Visual Cluster Analysis of Trajectory Data With Interactive Kohonen Maps

Tobias Schreck*
Interactive Graphics Systems Group
TU Darmstadt, Germany

Jürgen Bernard†
Interactive Graphics Systems Group
TU Darmstadt, Germany

Tatiana Tekušová‡
Interactive Graphics Systems Group
TU Darmstadt, Germany

Jörn Kohlhammer§
Fraunhofer Institute for Computer Graphics IGD
Darmstadt, Germany

Abstract

Visual-interactive cluster analysis provides valuable tools for effectively analyzing large and complex data sets. Due to desirable properties and an inherent predisposition for visualization, the *Kohonen Feature Map* (or Self-Organizing Map, or SOM) algorithm is among the most popular and widely used visual clustering techniques. However, the unsupervised nature of the algorithm may be disadvantageous in certain applications. Depending on initialization and data characteristics, cluster maps (cluster layouts) may emerge that do not comply with user preferences, expectations, or the application context.

Considering SOM-based analysis of trajectory data, we propose a comprehensive visual-interactive monitoring and control framework extending the basic SOM algorithm. The framework implements the general Visual Analytics idea to effectively combine automatic data analysis with human expert supervision. It provides simple, yet effective facilities for visually monitoring and interactively controlling the trajectory clustering process at arbitrary levels of detail. The approach allows the user to leverage existing domain knowledge and user preferences, arriving at improved cluster maps. We apply the framework on a trajectory clustering problem, demonstrating its potential in combining both unsupervised (machine) and supervised (human expert) processing, in producing appropriate cluster results.

Index Terms: H.4 [Information Systems]: Information Systems Applications; I.3.6 [Computing Methodologies]: Computer Graphics—Methodology and Techniques.

1 Introduction

Cluster analysis is a process for structuring and reducing data sets by finding groups of similar data elements [7]. It is regarded as one of the core tools to effectively analyze large data volumes. This process is usually unsupervised: Up to parameterization, most algorithms work fully automatic and the user has no further means to determine the clusters. However, only appropriate clusterings effectively support the user in analyzing large data sets. Visual cluster analysis is a specialization of general cluster analysis and relies on the appropriate visualization of clusters. Some of the most popular approaches perform a spatialization of the cluster centers to display space, trying to preserve essential relationships among the clusters, while visualizing additional data properties such as the number of represented data items or measures of cluster quality. To date, the Self-Organizing Map (SOM) algorithm [14] proposed by Kohonen is one of the most popular visual cluster algorithms. It effectively combines clustering and spatialization by learning cluster prototypes located on a grid structure embedded in low dimensional space. However, to the best of our knowledge none of the existing SOM implementations allows the user to monitor and steer the clustering process using visual-interactive means.

In this paper, we focus on trajectory data, which is a ubiquitous type of data important in many applications. For instance, enabled by tracking technology, it is possible to routinely collect large amounts of geo-referenced movement data. Also, trajectories consisting of observation sequences in arbitrary vector spaces, e.g., time-dependent observations in 2D diagram space can be regarded. Visual analysis in the trajectory data domain often faces very large data sets, which cannot be visualized effectively per se. Trajectory cluster analysis is a promising option to this end. In previous work [18], we applied the SOM algorithm to visually analyze sets of trajectories observed in diagram space (cf. Section 3). We observed that the fully automatic cluster analysis may yield meaningful cluster spatialization. However, we also recognized that there is a need to more closely integrate the expert user in the clustering process.

Based on these observations, we propose to extend the automatic (unsupervised) SOM algorithm by a visual-interactive control and analysis framework. The framework allows the analyst to *guide* the otherwise purely automatic Self-Organizing Map algorithm toward resembling user-defined trajectory cluster maps. Thereby, it allows the user to factor in domain knowledge, application needs, and user preferences. The framework allows the user to visually monitor and understand the otherwise black-box clustering process, and control it at an arbitrary level. The user can use it to obtain appropriate cluster maps from the full spectrum of maps generated either completely unsupervised or completely supervised.

2 Related Work

This work relates to a number of research strands. In general, this work follows the Visual Analytics idea of integrating automatic data analysis with human expertise, relying on visual-interactive means [19, 13]. Cluster analysis is one key data mining technique of which many automatic approaches exist [12, 11, 7]. Clusters may be found e.g., by centroid or medoid-based approaches, hierarchical models, or density-based approaches. Visualization is often key to understand otherwise possibly abstract clustering results. While certain clustering approaches implicitly yield visual representations (e.g., dendrograms or 2D mappings), for many other clustering techniques, appropriate visual representations need to be constructed as a post processing step. Projection-based approaches are common to this end [8, 5]. The Kohonen Map (SOM) algorithm [14] is a well-known approach suited for analysis of large volumes of high-dimensional data. The algorithm basically combines clustering and projection, and it is very amenable to visual analysis of high-dimensional data [21]. Its effectiveness has been demon-

*e-mail: tobias.schreck@gris.informatik.tu-darmstadt.de
†e-mail: juergen.bernard@gris.informatik.tu-darmstadt.de
‡e-mail: tatiana.tekusova@gris.informatik.tu-darmstadt.de
§e-mail: joern.kohlhammer@igd.fraunhofer.de

strated by its application on many different data types [9, 16, 4, 3]. The SOM may also be used in combination with other visual data analysis approaches. In [6], it has been integrated with several complementary visualizations, allowing the analysis of data showing high-dimensional as well as spatio-temporal characteristics.

