Visual Interactive Exploration of Spatio-Temporal Patterns

Radoslaw Rudnicki, Monika Sester, Volker Paelke

Abstract

In many applications it is of high interest to analyze spatio-temporal (ST) datasets for recurring patterns. In our approach we examine spatio-temporal datasets for movement patterns using efficient query techniques. In an iterative process combining search and interaction an expert can review search results and either draw direct conclusions and annotate the pattern or refine the search pattern. In this way a hierarchical visualization / generalization and interactive analysis of large ST datasets becomes viable. The key feature of the approach is that patterns are not specified in advance (as meaningful semantic patterns) but established from the data set.

1. Introduction

As more and better sensors become available to capture movement data over time the search for spatio-temporal patterns in large data sets becomes an increasingly relevant task in many applications. While some established patterns for moving objects exist and are widely agreed on (e.g. the "Flock" pattern) in many applications the patterns of interest are not known in advance or not well defined. To address this problem where an a-priori definition of patterns is not feasible or useful, we propose an experimental approach in which a human expert marks patterns of potential interest in a visualization of "temporal aggregates" of the raw data. These temporal aggregates can be viewed as proto-patterns - the human expert uses his knowledge of the problem domain to identify proto-patterns of potential interest. Similar "patterns" can then be identified automatically by searching the dataset for similar occurrences. The system is designed for interactive use in an iterative process. After a search the expert can review the results and either draw direct conclusions and annotate the pattern (if the results are matching the requirements) or refine the search pattern (if the results are not yet sufficient).

For the analysis of patterns it is first of all necessary to define the term "pattern". While there are some application domains in which relevant patterns are known and can be formally defined in advance (and thus searched for automatically) this is not the case for all applications. For our purpose we assume that patterns of interest are not known at the beginning of the process. The central characteristic of a pattern in our approach is that the same spatio-temporal formation of data elements occurs multiple times. However, not every coherent spatio-temporal data element is a pattern. Therefore we name potentially interesting patterns "proto-patterns", , but the importance of these patterns is a priori unknown. Finding patterns of straight movement on a linear street might not be as interesting as a criss cross trajectory in the same area. On the other side in another application analyzing crowd behavior in a stadium it could be more interesting to find the main stream of trajectories which leave a stadium after a match than some outliers.

Thus patterns in our sense are elementary parts and/or aggregates of trajectories in a grid approach. Patterns occur, when interesing constellations among these trajectories occur, where interesting in our case is defined by an information theoretic measure, namely the frequency of occurrence. The approach is a follows. For each pattern a descriptor is calculated, which is used as characteristics to determine similarity. This approach therefore presumes appropriate feature descriptors and given measures to describe similarity. In our case we use Fourier Descriptors to describe the shape of the patterns.

The paper is organized as follows: after a review of related work, we start with the description of the raw data and initial operations on it. Then we discuss how we handle large datasets and explain the problem of choosing the correct parameterization. After that we introduce two approaches to compare patterns, which are based on Fourier Transform. In Section 4 we describe our implementation in more detail. A discussion and an outlook on future work conclude the paper.

2. Related work

The interpretation of patterns is being investigated in different research directions. There are approaches to simplify and generalize patterns using a method adapted from graphic generalization, e.g. Douglas Peucker Algorithm (Yiu, 2008, [1]). Use of databases for spatio-temporal processing was recommended for example in Wolfson et al. , (1999) [2]. P. Laube et al., 2005 worked on discovering defined patterns, e.g. the Flock and Encounter-Pattern in [3]. Lin & Su, 2006, used space discretisation for similarity search of moving objects [4].

