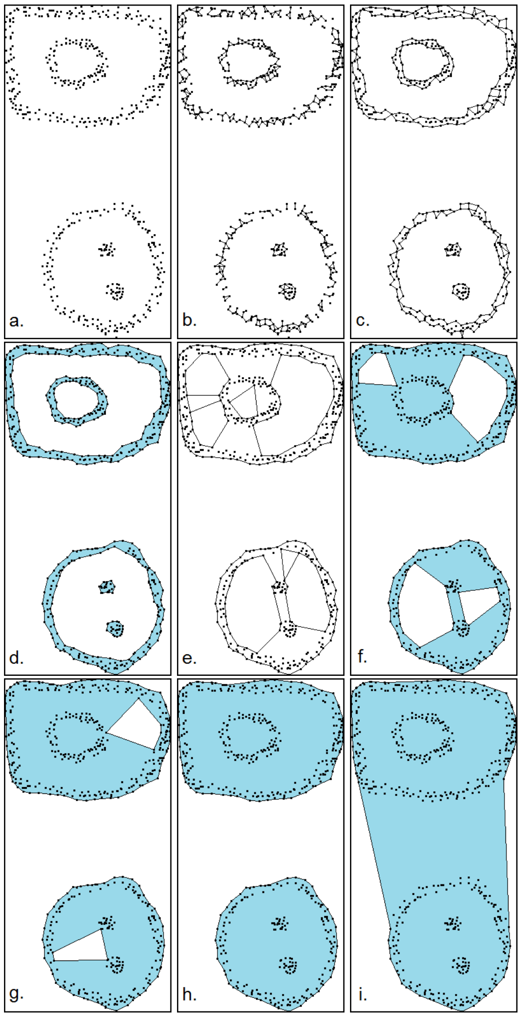


Fig. 3. The α-shapes obtained during different steps. Letters from (a)-(i) correspond to a step-wise increase of α. The input data is shown in (a). Clusters with simple polygonal cycles (i.e., clusters at breakpoints, when vertex degrees are 0 or 2) are colored in blue (d, f, g, h, i). Pictures with α-shapes at steps that are not breakpoints are left uncolored (b, c, and e).

α-shapes for different α's on the same dataset. In the second and the third column we added four outliers. The second column shows the resulting α-shapes; it is evident that the shapes are distorted significantly. The third column shows k-order α-shapes with $k = 1$. The four outliers are no longer inside the shape – the noise is removed. Note that other points (those near the dataset boundary) are excluded from the shape too; still, the shapes look much more similar to the ones in the first column. (Incrementing k further would shrink the shape even more; depending on the volume of the noise, different k's are suitable for denoising – the best choice of k is usually subjective and is left for the user).

Computational resources. We assume that the analyst will use relatively small values for k, i.e., that at no location the dataset contains too many outliers. Our experiments justified this assumption where using small k values were sufficient for detecting the cluster hierarchies. We begin by computing the resources for one k-order α-shape. Constructing the Delaunay triangulation takes $O(n \log n)$ time [11]. Since the triangulation has $O(n)$ edges, computing the edge α-intervals takes overall linear time (computation is local and takes constant time per edge); sorting the intervals endpoints to obtain the set of steps S takes additional $O(n \log n)$ time. We maintain the incident edges for the vertices (i.e., points of P) when traversing the sorted steps. By memoriz-

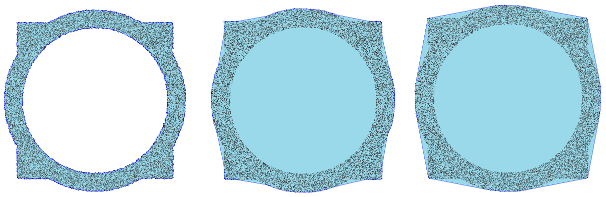


Fig. 4. From left to right: minimum α, minimum α-minimum cycles, maximum α. Although the two right options look similar, the shape of the maximum α contains far fewer edges yet has bigger voids - it is actually the convex hull of the points. Also that the three options are applied individually to each cluster; in particular, the set in the figure may be a single cluster in a larger dataset.

