Journal of Computational Information Systems 7: 11 (2011) 4137-4144
Available at http://www.Jofcis.com

Trajectory Outlier Detection Algorithm Based on Structural Features

Guan YUAN†, Shixiong XIA, Lei ZHANG, Yong ZHOU, Cheng JI

School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China

Abstract

As the development of location based service, such as GPS and RFID, a great volume of trajectory data can be collected. As a result, mining knowledge from these data has become an attractive topic. The trajectory outlier detection algorithm, proposed in this paper, can be used to detect trajectory outliers by comparing the structure between each trajectory segment pairs. Firstly, a new concept of trajectory structure, taking full advantage of trajectory implicit features, is introduced to analyze trajectory similarity. Secondly, an outlier detection algorithm is put forward to match the two segments by computing their structural similarity. Thirdly, segment and trajectory outliers are detected by analyzing their both external and internal features. Experiments on real dataset demonstrate the efficiency, effectiveness of the proposed algorithm, feature sensitivity can be adjusted by the parameters flexibly, and the detected outliers are more practical significant.

Keywords: Trajectory Outlier Detection; Corner; Structural Similarity (SSIM); Trajectory Match

1. Introduction

In recent years, with the rapid growth of GPS, RFID and wireless communication technologies, more and more trajectories can be collected and stored in database, and need an urgent analysis. An important branch of location based service is trajectory outlier detection[1]. A definition of trajectory outlier is that one or more trajectories in trajectory set are different from others in some features, and all these trajectories are called outliers [1, 9]. This broader definition brought a lot of algorithms, such as: univariate, parametric, and statistical method. According to their features, these algorithms are: Partition-based algorithm[2, 4], Distance-based algorithm [10, 11], Shape and time dependent algorithm[3]. The mismatch degrees of these algorithms are represented by the weighted sums of some differences in their independent attributes. However, those algorithms can't be directly applied to trajectory analysis[4]. Kreveld[3] is the first who introduced time dependent relation to shape based trajectory analysis. Knorr[10] represented trajectory by some key attributes, such as starting points, direction, speed, then computed the similarity according to the weighted distance. All these methods regard trajectory as a whole and ignore the local features, making it impossible to match the longer and more complicated trajectories. Lee[2] divided trajectories into *t*-partitions, and each *t*-partition is just the direct segment of two sampling points. Lee's method better solved the problems in comparison between complicated trajectories, however, the trajectory distance merely depends on its shape and although *t*-partitions are the approximate description of trajectory, and many local features losing still can't be prevented. Liu demonstrated the two shortages above in [4].

† Corresponding author.
Email addresses: yuanguan@cumt.edu.cn (Guan YUAN).

November, 2011

Borrowing the idea from Lee [2], Liu abstracted all trajectory segments of *k*-length as basic units, and proposed an algorithm called minimum Hausdorff distance under translation to compute local similarity, and then identified the global similarity of trajectories. However, the selecting of *k*-length segments lacks for theoretical support.

Some current trajectory mining algorithms treated trajectory as static data sequence[1-7, 9], and did not take valuable features into consideration, for example, the starting time of trajectory may effect on trajectory shape, location, and other features. In this paper, more trajectory features will be considered, at the same time, the importance of features can be adjusted by their weights. Therefore the match degree can be measured by computing their structure features instead. By this way, structural similarity not only can better reflect the differences of both internal and external features, but also can strengthen analysis effect.

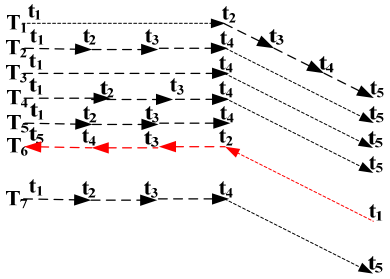


Fig. 1 Examples of Hurricane Trajectories.