Dodge, Weibel & Lautenschütz (2008) [5] proposed a taxonomy to classify and categorize movement patterns. Johnson & Hogg (1995) [6] describe a method to use vector quantisation to learn tpical trajectories and ST events. In the domain of information retrieval there are approaches to index large collections of data using highly unique and invariant features. A good example are SIFT features to describe scale invariant features in digital images. In order to determine the similarity of geometric features, appropriate measures have to be devised. Popular methods are geometric moments. This is proposed by Heinzle et al. (2006) [7], where it is used to define the similarity of a shape with a circle in order to determine circular roads in vector data. In order to determine the ocorrect parameters Data Mining tools there used (Witten & Frank, 2000, [8]). In a similar way, Hild (2001), [9] used affine invariant moments and Fourier Descriptors (Lee et al. 2004 [10], B. Srinivasa Reddy and B. N. Chatterji, 1996 [11]) to describe geometric features that are used as ground control features for automatic image registration. Andrienko & Andrienko (2008) [12] use aggregation of ST patterns mainly to

enhance their visual inspection, especially when it comes to handle large data sets.

3. Proposed Method for describing and indexing ST-patterns

The base information in our application are trajectories. Figure 1 gives an example of trail of playing children.

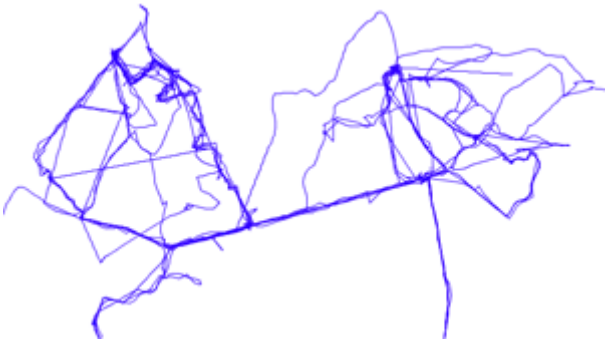

Figure 1. ***Example of trajectories***

We are interested in patterns that arise at intersections of trajectories at highest level of detail, as well as in more aggregated generalizations of the data. Trajectories consist of coordinates with timestamps. The coordinates can come from GPS sensors, computer vision and other systems. Timestamps often appear in irregular intervals and can be desynchronised between different observed objects. Therefore the first step is to resample and align the data. The sampling rate determines the level of detail of the trajectory. Of course there can only be an approximation of the data between real measured points. The goal is to resample the trajectories without changing their characteristics.

For our use case, linear interpolation is a sufficient approximation, because minor changes in the coordinates of a point (at the scale of a few centimeters) do not have a significant effect on the resulting patterns and are typically below the threshold of the accuracy of the sensors used in the acquisition of the trajectories. For practical purposes linear interpolation has the benefit that it is easy to handle and lends itself to fast implementation. Other interpolation techniques could be substituted should the need arise.

Figure 2 shows a typical trajectory dataset resampled at a rate of 1 point every 30 seconds. After the resampling the two datasets are synchronized with identical time intervals. Such a sysnchronized dataset simplifies the analysis of a situation at an arbitrary point in time. The human operator should adjust the resampling rate depending on the spatial and temporal extent of the patterns of interest, taking the sampling theorem into account.

In addition to synchronizing trajectory datasets the issues of high data volume and noise in the measurements must be addressed. Therefore, methods to simplify datastreams are required. A simple and efficient approach that works well for typical applications is to discretize the data not only in the temporal domain (as in the resampling step) but also also in the spatial domain.

From the original trajectory data, which only includes the positions and timestamps, other information of the observed objects like direction and speed can be derived at any time step. Sometimes it is more useful to define patterns of interest in this domain, therefore this option should be available to the user. We therefore also support the selection of such derived data sets in the following spatial discretization step. This allows to aggregate the data inside a cell and simplifies the