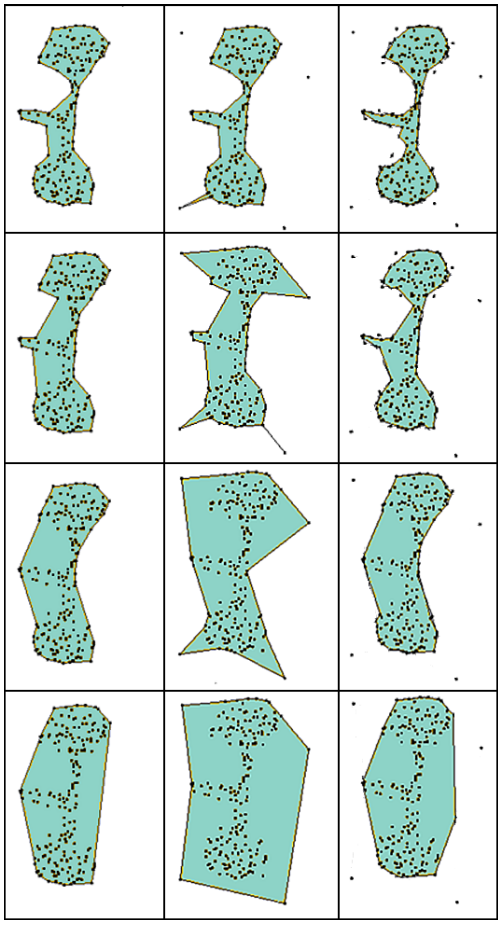


Fig. 5. k-order α-shape (in blue) with different α's and k's. Increasing α values are shown in rows. The first column shows the α-shapes ($k = 0$). In the second and third columns four outliers were added. The k in the second and third column equals 0 and 1 respectively.

ing the vertices whose degree is not 0 or 2 and those who were not connected to any edge, we can determine when a potential breakpoint is found. Then by tracking cluster changes (an edge of two vertices that belong to different clusters is added) and cycles, we find our desired breakpoints. Whenever a breakpoint is found, we need to compute its polygons. We do this by sweeping the plane in $O(n \log n)$ time. Let M be the number of cluster partitions we detect. Summing up the times of each of the steps above, we get a total of $O(Mn \log n)$ time. Multiplying by k, we get $O(kMn \log n)$ time. We conjecture that $M = O(\log n)$. If our conjecture is correct, then the entire process takes $O(kn \log^2 n)$. Based on the data structures we use, it follows that we require $O(kn)$ space.

3.5 Algorithmic Flow

The overall algorithmic flow is initiated by the spatial point data, which is first preprocessed using two major modules: the Dual graphs and the k-order α-shape computation. In the subsequent step, the heuristic is defined and applied to select α-breakpoints that result in interesting cluster-shapes. Once clusters are generated, users can review all cluster sets and alter the value of α according to their preferences, also redefining the heuristic for generating clusters if needed. If the user is not satisfied with the result in the sense that too much noise harm the result, he will increment k and reiterate. Figure 6 shows the overview of the algorithm flow. Its individual steps were discussed in previous sections.

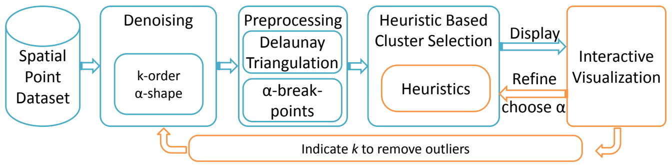


Fig. 6. Overview of the algorithmic flow shows an interactive loop for the user to define a heuristic and alter the parameters of the algorithm. Blue boxes indicate algorithmic computation, whereas orange boxes indicate user input and interaction.

3.6 Interactive Visualization

Interactive visualization is an indispensable part of our user-centered design. Visualization aims at making the results of the clustering accessible to users. Currently, the first screen displayed to the user only contains the input data as single points, which get connected as clusters emerge. We use color to encode individual clusters along the different hierarchy levels, as suggested by Color Brewer [4]. Colors were used for both drawing the edges between points and also as fill color in clusters closed into polygons. Colors were, however, applied only when the heuristic indicated potentially interesting clustering results. In all other cases, when the input was altered by the user, clusters were drawn in black and could be colored when finally exported or selected. Visualization in the proposed approach has two major functions: (1) to create a visual feedback on the obtained clustering results and (2) to interact with the algorithm though the visual interface. For the visual feedback, we used shape borders and fill-color. Shape borders are essential, as they give the clusters a form that can be related to, and fill color supports the information on hierarchy levels. The resulting images create topology of clusters that users can interact with.

Users are able to interact with the algorithms and with the visualization. This interaction is critical for the search for interesting clustering results. As described in previous sections, the heuristics create a set of interesting clustering constellations (at breakpoints) that could be browsed through with dedicated step-forward or step-backward buttons. The corresponding α-value is mapped on a sequential slider. To use interim stages, between two heuristically selected α-values, we added a second dedicated step-over button for forward and backward steps, which allowed for browsing through all the *steps*, even ones not selected by the heuristics as the breakpoints. Finally, at any stage, users can move the slider to any α-value, regardless of breakpoints. In addition, a second slider is dedicated to the setting of k for the $k-$order α-shape. At the beginning, the value of k is set to 0, indicating that no noise reduction was conducted. An increase of k starts the computation from the beginning and reduces the noise more and more.

For an illustrative example of interaction with the user, refer to Figure 3. Clusters in subfigure (h) are shapes without holes; however, subjectively, the dataset is better represented by shapes in subfigure (d). Note that to a computer, the clusterings in subfigures (d) and (h) look equally good; human supervision is needed to make the correct choice. On the other hand, if in the users specific domain it is more intuitive to have clusters without holes, the user may prefer to choose the clustering on subfigure (h); the possibility of choice is brought up by the interactive visualization.

Note that since we pre-compute and store the edges for each α and k, we get the interactivity for free, as the polygons only have to be rendered (and not computed) when required.