Example 1: There are 7 incompletely identical hurricane trajectories in Figure 1. Traditional algorithms may think the 7 trajectories have the same moving pattern. In terms of shape, all these trajectories are similar besides T_5. From the visual point of view, T_1, T_2 and T_6 are apparently dissimilar trajectories, but some shape-dependent algorithms would not think so, this is due to the overlook of the lost sample points. For T_3, distance between t_1 and t_4 become too long due to the lost of t_2 and t_3. However, T_2 and T_3 are dissimilar trajectories in density-based algorithms. In some location insensitive algorithms, T_7 and T_2 may be considered similar although they are far away from each other. T_4 and T_5 have similar local features, and they may be considered similar in some cases.

In allusion to the problems above, a Trajectory Outlier Detection algorithm based on Structural Similarity (TOD-SS) is proposed in this paper. Firstly, TOD-SS computes all the corners at each sampling point and partitions trajectory into segments according to the corner threshold. Secondly, TOD-SS borrows the idea of structural similarity from [12], and abstract trajectory features. Thirdly, segment outlier can be detected on the basis of SSIM matrix, and trajectory outliers can be detected based on the proportion of its segment outliers. Finally, experiments on real dataset demonstrates the efficiency, effectiveness, and flexibility of TOD-SS, and the detected outliers are more practical significant.

2. Relate Concepts

Formal descriptions of trajectories and related attributes are given to express TOD-SS. The definitions of trajectory given in [2] are also available in this paper. *TD* denotes trajectory set, $TD=\{TR_1, TR_2, \ldots, TR_n\}$, and TR_i is the *i*-th trajectory. Each TR_i ($1\le i\le n$) in *TD* may include one or more outlying segments, and the outlier is denoted by $O=\{O_1, O_2, \ldots, O_n\}$. A trajectory is a chronological sequence consisted of

multidimensional locations denoted by $TR_i=\{P_1, P_2, \ldots, P_m\}(1\le i\le n)$. P_j $(1\le j\le m)$, is a point in the trajectory, denoted by $<Location_j, T_j>$, which means that the location of the object is $Location_j$ at time of T_j, where $Location_j$ is a multidimensional point. The sub-trajectory $P_{c1}, P_{c2}, \ldots, P_{ci}$ $(1\le c1<c2<\ldots<ci\le m)$ represents a trajectory segment, which is denoted by *TS* (Trajectory Segments), $TS_i=\{L_{i1}, L_{i2}, \ldots, L_{inum}\}$.

2.1. The Partition of Trajectory

The measurement of structural similarity is one of the key points in outlier detection. However, the accuracy of SSIM is determined by the partition of trajectory, which aims at finding feature points where behavior varies rapidly. Trajectory partitioned at these points, can be helpful to maintain relative smooth structural features. The trajectory corner is the turn angle of two adjacent sections, reflects the moving tendency of a trajectory, as shown in Figure 2.

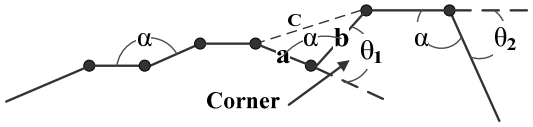


Fig. 2 Trajectory Corner

The included angle of two segments is denoted by α and the corners are signed as θ_1 and θ_2. In order to simplify the calculation, corner inside θ_1 is marked as a positive, and corner outside θ_2 is marked as a negative. As shown in Figure 2, there 3 edge *a*, *b* and *c*, represents adjacent sides, and the opposite side of the included angle α respectively. The calculation of α is given in Eq. (1):

$$\alpha = \arccos\left((a^2 + b^2 - c^2)/2ab\right) \tag{1}$$

According to Eq. (1), the calculation formula of corner θ is defined in Eq. (2):

$$\theta = \begin{cases} \pi - \alpha, if\ (\vec{a} \times \vec{b} \ge 0) \\ \alpha - \pi, if\ (\vec{a} \times \vec{b} < 0) \end{cases} \tag{2}$$

The trajectory partition algorithm consists of four parts: 1) scan point sequence in trajectory; 2) compute corner through Eq.(1) and (2); 3) partition trajectory at points where $|\theta|>\omega$ according to corner threshold ω; 4) merge continuous short segments. The time complexity of trajectory partition algorithm is $O(n)$, where n is the count of sampling points.