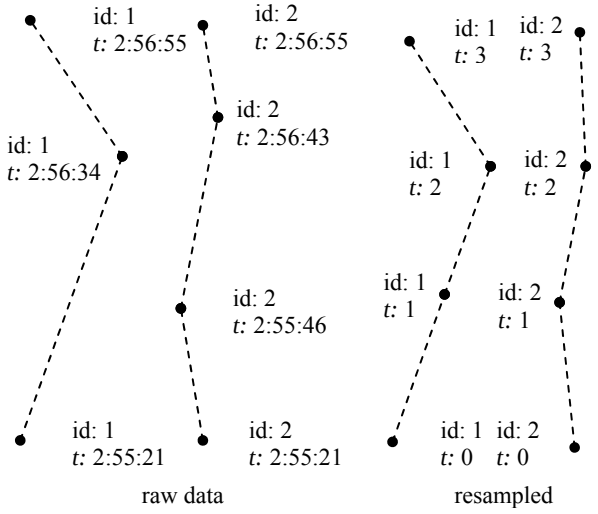

Figure 2. ***Comparison of feature similarity between original and rotated object with and without smoothing***

following operations and calculations by reducing the size of large datasets to manageable proportions. The aggregation removes details and helps to focus on the essential characteristics required to describe spatio temporal patterns. It is obvious that the selection of an appropriate cell size by the user is essential to achieve a suitable balance between data reduction and the potential loss of information that is essential for the following analysis.

To help users with the selection of a suitable parameterisation for the discretization in space and time an interface must be provided that makes it easy to adjust the settings and provide feedback on the tradeoffs being made: The smaller the cell size, the less the spatial data is aggregated and the data cells provide a closer approximates the original trajectory data. The bigger the cell size, the more trajectories are integrated, which reduces the amount of data but also brings about the possible loss of essential information so that patters that were part of the original dataset can no longer be detected.

In very small time intervals the patterns are nearly the same, because in short time it is almost allways a short step straight forward. There is chance to build complex patterns. In very long time intervals the data becomes more chaotic, and it

is hard to find any structure in it. So, the best option is somewhere in between, depending on data and intention.

For each cell all crossing trajectories are considered and suitable aggreagates of speed, direction and density information within a given time interval are calculated and stored as attributes of the cells..

An aggregtion of neighboring filled cells defines proto-patterns. The boundary of these aggregations describe the spatial contour of the pattern. The movement information (captured by the different aggreagates of speed, direction and density) characterize the inside. To identify similar patterns, it is important to define a measure of similarity. By working on the cell based aggreagtes we can focus on geometric and statistic similarity measures. We tried out two approaches for similarity analysis. The first approach just uses geometric properties of the data, and ignores the attributes speed, density and direction. After observing that it was not possible to cover all cases of similarity and dissimilarity that we assumed, we tried a second approach which takes the available statistics data into account.

In comparing such patterns it is typically useful to define similarity measures that are independent of scale, specific location and orientation. In the first approach, we experimented with the one dimensional discrete fourier transformation (1D DFT (see e.g. Lee et al. [10]). In order to achieve invariance in shape, size and positioning all objects contours are resized to a specific length and shifted to the point of origin. By resampling the outline and interpreting the *x* and *y* values as reel and imaginary part of a complex number, it is possible to determine a vector which represents this shape. After normalising the vectors the euclidian distance represents a similarity measure similarity between shapes.

This method produces results that comply with human assessment of similarity if the cell size is small enough and the speed, direction and density values can vary without having influence on the importance of patterns. The problem with the cell size is that a rotated shape can optically depart from the original one, because the impact of discretization becomes too strong. Therefore it is useful to smooth the contours to keep rotation also invariant. Figure 3 illustrates the problem with a simple shape (1.), which is rotated about 45° (2.). It looses a lot in contour similarity. 3. Appears after filling triangles in free space, where two sides are in touch with the original shape and removing triangles without neighbors. It looks more similar to 1. and the measure of 0.95 confirms this.

Another problem with this approach is, that not all shapes can be differed reliably. Because of sampling the contour, holes cannot be detected. The algorithms does not differ shapes, if their outer contour is equal. And there can be some deviations in similarity depending on the starting position for sampling the contour and the sampling rate. Of course, the sampling rate should be at least as double as high as the highest frequency in this shape (sampling theorem). But the higher the rate, the longer the algorithm takes. With a relatively low rate, the corners can not be represented correctly, which worsens the result.