4 Evaluation

To evaluate the performance of our method, we experimented with the structure shown in Figure 8. This structure consists of 12 visible clusters of points, and there are no outliers.

Figure 8 shows nine different partitions of the points into clusters. To simplify the visualization effect, we chose the maximum-α option, which minimizes the number of edges of the α-shapes. Starting from subfigure (a) with 12 clusters, we gradually show how clusters merge until all points belong to one cluster (subfigure (l)). All cluster-levels are colored differently in each subfigure using a sequential color palette by decreasing from red to yellow as clusters merge together. The transitions depicted in subfigures (a), (c), (d), (f), (g), (h), (j), (k) and l show how clusters are merged. Further in this process, voids appear, transform, and vanish. A good example is the void below cluster H, which appears in subfigure (h), transforms in subfigures (j) and (k), and finally disappears in subfigure (l). The dynamics of the hierarchical clustering is illustrated in Figure 7 for clarity. Subfigures (b), (e) and (i) depict values with which no interesting partition occurred. As opposed to the other subfigures, the configurations depicted in those figures are not reported by our heuristics.

As shown in Figure 8, as α is increased from subfigure (a) to (l), the voids shrink because the corresponding discs can no longer reach areas inside the voids. Another visual effect is the how the areas are monotonically captured by the α-shapes: from subfigures (a) to (l), clearly more and more area is captured. A region captured by some α-shape will never belong to voids again.

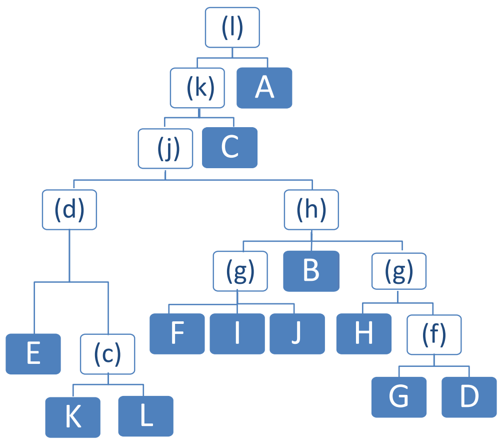


Fig. 7. The hierarchical clustering structure of the data shown in Figure 8. Filled boxes with capital letters indicate the clusters as marked in Figure 8(a). Empty boxes with letters in parentheses represent splitting points in the hierarchy as shown in the subfigures in Figure 8, in reverse order, from (l) to (a).

Figure 8(a) also illustrates the importance of supporting holes inside the shapes. The shapes of several of the clusters (such as F and G) must contain holes in order to separate them from other clusters (for instance, to separate I and J from F). In addition, when it is important to get tight shapes, eliminating voids inside the clusters can be achieved only by introducing holes (see, e.g., the left shape in Fig. 4).

For a quantitative evaluation we generated seven data sets of different sizes. All points reside within the same regions shown in Figure 8, but in each dataset the point density is different. We tested each dataset with all three heuristic options. The results are summarized in Table 1.

The time it took for each set to pre-compute the edges and to build the cluster shapes for all values of α's seems to satisfy the asymptotic complexity of our method ($O(Mn\log n)$, where n is the input size and M is the number of cluster partitions. Note that in each set, we obtained different numbers of partitions. The reason for this is that in denser datasets the regions are better separated as bigger α's capture them.

We compared the geometric differences obtained with our three options. The geometric measures we tested correspond to the collection of partitions our method produced: (1) the number of cycles in the different α-shapes and (2) the sum of the α-shape sizes. The baseline is the *maximum-α*, as it clearly contains the minimum number for both measures. Thus we refer to the measures of this option as 100%. Note that the number of cycles obtained with the *minimum α-minimum cycles* option is the same as the one for the minimum α. As expected, the other values are bigger for the other measures; the differences became as big as 56%, when comparing the cycles' quantity, and as big as 15%, when comparing the output size. These findings confirm our expectations and can guide the user with regards to option selection if the output size is important.

Finally, to show the effect of the interaction with the α-values and how the heuristics are applied, we have recorded a small video for illustration purposes. First we show α-shapes for increasing α; 47 shapes are shown in total, each for 0.5secs. As expected, the α-shape for $\alpha = 0$ is just the points; the α-shape for $\alpha = \infty$ is the convex hull of all the points (the rectangle). Nest we show the α-shapes at the breakpoints (i.e., values of α for which the shapes form polygons), with clusters shown in colors; 9 clusterings are shown in total, each for 3secs. $\alpha = 0$ is always a breakpoint because the shapes do form polygons, albeit degenerate – each polygon is empty (there are no edges, i.e., the degree of every point incident to an edge is 0) and each point is in its own cluster; $\alpha = \infty$ is also a breakpoint (since the convex hull is a polygon) with all points being in the single cluster. We conclude the video with a sequence of images each showing empty α-circles for different values of α.