Trajectory data lately has attracted much research interest. Due to advances in sensor and other techniques, increasingly large amounts of trajectory data arise, and consequently, techniques for their analysis are being developed. Trajectory data may be observed in real-world coordinates on various scales [1, 10]. Also, trajectories may be regarded in more abstract spaces, e.g., 2D diagram space [18]. Trajectory mining research considers analysis and description of important properties in trajectory data. Of primary concern are methods to define appropriate similarity functions to query, compare, and cluster trajectories [2, 17], and support the detection of interesting patterns [20].

3 SOM-Based Clustering of Trajectory Data

In this section, we discuss the clustering of trajectory data using Self-Organizing Maps. We briefly recall the basic mechanism of the unsupervised SOM algorithm in Section 3.1, followed by a sketch of its application to trajectory data in Section 3.2. In Section 3.3, we then motivate the need for integrating the user in the clustering procedure using visual-interactive facilities.

3.1 Self-Organizing Map Algorithm

The SOM algorithm is a neural network type learning algorithm. It iteratively trains a network of prototype vectors to represent a set of input data vectors. The network is usually given in the form of a 2-dimensional regular grid. During training, the algorithm iterates over the input data vectors; finds the best matching prototype vector; and adjusts the best matching prototype and a number of its network neighbors toward the input vector. In the course of the process, the size of the considered neighborhood and the strength of the adjustment process are reduced.

In practice, two key effects are achieved by this process. Firstly, a set of prototype vectors (or clusters) is obtained representing the input data. And secondly, a low-dimensional arrangement (sorting) of the prototypes is obtained, given by the grid structure. The main parameterization required by the algorithm includes the initialization of prototype vectors and the specification of learning parameters. The latter include the duration of the training process, the definition of the neighborhood kernel, and the degree of vector adjustment (the learning rate). While a number of rules of thumb exist for the parameter setting, finding good settings for a given data set usually requires experimentation and evaluation by the user.

3.2 Simple Trajectory Data Model for SOM Analysis

Application of the SOM algorithm to trajectory data requires a suitable vector representation of the trajectory data items. The vector representation should capture relevant trajectory characteristics and allow meaningful interpretation of vector distances as a measure for dissimilarity of the corresponding trajectories. Generally speaking, a trajectory feature selection problem has to be solved before the SOM algorithm can be applied. Many different trajectory features are candidates for a vector representation. For instance, features such as position, orientation and direction, curvature, and changes thereof may be considered. Also, sampling and normalization aspects are usually an integral part of the feature selection process.

Based on previous work [18], we consider a simple trajectory vector representation constructed from normalized trajectory sample points. To obtain the vector representation, we first normalize each trajectory by scaling it into the unit square $[0,1]^2$, and then sample n uniformly spaced (x,y) coordinates spanning the trajectory from its start point to its end point. The concatenation of the sample coordinates in their sequence along the trajectory yields the vector representation of length $2n$. By definition this representation ignores features, which might be important in certain applications. For instance, it ignores the trajectories' absolute positions and scale in space, and, depending on the number of samples, may loose trajectory details or introduce sampling artifacts. The key advantage of this representation in context of this work is that it has a direct geometric interpretation and that it can serve as the basis for visualization of and interaction with cluster prototype vectors produced by the SOM algorithm. Therefore, it is an integral component of the framework developed in Section 4. Besides, this vector representation is simple to obtain and allows a straightforward interpretation of vector distances.

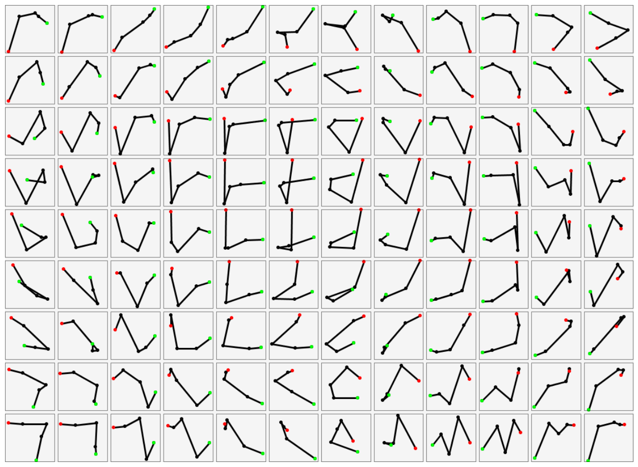
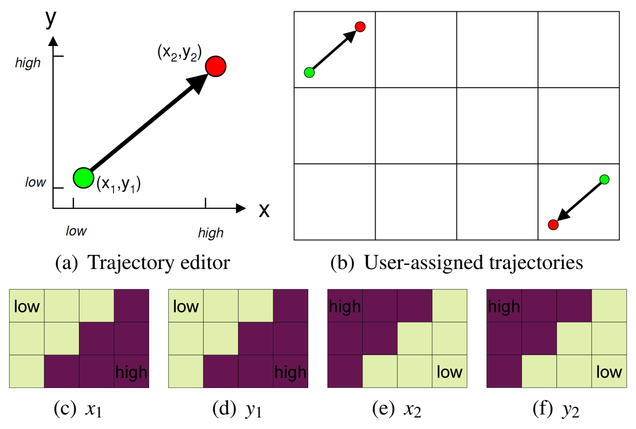


Figure 1: Self-Organizing Map of trajectory data, trained in unsupervised mode. Start and end points of trajectories are indicated by green and red dots, respectively.

3.3 Requirement Analysis

As an example following [18], we consider a data set from the financial data analysis domain (cf. also Section 5.1). The data set consists of time-dependent observations of *risk* and *return* measurements of financial assets. Specifically, we consider consecutive observations in this 2-dimensional space as sample points describing trajectories in an abstract (diagram) space. By taking daily samples and observing whole trading weeks (Monday through Friday), we arrive at 5 sample points and 10-dimensional trajectory vector representations, describing the movement of asset characteristics over time in risk$\times$return diagram space. Figure 1 shows the reference vectors of a 12×9 Self-Organizing Map trained from 5.500 trajectories. Note that this SOM was obtained by standard unsupervised training.