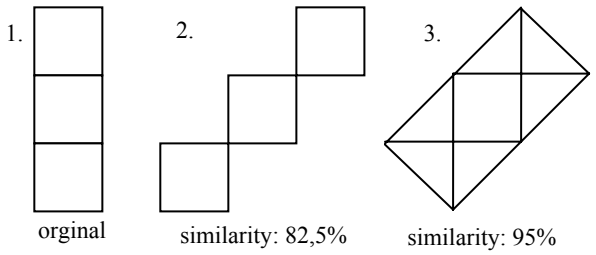

Figure 3. ***Comparison with and without smoothing***

However, this first approach does not take all requirements and options into account. Speed, direction and density have no influence on the result. There are just filled and not filled cells during a considered time interval. It is like a 2D-region of *true* and *false* values from which the contour is extracted. To improve the compliance of the requirements and to exploit the underlying data in its richness, there is the need of bringing the further available data into this approach. Therefore we extend the 2D-regions from a binary representation (a cell is intersected or not) to higher dimensional region including more information, namely density, speed and direction. This was developed in a second approach. The central idea is to use the different features as multiple channels of overlapping information, similar to color channels in a multicolor picture. A pixel is represented by a cell, including different data channels. In our case we have the speed, direction and density as “colors” of the picture instead of RGB. With this special view on the data, a 2D discrete fourier transformation (DFT) can be used. Just a 2D DFT on a picture is not invariant to rotation, but this can be solved by an additional intermediate step in an extension of the 2D DFT known as the Fourier Mellin transform. The rotation invariance can be seen in Figure 4.

The first row shows two original data sets, a horizontal and a vertical line. In the second row is their logarithmic visualization of the frequency space after DFT. Due to the different orientations, the visualization of the frequencies also is rotated. The third row shows nearly the identical image, indicating the rotation invariance. This special way to represent DFT data in polar coordinates is called “Fourier Mellin Transformation” (FMT).

This second approach includes all requirements: the similarity measure is invariant to changes in scale, position and orientation, and all characteristics of speed, direction and density are taken into account. A key improvement is that a distinction of patterns that are of similar shape in their spatial aggregates can be made between those with and without holes. If the cell size is small enough this approach produces promising results. For bigger cells, there is still the need to smooth the contours.

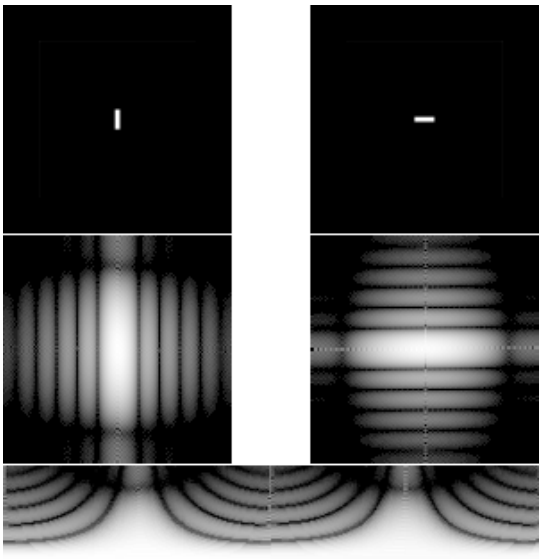

Figure 4. ***Steps of a Fourier Mellin Transformation***

To do a similarity measurement , the two dimensional data is filled up to have a $2^n \times 2^m, n, m \in \mathbb{N}$ image, which is converted to a one dimensional vector. This allows the use of a fast fourier transformation (FFT) algorithm. Thereafter the vector is reconverted to a $2^n \times 2^m$ matrix. Then the values are reorganised into polar representation.

These representations are used to calculate the differences between different proto-patterns. Therewith the closest matches of a pattern can be found.

4. Implementation

In this chapter we describe how the approach was implemented as a working system consisting of components for resampling, spatial discretization into cells, similarity measures for cell constructs and a user interface that allows to search for patterns using an interactive visualization of detected proto patterns.