4.1 Comparison with Convex Hulls

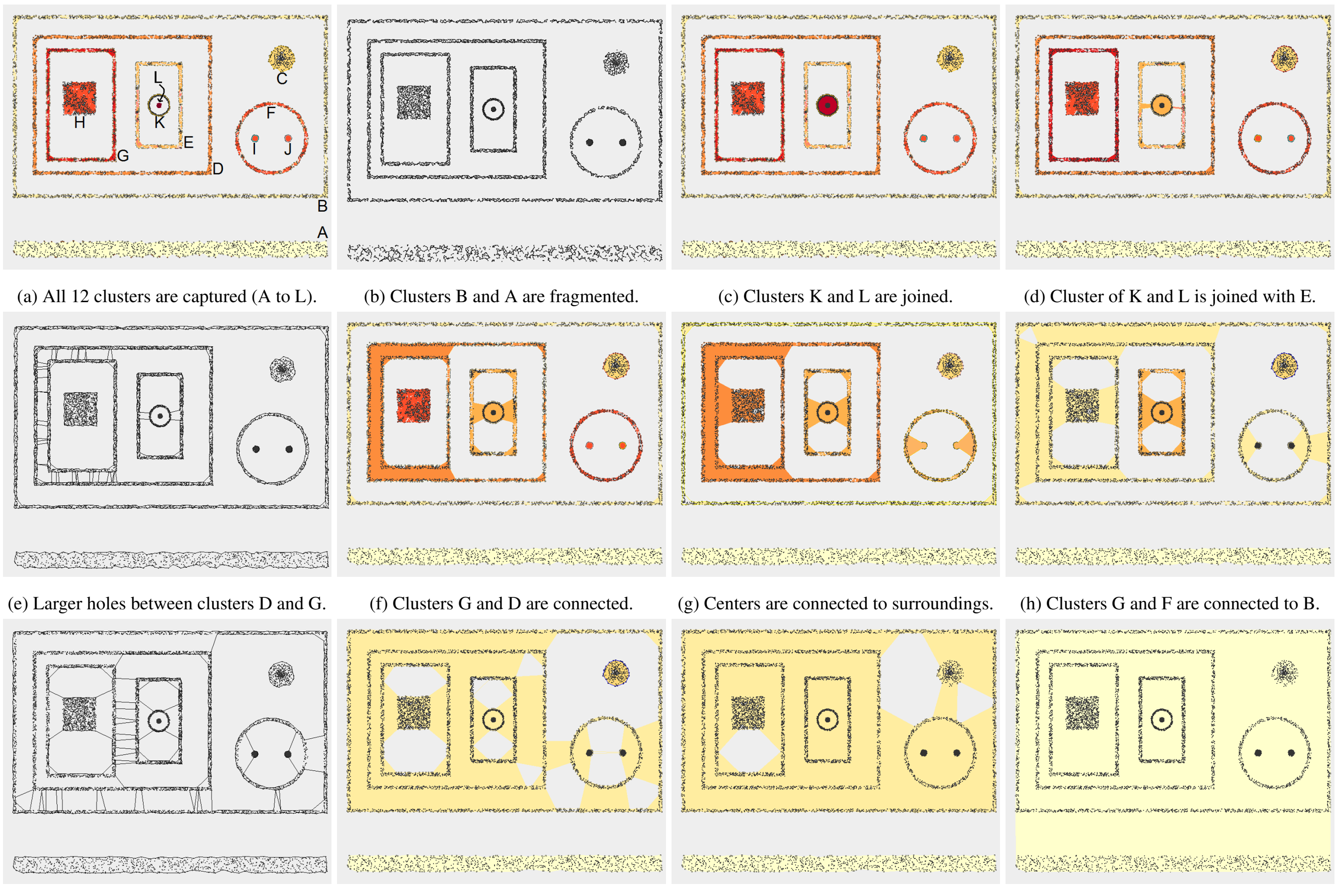
An easier alternative to hierarchical clustering would be to use *nested convex hulls* so that each convex hull will contain the points of its cluster. However, using convex hulls would be inferior to α-shapes for several important reasons. The first is that unless the points lie densely and uniformly within a convex area, the convex hull will contain voids and thus might not provide plausible visualization. Even worse, nested convex hulls will fail to separate different clusters in many cases where the α-shape succeeds (see Fig. 9 for a simple example).

4.2 Processing bridges

A well known clustering challenge is to handle "bridges" between clusters (a bridge is a single-link chain) which is basically a linearly connected point set that connect two clusters. One difficult case in spatial clustering is when two clusters are connected by a thin "bridge"; the user may want to view such datasets as having one or two clusters, depending on the context. Our application provides both options by varying k; Figure 10) gives an example. For a certain α, with $k = 0$ the two balls are connected by the bridge and form one cluster; with $k = 1$, the bridge points are filtered out and do not belong to any cluster, so the balls are disconnected and form two clusters.

