Interactive Visual Clustering of Large Collections of Trajectories

Gennady Andrienko[1], Natalia Andrienko[1], Salvatore Rinzivillo[2], Mirco Nanni[2], Dino Pedreschi[3], Fosca Giannotti[2]

[1] Fraunhofer Institute IAIS (Intelligent Analysis and Information Systems), Sankt Augustin, Germany
[2] KDD Lab –ISTI –CNR, Pisa, Italy
[3] University of Pisa, Pisa, Italy

ABSTRACT

One of the most common operations in exploration and analysis of various kinds of data is clustering, i.e. discovery and interpretation of groups of objects having similar properties and/or behaviors. In clustering, objects are often treated as points in multi-dimensional space of properties. However, structurally complex objects, such as trajectories of moving entities and other kinds of spatio-temporal data, cannot be adequately represented in this manner. Such data require sophisticated and computationally intensive clustering algorithms, which are very hard to scale effectively to large datasets not fitting in the computer main memory. We propose an approach to extracting meaningful clusters from large databases by combining clustering and classification, which are driven by a human analyst through an interactive visual interface.

KEYWORDS: Spatio-temporal data, movement data, trajectories, clustering, classification, scalable visualization, geovisualization.

INDEX TERMS: H.1.2 [User/Machine Systems]: Human information processing – Visual Analytics; I.6.9 [Visualization]: information visualization.

1 INTRODUCTION

Nowadays our civilization faces an explosion of various kinds of space-related data, such as measurements from static and mobile sensors, GPS tracks, or georeferenced photos put on the Web by general people. As potential sources of valuable information and knowledge, these data call for scalable methods of analysis, which must take into account the particular features of the geographical space: its heterogeneity, variety of properties and relationships, spatial and temporal autocorrelation, anisotropy, and scale dependence. As all these features cannot be adequately modeled (yet) for fully automatic processing, the analysis relies heavily upon the human analyst's sense of the space and place and tacit knowledge of their inherent properties and relationships [4]. These are incorporated in the analysis through the use of appropriate visual representations of the space (in particular, maps) and interaction techniques.

Clustering is one of the general approaches to exploring and analyzing large amounts of data since it allows an analyst to consider groups of objects rather than individual objects, which are too numerous. Clustering associates objects in groups (clusters) such that the objects in each group share some properties that do not hold (or hold much less) for the other objects. Spatial clustering builds clusters from objects being spatially close and/or having similar spatial properties (shapes, spatial relationships among components, etc.).

http://geoanalytics.net/and andrienko@geoanalytics.net

IEEE Symposium on Visual Analytics Science and Technology
October 12 - 13, Atlantic City, New Jersey, USA
978-1-4244-5283-5/09/$25.00 ©2009 IEEE

In clustering, objects are often represented by feature vectors (in other words, points in multi-dimensional space of properties). However, structurally complex objects such as spatial time series, trajectories of moving entities, or spatial distributions cannot be adequately represented in this way. Therefore, the most widely used clustering algorithms, such as K-Means, KD-Tree, or SOM (self-organizing map), are not applicable. It is necessary to use special methods, which may be very complex computationally.

A problem of currently existing clustering methods devised for structurally complex objects is their lack of scalability with respect to the size of the data. Clustering involves numerous comparisons between objects, and the comparison of complex objects is by itself computationally complex and time-consuming. Implementations of clustering algorithms typically work only with objects loaded in the main computer memory, which is often impossible for real datasets. Out-of-memory implementations are technically possible but extremely time-consuming. This might not be a very big problem if the clusters they produce were the final, unmodifiable outcomes, but this is not the case. Clustering results by themselves have no meaning and value until a human analyst interprets them. However, all clustering techniques involve parameters, and different parameter settings lead to diverse results, which may be more or less meaningful to a human or may provide different complementary meanings. Hence, the analyst needs to run clustering several or even many times with different settings, which requires the reaction time to be short.

We suggest an approach that allows interactive cluster analysis of large numbers of structurally complex objects. The essence of the approach can be shortly described as follows. First, the analyst takes a manageable subset of the objects and applies clustering to it. The analyst experiments with the clustering parameters for gaining meaningful results with respect to the analysis goals. Then, the analyst builds a *classifier*, which can be used for attaching new objects to the existing clusters. The analyst may also modify the clusters for their better understandability and/or conformance to the goals. The produced classifier is applied to the whole dataset. Each object is either attached to one of the clusters or remains unclassified, if it does not fit in any cluster. When necessary, the analyst may repeat the procedure (take a subset – cluster – build a classifier – classify) to the unclassified objects.

This paper explains what *classifier* is and how it is built. We introduce the concept by example of clustering and classification of trajectories of moving entities; however, the approach is generic and can be applied to other types of structurally complex spatial and non-spatial objects.

2 RELATED WORK

There are two main approaches to clustering complex data: (i) defining ad hoc notions of clustering and clustering algorithms tailored to the specific data type; and (ii) applying generic notions of clustering and generic clustering algorithms by defining some distance function, which measures the similarity between data items. In the second case, the specifics of the data are completely encapsulated in the distance function.

Both approaches are applied in developing clustering methods for trajectories of moving objects. Examples of the first approach include grouping together objects which are likely to be generated from a common core trajectory by adding Gaussian noise [11][13], building (hidden) Markov models that try to explain the transitions between positions [2], and methods based on the search and measurement of simple (for instance linear) sub-segments of trajectories that match sufficiently well [15][16].

The second approach is taken, for example, in [18]: a generic density-based clustering algorithm is used with a distance function computing the average spatial distance between the trajectories within a time interval, which is automatically chosen through an iterative optimization procedure. Generic clustering algorithms can be implemented in such a way that the distance function itself becomes a parameter [20]. Hence, the clustering problem is reduced to choosing an existing algorithm that satisfies some general requirements (for instance scalability, tolerance to noise, ability to detect non-convex clusters, etc.) and designing the distance function that best suits the specifics of the data and the purposes of the analysis. A variety of distance functions have been proposed for trajectories, including the basic Euclidean distance (assuming that trajectories are represented by vectors of fixed length), spatial Euclidean distance average along the time [18], time series-inspired functions such as (dynamic) time warping distance [8][25] and Least Common Sub-Sequence (LCSS) measure [1][9], and direction-oriented distances [19][26].

Paper [20] describes a clustering tool that includes a density-based clustering algorithm OPTICS [6] equipped with a library of semantically and computationally different distance functions. The tool supports a step-wise analytical procedure called "progressive clustering": a simple distance function with a clear meaning is applied at each step, while successive application of different functions yields sophisticated interpretation of clusters. Visualization and interaction techniques play here a crucial role.