2.2. Trajectory Structure

The essence of outlier detection is to compare the structural similarity (*SSIM*) of each segment pair. Therefore, in this section, we give the detailed discuss of trajectory structure computing. The trajectory structure is a set of external and internal features of a trajectory, and all these features can represent a whole trajectory from relative full aspects.

Trajectory structure includes 4 features: *Trajectory Structure <Direction, Speed, Angle, Location>* represents both internal and external features in a trajectory. We also define $W=\{W_D, W_S, W_A, W_L\}$ as feature weights vector, and each item corresponds to trajectory structure vector respectively. Weights vector establishes requirements that: 1) $W_D \ge 0$, $W_S \ge 0$, $W_A \ge 0$, and $W_L \ge 0$; 2) $W_D+W_S+W_A+W_L = 1$. The sensitive

of these features can be adjusted by feature weights flexibly, for example: Location sensitive method can be implemented by set weights to ($W_D = W_S = W_A = 0, W_L = 1$), and the idea of this case is similar to density based trajectory analysis.

2.3. Computation of Structural Similarity

The measurement of *SSIM* is represented by structural distance (*SDIST*). *SDIST* consists of 4 comparisons: 1) $DirDist(L_i, L_j)$ denotes comparison of direction; 2) $SpeedDist(L_i, L_j)$ denotes comparison of speed; 3) $AngleDist(L_i, L_j)$ denotes comparison of angle; 4) $LocDist(L_i, L_j)$ denotes their location distance; where L_i and L_j are two segments, and calculation formulas are given in Eq. (3), (4). Where $N(\ldots)$ is a function of distance normalization.

$$SDIST(L_i, L_j) = (DirDist \times W_D + SpeedDist \times W_S + AngleDist \times W_A + LocDist \times W_L) \quad (3)$$

$$SSIM(L_i, L_j) = 1\text{-}N(SDIST(L_i, L_j)) \quad (4)$$

In this paper, *SSIM* of segments is computed first, and then a whole trajectory should be matched with structural comparison. Through similarity computation, the match degree of segment pairs not in the same trajectory is well represented by *SSIM*. Therefore, the larger *SSIM* is, the more similar trajectories are. SSIM of a whole trajectory is denoted by (*MSSIM*), and the formula is given by Eq. (5):

$$MSSIM(TR_i, TR_j) = \frac{1}{\min(|L_i|, |L_j|)} \sum_{i=0, j=0}^{\min(|L_i|, |L_j|)} SSIM(L_i, L_j) \quad (5)$$

1) Comparison of direction: $DirDist(L_i, L_j)$ represents the direction deviation in moving tendency of L_j compared to L_i, where φ is their included angle. The formula is given in Eq. (6):

$$DirDist(L_i, L_j) = \begin{cases} \min(\| L_i \|, \| L_j \|) \times \sin(\varphi), if\ (0 \le \varphi \le 90) \\ \min(\| L_i \|, \| L_j \|), if\ (90 \le \varphi \le 180) \end{cases} \quad (6)$$

The best case of *DirDist* is that φ is so small that it can be viewed as nearly parallel in the same direction ($DirDist \approx 0$). The worst case is that φ is so large that it can be viewed as nearly parallel in the opposite direction (*DirDist* is the length of the short one).