5 Instantiation of the Method

To demonstrate the utility of the proposed method, we selected a real-world dataset from the domain of urban public transportation, in which data is publicly accessible and available in large quantities. Public transportation is not only the heartbeat of urban life but is also interesting and challenging to examine from both a research and a business perspective. The task at hand was to explore and define stop patterns in the city of Helsinkis public transportation system. The stops in public transportation are of critical value. Planned stops are required at certain junctions and stations, and unplanned stops are often unavoidable. Although precisely predicting the duration and location of both types of stops can be difficult, they should be carefully calculated since they represent the key performance indicator for the public transportation service. The first step in optimizing public transportation is to understand the spatial distribution of the stops.



(a) All 12 clusters are captured (A to L). (b) Clusters B and A are fragmented. (c) Clusters K and L are joined. (d) Cluster of K and L is joined with E.

(e) Larger holes between clusters D and G. (f) Clusters G and D are connected. (g) Centers are connected to surroundings. (h) Clusters G and F are connected to B.

(i) Holes between clusters E-G and D-B. (j) Cluster B shows holes, C is still disjoint. (k) Cluster B contains all inner clusters. (l) Convex hull contains all the points.

Fig. 8. Obtained clustering results are shown by increasing α-values from left to right and top- to bottom-row. Colors encode individual clusters as they are joined together into higher level clusters. Subfigure (a) indicates the highest resolution of clusters with letters assigned for referencing, and (l) shows the convex hull. Subfigures (b), (e) and (i) not colored show results that were not selected by the heuristics.

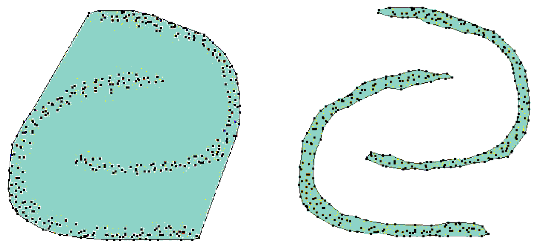


Fig. 9. Two C-shaped clusters. It is obvious how the α-shape (right) nicely clusters and shapes the two sets while the convex hull (left) fails to do so. Moreover, even if we could separate the points into the two clusters, their convex hulls would overlap and provide poor shaping and visualization.

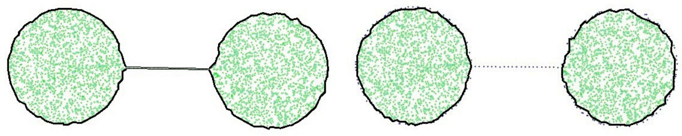


Fig. 10. Two ball-shaped sets connected with a thin bridge. With $k = 0$ for a certain α we obtain one cluster (left); changing k to 1, we obtain two clusters (right).

To gather traffic data, we used the Helsinki Regional Transports HSL Live web service [25]. We connected to the HSL Push interface that sends one record per second per active bus and tram, for the duration of 24 hours. The result was a list of the locations of all active buses and trams within a predefined geographic area. We parsed and saved these locations as latitude and longitude by the identity of the vehicles. From these data we defined stops and extracted them as single events. A stop was defined as such, when a vehicle stood still (0 km/h) for at least 10 seconds. From these stops we removed planned stops, at which the bus stopped at a station; this information was included in retained dataset from the public API. At the end of this process we gathered about 7,000 unplanned stops over a day. The task for our algorithm was to reveal spatial patterns of unplanned stops taking their hierarchical nature and the noisiness of the data into account.

From the application perspective stops are very important to understand how a conjunction spreads from junction to junction and what street-segments are more vulnerable than others. It is therefore a key question to assess the stop occurrences in a hierarchical manner, so hot spots and consecutive affected areas can be distinguished. The final results of the heuristic approach are shown in Figure 11. The heuristic revealed several levels of hierarchies for clustering. We used a sequential color palette to color the different levels of hierarchy, pale-yellow colors for higher levels, and intensive red colors for lower levels of the hierarchy.

Pattern (a) in Fig. 11 shows the junctions next to the Kansanelaekelaitos (social security institute), where the stops are distributed from and towards the center in both directions. It is evident form the vi-