Interactive, visually-aided clustering procedures have been suggested also for other purposes [10]. Thus, the analyst may wish to tune clustering outcomes to his/her background knowledge by interactive post-processing, e.g. moving objects between clusters [7][17]. The analyst may direct the work of the algorithm, for instance, the training of a self-organizing map [21]. Interactive visualization may allow the analyst to compare results of several runs of clustering and investigate the sensitivity to parameters [23]. Clustering techniques are often included in visualization systems and toolkits, so that the analyst may, on the one hand, use visualization for examining and interpreting results of clustering, on the other hand, use results of clustering for further analysis by means of interactive visual techniques [14][22].

3 Approach

The approach we suggest involves the use of a generic density-based clustering algorithm such as OPTICS [6], which belongs to the DBSCAN family [12]. Advantages of these methods are tolerance to noise and capability to discover arbitrarily shaped clusters. A brief description of OPTICS is given in [20]. The clustering algorithm is used with a distance function suitable for the type of objects under analysis.

3.1 General procedure

The basic idea of the approach is to apply the clustering algorithm to a small subset of objects and then to attach the remaining objects to the clusters that have been discovered at the first stage. For this purpose, in each discovered cluster one or several *prototype objects* (or, shortly, *prototypes*) is (are) selected such that the distance of any other cluster member to one of these objects is below a certain threshold. The distance is measured by the distance function, which has been used for the clustering. The prototypes of the clusters, the respective distance thresholds (which may be prototype-specific), and the distance function form together a *classifier*.

Attaching new objects to the so defined clusters is done by comparing the objects to the cluster prototypes, i.e. finding the distances by means of the distance function. An object is attached to a cluster if its distance to one of the prototypes is below the respective threshold. If an object is close to prototypes of two or more clusters, the closest prototype is chosen. If an object is not sufficiently close to any of the prototypes, it remains unclassified.

Given a database ***D***, the whole process can be formalized as follows (Algorithm I):

1. Extract a subset ***D′*** of objects from ***D*** (see Section 3.2).
2. Apply the density-based clustering algorithm with a suitable distance function ***d*** and get a set of clusters $\{\mathbf{C_1}, \mathbf{C_2}, \ldots, \mathbf{C_m}\}$
3. For each cluster $\mathbf{C_i}$
   - Select q prototypes in $\mathbf{C_i}$, with $1 \le q < |\mathbf{C_i}|$, namely $\{ p_i^1, p_i^2, \ldots, p_i^q \}$, with corresponding distance thresholds $\{ \varepsilon_i^1, \varepsilon_i^2, \ldots, \varepsilon_i^q \}$ such that the cluster $\mathbf{C_i}$ may be described as the set of objects in ***D′*** whose distance to one of the prototypes p_i^j is less than the corresponding threshold ε_i^j, i.e.
     $C_i = \{ o \in \boldsymbol{D'} \mid \exists\, j, 1 \le j \le q, \text{ such that } \boldsymbol{d}(o, p_i^j) < \varepsilon_i^j \}$
     (see Section 3.3 for the details on how to choose prototypes for each cluster).

   The set of the prototypes for all clusters p_i^j together with their distance thresholds ε_i^j and function ***d*** form a classifier.
4. Visually inspect and refine the classifier; possibly, modify the clusters (see Section 3.4).
5. Apply the classifier to the remaining objects in ***D***: for each object $o \in \boldsymbol{D}$, $o \notin \boldsymbol{D'}$
   - Find all close prototypes, i.e. p_i^j, $1 \le i \le m$, such that $\boldsymbol{d}(o, p_i^j) < \varepsilon_i^j$. If only one close prototype p_i^j exists, attach o to the cluster $\mathbf{C_i}$ represented by p_i^j. If two or more close prototypes $p_{i1}^{j1}, \ldots, p_{iN}^{jN}$ exist, select the closest of them, i.e. such prototype p_{ik}^{jk}, that $\boldsymbol{d}(o, p_{ik}^{jk}) < \boldsymbol{d}(o, p_{in}^{jn})$ for $\forall n : 1 \le n \le N$, $n \ne k$; attach o to the cluster $\mathbf{C_{ik}}$ represented by p_{ik}^{jk}. If no close prototypes exist, the object remains unclassified.
6. Possibly, exclude the original and new members of clusters $\{\mathbf{C_1}, \mathbf{C_2}, \ldots, \mathbf{C_m}\}$ from ***D*** and restart the whole process again.

The computational time required for the classification (step 5) depends linearly on the number of objects in ***D***: each object is compared with a constant number of cluster prototypes (unlike clustering, where each object needs to be compared with all others). Hence, the algorithm is quite scalable with respect to the database size. Although step 5 may take minutes or even hours for a very big dataset, it does not require the involvement of a human analyst. Since the analyst can previously obtain meaningful, goal-oriented clusters by running the clustering method with different settings at step 2 and interactively refining the outcomes at step 4, the results of the following cluster-based classification will also be meaningful and conform to the goals of the analysis.

3.2 Selection of a subset

Algorithm I starts with a selection of a subset of the original dataset. The subset should have a manageable size and at the same time be representative of the dataset as a whole. An ideal sampling strategy must preserve the actual distribution of the objects in the original dataset. This would require the knowledge of this distribution, which is not always possible.

Uniform sampling from the database is a reasonable strategy when a density-based clustering algorithm is used: dense regions in the original dataset remain (relatively) dense also in the sample, and hence can be discovered by the algorithm. In a case when a dense region becomes too sparse in the sample, there is still a possibility of detecting it in the successive iteration of the process.

Specifics of the data and/or goals of the analysis may call for a certain way of selecting the subset, as will be seen in the description of an example analysis scenario (Section 4).

3.3 Selection of cluster prototypes

Selection of prototypes from density-based clusters is a non-trivial problem. In a density-based cluster, each object is close to a certain minimum number of other objects (this is a parameter of the algorithm). However, two arbitrary cluster members may be quite distant from each other; therefore, a cluster may have rather high internal variation. Fig.1A shows how a cluster of points may look like (we use points only for illustration purposes, to make the argument, which applies to any kind of objects, easier to understand). Fig.2 gives examples of density-based clusters of trajectories according to the similarity of their routes (the small hollow squares and the larger solid ones mark the starting and ending points of the trajectories, respectively).

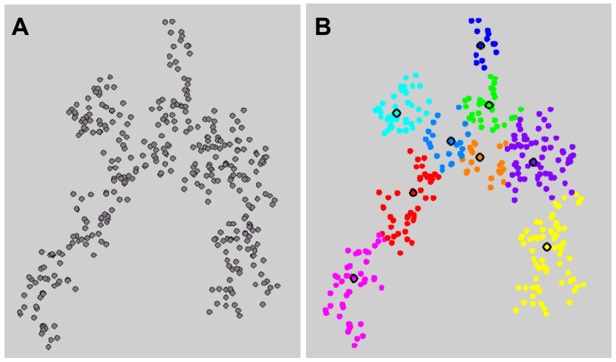


Figure 1. A) A density-based cluster of points. B) The cluster has been divided into "round" subclusters.