2) Comparison of speed: $SpeedDist(L_i, L_j)$ reflects the comprehensive divergence from three aspects: maximum, minimum and average speed. Formula is given in Eq. (7).

$$SpeedDist(L_i, L_j) = \frac{1}{3}\left(S_{\max}(L_i, L_j) + S_{avg}(L_i, L_j) + S_{\min}(L_i, L_j)\right) \quad (7)$$

$S_{max}(L_i, L_j) = |V_{max}(L_i) - V_{max}(L_j)|$, and exemplifies the absolute difference of their maximum speed. Correspondingly, S_{avg} and S_{min} are the absolute divergence of average and minimum speed.

3) Comparison of angle: $AngleDist(L_i, L_j)$ reflects the change of internal fluctuation direction in trajectory. According to Eq. (2), the value of each angle is decided by trajectory direction.

$$AngleDist(L_i, L_j) = \frac{\sum_{1,1}^{P(L_j), P(L_j)} ((|\theta_i - \theta_j|)/(|\theta_i| + |\theta_j|))}{P(L_i) + P(L_j)} \quad (8)$$

In the best case, *AngleDist* is 0 when L_i is matched with L_j at each angle, and in the worst case, *AngleDist* is 1 when L_i is in opposite direction match with L_j at each angle.

4) Comparison of location: In order to simplify the calculation, Hausdorff distance is adopted, to

measure the distance between two segments.

$$LocDist(L_i,\ L_j) = \max(h(L_i,\ L_j),\ h(L_i,\ L_j)) \quad (9)$$

The equation $h(L_i, L_j) = \max_{a \in L_i}(\min_{b \in L_j}(dist(a,b)))$ in the Eq.(9) is the direct Hausdorff distance of L_i and L_j, and $dist(a, b)$ is the Euclidean distance between sampling points.

2.4. Definitions of Trajectory Outlier

Trajectory outlier is determined by its similarity to others [2], while the decision of whether a trajectory is an outlier mainly depends on the number of its neighbors.

Definition 1. ε-Neighbor: for a given SSIM threshold ε, and segment L_i, if there exists segment L_j (i≠j), and it satisfies SSIM(L_i, L_j)≥ε, then L_j belongs to the ε-neighbor of L_i, denoted by: $L_j \in N_\varepsilon(L_i)$.

The ε-neighbor, given in definition 1, reflects the singular degree of a segment. According to ε-neighbor, the definitions of segment outlier and trajectory outlier are given as follows.

Definition 2. Segment Outlier: for a given segment L_i, if its ε-neighbor count $|N_\varepsilon(L_i)| < \sigma$, then L_i is called ε, σ related segment outlier, marked with $SO_{\varepsilon,\sigma}(L_i)$, where σ is the neighbor threshold specified by users.

In definition 2, segment outlier is determined by the two thresholds ε and σ. Combined with the definition of segment outlier, definition of trajectory outlier is given below.

Definition 3. Trajectory Outlier: according to definition 1 and 5, a trajectory TR_i is an outlier only if it meets the following two conditions: 1) The proportion of outlier segments in TR_i should be more than the given proportion threshold ξ; 2) The total similarity (MSSIM) to others should be less than threshold F. If so, TR_i is denoted by $TO_{\xi,F}(TR_i)$, and called ξ, F related trajectory outlier.

3. Trajectory Outlier Detection Algorithm

To better describe TOD-SS, notation of basic symbols is given in Table 1.

Table 1 Basic Symbols Notation.

Parameters	Description
TD, TS	Trajectory and Segment set
$\|\| TR_i \|\|$	The length of TR_i, and $(i \in N) \wedge (1 \le i \le \|TD\|)$
$\|\| L_j \|\|$	The length of L_j, and $j \in N \wedge 1 \le j \le \|TS\|$
$P(x)$	The number points of a trajectory or a segment, and $x \in \{TD\} \| x \in \{TS\}$
$V_{flag}(L)$	Some kinds speed of L, where *flag* is *s*, *e, max*, *min*, *avg*, and represents start, end, maximum, minimum, average speed respectively.
$SO_{\varepsilon,\sigma}(L)$	L is the ε, σ related segment outlier
$TO_{\xi,F}(TR)$	TR is the ξ, F related trajectory outlier
θ, ω, ε, σ	Corner, corner threshold, SSIM threshold, and neighbor threshold respectively.
ξ, F	Proportion threshold and structural similarity threshold