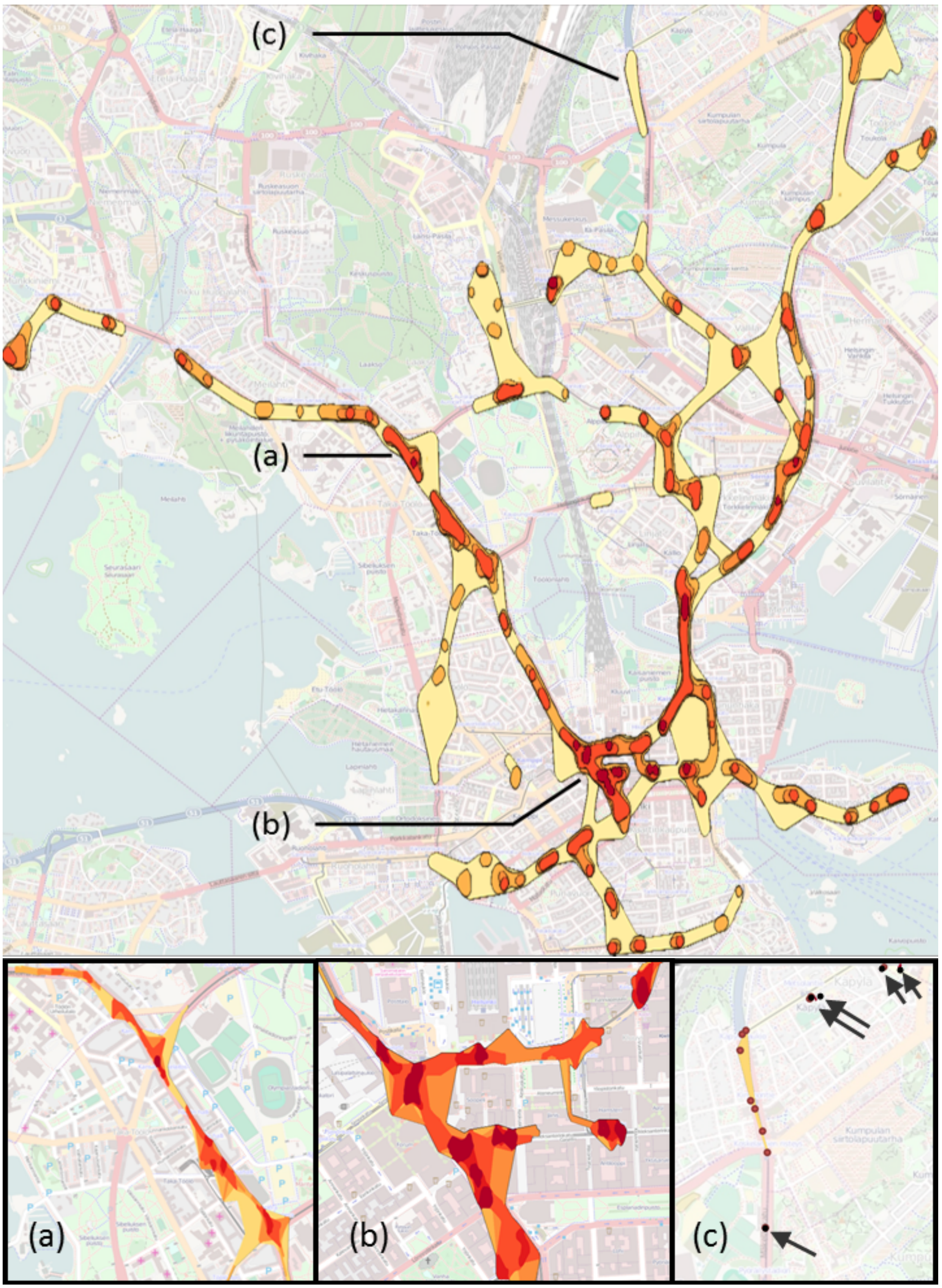


Fig. 11. Clustering results on a real world dataset using the proposed heuristic approach. The cluster-heatmap shows the distribution of public transportation vehicles in five selected levels of hierarchy. More intensive red colors indicate higher concentration of stops. Different distribution pattern are shown in subfigures (a) and (b). Subfigure (c) shows the stops included into the cluster, along with the outliers that were disregarded by the k-order α-shape with arrows.

Table 1. Performance results of our method with the artificial dataset, showing dependence on the number of data points and behavioral differences depending on the different heuristic options. The data are also visualized in Fig. 8.

# points	# partitions	Precomputation time (sec.)	%cycles (to base)			%output size (to base)		
			maximum α	minimum α-min cycles	minimum α	maximum α	minimum α-min cycles	minimum α
41600	12	67		100%	130%		110%	116%
36400	11	47		100%	129%		108%	113%
31200	11	31		100%	129%		108%	113%
26000	11	24	100%	100%	124%	100%	105%	119%
20800	10	18		100%	139%		106%	111%
15600	9	11		100%	124%		105%	113%
10400	8	8		100%	156%		107%	115%

sualization that the center of the distribution is at the station called “Kansanelaekelaitos” itself, indicated by the darkest color. However, when taking higher levels of cluster hierarchies into account, we can see how the stops occur along the main road from north-west to the south. The conjunction also reaches to the north-east forming a rotated ”T”, even though buses do cross the junction to the south west. Pattern (b) Fig. 11 refers to the center of the city, next to the central railway station. This is known to be the busiest traffic area at any hour of the day. The area right (East) of the railway station is a bus terminal – large square with many bus stops, almost all of them are terminal stops – the North-Eastern and Northern traffic from there stops at two traffic lights when leaving the terminal, and then has a stretch with no traffic lights. It is evident from our visualization that the stops surround the area, but the hierarchies show that the western semi-circle is the busiest at two-to-four traffic lights, whereas the eastern semi-circle opens up for a few hundred meters without traffic conjunction. Right after it joins into a busy north-directed traffic, with long continuous stopping areas. Pattern (c) in Fig. 11 is highlighted to show the ability of k-order α-shape to remove outliers. In the upper large image, the stop events were removed, but in the highlighted pattern on the bottom, we added them back, so the viewer can realize that the pattern consist of 6-9 stop occurrences at a low hierarchy level only. The remaining 5-8 stops were not included in the cluster constellation and regarded as outliers. From experience, we know that this area has several schools and shopping opportunities, and is considered only at certain hours of the day as busy traffic area.