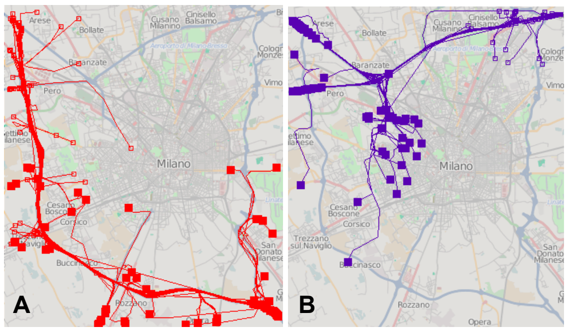


Figure 2. Examples of density-based clusters of trajectories according to the distance function "route similarity" [5].

To find appropriate prototypes in a density-based cluster, we suggest dividing it into "round" subclusters. A round (sub)cluster is a set of objects $\mathbf{S}=\{o_1, o_2, \ldots, o_k\}$ for which there is a special object o' and distance ε such that $\boldsymbol{d}(o_i, o')<\varepsilon$, $1 \le i \le k$, and for any other object $o \notin \mathbf{S}$, $\boldsymbol{d}(o, o') \ge \varepsilon$. The object o' (real or theoretical) is called *center* of the (sub)cluster **S**. The maximum distance among $\boldsymbol{d}(o_i, o')$, $1 \le i \le k$, is called *radius* of the (sub)cluster **S**. In a case when the objects are points, as in Fig.1, and the distance function $\boldsymbol{d}$ is Euclidean distance, the notions of center, radius, and round cluster can be understood literally. In a case of structurally complex objects (e.g. trajectories, as in Fig.2) and arbitrary $\boldsymbol{d}$ (e.g. a function measuring the similarity between the routes), these notions need to be understood metaphorically.

For complex objects and distance functions, finding the true center of a round (sub)cluster is a complex problem, not only computationally but also conceptually. However, for the purposes of building a classifier, the true centers are not really needed. They can be substituted by *medoids*. A *medoid* is a member of a subcluster having the smallest mean distance to all other members. Medoids may be used as cluster prototypes.

Formally, the problem of selecting cluster prototypes may be stated as follows: given a cluster **C**, a distance function $\boldsymbol{d}$, and a maximum distance threshold ε^{max}, divide **C** into subclusters $\{\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_n\}$ where for $\forall \mathbf{S}_i\ \exists m_i \in \mathbf{S}_i$ (medoid) and $\exists \varepsilon_i \le \varepsilon^{max}$ such that for $\forall o \in \mathbf{S}_i$, $\boldsymbol{d}(o, m_i)<\varepsilon_i$. For solving the problem, we suggest the algorithm described below (Algorithm II). At each stage, the status of the algorithm is represented by a list **L** where each entry consists of a subcluster and its corresponding medoid $<\mathbf{S}_i, m_i>$.

1. Create an empty list **L**.
2. Take the next object $o \in \mathbf{C}$. Try to find $<\mathbf{S}_i, m_i> \in \mathbf{L}$ such that $\boldsymbol{d}(o, c_i)<\varepsilon^{max}$. If found, go to 3, otherwise go to 4.
3. Let $\mathbf{S}_i' = \mathbf{S}_i \cup \{o\}$. Find the medoid m_i' of $\mathbf{S}_i'$. For each member $x \in \mathbf{S}_i'$, test the condition $\boldsymbol{d}(x, m_i')<\varepsilon^{max}$. If $\boldsymbol{d}(x, m_i')<\varepsilon^{max}$ holds for $\forall x \in \mathbf{S}_i'$, go to 3a, otherwise go to 4.
   a) Remove $<\mathbf{S}_i, m_i>$ from **L**.
   b) Put $<\mathbf{S}_i', m_i'>$ in **L**. Go to 2.
4. Put $<\{o\},o>$ in **L** (i.e. create a new subcluster with a single member o, which is also the medoid of the subcluster). Go to 2.

At the end, the list **L** represents a partitioning of the cluster **C** into round subclusters. The medoids of the subclusters become the prototypes of the original clusters. The maximum distance from a medoid to the members of its subcluster is taken as the distance threshold for this prototype. Fig.1B illustrates, by example of points, a possible outcome of Algorithm II. The subclusters are indicated by different colors of the circles representing their members. The medoids are marked by thick black boundaries.

Although the computational complexity of Algorithm II is $O(n^2)$, where n is the number of objects in cluster **C**, this is not critical due to the relatively small sizes of density-based clusters that can usually be discovered in a not so big subset $\boldsymbol{D'}$ of the database $\boldsymbol{D}$. Besides, the distances between the objects, which are needed for Algorithm II, are computed at the stage of density-based clustering (step 2 in Algorithm I) and can be later re-used, which substantially reduces the computation time.

It may seem that a standard clustering method like K-Means could be used to divide a cluster into round subclusters. There are two problems here. First, K-Means [24] and some other methods require computing the mean of multiple objects, which is too difficult for complex objects. K-Medoids [24] could suit better as it uses medoids instead of means. However, the second problem, common for K-Means and K-Medoids, is that the number of subclusters must be known in advance, which is not the case.

3.4 Visual inspection and refinement

There are at least two motives for revising the automatically built classifier. First, density-based clusters with high internal variation are difficult to understand. The analyst may wish to refine them by dividing into parts with smaller internal variation and/or by removing some of the members. Second, the analyst may wish to tune the selection of cluster prototypes and distance thresholds to his/her understanding of the distinctive properties of the clusters.

Furthermore, the analyst needs to make sure that the classifier will correctly assign new objects to the defined clusters. This can be tested by applying the classifier to $\boldsymbol{D'}$. Since the assignment of objects to clusters is done in different ways in the classification and in the density-based clustering, the outcomes of the classification may differ from the original clusters. Some of the original members of a cluster may not be there any more (such objects will be called false negatives), and/or some new objects may be put in the cluster (such objects will be called false positives). This discrepancy is not necessarily disadvantageous. It may happen that a false negative is too dissimilar to the other objects in the cluster and should rather not be there, and it may also happen that a false positive is sufficiently similar to the core objects of the cluster and should rather be there. Hence, each case of divergence between the two assignments of the objects to the clusters needs to be inspected by the analyst. If the analyst is not satisfied with the new assignment, he/she should be able to refine the part of the classifier responsible for the misclassification.