3.1. Description of TOD-SS

Based on the analysis above, TOD-SS can be divided into two parts: 1) similarity computing as in Figure 3; 2) outlier detection as in Figure 4. In the first part, the corner θ at each point is calculated, and trajectories

are partitioned into *TS* according to the ω firstly (line 01~02); then similarities of segment pairs are computed (line 03~06), and segment similarity matrix M_L and trajectory similarity matrix M are formed (line 07~10). According to these two matrixes, outliers can be well detected. The outlier detection part can be divided into 3 phases: 1) identify the neighbor of each segment according to neighbor threshold ε (line 01~03); 2) detect segment outlier(line 04~06); 3) detect trajectory outlier(line 07~09).

Algorithm: Similarity Computing
Input: Trajectory set $I=\{TR_1, TR_2, \ldots, TR_n\}$, ω (corner threshold), W (feature weights)
Output: Similarity matrixes M_L (Segments) and M (Trajectories)
/*First phase: Trajectory Partition*/
01: for each $TR_i \in I$ **do**
02: $TS \leftarrow$ Partition TR_i according to ω; /*Sencond phase: Computation of Structural Similarity */
03: for each $L_i, L_j \in TS \wedge i \neq j \wedge L_i.TR \neq L_j.TR$ **do**
04: Initialize SSIM(L_i, L_j);
05: SSIM_Computer(L_i, L_j)* {W}; /* multiply by weighs */
end for
06: Generate Segment Matrix M_L;
07: for each TR_i, $TR_j \in I \wedge i \neq j$ **do** /* compute similarity matrix*/
08: Initialize MSSIM(TR_i, TR_j);
09: Calculate MSSIM(TR_i, TR_j);
end for
10: Generate Trajectory Matrix M;
End.

Fig. 3 Pseudo-code of Similarity Computing.

Algorithm: Outlier Detection
Input: Similarity matrixes M_L (Segments), M (Trajectories), ε (neighbor threshold), σ, ξ, F
Output: Segment Outliers, Trajectory Outliers
01: for each *Segment Pair*(L_i, L_j) $\in M_L$ **do**
02: **if** $SSIM(L_i, L_j) \geq \varepsilon$ **then**
03: $N_\varepsilon(L_i) \leftarrow L_j$; /*set the ε-neighbor of L_i */
04: for each $L_i \in TS$ **do**
05: **if** $|N_\varepsilon(L_i)| \leq \sigma$ **then**
06: Set $SO_{\varepsilon,\sigma}(L_i) \leftarrow L_i$; /*mark L_i as an outlier*/
07: for each $TR_i \in I$ **do**
08: **if** Count(L is an outlier $\wedge$ $L \in TR_i$)/ TR_i.SegmentCount $> \xi \wedge$ MSSIM(TR_i) $< F$ **then**
09: Set $TO_{\xi,F}(TR_i) \leftarrow TR_i$; /*mark TR_i as an outlier*/
End.

Fig. 4 Pseudo-code of Outlier Detection.

3.2. Algorithm Performance Analysis

For parameters in TOD-SS, ω and ε are key factors that affect the effectiveness of the algorithms. When making experiments, we find that ω can't be set too small, or, some details of outliers will be lost. Similarly, ω can't be set too large, or, sampling exceptions will be included, and the performance will be brought down. With priori knowledge, domain experts can find a proper ω to solve this problem. Because materialized technology is used to store data and distances matrix, the retrieval effectiveness is improved by trading space for time. The time complexity of TOD-SS is $O(n\log n)$, where n is the segments count.