Overall, our proposed clustering approach revealed interesting cluster constellations and showed great evidence for the usefulness of our method. The fact that the topology of the city was reflected in our geometry-based heuristics for clustering public transportation data was very interesting and informative. Our method was able to reveal at certain resolution levels in the hierarchy the two sides of a junction, but on lower resolution levels show the whole street as one segment. As urban traffic and transportation environments are indeed sensitive to different hierarchy levels, the current application domain could reveal the strength of the proposed approach.

6 Conclusion and Discussion

In this paper, we presented a novel approach to apply visual analytics to spatial clustering. Our approachs main feature are heuristics that allow selecting interesting cluster constellations and indicate the corresponding input parameter settings of the algorithm. As a result, a user is able to first screen those input parameters that apply to his definition of interestingness and then tune the algorithm at these levels. To enable this heuristic approach, we had to create a system for visual feedback of particular parameter settings and clustering results. We achieved this using a visual display showing the resulting clusters as colored. We evaluated our algorithm and also our heuristics on a suitable artificial dataset with many relevant geometric and topological features that occur in data. Finally, we demonstrated the usefulness of our approach on a real-world dataset that closely relates to the client data we are exposed to in our real-world scenarios.

In summary, our approach entails the following features that make up a comprehensive clustering solution with user-supported exploration: (1) We provide different levels of meaningful partitions of the data space. (2) User can select among several possible visual shapes for the clusters. (3) Clustering is conducted fully automatically with immediate visual responses. (4) We introduced the possibility to seamlessly apply multiple heuristics. (5) The user is provided with a meaningful setting from which to start the clustering exploration. Even though some of these features are present in alternative techniques, we believe that we have made a solid contribution with the introduction of a comprehensive heuristic-based approach for user-guided exploration.

Nevertheless, we must mention the shortcomings and weaknesses we discovered in our approach. We have not sufficiently addressed the option of making α-shapes part of other clustering types, as described in the related work. Also, more sophisticated ways exist to remove noise, as the literature mentions. In our method, we used the k-order α-shapes for efficient noise reduction, but we have not provided more technical details on the approach of noise removal itself, because it would have changed the scope of the paper significantly. While alternative approaches for noise reduction could be considered, the k-order α-shape is the most appropriate one, since it was specifically designed to make α-shape robust to outliers. In certain cases, the users may consider finding outliers even more interesting than the clustering results themselves. The current implementation allows for moving the k-slider back and forth to highlight vanishing clusters, but a more sophisticated interaction and visualization for these clusters should of course be accounted for.

Regarding usability, we must and will consider further approaches to address user needs with respect to the expression of interestingness and its translation into heuristics. In a real-world application, enabling users to create their own frameworks for this task would allow them to change the heuristics during the exploration, depending on the context, task, and data properties. As we have not claimed a contribution on the visualization of clustering hierarchies, this line of research will need to be followed up on. Such research could lead to revealing richer information to the user besides the levels and the cluster shapes. In this line of research, the aim would be to explore more comprehensive and novel ways to display the changes of clustering themselves, as functions of the input parameters.

Some of the limitations of our approach naturally lie in the choice of the methodology and algorithmic bases. As such, we have focused and implemented one typedistance-based computation of a hierarchy of cluster constellations. Our choice might limit the problem space we claimed to have solved; however, we believe that on a higher abstraction level, our approach could be successfully applied and easily adopted to other types of clustering techniques. Consequently, our contribution can be appreciated as a visual analytics approach as it combines complex computational geometry with interactive visualization - taking the best of both worlds.

Acknowledgments

Valentin Polishchuk is funded by the Academy of Finland grant 1138520. Mikko Nikkilä is funded by the Research Funds of the University of Helsinki. Images of the instantiation of the method have been created using IBM Interactive Maps Technology.

References

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. of the ACM SIGMOD, Int. Conf. Management of Data*, volume 27, pages 94–105. ACM, 1998.

[2] B. Alper, N. Riche, G. Ramos, and M. Czerwinski. Design study of linesets, a novel set visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2259–2267, 2011.

[3] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[4] C. Brewer and M. Harrower. Color brewer 2.0. 2012.

[5] E. T. Brown, J. Liu, C. E. Brodley, and R. Chang. Dis-function: Learning distance functions interactively. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*, pages 83–92. IEEE, 2012.

[6] K. Chen and L. Liu. A visual framework invites human into the clustering process. In *Scientific and Statistical Database Management, 2003. 15th International Conference on*, pages 97–106. IEEE, 2003.

[7] K. Chen and L. Liu. Vista: Validating and refining clusters via visualization. *Information Visualization*, 3(4):257–270, 2004.