The resampling of the trajectories is relatively easy. Assume that we have a starting point p_i, two positions p_n and p_{n+1} with timestamps t_n and t_{n+1} and the next synchronized timestep t_a is between these both, a new synchronised point is created by

$$p_{new} = p_i + t_a \cdot (p_{n+1} - p_n), n \geq 0, 0 \leq t_a \leq 1.$$
$$p_i = p_n$$

For an efficent implementation of resampling, it is useful to have quick access to data belonging to a certain trajectory at a requested timestamp. Because of typically large datasets a method is needed which allows to work on it without keeping it all in RAM. Reading the data from simple files allows to use the full diskspace, but is relatively slow. Therefore a MySQL database is used. It allows to use index structures to read data, saves the data efficently and may use more space than RAM allows. So, a table *trajectory* with attributes *id*, *position_x*, *position_y*, *timestamp* with the *id* as primary key and a BTree index on *timestamp, position_x and position_y* is created. A table *trajectory_sampled*, with the same attributes and indexes keeps the resampled data also efficient accessable.

To find passed through grid cells intersection points with the trajectories were calculated. In a further table *grid* with attributes *x*, *y*, *time_step*, *trajectory_id*, *speed* and *direction* the intersected cells are saved including the time step information. *x* and *y* do not represent the position itself, but the grid cell. Indexes are used on *x*, *y* and *time_step*. That accelerates the next steps in processing the data.

Knowing when which grid cell is intersected by which trajectory, enables the aggregation inside the cells. Therefore the speed and direction information of different trajectories inside a cell are averaged. To average the direction, the vector of movement during the considered time interval is transformed to polar coordinates and reduced to its angle α. But there is an exception while averaging the angles. If the difference d_1 between two angles become bigger than π the smaller angle $d_2 = 2\pi - d_1$ has to be considered to take the mean.

$$\alpha_1 > \alpha_2, \alpha_1 - \alpha_2 > \pi$$

then

$$\alpha_3 = \left(\frac{(\alpha_2 + 2\pi - \alpha_1)}{2} + \alpha_1\right) mod\ 2\pi$$

The averaged angle has to be between 0π and 2π . Therefore a modulo is used. The following example in Figure 5 should help to understand this special case. The red arrow symbolises the arithmetic mean, the green one the correct value.

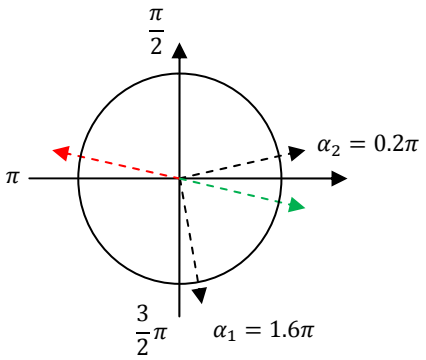

Figure 5. ***average direction***

The arithmetic mean would be 0.9π, but the right direction is represented as angle $\alpha_3 = 1.9\pi$. Speed can be averaged

easily. The result of this last step are saved in a table *aggregated_data*. As additional new information, the density of trajectories inside a cell is added. So we have the attributes *x* and *y* representing the cell, *angle*, *speed*, *density* and *time_step*.

To identify which intersected grid cells at a time step t_n belong together to form a shape, a random intersected grid cell is taken from the table *grid* and examined, with respect to its neighbours (N8-neighborhood) which are also marked as intersected at this moment or time interval, respectively. All localised new cells again examine their neighbours, recursivley, in a region growing fashing. In this way the composition of cells, that form a pattern can be found. This is repeated until all intersected cells from the database are allocated to a proto-pattern. Because of the indexes on the attributes *x, y* and *time_step* the neighbours can be checked very efficently. That accelerates the detection of marked neighbours a lot. To check, if a cell is already allocated to a pattern is also supported by the database indexes. Combined with the content from *aggregated_data* a new table *pattern* is filled. It is the first table, that uses aggregated data for inside the cells and knows, which cells belong to a proto-pattern at which time step. Following from that, the attributes are *pattern_id, x, y, speed, angle, density* and *time_step*.