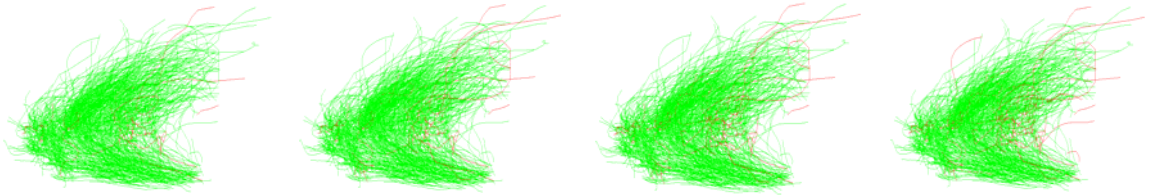
4. Experiments and Results Analysis

In order to valid TOD-SS, a trajectory mining system (TrajMiner) is developed, using Borland C++ 6.0. Environment of experiments includes: Windows XP, PC (Duro 2.8G CPU, 1G Ram). Atlantic hurricane data is adopted (http://weather.unisys.com/hurricane/index.html) as experiment data set. Hurricane data is divided into two subsets: 1) larger subset from 1950 to 2009 includes 639 trajectories consisting of 19750 points; 2) smaller subset 1990 to 2009 includes 252 trajectories consisting of 8111 points.

4.1. Parameter Selection and Results Comparison

As discussed above, the cost of TOD-SS mainly depends on ω, ε and σ. While the larger the ω is, the fewer segments are, and the longer a segment may be, therefore the more outliers will be. While, the larger of the

ε is, the fewer neighbors of L_i will be, and the greater outlier probability L_i is. Similarly, the larger the σ is, the less segments which satisfy $|N_\varepsilon(L_i)| \leq \sigma$ are, and the greater outlier probability L_i is. Figure 5 shows the outlier where ε is in 5, 6, 7 and 8. Without loss of generality, the features weight in this paper is set to: W={0.25, 0.25, 0.25, 0.25}.



ε=5 ε=6 ε=7 ε=8

Fig. 5 Outliers with Different SSIM Threshold ε.

In Figure 5, the number of outliers is found to increase with the increment of ε. For the segment L_i, with the same other parameters, the larger ε is, the less neighbors of L_i, and at fixed values of σ, the more outliers will be detected. Similarly, according to definition 2 and 3, the larger of σ is, the less segments satisfy $|N_\varepsilon(L_i)| \leq \sigma$ are, and the more outliers will be detected.

There are many differences of TOD-SS compared with algorithms in [2] and [4], for example, the algorithm proposed in [4] represents local features of the trajectory through an intertwined pattern, however, the method has questions itself. Although, the two phase partition algorithm proposed in [2] is similar to ours in certain degree, however, it is difficult to implement using our algorithm. So it is hard to compare with others. However, the effectiveness can be matched with others. Outliers detected by TOD-SS on larger dataset are given in Table 3.

Table 3 Number of Outliers and CPU Time Under Different Parameters on Larger Dataset.

ω=30, σ=4, ξ=0.5, F=15.5								
ε	1	2	3	4	5	6	7	8
time(s)	1029.6	991.3	973.1	880.3	802.4	711.2	631.8	610.2
$Num_{segments}$	956	956	956	956	956	956	956	956
$Num_{outliers}$	16	31	48	83	109	132	166	204
ε=1, σ=4, ξ=0.6, F=15.5								
ω	30	33	36	38	40	43	45	50
time(s)	1356.9	1292.6	1240.0	1238.3	1103.4	1073.1	944.2	814.5
$Num_{segments}$	643	731	785	801	719	594	502	463
$Num_{outliers}$	16	22	23	20	20	20	18	14
ε=2, ω=33, ξ=0.6, F=18								
σ	4	5	6	7	8	9	10	11
time(s)	1121.9	1112.7	1040.2	1058.3	1008.6	973.1	944.1	94.46
$Num_{segments}$	843	843	843	843	843	843	843	843
$Num_{outliers}$	18	21	24	30	30	33	34	42