Generally, the result of the Self-Organizing Map algorithm depends on input data characteristics, initialization of the map reference vectors, and the set learning parameters. For effective SOM-based visual trajectory analysis, it is important that the overall cluster map is (a) meaningfully interpretable in terms of the location of reference trajectories, and (b) is stable with respect to data updates. It is desirable that the position of the reference trajectories also corresponds to specific features and transitions of the underlying trajectories. Thereby, the spatial memory of the human analyst can be fully utilized, and meaningful interpretation can be supported even for changing data sets. Also, the presentation of the results is made easier if the layouts meet the common expectations of the target audience. For example, it might be desirable that the left-hand side of the SOM holds low values of the start points, while the right-hand side holds high end values (both in terms of (x,y) coordinates of the trajectory control points). On the other hand, it could be desirable



(a) Trajectory editor (b) User-assigned trajectories

(c) x_1 (d) y_1 (e) x_2 (f) y_2

Figure 2: Supervised initialization of the SOM prototype grid using the *trajectory editor* concept. (a) An example trajectory consisting of two control points (x_1, y_1) (start point; marked green) and (x_2, y_2) (end point; marked red). (b) Two example trajectories specified on a 4×3 SOM grid. (c-f) Interpolated component planes for the x_1, y_1, x_2, and y_2 components. Bright (dark) colors indicate *low* (*high*) component values.

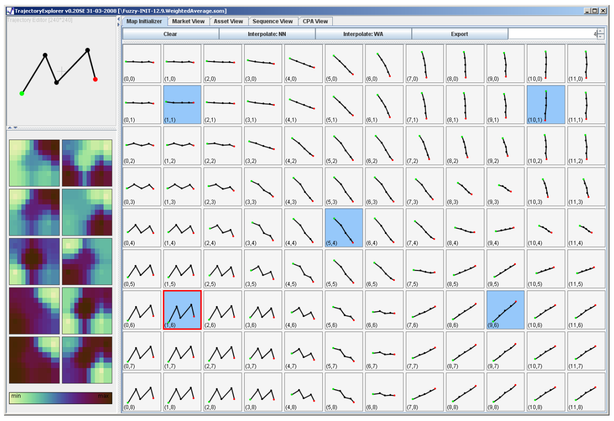


Figure 3: Editor-based initialization of a 12×9 SOM trajectory grid, using 5 user-defined example trajectories (marked blue) in conjunction with weighted average interpolation. Component distributions (x_1, y_1) to (x_5, y_5) are shown in the left panel.

that the four corners of the SOM contain reference trajectories resembling trajectories in diagonal direction. Standard SOM training usually cannot guarantee this, as it performs the learning process strictly unsupervised, and often the SOM algorithm is applied in a "black box" manner. What is required from the user perspective are efficient means of guiding the otherwise fully automatic learning process toward the desired trajectory cluster layout.

4 Trajectory Cluster Map Learning Framework

We propose a comprehensive framework for supervised-interactive SOM-based clustering of trajectory data. It consists of three main visual-interactive extensions to the otherwise fully automatic SOM learning algorithm. The framework was designed to be systematic with respect to the SOM clustering algorithm, and to incorporate visual-interactive monitoring and control facilities considered useful in guiding the clustering process.

We point out that we do not expect every single control option discussed in this section to be required in every data analysis scenario. Rather, depending on the application, an appropriate *combination of controls* from the framework is best suited to support achieving a given analysis goal.

4.1 Map Initialization Based on Trajectory Editor

Before the SOM training process can start, the grid of cluster prototypes needs to be initialized. The initialization guides the training process, and often influences the overall layout of the emerging cluster map. In the standard approach, two initialization methods are common: Random initialization, and initialization based on a Principal Component Analysis of the input data set [14]. Both methods are unsupervised in nature.

We propose a more user-oriented approach to control the initialization process. We base the approach on the fact that our trajectory data representation has a straightforward geometric interpretation: The vectors directly encode the trajectory geometry (the sequence of trajectory control points), and can therefore be readily visualized and manipulated interactively. To do so, we provide an interactive *trajectory editor* that lets the user draw example trajectories into chosen SOM grid positions. Reference trajectories may be input at distinct map locations, thereby specifying a model for the overall SOM cluster layout desired. Starting from a user-provided set of example trajectories, we initialize the full grid of SOM trajectory prototypes as follows:

- For the grid nodes for which the user has provided example trajectories, we set the initial value of the SOM prototype vector equal to the vector representation of the drawn trajectory (simply a sequence of (x,y) coordinates).
- For the unassigned grid nodes, we interpolate between the assigned example vectors.

Figure 2 illustrates the trajectory editor concept. Figure 2 (a) shows a simple trajectory consisting of two control points: one (green) start and one (red) end point. Figure 2 (b) illustrates a 4×3 SOM grid, into which two example trajectories have been drawn by the user. Interpolation of the unassigned nodes takes place on a component-by-component basis, determined by the assigned values and an appropriate interpolation function. Figures 2 (c) to (f) illustrate the resulting distribution of components over the SOM grid. Consider e.g., Figure 2 (c) showing the distribution of the x_1 component over the SOM grid. The top left cell corresponds to *low* value, and the bottom-right cell corresponds to *high* value of this component. This is in accordance with the fact that the x_1 coordinate (the x coordinate of the start point) of the two entered trajectories is low for the top left example, and high for the bottom right example. In this example, nearest neighbor interpolation was used, but other schemes such as weighted average are possible.