To enable the revision and refinement of the classifier, we suggest the following operations:

1. Exclude one or several subclusters from a cluster and perform one of the following actions:
   a) make a new cluster as a union of these subclusters;
   b) turn each subcluster into a new cluster;
   c) discard the subclusters, i.e. treat their members as not belonging to any cluster.
2. Divide a subcluster into two or more smaller subclusters.
3. Merge two or more subclusters into a single larger subcluster.
4. "Dissolve" one or more subclusters, i.e. distribute their members among the remaining subclusters.
5. Change the distance threshold of a selected prototype.

The operations 2, 3, and 4 involve automatic re-computing of the medoids of the subclusters. For dividing a subcluster into smaller subclusters (operation 2), the K-Medoids method may be applied. It may be modified so that the analyst could select the initial seeds for the new subclusters.

The process of reviewing and revising the classifier is supported by appropriate visual representations of the subclusters and their medoids and interactive facilities for

– focusing on a cluster as a whole or on one or more subclusters,
– selecting one or more subclusters for a desired operation, and
– selecting candidate seeds for dividing a subcluster.

After any operation, the analyst visually inspects the results and, possibly, runs the test of the classifier on $\boldsymbol{D'}$. If the results are not satisfactory, the analyst may revert to the previous state. All operations are logged, and it is possible to trace how the current state of the classifier has been derived. This also helps to explain the process to others and to re-produce the result when necessary.

4 An Example Analysis Scenario

4.1 Analytical environment

The example scenario is about analyzing a large set of trajectories of moving entities (while the suggested method is generic and applicable also to other types of data). We use our implementation of the method, which is incorporated in a visual analytics toolkit for spatial and spatio-temporal data. The implementation takes advantage of the general visualization and interaction techniques available in the toolkit (cartographic visualization, interactive filtering, interactive selection of objects in a display, etc.) as well as specific visual and computational techniques and database queries oriented to trajectories. The data are stored in a standard relational database (Oracle) in the form of position records <entity_id, trajectory_id, time, position, {attributes}>, where the position consists of two coordinates, geographical (longitude and latitude) or Cartesian (X and Y). When the data are loaded in the toolkit, position records with the same trajectory_id are used to construct objects of a special type representing trajectories [5].

The initial clustering of a subset of trajectories is done by our implementation of the density-based algorithm OPTICS with a set of trajectory-oriented distance functions [20]. In the scenario, we use the function "route similarity", described in [5]. The distance (amount of disparity) between two trajectories depends on the similarity of the geometric shapes of their footprints and the closeness of their spatial positions and orientations. Fig.2 shows examples of clusters detected by OPTICS with "route similarity" function. Note that density-based clusters do not necessarily have so high internal variation as in these specially selected examples. Another important note is that density-based clustering does not put every object in some cluster. When an object is not similar enough to a certain number of other objects, it is treated as "noise", i.e. stays outside of any cluster. The required minimum number of similar objects and the maximum distance to each of them are the parameters of the clustering method [20].

In Fig.2, clusters of trajectories are visualized by drawing their individual members on a map. Each cluster is assigned a specific color, in which its members are painted. There is an interactive tool allowing the user to choose which clusters will be visible. Clusters can also be represented in a summarized way, as described in [5] and [3]. The user may get a visual overview of clustering results in the form of a panel with multiple small maps each representing one cluster in a summarized form. This may be stored as an HTML page. When building a classifier, the user may document the process using the available logging tools, which produce a collection of linked HTML pages.

4.2 Data and analysis task

In this imaginary but realistic scenario, we play a role of analysts who received a task from the mobility and transportation planning department of a big city. The department has obtained a dataset collected by tracking about 17,200 cars that moved in the city during one week. The mobility managers used to do their analyses and planning with the help of traffic models, which are created on the basis of costly and time-consuming population surveys carried out once in several years. Now there is an opportunity to use large amounts of cheap and up-to-date data, and, consequently, a need in methods for extracting useful information from these data.

In our scenario, the mobility managers gave us the data with the task to extract the typical (i.e. frequently occurring) routes of commuters in the city. The knowledge is needed for planning improvements in the traffic infrastructure and/or public transport. The task may be re-formulated as finding big groups of trajectories following the same or very similar routes. This task can be accomplished by means of cluster analysis using the distance function "route similarity". However, the dataset consisting of about 176,000 trajectories (over 2 million position records) is too big for loading RAM and direct application of a clustering algorithm; hence, this is the case for our Algorithm I.

4.3 Subset selection and clustering

The initial subset of trajectories must be manageable in terms of the size but representative in terms of the probability of finding the clusters we are interested in. According to our knowledge, the most intensive movement of commuters occurs in the mornings of working days. Hence, we take a typical working day such as Wednesday (the middle of the week) and extract from the database all trajectories that occurred from 6 till 10 AM on that day. This gives us 6,591 trajectories, which is quite manageable.

For the clustering, we use the distance threshold (maximum deviation) 500 meters for treating two trajectories as similar and set 3 as the required minimum number of neighbors (similar trajectories) for each trajectory in a cluster. The first parameter is responsible for the amount of internal variation in the clusters. The value 500 has been chosen after several experimental runs of the tool with different threshold values. The second parameter determines the density of the clusters. Although we are interested in dense clusters, we keep in mind that we deal now with a small subset (about 3.5%) of the whole set. Dense clusters existing in the whole dataset may be represented by much sparser clusters in the subset. Hence, if we want the clusters to be detected, we should not choose a very high value for the second parameter.

The application of the clustering tool to the subset of trajectories gives us 138 clusters with the sizes ranging from 4 to 102; 4,708 trajectories (71.48%) are labeled as "noise". We look at the cluster overview display, where the clusters are represented in a generalized and summarized form (by arrows with the thickness proportional to the number of corresponding moves; see Figs. 3 and 4). We notice three major types of routes, illustrated in Fig.3: routes that pass the city by a belt road without entering the inner city (Fig.3A), very short trajectories (Fig.3B; summarized movements within small areas are represented by rings with the thickness proportional to the number of the trajectories), and the routes entering the city (Fig.3C). Only the latter type of route is relevant to the analysis task.

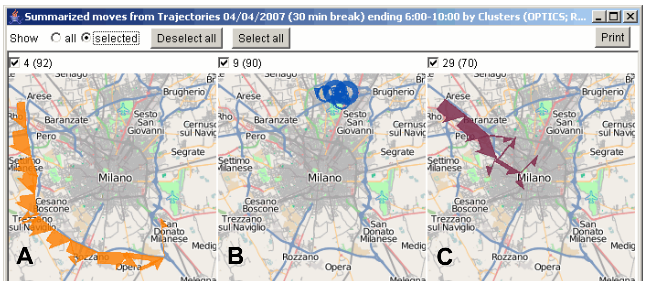

Summarized moves from Trajectories 04/04/2007 (30 min break) ending 6:00-10:00 by Clusters (OPTICS; R...
Show
all
selected
Deselect all
Select all
Print
4 (92)
9 (90)
29 (70)
Milano
A
B
C

Figure 3. A fragment of the cluster overview panel.