Table 3 shows a full analysis of TOD-SS with different parameters. In the first part of the table, the change of ε is analyzed from 1~8, and the second part of analyzes the change of ω from 30~50, and the last

of the table analyzes the change of σ from 4~11. It can be seen from the table, the larger the ω is, the fewer segments are, and the more outliers may be. However, with the increase of ε, the *SSIM* of segment pairs will also increase, which leads a decrease of outliers. Trajectories in sparse region are always viewed as outliers in traditional algorithms. While, trajectories with different direction, speed, and angle are seldom considered to be outliers. From Figure 5, we can see that some trajectories with too slow or fast speed are detected as outliers, and those with complicated angles are also viewed as outliers.

5. Conclusion

In this paper, with the concept of trajectory structure introduced with full consideration on features, TOD-SS is proposed to partition trajectory firstly. Secondly structure feature are abstracted and similarity is computed. Finally, outliers are detected according to neighbor threshold and other related parameters. Experiments demonstrate the efficiency, effectiveness and flexibility of TOD-SS, and the detected outlier are more practical significant. With time varying, trajectory movement is apparently periodic, and worth studying. In future work, we will do study on trajectory dynamic periodic discovery.

Acknowledgement

This work was supported by the National Nature Science Foundation of China (No. 50674086), the Fundamental Research Funds for the Central Universities (No. 2010QNB21) and the Postgraduate Innovation Project of Jiangsu (No. CXZZ11_0294).

References

[1] X. L. Li, Z. H. Li, J. W. Han and J. G. Lee. Temporal outlier detection in vehicle traffic data. In *Proceedings of the 25th International Conference on Data Engineering*, pages 1319-1322, 2009.

[2] J. G. Lee, J. W. Han and X. L. Li. Trajectory outlier detection: a partition and detect framework. In *Proceedings of the 24th International Conference on Data Engineering*, pages 140-149, 2008.

[3] M. Kreveld and J. Luo. The definition and computation of trajectory and subtrajectory similarity. In *Proceedings of the 15th International Symposium on Advances in Geographic Information Systems*, pages 324-327, 2007.

[4] L. X. Liu, S. J. Qiao, B. Liu, J. J. Le and C. J. Tang. Efficient trajectory outlier detection algorithm based on r-tree. *Journal of Software*. 20(9):2426-2435, 2009.

[5] N. Pelekis, I. Kopanakis, G. Marketos, I. Ntoutsi, et. al. Similarity search in trajectory databases. In *Proceedings of the 14th International Symposium on Temporal Representation and Reasoning*, pages 129-140, 2007.

[6] D. Chetverikov and Zsolt Szabo. A simple and efficient algorithm for detection of high curvature points in planar curves. In *Lecture Notes in Computer Science*, pages746-753, 2003.

[7] M. Vlachos, M. Hadjieleftheriou and D. Gunopulos. Indexing multidimensional time-series. *International Journal on Very Large Data Bases*, 15(1):1-20, 2006.

[8] S. Tsumoto and S. Hirano. Behavior grouping based on trajectory mining. In *Social Computing and Behavioral Modeling*, pp.219-226, 2009.

[9] X. L. Li, J. W. Han, J. G. Lee, et. al. Traffic density-based discovery of hot routes in road networks. In *Proceedings of the 10th Conference on Advances in Spatial and Temporal Databases*, pages 441-459, 2007.

[10] E. M. Knorr, R. T. Ng and V. Tucakov. Distance-based outliers: algorithms and applications. *the VLDB Journal*, 8(3):237-253, 2000.

[11] M. Nanni and D. Pedreschi. Time-focused density-based clustering of trajectories of moving objects. In *Proceedings of the 18th International Conference on Data Engineering*, pages 285-289, 2002.

[12] G. Z. Wan, A. C. Bovik and H. R. Sheikh. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600-612, 2004.