Figure 3 shows an example of the trajectory editor for initialization of the SOM prototype vectors. Five reference trajectories were assigned by the user, and the remaining prototype vectors were filled in by weighted average interpolation. With this concept, the user is able to efficiently initialize a SOM prototype map with a coarse template of a desired layout.

4.2 Online Visualization and Control of the Map Training

In the standard approach, the SOM clustering is produced by an unsupervised training process which ends once a fixed number of iterations has elapsed or the quantization error meets a predefined threshold [14]. In our approach, we aim to produce SOM cluster results that are both good with respect to quantization error, and at the same time reflect user- or application-desired prototype patterns and layout criteria. We therefore extend the unsupervised training process (a) by online visualization, and (b) by control functionality. Visualization of online training and optional user intervention are coupled. At any time during the training, the user is able to pause the training, update training parameters, and resume the training.

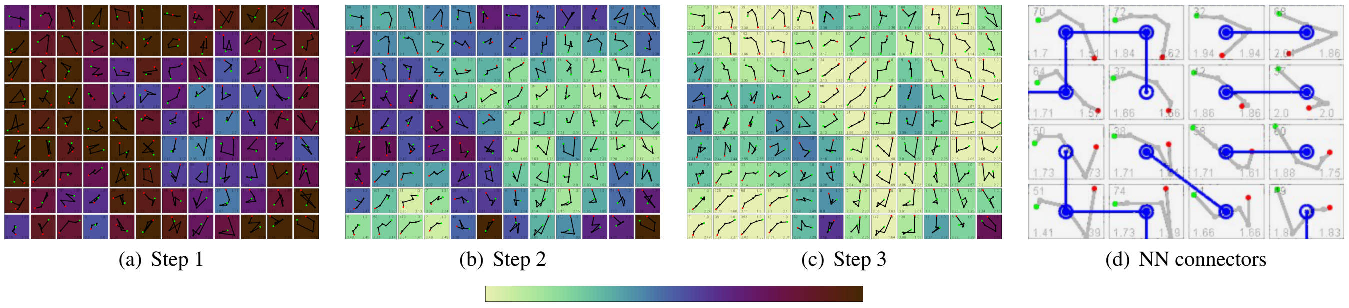

(a) Step 1 (b) Step 2 (c) Step 3 (d) NN connectors (e) Colormap

Figure 4: Visualization of the online learning process by color-coding of the quantization error (a-c; brighter is better). Nearest neighbor connectors (d) are an optional overlay indicating the smoothness of the trajectory pattern transitions over the trajectory map. The connectors show the nearest neighbor relationships between the reference trajectories (shorter is better).

4.2.1 Visualization of the Training Process

Recall that in our application, the data vectors have an immediate geometric interpretation. Therefore we are able to visualize the online training process by showing a continuously updated display of prototype trajectories. Specifically, the user can observe the effect of the provided trajectory initialization (cf. Section 4.1) on the subsequent training process. In addition to visualizing the emerging trajectory patterns within the SOM cells, we optionally superimpose certain cluster map quality metrics using color-coding and nearest neighbor connectors (cf. Figure 4):

1. Color-coding of the current quantization error of the emerging maps: For each prototype vector, we calculate the average Euclidean distance between the prototype, and the trajectory data samples it represents.

2. Color-coding of the average Euclidean distance between each SOM prototype vector and its immediate prototype vector neighbors on the grid (also known as U-Matrix color coding) [21]).

3. Nearest-neighbor connectors indicating the nearest neighbor relations between the SOM prototype vectors. This visualization reflects the smoothness of the pattern transitions over the map (smoother transiting prototype layouts show shorter connectors).

By means of these visualizations, the user can observe both the emerging organization of the pattern layout, as well as the quality of the representation of the obtained clustering. Figure 4 illustrates the online training visualization with snapshots of the quantization error during training of a 12×9 SOM of trajectories (a-c) and a zoom into a connector display (d).

4.2.2 Control of the Training Process

The framework supports a set of interaction facilities for control of the training process. At any time, the user can suspend the training process and, depending on preferences and experience, exert one or more of the following controls:

1. Adjust single prototype trajectories by directly editing them with the trajectory editor.

2. Adjust the map by editing a selection of prototypes and replace the remaining prototypes by interpolating between the selected prototypes.

3. Update the training parameters at global granularity: Adjust the number of remaining iterations, learning rate, and neighborhood kernel.

4. Manipulate learning parameters at local granularity: Set different learning rate and radius for selected grid cells.

5. Reinforce training of selected patterns.

These controls serve to guide the learning process toward user desired results, if required. Control 4 particularly allows the specification of smaller or even zero learning rates for selected patterns. This allows to explicitly enforce selected patterns on the map. Control 5 is another option we implemented to smoothly place example patterns into the map as follows. If this option is set, the system monitors the evolution of the assigned example patterns during the training process. Once the Euclidean distance between the prototype vector and the user-assigned trajectory grows too high, we repeatedly inject (update) the assigned prototype onto the respective grid position with the current training parameters. This has the effect that the otherwise freely adapted patterns do not deviate too strongly from the assigned patterns during training, and that the map neighborhood smoothly accommodates the assigned pattern.

While options 1 and 2 are basic controls, options 3 to 5 are more advanced controls of the training process, designed for users requiring fine-grained control of the training. However, we expect that it should also be possible to wrap the more advanced controls by easy-to-use high-level commands, such as setting an ‘enforce this pattern’ flag which can be set inside the trajectory editor. Thereby, the more advanced options can also be easily used by less experienced users. After updates to the training process have been manually entered, training is resumed and the user can continue to observe the effects. Usually, experimentation with different parameter settings is required for optimizing results on a given data set and analysis task. The experimentation process is supported by an *undo* operation, which rewinds the training effect of the most recent update.