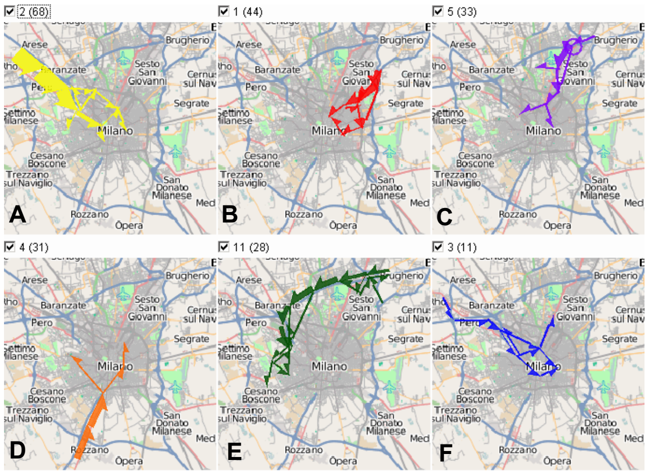


Figure 4. The biggest clusters of trajectories ending in the city.

To get rid of numerous irrelevant clusters, we apply filtering to the subset of trajectories. Interactively (by drawing on the map), we define the area of inner city and select only the trajectories ending in this area. We also filter the trajectories by their length, which must be at least 3 km. 2,028 trajectories satisfy both filters. We apply clustering to these trajectories and obtain 21 clusters with the sizes ranging from 4 to 68; 1,720 trajectories (84.8%) go to "noise". The routes, presented in the cluster overview panel (see a fragment in Fig.4), correspond quite well to our idea of possible commuter routes. Now we shall build a classifier to find out how frequent these routes are in the entire database.

4.4 Building the classifier

A starting version of the classifier is generated automatically. For this, we specify the maximum distance threshold for a cluster prototype (1000 meters). The tool divides the clusters into round subclusters (the number ranges from 1 to 7) and computes the medoids. Now we shall inspect and refine the classifier.

To have a convenient overview of the classifier, we use the documenting function, which generates a set of HTML pages displayed in a browser. The first page includes small images of all clusters and a summary table informing us about the size of each cluster, the number of subclusters, and the maximum distance threshold. An image of a cluster (e.g. Fig.5A) shows all trajectories of this cluster by thin, neutrally colored lines and the cluster prototypes (i.e. medoids of the subclusters) by thicker lines painted in the color of this cluster. The image serves as a hyperlink to the page describing the division of the respective cluster. The latter page contains a table listing all cluster prototypes, their distance thresholds, the subcluster sizes, and the mean distances from the prototypes to the subcluster members. The page also includes a bigger image of the whole cluster and small images of all its subclusters. An image of a subcluster includes the trajectories of the subcluster and the medoid, which are shown in the same way as in an image of a cluster. Such an overview page allows us to grasp immediately how the cluster has been divided and how the subclusters differ from each other.

By viewing the pages, we find that editing is needed in the parts of the classifier representing the clusters with high internal variation. All such clusters have multiple subclusters. Thus, cluster 1 (Fig.4B) has 7 subclusters. Fig.5A presents the cluster with its prototypes. We notice spatial separation among the prototypes: a group of 5 prototypes (Fig.5B) lies northward of the remaining 2 prototypes (Fig.5C). As these two groups represent different routes toward the city center, it is reasonable to have them in two distinct clusters. Hence, we select the two subclusters representing the southern route and use operation 1a (Section 3.4) to extract them from cluster 1 and produce a new cluster.

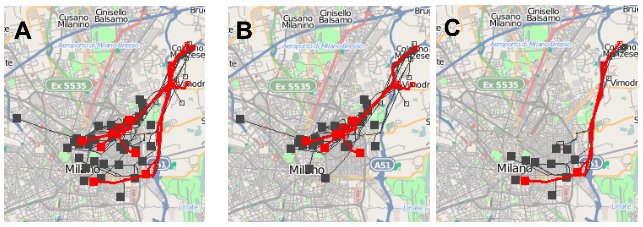


Figure 5. Cluster 1 should be split in two clusters.

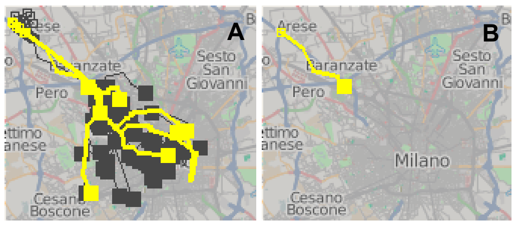


Figure 6. Cluster 2 (A) and its singular prototype (B).

Cluster 2 (Fig.4A) has been also divided into 7 subclusters, which reflects the branching of the tracks inside the city (Fig.6A). One subcluster consists of a single trajectory (Fig.6B). We find that this trajectory should rather not be a cluster prototype: it is not much interesting as a commuter route since it does not enter the inner city but ends near its boundary. Hence, we remove this subcluster with a single member from the cluster and discard it (operation 1c). In a similar way, we edit cluster 5 (Fig.4C). We discard a subcluster consisting of a single self-intersecting trajectory that can hardly be treated as a typical commuter route.

In cluster 6 (Fig.7A), one of the 3 prototypes has a notably higher distance threshold (1000) than the distance thresholds in all other clusters. To refine the corresponding subcluster (Fig.7B), we divide it into two subclusters using operation 2. The tool finds a possible division and presents it to us as a suggestion. The division takes place after our approval (Fig.7C); the distance thresholds of the 2 new prototypes are 252 and 262.

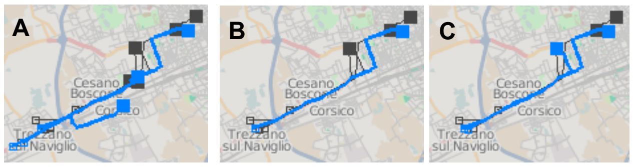


Figure 7. A) Cluster 6 with its 3 prototypes. B) The subcluster with a big radius. C) The subcluster has been refined.