The concept is implemented as a prototype written in Java. It allows to load the raw information and do the processing on the database. Additionally, it allows to have the data animated by incrementing the time step. Because of the indexes used in the database just the current viewed data is written into RAM for each step. The time interval and the grid size can be easily controlled with sliders. To visualize the different components of a cell, the density of the trajectories are visualized using the saturation of their background color. The direction of the aggregated data is shown by an arrow. The size of this arrow shows the average speed inside this cell. Thus all information is visible at the same time.
The software also allows to have a look at all proto-patterns and by clicking them to jump to the time step in which they appeared. After chosing a proto-pattern there is the option in the software to find most similar patterns.
The screenshot in Figure 6 shows the GUI of the software prototype. The main panel shows a part of the considered andby cells subdevided area. In the bottom there are controller to set up cell size, the time inverval and the speed of animation. The right side informs about proto-patterns. It lists all available proto-patterns which were found during processing and allows the user to select them for further analysis. The user can compare the selected pattern with other patterns and order them by similarity. By users decision from which pattern the similarity is too bad, a threshold is set up. The number of similar proto-patterns shows, if this data composition appears many times or is an exception.

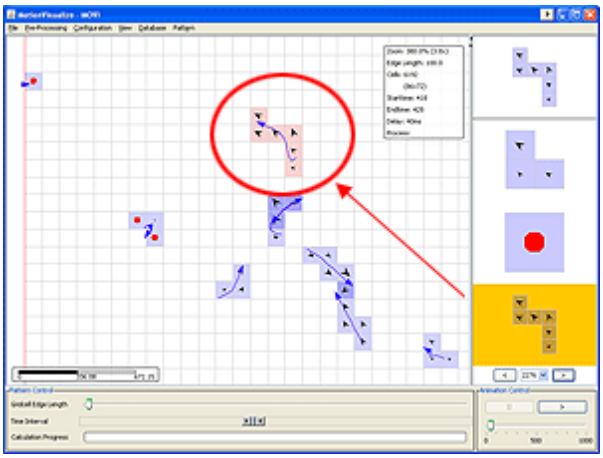

Figure 6. *Visualisation*

5. Observations and Discussion

The use of the database and especially its index structure proved to be a good backend of the implementation. Besides the efficient disk space use and quick return of requested data, more efficient use of multi-core processors is made. While one core works on processing the data, another core is appropriated for reading and writing the database. Import and export of data works faster as well. Also workload of RAM can be reduced, because it is not necessary to keep everything in it. Alternatively it is possible to create own index structures adjust on this problem. But the database worked good with our dataset.

In the two approaches, different ways of using the frequency space were described. The first approach works with the contour of a shape. The second idea, to consider the data matrix as picture is more complex and needs additional procedures to keep rotation invariant, but can differ proto-patterns more precise as Figure 7 shows. The first row of this figure shows the different behavior of the first and second approach, by analyzing shapes with and without holes. The first approach is blind for holes, because it just considers the outline. The second row demonstrates how the algorithms evaluate same shapes with different aggregated information about i.e. direction. Again the first approach does not react on this significant difference.

The first approach therefore works faster. The main idea is based on a 1D DFT. In the second approach the two dimensional data is transformed to a one dimensional vector to use a fast fourier implementation as well. The transforming from 2D to 1D and back to 2D and the mellin transformation produces additional processing costs.

One important observation when comparing patterns with Fourier Descriptors and differences in their representive

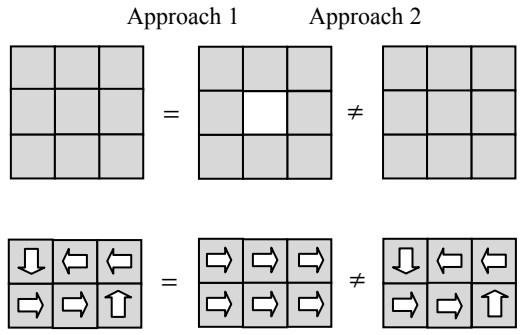

Figure 7. Advantage of Fourier Mellin Transformation compared to 1D Fourier Transformation

vectors of the coefficents is that there is no value which can be used as threshold for similarity. In some datasets a value of 87% seems to indicate high similarity, in other it does not. The range of values isn't fully exploited because, there is always any kind of similarity. In our special case the smalles common shape is a box which can be found in every pattern. Typically we had similarities between 75% and 100% in our dataset. A person is still needed to decide from which value on there is no adequate similarity given anymore.