Note the idea of fixing selected data vectors to given SOM grid locations during training is not new per se. For instance, the SOM-PAK implementation includes an option for doing so [15]. We point out that our interactive training controls extend beyond a simple fixing of vector assignments. Not only basically any training parameter may be edited at runtime, but also, the reference vectors may be interactively modified during training using the trajectory editor.

We also point out that, in principle the control framework allows a user to produce any prototype layout desired, possibly influencing the reliability of the obtained results. Generally, we expect that an application- or user-dependent tradeoff will have to be found between supervised and unsupervised training of the reference map. Clustering quality visualization is recommended for appropriately balancing the tradeoff between the precision of the clustering (in terms of quantization error and nearest neighbor transitioning) on the one hand, and supervised preassignment of the reference layout on the other hand.

4.3 Map Postprocessing

Usually, the final trajectory map yielded by the training will be the basis for subsequent visual analysis of the obtained clustering and the underlying data. Depending on the nature of the analysis task, it may be useful to post process the obtained trajectory map. The framework therefore supports the following trajectory map post processing interactions:

1. Merging of multiple trajectory prototypes. This allows aggregation of similar prototypes and reduces the size of the map. The new prototypes are formed by averaging the original prototypes.

2. Expansion of trajectory prototypes. This allows finer grained visual analysis of prototypes that perform too much aggregation. The expansion is achieved by training a sub map of refined prototypes based on the data represented by the original trajectory prototypes.

3. Editing, creation, and deletion of trajectory prototypes. The user can manually edit existing trajectory prototypes, or add new prototypes to the map using the trajectory editor. Also, existing prototypes can be deleted from the map.

4. Swapping of prototypes. The user is allowed to rearrange the layout of the prototypes by position swap operations.

These operations are optional, yet useful in certain situations. For instance, manual addition of possibly non-represented, sparse patterns to the map may be very helpful in situations where certain patterns are important from the analysis perspective, but underrepresented in the data set and therefore, were not trained by the SOM algorithm. Note that like manual control of the online training process, an interactive post processing operation may incur a loss of quantization precision or pattern transition smoothness, compared to a SOM trained in a completely unsupervised way. Again, referring to the quality visualizations, it is left to the discretion of the user to balance this tradeoff.

5 Application

We apply our supervised Self-Organizing Map framework on a data set of trajectories. In Section 5.1, we describe how an unsupervised reference SOM clustering was obtained. In Section 5.2, we then apply our framework to produce several different target layouts, demonstrating the functionality of the framework for generating supervised clusterings.

5.1 Data Set and Unsupervised Clustering

We consider the same data set as in [18] (cf. also Section 3.2). An unsupervised reference SOM was trained from this data set, consisting of a rectangular grid of 12×9 trajectory prototypes. The training was done as follows. We first iterated 100 times over the data set, initially setting the learning rate to 5% and the learning radius to 15 using a bubble neighborhood kernel. We then refined the map by a second run, iterating 200 times over the data set, initially setting the learning rate to 2%, and the neighborhood radius to 5. We considered both random and linear initializations of the prototype vectors, obtaining both times approximately the same end result, which is shown in Figure 1.

5.2 Supervised Clustering Experiments

We next present a series of experiments applying our framework to produce user-guided trajectory maps. The series addresses training runs relying on re-usage of unsupervised prototype trajectories, as well as training user-defined abstract trajectory patterns.

5.2.1 Adaptation of Unsupervised Trajectory Map

In the first experiment, we show how the framework can be used to adapt a given trajectory map to reflect the users' global layout preferences. Assume that the user has inspected the fully unsupervised map shown in Figure 1. While the user agrees with the obtained cluster prototypes, another global map layout is desired. The user proceeds to initialize a new map by a number of example prototypes taken from the unsupervised map. Figure 5 (a) shows the initialization: Four example trajectories were selected and assigned to the corner regions of an initial map; the unassigned prototypes were filled in using weighted average interpolation. Then, training using the Self-Organizing Map algorithm takes place. To reinforce the assigned example trajectories, control 5 described in Section 4.2.2 is applied to the preassigned reference trajectories. Figures 5 (b-f) show how the map converges toward a stable layout. The map layout basically represents the patterns contained also in the original unsupervised map, but this time, also the user-intended global cluster map layout is obtained.

5.2.2 Abstract Reference Map

In this experiment, we assume that the user is interested in a couple of rather different, dissimilar trajectory patterns. The patterns are assumed to carry an application-specific important meaning, and therefore need to be reflected in the map. The analyst starts the training by assigning these patterns. Figure 6 (a) shows the initialization of a cluster map based on six abstract user-defined patterns, along with nearest neighbor interpolation. A short training interval consisting of a small number of iterations, in conjunction with reinforcement of example patterns, yields the smoothly transitioning cluster maps shown in Figures 6 (b) and (c). The clusters adapt to reflect the data distribution, at the same time, keeping up the types of patterns preassigned, as well as their positions. Figures 6 (d-f) visualize the emerging smooth transitions between the trajectory prototypes. Overlaid by color-coding are the normalized average distances between the prototype vectors (the second SOM metric in Section 4.2.1).