The remaining parts of the classifier do not seem to require editing. We run the automatic test of the classifier, in which the classifier is applied to the subset of 2,028 trajectories used in the initial clustering. In the test, the classifier assigns the trajectories to the clusters, and the assignment is compared with the original membership of the trajectories in the clusters. As mentioned earlier, two types of discrepancy are possible: false negative and false positive. The documenting tool produces a summary table with test results, where we can see that 11 out of 22 clusters have from 1 to 7 false positives and 2 clusters have 1 false negative each. These cases need to be inspected. The interactive facilities allow us to focus on the false positives or false negatives of each cluster and to compare them with the cluster prototypes and the true members of the cluster. We see that in all but one cases the false positives are quite similar to the original members of the clusters. For example, Fig.8 presents the false positives of clusters 1 and 6 (portrayed by thin dark grey lines) together with the prototypes of these clusters (shown by thick colored lines). The false positives are consistent with the cluster prototypes and the other cluster members (visible in Fig.5B and Fig.7A). These and other cases do not require any corrective means.

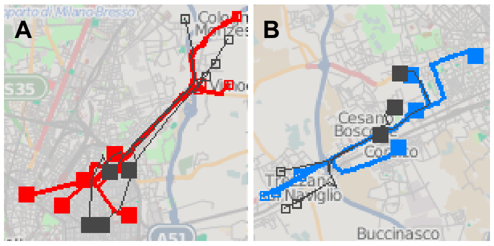


Figure 8. The false positives of clusters 1 (A) and 6 (B) shown together with the prototypes of these clusters.

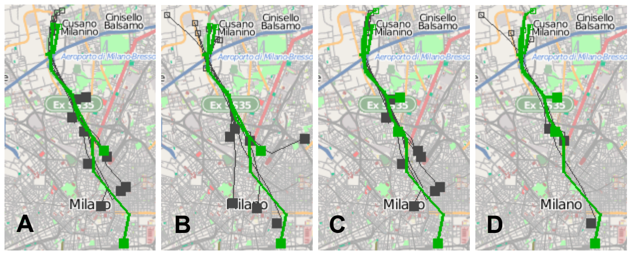


Figure 9. A) Cluster 17 with 2 original prototypes; B) the false positives; C) cluster 17 with 3 prototypes after a refinement; D) the false positives after the refinement.

However, cluster 17 (Fig.9A) has acquired 7 false positives (Fig.9B), some of which deviate quite much from the original cluster members. To improve the situation, we try a different division of cluster 17 into subclusters. We merge the original 2 subclusters (operation 3) and divide the cluster into 3 subclusters (operation 2). The resulting prototypes are in Fig.9C. We run the test again, and the cluster acquires only 3 false positives (Fig.9D), which are consistent with the original cluster members.

The cases with false negatives can be handled in two possible ways: (1) by increasing the distance thresholds of the prototypes of the subclusters in which these trajectories have been originally; (2) by refining the subclusters so that the false negatives become additional cluster prototypes. We choose the second approach and use the possibility to select candidate prototypes for the resulting subclusters. In both cases of false negatives, we obtain additional subclusters with singular members and radii equal to 0. We have to specify explicitly the desired distance thresholds for the respective prototypes. We choose 200m, which equals the minimum threshold among the other prototypes. Then we run the test once again and see that there are no false negatives any more and no additional false positives have appeared.

Now we are satisfied with the classifier. We use the documenting tool for producing a set of HTML pages presenting the final state of the classifier and the results of the test. We also save the classifier in a file in a special XML format, so that the classifier can be loaded and used in another session.

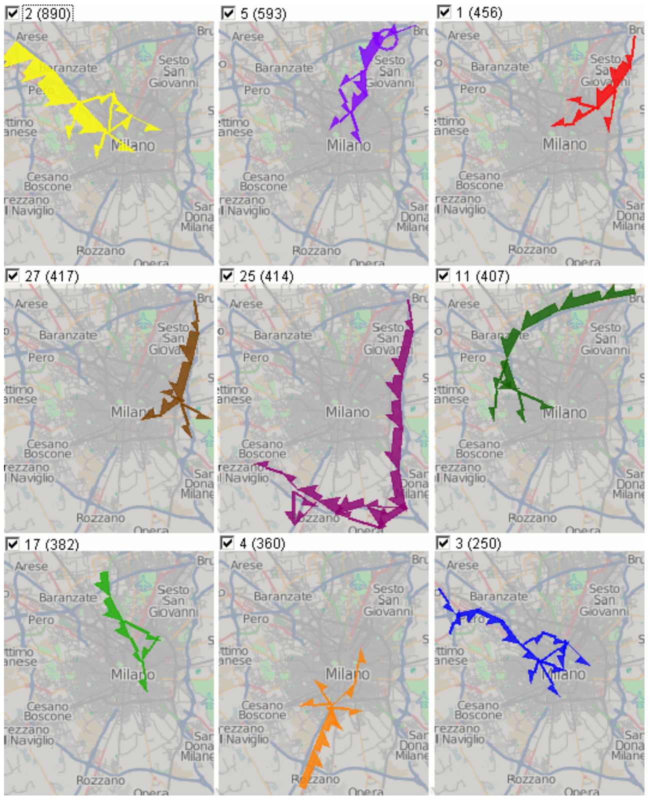
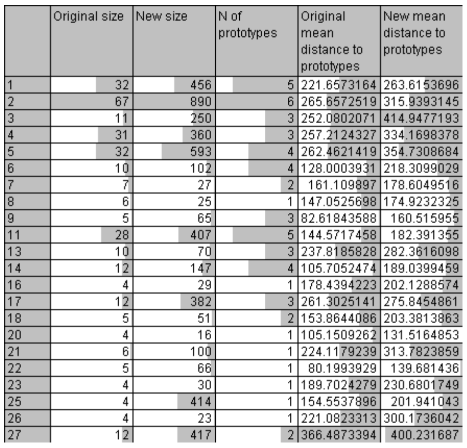


Figure 10. The graphical summaries of the biggest clusters obtained by applying the classifier to the entire dataset.

4.5 Applying the classifier to the whole database

A classifier is applied to the whole database in the following way. We remind that our database does not contain trajectories as special constructs but only position records. These records are loaded in the visual analytics system by small portions. The system constructs trajectories from the loaded records and applies the classifier to each trajectory. To store the results, the system creates a lookup table in the database, where it makes a record for each trajectory with its identifier and the number of the cluster it has been attached to or -1 if it does not fit in any cluster. During the classification process, the system also incrementally builds graphical summaries of the clusters, analogous to those of the initial clusters (Fig.4). When a trajectory is attached to a cluster, the graphical summary of this cluster is updated. The processed trajectories are discarded.

In our scenario, we apply the classifier to the whole database, which defines 175,890 trajectories. Although it is possible to exclude the initial subset of trajectories from the classification, we do not use this option since we want the lookup table to contain records for all trajectories. The classification together with the graphical summarization takes about 14.5 minutes. Fig.10 presents a fragment of the display with the graphical summaries of the clusters. Fig.11 shows the table display of the statistics about the clusters including the initial and new sizes and the original and new mean distances of the cluster members to the prototypes.