[8] J. Choo, H. Lee, J. Kihm, and H. Park. ivisclassifier: An interactive visual analytics system for classification based on supervised dimension reduction. In *Visual Analytics Science and Technology (VAST), 2010 IEEE Symposium on*, pages 27–34. IEEE, 2010.

[9] R. Cole, M. Sharir, and C. K. Yap. On k-hulls and related problems. *SIJCOMP*, 16(1):61–77, 1987.

[10] C. Collins, G. Penn, and S. Carpendale. Bubble sets: Revealing set relations with isocontours over existing visualizations. *IEEE Trans. on Visualization and Computer Graphics (Proc. of the IEEE Conf. on Information Visualization)*, 15(6), 2009.

[11] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Heidelberg, Germany, 2004.

[12] L. Duan, L. Xu, F. Guo, J. Lee, and B. Yan. A local-density based spatial clustering algorithm with noise. *Information Systems*, 32(7):978–986, 2007.

[13] H. Edelsbrunner. Alpha shapes - a survey. *in Tessellations in the Sciences; Virtues, Techniques and Applications of Geometric TilingsVisualization and Data Analysis*, 2010.

[14] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Trans. Inform. Theory IT-29*, pages 551–559, 1983.

[15] A. Endert, C. Han, D. Maiti, L. House, S. Leman, and C. North. Observation-level interaction with statistical models for visual analytics. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pages 121–130. IEEE, 2011.

[16] M. Ester, H. Kriegel, and X. Xu. Knowledge discovery in large spatial databases: Focusing techniques for efficient class identification. In *Advances in Spatial Databases*, pages 67–82. Springer, 1995.

[17] V. Estivill-Castro and I. Lee. Amoeba: Hierarchical clustering based on spatial proximity using delaunay diagram. In *Proceedings of the 9th International Symposium on Spatial Data Handling. Beijing, China*. Citeseer, 2000.

[18] S. Guha, R. Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. In *ACM SIGMOD Record*, volume 27, pages 73–84. ACM, 1998.

[19] D. Guo. Coordinating computational and visual approaches for interactive feature selection and multivariate clustering. *Information Visualization*, 2(4):232–246, 2003.

[20] D. Guo, D. Peuquet, and M. Gahegan. Opening the black box: interactive hierarchical clustering for multivariate spatial patterns. In *Proceedings of the 10th ACM international symposium on Advances in geographic information systems*, pages 131–136. ACM, 2002.

[21] D. Guo, D. Peuquet, and M. Gahegan. Iceage: Interactive clustering and exploration of large and high-dimensional geodata. *GeoInformatica*, 7(3):229–253, 2003.

[22] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2):107–145, 2001.

[23] M. Han, J.; Kamber and A. Tung. Clustering methods in data mining: A survey. In *Geographic Data Mining and Knowledge Discovery*, pages 1–29. Taylor and Francis, 2001.

[24] A. Hinneburg and D. Keim. An efficient approach to clustering in large multimedia databases with noise. pages 58–65, 1998.

[25] HSL. Helsinki region transport - live vehicle api documentation. Website, 2011. `http://developer.reittiopas.fi/pages/en/other-apis.php`.

[26] A. Jain, M. Murty, and P. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[27] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):881–892, 2002.

[28] G. Karypis, E. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[29] K. Koperski, J. Adhikary, and J. Han. Spatial data mining: progress and challenges survey paper. In *Proc. ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, Montreal, Canada*, pages 1–10. Citeseer, 1996.

[30] D. N. Krasnoshchekov and V. Polishchuk. Robust curve reconstruction with k-order alpha-shapes. In *Shape Modeling International*, pages 279–280, 2008. Full version: `http://www.cs.helsinki.fi/u/polishch/pages/koas.pdf`.

[31] A. Lucieer and M.-J. Kraak. α-shapes for visualizing irregular-shaped class clusters in 3d feature space for classification of remotely sensed imagery. *Proc. SPIE 5295, Visualization and Data Analysis*, 16, 2004.

[32] R. Maciejewski, R. Hafen, S. Rudolph, S. G. Larew, M. A. Mitchell, W. S. Cleveland, and D. S. Ebert. Forecasting hotspots — a predictive analytics approach. *Visualization and Computer Graphics, IEEE Transactions on*, 17(4):440–453, 2011.

[33] L. Mu and R. Liu. A heuristic alpha-shape based clustering method for ranked radial pattern data. pages 621–630. Applied Geography, Vol. 31, No. 2., 2011.

[34] R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases*, page 144. Morgan Kaufmann Pub., 1994.

[35] J. Sander, M. Ester, H. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, 1998.

[36] R. Sibson. SLINK: an optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973.

[37] R. van de Weygaert, E. Platen, G. Vegter, B. Eldering, and N. Kruithof. Alpha shape topology of the cosmic web. In M. A. Mostafavi, editor, *ISVD*, pages 224–234. IEEE Computer Society, 2010.

[38] J. Wang, W. Peng, M. O. Ward, and E. A. Rundensteiner. Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets. In *Information Visualization, 2003. INFOVIS 2003. IEEE Symposium on*, pages 105–112. IEEE, 2003.