5.2.3 Circular Flow-Like Map

As a further abstract supervised target layout, we consider a circular flow-like layout. Figure 7 (a) shows an initialization given by eight control trajectories in conjunction with weighted average interpolation. Figure 8 compares training of that reference layout on the data set with and without reinforcement (cf. control 5 described in Section 4.2.2) of the assigned patterns. We observe that as expected, reinforcement of the assigned patterns (top row in Figure 8) holds them fixed on the map, and adapts neighboring patterns accordingly. Without reinforcement of assigned patterns (bottom row in Figure 8), these too are subject to adaptation by the SOM training, and evolve together with the overall map of reference trajectories.

6 Observations Made and Discussion of Limitations

The overall goal of our SOM visualization and control framework is to guide the otherwise unsupervised algorithm to produce maps of user-preferred trajectory clusterings. While we did not perform a formal user study, experience obtained from our experiments indicates that the implemented visual-interactive SOM controls support quite efficient and effective parameter setting by the user.

Usually, the more the trajectory clustering aimed at by the user differs from the result achievable by the purely unsupervised algorithm, the less aggressive the training parameters need to be set, to retain the main characteristics of the predefinition. This is in accordance with practical recommendations for SOM training, suggesting to use moderate training parameters during a *fine-tuning* phase after a preceding *global organization* phase [15] has taken place. In our system, the global organization phase is replaced by interactive map initialization using the trajectory editor, and the fine-tuning is done by application of a number of interactive SOM training iterations.

By controlling the training process, in the extreme case the user is able to achieve any clustering desired, no matter how precise (and thereby meaningful) this clustering result may be. Balancing the tradeoff between optimizing a formal clustering quality metric (e.g., quantization error) and the user-desired trajectory clustering, will ultimately be the responsibility of the user. While formally evaluating this tradeoff is considered to be difficult, we believe the SOM quality visualization options implemented, including the nearest neighbor connectors visualization such as illustrated in Figure 7 (b), support achieving a good tradeoff. More evaluation in this direction is considered interesting future work.

Regarding the supported data model, our framework is applicable to trajectory data of constant length described in a simple geometry-based vector representation. Currently not included are position- and scale-dependent geometric features, features for very long trajectories, or more abstract and non-geometric trajectory features. Some of these features are expected to be easy to incorporate by an extended vector representation. Other trajectory features are expected to be more difficult to represent by the vector model, and also more difficult to visualize and interact with. Generally, the inclusion and evaluation of a richer set of trajectory features into our framework constitutes interesting future work.

7 Conclusion

We defined a visual-interactive framework for guiding the otherwise unsupervised Self-Organizing Map algorithm by a user, customized to operate in conjunction with a simple trajectory data model. The framework enables the user to visually monitor the clustering process and control the algorithm at an arbitrary level of detail. A number of interaction facilities were proposed, an integral part of them being the trajectory editor for interactive initialization of the clustering process, and interaction facilities to manipulate the training parameters during runtime. The framework was applied to a number of trajectory clustering tasks.

The framework is regarded as one step toward better fitting this popular, yet largely unsupervised clustering algorithm toward user supervision. A number of options for future work have been identified, including extension of the simple trajectory data model currently supported. Based on a flexible set of trajectory features, also the implementation of a hierarchical SOM algorithm, using different trajectory properties to organize the data on different hierarchy levels, could be realized. To this end, appropriate interaction techniques for specification of the layouts on the different levels will have to be developed.

Acknowledgements

We thank the anonymous reviewers for provision of detailed comments that were helpful in improving this work.

References

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

[2] N. Andrienko and G. Andrienko. Designing visual analytics methods for massive collections of movement data. *Cartographica*, 42(2):117–138, 2007.

[3] B. Bustos, D. A. Keim, C. Panse, and T. Schreck. 2D maps for visual analysis and retrieval in large multi-feature 3D model databases. In *Proc. IEEE Visualization Conference (VIS)*, 2004. Poster paper.

[4] G. Deboeck and T. K. (Editors). *Visual Explorations in Finance: with Self-Organizing Maps*. Springer, 1998.

[5] I. Dhillon, D. Modha, and W. Spangler. Class visualization of high-dimensional data with applications. *Computational Statistics and Data Analysis*, 4(1):59–90, 2002.

[6] D. Guo, J. Chen, A. M. MacEachren, and K. Liao. A visualization system for space-time and multivariate patterns (VIS-STAMP). *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1461–1474, 2006.

[7] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kauffman, 2nd edition, 2006.

[8] A. Hinneburg, M. Wawryniuk, and D. A. Keim. HD-eye: Visual mining of high-dimensional data. *IEEE Computer Graphics & Applications Journal*, 19(5):22–31, September 1999.

[9] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen. WEBSOM—self-organizing maps of document collections. In *Proc. Workshop on Self-Organizing Maps (WSOM)*, pages 310–315. Helsinki University of Technology, 1997.

[10] Y. Ivanov, C. Wren, A. Sorokin, and I. Kaur. Visualizing the history of living spaces. *Transactions on Visualization and Computer Graphics*, 13(6):1153–1160, 2007.

[11] A. Jain, M. Murty, and P. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

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

[13] D. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler. *Visual Analytics: Scope and Challenges*. Springer, 2008. Lecture Notes in Computer Science (LNCS).

[14] T. Kohonen. *Self-Organizing Maps*. Springer, Berlin, 3rd edition, 2001.

[15] T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen. Som_pak: The self-organizing map program package. Technical Report A31, Helsinki University of Technology, 1996.

[16] J. Laaksonen, M. Koskela, S. Laakso, and E. Oja. PicSOM—content-based image retrieval with self-organizing maps. *Pattern Recogn. Lett.*, 21(13-14):1199–1207, 2000.