	Original size	New size	N of prototypes	Original mean distance to prototypes	New mean distance to prototypes
1	32	456	5	221.6573164	263.6153696
2	67	890	6	265.6572519	315.9393145
3	11	250	3	252.0802071	414.9477193
4	31	360	3	257.2124327	334.1698378
5	32	593	4	262.4621419	354.7308684
6	10	102	4	128.0003931	218.3099029
7	7	27	2	161.109897	178.6049516
8	6	25	1	147.0525698	174.9232325
9	5	65	3	82.61843588	160.515955
11	28	407	5	144.5717458	182.391355
13	10	70	3	237.8185828	282.3616098
14	12	147	4	105.7052474	189.0399459
16	4	29	1	178.4394223	202.1288574
17	12	382	3	261.3025141	275.8454861
18	5	51	2	153.8644086	203.3813863
20	4	16	1	105.1509262	131.5164853
21	6	100	1	224.1179239	313.7823859
22	5	66	1	80.1993929	139.681436
23	4	30	1	189.7024279	230.6801749
25	4	414	1	154.5537896	201.941043
26	4	23	1	221.0823313	300.1736042
27	12	417	2	366.4873394	400.231687

Figure 11. The statistics about the classification results.

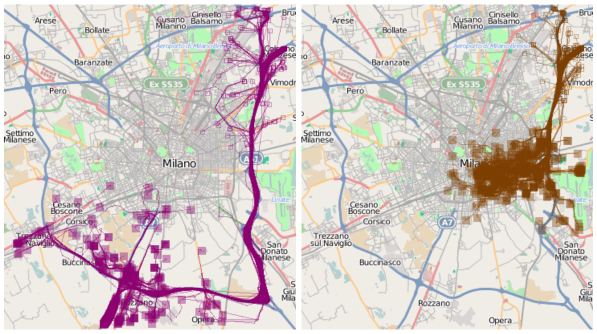


Figure 12. Clusters 25 (left) and 27 (right).

As may be seen, the biggest clusters discovered in the subset of trajectories remain among the biggest ones in the entire set. The graphical summaries of the clusters extracted from the whole set (Fig.10) are consistent with those of the original clusters (Fig.4). The small clusters from the subset mostly remain small in the whole set, but a few exceptions require investigation. Thus, the graphical summaries of clusters 27 and 25 (left and middle in the second row in Fig.10) make us think that the clusters may include trajectories that do not end in the inner city. The system allows us to load the trajectories belonging to selected clusters (the lookup table is used for this purpose). We load the trajectories of clusters 25 and 27 (Fig.12). Indeed, very many trajectories of cluster 25 and some trajectories of cluster 27 end outside of the city. We apply the spatial filtering, like we did for the initial subset (Section 4.3), and see that only 107 of the 414 trajectories (25%) of cluster 25 end in the inner city. The reason is that the cluster is represented in the classifier by a single prototype trajectory ending close to the boundary of the inner city. This prototype turns out to be similar to many trajectories ending outside. Hence, it is appropriate to choose another prototype or set of prototypes for the route represented by cluster 25. For cluster 27, the ratio is 347 to 417 (83%), which is not as bad; however, the choice of the prototypes can also be improved.

4.6 Further iterations

To continue the analysis, we create a new database table with the data for the unclassified trajectories, as indicated in the lookup table. From the new table, we extract the trajectories from the morning of another working day (Tuesday). We handle this new subset like the initial subset (Section 4.3). Clustering of the new subset gives us only 7 clusters with the sizes from 4 to 6, which makes us think that the most significant clusters have been discovered in the previous iteration. Still, we build a classifier and apply it to the table with the unclassified trajectories. This time, two biggest clusters contain 126 and 86 trajectories, and the remaining clusters are much smaller. We repeat the procedure also for Monday. The subset contains 6 clusters with sizes from 4 to 7, but one of them grows to 204 in the result of the cluster-based classification. A closer look at these trajectories reveals that they mostly occur in the afternoons, which explains why the cluster was so small in the subset composed from morning trajectories.

We do a couple of additional iterations to be sure that we have not missed any significant clusters. At the end, we take the largest clusters discovered during the whole process (17 clusters with the sizes from 95 to 890) as representing the most frequent routes. For these routes, we compute the frequencies of their occurrence by days of the week and by times of the day and visualize them by segmented histograms. We make a final report using the HTML pages generated in the course of the analysis.

Classifiers created in different iterations can be combined into one classifier, which can be stored externally for further uses. It may be sensible to re-apply the combined classifier to the whole dataset for having continuous numbering of the clusters and one lookup table with all results.

5 DISCUSSION

The suggested approach is generic, i.e. applicable to different types of structurally complex objects. However, to make it work for a particular type of objects, certain type-specific components are necessary: (1) a database representation of the objects; (2) a distance function; (3) a visual representation of the objects; (4) optionally, methods for graphical summarization of clusters. This is demonstrated by the example scenario of analyzing a large set of trajectories. The visual representation of the objects should correspond to the distance function, i.e. exhibit the properties of the objects accounted for in the distance function. The same applies to the summarized representation. Thus, in the example scenario, the distance function compares trajectories according to their routes; hence, the visualization should exhibit the routes. The cartographic representations of individual trajectories and clusters that we have used are suitable for this purpose. However, in a case of distance function that takes into account also the temporal aspect of the trajectories, these representations are inadequate. It is necessary to exhibit the temporal component, for example, by involving an additional display dimension to represent time.

An essential feature of the approach is the division of labor between computer and human and a true synergy where each side helps the other. Not only the computer gives its computational power to the human but also human's knowledge and reasoning capabilities help the computer. Thus, clustering algorithms do not scale to very large sets of structurally complex objects. In our approach, cluster analysis of very large datasets becomes possible owing to the human analyst, who directs the work of the computer to the discovery of meaningful, relevant clusters. The direction is realized through the following activities: (1) selection of a subset for the initial clustering on the basis of the analyst's knowledge of the domain, data, and problem to be solved; (2) selection of appropriate clustering parameters, by trying different variants and evaluating the results; (3) editing of the automatically built classifier, which involves interpretation, evaluation, and adaptation to the goals of analysis (it may be said that the analyst imbues the classifier with meaning). The computer, from its side, supports these activities by visualization and tools for interaction. The computer also helps in producing reports by documenting states, operations, and results.

However, not only a report is the outcome of the analysis. The classifier itself is a valuable material result, which can be used for further analyses. Thus, the classifier can be applied to a different dataset (e.g. from another time period). It can efficiently classify new data coming in real time. Furthermore, the distance function in the classifier can be modified so as to be able to assess the similarity of an object to a prototype having only partial information about the object (e.g. a fragment of a trajectory). With such a distance function, the classifier can be used for real-time prediction (e.g. prediction of the future movement).