This is why in our approach the human operator is in the loop to identify similarities of patterns. Based on this, the corresponding similarity values can be calculated and used as thresholds in subsequent processes.

6. Summary and Outlook

The basic idea of our approach is to identify "interesting" patterns in trajectories and store and index them in a database. From the frequency of occurrence of a pattern applications dependent inferences can be drawn: when looking for dominant patterns, the ones with a high frequency of occurrence are chosen; when looking for non-conform patterns, e.g. unusal movement patterns in a large crowd, then the most seldom patterns are important.

Our concept of aggregating information in a grid naturally leads to a hierarchical representation and processing of the data. In this way it is possible to identify similar patterns in different aggregation levels, e.g. the intersection of two trajectories on a high level is similar to the small scale intersection of larger collections of objects.

It remains to be seen, if the ranking of all patterns agree with human evaluation in detail. Therefore tests with some experts are necessary.

7. References

[1] H. J. M. L. Yiu, X. Zhou, C. S. Jensen, H. T. Shen (2008): Discovery of convoys in trajectory databases, in: Proceedings of the VLDB Endowment Volume 1 , Issue 1 (August 2008)

[2] Ouri Wolfson, Prasad Sistla, Bo Xu, Jutai Zhou, Sam Chamberlain, (1999): "DOMINO: Databases fOr MovINg Objects tracking", ACM SIGMOD Record Volume 28, 547 – 549

[3] Laube, P., Imfeld, S., & Weibel, R. (2005). "Discovering relative motion patterns in groups of moving point objects", International Journal of Geographical Information Science, 19(6), 639–668.

[4] Bin Lin, Jianwen Su, (2007): "One way Distance: For Shape based Similarity Search of Moving Object Trajectories", GeoInformatica, Springer Netherlands, 1384-6175 (Print) 1573-7624 (Online), 117-142

[5] Dodge, S., R. Weibel & A.-K. Lautenschütz (2008): "Towards a taxonomy of movement patterns. Information Visualization", 7, 240-252.

[6] Johnson, N. & D. Hogg (1995), "Learning the Distribution of Object Trajectories for Event Recognition", Proc. British Machine Vision Conf., D. Pycock, ed., pp. 583-592, Sept. 1995.

[7] F. Heinzle, K.-H. Anders and M. Sester, (2006): "Pattern Recognition in Road Networks on the Example of Circular Road Detection", Geographic Information Science , no. 4197 , p. 253-267 , Münster

[8] Witten I.H., Frank E., (2000): "Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations." Morgan Kaufmann Publishers.

[9] Hild, H.: Automatic Image-To-Map-Registration of Remote Sensing Data, in: D. Fritsch & R. Spiller, eds, 'Photogrammetric Week '01', Herbert Wichmann Verlag, Heidelberg, pp. 13–23. 2001

[10] D.J. Lee, S.Antani and L. R. Long, (2004): "Similarity Measurement Using Polygon Curve Representation and Fourier Descriptors for Shape-based Vertebral Image Retrieval"

[11] B. Srinivasa Reddy and B. N. Chatterji (1996): "An FFT-Based Technique for Translation, Rotation, and Scale-Invariant Image Registration, IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 5, No. 8

[12] Andrienko, G., & Andrienko, N.: "Spatio-temporal aggregation for visual analysis of movements", In Proceedings of IEEE Symposium on Visual Analytics Science and Technology (VAST 2008), IEEE Computer Society Press, 2008, pp.51-58.