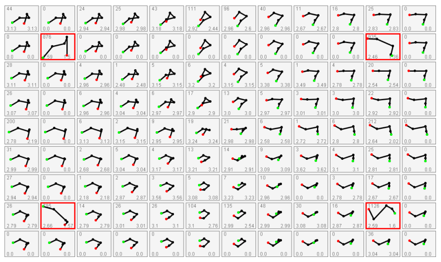
[17] N. Pelekis, I. Kopanakis, G. Marketos, I. Ntoutsi, G. Andrienko, and Y. Theodoridis. Similarity search in trajectory databases. In *Proc. Int. Symposium on Temporal Representation and Reasoning*, 2007.

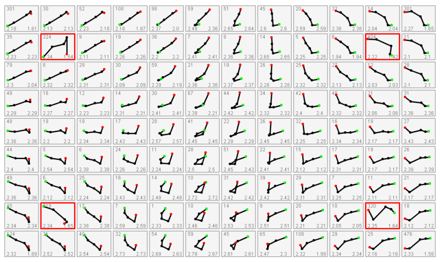
[18] T. Schreck, T. Tekušová, J. Kohlhammer, and D. Fellner. Trajectory-based visual analysis of large financial time series data. *SIGKDD Explorations*, 9(2):30–37, December 2007.

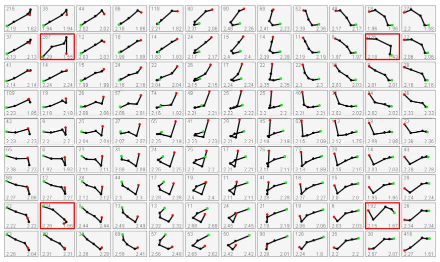
[19] J. Thomas and K. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society, 2005.

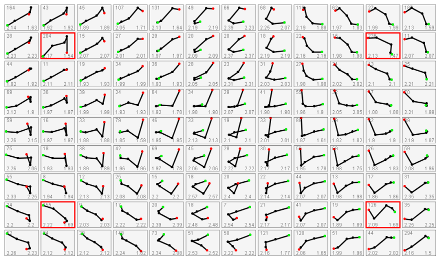
[20] A. Tietbohl, V. Bogorny, B. Kuijpers, and L. Alvares. A clustering-based approach for discovering interesting places in trajectories. In *Proc. ACM Symposium on Applied Computing, Advances in Spatial and Image-Based Information Systems Track*, 2008.

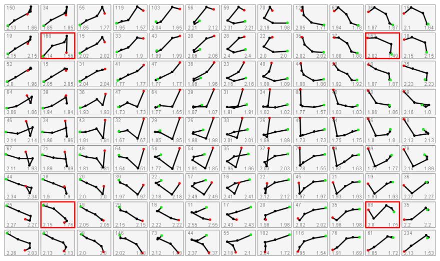
[21] J. Vesanto. SOM-based data visualization methods. *Intelligent Data Analysis*, 3(2):111–126, 1999.


(a) Initialization


(b) 2 iterations


(c) 4 iterations


(d) 6 iterations


(e) 8 iterations

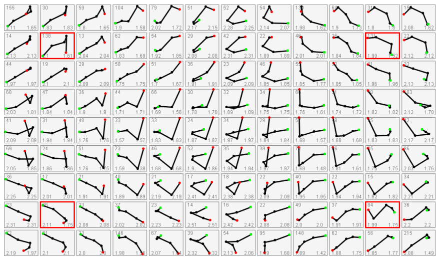
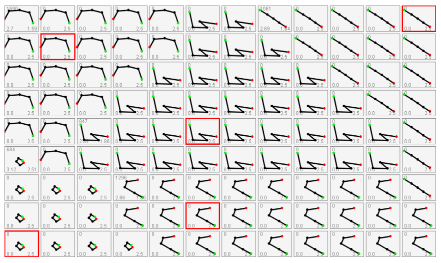
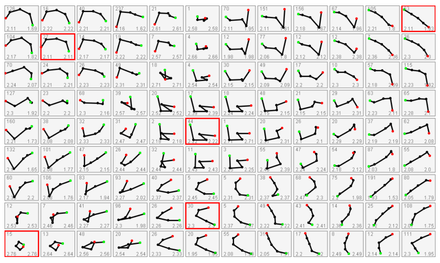
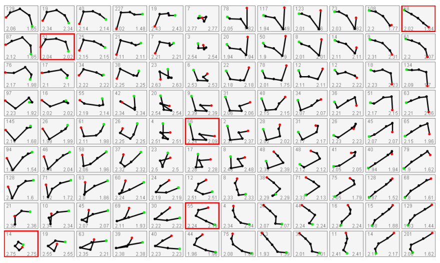
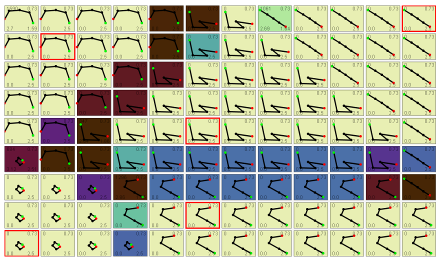

(f) 10 iterations

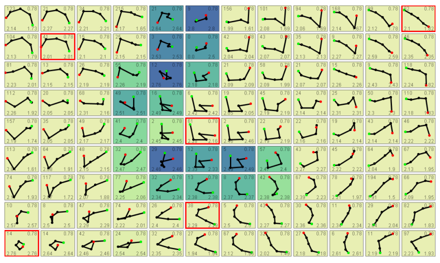
Figure 5: Generation of a trajectory map based on trajectory patterns re-used from a preceding unsupervised SOM training run (the re-used trajectory prototypes are marked in red). The layout converges against the desired global layout. In each iteration, each data vector from the data set was used once to update the cluster map.