Editing of a classifier may require significant human's effort. More specifically, the editing effort is high for big clusters with high internal variation. Usually, such clusters need to be refined by splitting and/or by removing inconsistent objects. The reward is "clean", easily understandable clusters extracted from the whole dataset. When a cluster is originally coherent, little or no editing of the respective part of classifier is needed. Our experience shows that, with an appropriate subset selection, big and "dirty" clusters mostly appear at the first iteration of the analysis. At the following iterations, the discovered clusters tend to become much smaller due to the decreasing density of the data. The internal variation in the clusters is also small. The number of discovered clusters also decreases. Hence, the editing effort significantly lessens with each iteration. Thus, in our experiments, we typically spent 30-45 minutes for reviewing and editing the classifier at the first iteration and only 5-10 minutes at the following iterations (mainly reviewing, almost no editing was required).

It might be appropriate to compare the clusters built by means of our interactive and iterative method with results of automated clustering of the complete dataset. However, this is currently not possible because of the lack of a scalable clustering algorithm suitable for trajectories. As noted in the introduction, the existing scalable methods like KD-Tree or SOM can only be applied to feature vectors while trajectories of moving objects cannot be adequately represented in this form.

6 Conclusion

Finding clusters in very large sets of structurally complex objects, such as spatial and spatio-temporal objects, is a complex problem. Existing clustering algorithms are not scalable to very large datasets. We suggest a visual analytics approach to solving this problem at the cost of involving a human analyst, who directs the work of the computer towards the discovery of meaningful, relevant clusters. The approach is generic, i.e. can be applied to different types of complex objects. By an example analysis scenario, we have demonstrated a possible use of the approach for the analysis of trajectories of moving objects.

References

[1] Agrawal, R., Lin, K.-I., Sawhney, H. S., and Shim, K.: Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *Proc. VLDB* 1995, 490–501.

[2] Alon, J., Sclaroff, S., Kollios, G., and Pavlovic, V. Discovering clusters in motion time-series data. In *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR'03)*, IEEE, 2003, 375–381.

[3] Andrienko, G., and Andrienko, N.: Spatio-temporal aggregation for visual analysis of movements. In *Proc. IEEE VAST 2008*, 51-58.

[4] Andrienko, G., Andrienko, N., Dykes, J., Fabrikant, S., and Wachowicz, M.: Geovisualization of dynamics, movement and change: key issues and developing approaches in visualization research. *Information Visualization*, 2008, v.7 (3/4), 173-180.

[5] Andrienko, G., Andrienko, N., and Wrobel, S.: Visual analytics tools for analysis of movement data, A*CM SIGKDD Explorations*, 9(2), 2007, 38-46

[6] Ankerst, M., Breunig, M., Kriegel, H.-P., and Sander, J. Optics: Ordering points to identify the clustering structure. In *Proc. ACM SIGMOD* 1999, 49–60.

[7] Assent, I., Krieger, R., Müller, E., and Seidl, T. VISA: visual subspace clustering analysis. *ACM SIGKDD Explorations*, 9(2), 2007, 5-12

[8] Berndt, D., and Clifford, J. Using dynamic time warping to find patterns in time series. In *Proc. Knowledge Discovery and Delivery Workshop,* 1994, 359–370.

[9] Bozkaya, T., Yazdani, N., and Özsoyoglu, Z. M. Matching and indexing sequences of different lengths. *Proc. CIKM* 1997, 128–135.

[10] Ceglar, A., Roddick, J.F., and Calder, P. Guiding knowledge discovery through interactive data mining. In Pendharkar, P.C. (ed.) *Managing data mining technologies in organizations: techniques and applications*, Idea Group Publishing, 2003, 45-87

[11] Chudova, D., Gaffney, S., Mjolsness, E., and Smyth, P. Translation-invariant mixture models for curve clustering. In *ACM KDD* 2003, 79–88.

[12] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. ACM KDD* 1996, 226-231.

[13] Gaffney, S., and Smyth, P. Trajectory clustering with mixture of regression models. In *Proc. ACM KDD* 1999, 63–72.

[14] Guo, D., Chen, J., MacEachren, A.M., and Liao, K. A visualization system for spatio-temporal and multivariate patterns (VIS-STAMP), *IEEE TVCG*, 12(6), 2006, 1461-1474

[15] Hwang, S.-Y., Liu, Y.-H., Chiu, J.-K., and Lim, E.-P. Mining mobile group patterns: A trajectory-based approach. In *Proc. 9th Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'05)*, Springer, 2005, 713–718.

[16] Li, Y., Han, J., and Yang, J. Clustering moving objects. *In Proc. ACM KDD* 2004, 617–622.

[17] Nam, E.J., Han, Y., Mueller, K., Zelenyuk, A., & Dan, I. ClusterSculptor: A visual analytics tool for high-dimensional data. In *Proc. IEEE VAST 2007*, 2007, 75-82

[18] Nanni, M., and Pedreschi, D. Time-focused density-based clustering of trajectories of moving objects. *Journal of Intelligent Information Systems*, 27(3), 2006, 267–289.

[19] Pelekis, N., Kopanakis, I., Marketos, G., Ntoutsi, I., Andrienko, G., and Theodoridis, Y. Similarity search in trajectory databases. In *TIME*, 2007, 129–140.

[20] Rinzivillo, S., Pedreschi, D., Nanni, M., Giannotti, F., Andrienko, N., and Andrienko, G.: Visually–driven analysis of movement data by progressive clustering, *Information Visualization*, 7(3/4), 2008, 225-239.

[21] Schreck, T., Bernard, J., Tekusova, T., and Kohlhammer, J. Visual cluster analysis in trajectory data using editable Kohonen maps. In *Proc. IEEE VAST 2008*, 2008, 3-10.

[22] Seo, J., and Shneiderman, B. Interactively exploring hierarchical clustering results. *IEEE Computer*, 35(7), 2002, 80-86

[23] Sharko, J., Grinstein, G., Marx, K. A., Zhou, J., Cheng, C., Odelberg, S., and Simon, H.-G. Heat map visualizations allow comparison of multiple clustering results and evaluation of dataset quality: application to microarray data. In *Proc. International Conference Information Visualization*, 2007, 521-526

[24] Tan, P.-N., Steinbach, M., and Kumar, V. *Introduction to Data Mining.* Addison-Wesley, 2006

[25] Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., and Keogh, E. J. Indexing multi-dimensional time-series with support for multiple distance measures. In *Proc. ACM KDD* 2003, 216–225.

[26] Vlachos, M., Kollios, G., and Gunopulos, D. Discovering similar multidimensional trajectories. In *Proc. 18th Int. Conf. on Data Engineering (ICDE'02)*, IEEE, 2002, 673–684.