(a) Initialization


(b) 1 iteration


(c) 2 iterations


(d) Initialization


(e) 1 iteration

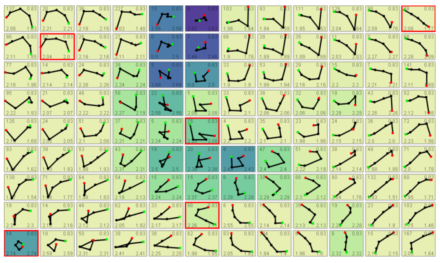
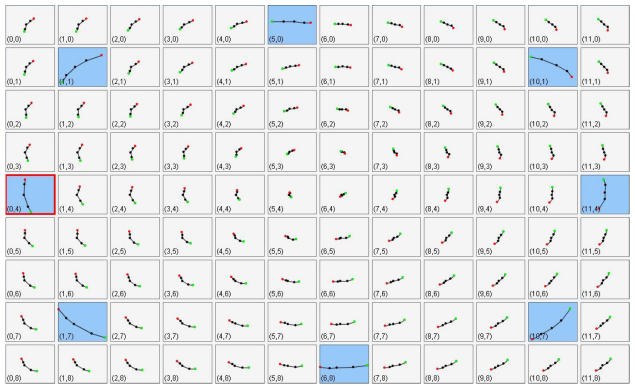

(f) 2 iterations

Figure 6: Trajectory map trained from six rather abstract, supervised trajectory patterns. The top row shows the prototype vectors only. The bottom row also includes color-coding of the vector space distance between neighboring prototype vectors (cf. the second SOM metric from the list in Section 4.2.1). Note that during training, the sharp differences between the initially assigned reference patterns are reduced in the course of the training process.


(a)

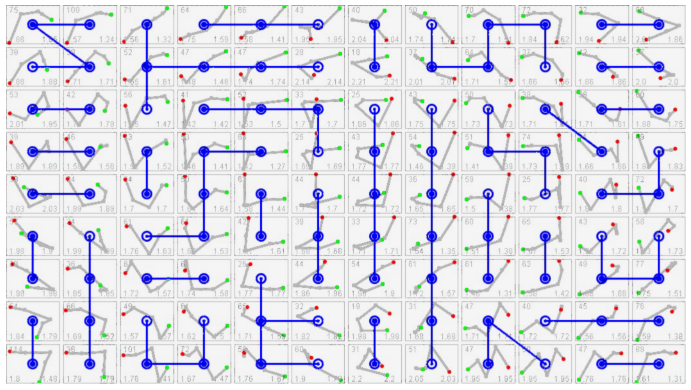
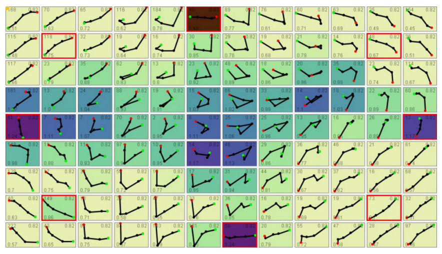
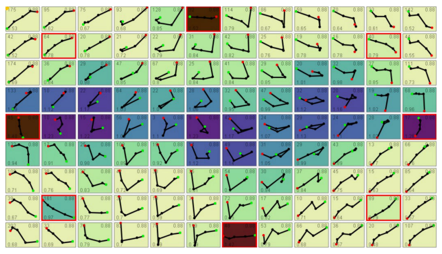
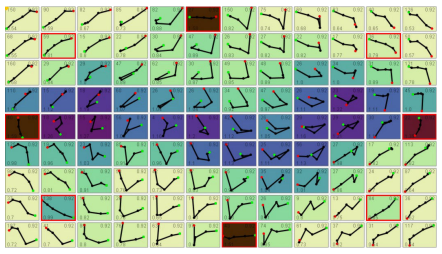
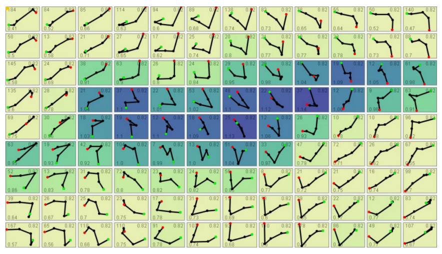

(b)

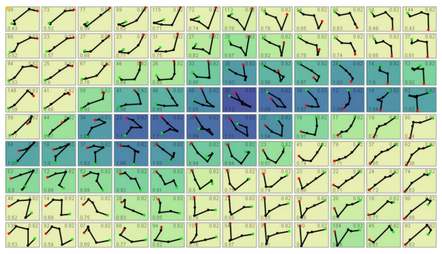
Figure 7: (a) Initialization of a circular-flow cluster map. (b) Visual inspection of the quality of a trajectory map is optionally supported by the nearest neighbor connector visualization.


(a) 1 interation


(b) 2 interations


(c) 3 interations


(d) 1 interation


(e) 2 interations

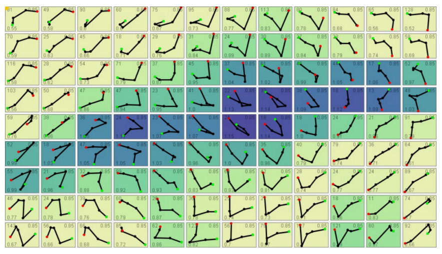

(f) 3 interations

Figure 8: Training based on the circular supervised reference layout from Figure 7 (a), using reinforced reference patterns (top row) and free-floating patterns. Like the bottom row of images in Figure 6, the color-coding indicates the average distance between SOM prototype vectors. The visualization indicates that several different trajectory regions evolve. The reinforced map shows larger differences between trajectory regions; specifically, the reinforced patterns produce larger differences to their neighborhood trajectory patterns.