Using Relative Distance and Hausdorff Distance to Mine Trajectory Clusters

Bo Guan, Liangxu Liu*, Jinyang Chen
Ningbo University of Technology
Cuibai Road 89#, Haishu District, Ningbo China, 86-0574-887081232
*corresponding author, e-mail: luransh@126.com

Abstract

Along with development of location service and GPS technology, mining information from trajectory datasets becomes one of hottest research topic in data mining. How to efficiently mine the clusters from trajectories attract more and more researchers. In this paper, a new framework of trajectory clustering, called Trajectory Clustering based Improved Minimum Hausdorff Distance under Translation (TraClustMHD) is proposed. In this framework, the distance between two trajectory segments based on local and relative distance is defined. And then, traditional clusters algorithm is employed to mine the clusters of trajectory segment. In additional, R-Tree is employed to improve the efficiency. The experimental results showed that our algorithm better than existing others which are based on Hausdorff distance and based on line Hausdorff distance.

Keywords*: trajectory clustering, movement patterns, hausdorff distance*

1. Introduction

Along with the development of GPS and Location Service, more and more location data are collected in application servers, such as, traffic control, weather monitor, intelligent navigation, biomedicine, business decisions and anti-terrorism. How to mine information from these datasets becomes increasingly broad and important research [1-2].

As clustering is one of researching hot-point of data mining, there are many researchers trying to mine clusters from location data, and lots of approaches in location data clustering are proposed. According to clustering target, existing approaches could be divided into moving object clustering and trajectories clustering. The former is focused on the cluster of moving objects [3-7], and the latter is trajectories [8-10]. In this paper, we focus on the latter.

As the image, Location data is stored by point sets in most applications. Some approaches engaged Hausdorff Distance and its variants, which is popular distance metrics in computer graphics and pattern recognition, to measure the similarity between the trajectories, such as, line Hausdorff distance [7], Hausdorff distance [12-14]. But, these methods were all based on absolute distance. Obviously, it is not fitted to measure the dissimilarity between trajectory data.

Example 1: As shown in Figure 1, given seven trajectory segments T_1, T_2, T_3, T_4, T_5, T_6, and T_7. From the graph, we can find: there are same moving pattern in T_1, T_3, T_5, and T_7, and same moving pattern inT_2, T_4, and $T6$. However, existing algorithms based on absolute distance, such as TRAOD, are difficult to distinguish between them.

In order to solve the defects of the approaches, this paper proposes a new similarity measure between trajectory segments, which is based on relative distance. In which, each trajectory is firstly divided into the segments. And then each segment is compared to the segments from other trajectories to obtain the clusters.

2. Similarity Definition of Trajectory Segments

Hausdorff Distance (HD for short) is used to measure the similarity between two point sets, which is used to measure the shape between binary images in pattern recognition. And

Received September 28, 2012; Revised November 22, 2012; Accepted November 30, 2012

then, HD was employed to measure the similarity between trajectories [5]. However, being different from image's disorder, the points in the trajectories are ordered. Therefore, some changes must be made in HD to fit the trajectories. For the convenience of discussion, two definitions are introduced firstly.

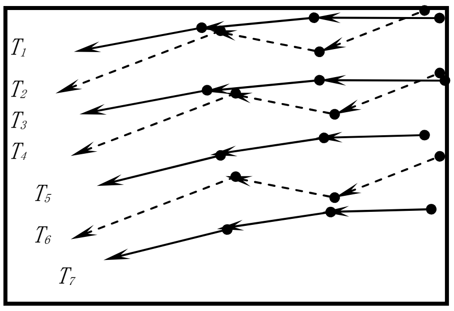


Figure 1. The Example in Trajectory Data

Definition* 1**: when one trajectory segment is compared to another one, the first segment is called ***The Target, and another one is called ***The Compared***.

***Definition* 2**: *k*-comparing unit (*k*-units) consists of k-continuous points in the trajectory.

Let $T = \{T_i | \ 0 \le i < l\}$ be the set of trajectories, and l is the trajectory number, n_i is point number of T_i, and $S_{im} = \{p_{im}, \ldots, p_{i(m+k-1)}\}$ is one k-unit of Trajectory T_i. Given two trajectories T_i, T_j, S_{im} is one *k*-unit of T_i, S_{jn} is one *k*-unit of T_j. the pair (S_{im}, S_{jn}) is called k-unit pair.

Minimum Hausdorff Distance under Translation is a measure used to measure relative distance between two points set, and in the field of pattern recognition it is often used in comparing shapes based on the binary image. Although the trajectory and the image have the same representation - point set, generally, the image is disordered, and trajectory point is disorder. HD is aimed to measure the distance between disorder point sets. However, trajectory point is the order. As a result, we made a change in HD, and each point of k-unit is compared each one of other k-unit one by one. Next, the feature of trajectory point represents moving pattern of objects, and moving pattern owns local similarity, that is to say, trajectory point only is similar to its neighboring points.

***Definition* 3:** Given a neighboring threshold ω, point p_i. if point p_j belongs to *ω*- neighboring set of p_i only if $dist(p_i - p_j) < \omega$, denotes $p_j = N_\omega(p_i)$.

Our idea is according to Minimum Hausdorff Distance under Translation, the distance between S_{im} and S_{jn} is as follows: let S_{im} is fixed, S_{jn} is translated to make them to the closest, and Improved Minimum HD under Translation can be written as:

***Definition* 4**: given *k*-units S_{im}, S_{jn}, local threshold λ, and neighbouring threshold *ω*. The relative distance is defined as:

$$d(t_{ix}, t_{jy}) = \max_{r=0,\ldots,k-1} \{d(p_{i(x+r)}, p_{j(y+r)} - t)\} \tag{1}$$

where

$$t = \frac{1}{k} \sum_{r=0}^{k-1} dist(p_{i(x+r)}, p_{j(y+r)})$$

$$d(p_{ix}, p_{jy} - t) = \begin{cases} dist(p_{ix}, p_{jy} - t), dist(p_{ix}, p_{jy}) \le \lambda \\ \infty \quad , dist(p_{ix}, p_{jy}) > \lambda \end{cases}$$

$$dist(p_{ir}, p_{jr}) = \sqrt{\left(p_{ir}.x - p_{jr}.x\right)^2 + \left(p_{ir}.y - p_{jr}.y\right)^2}$$

t is average distance between each point pair ($p_{i(m+x)}$, $p_{j(m+x)}$) ($0 \le x < k-1$). $d(p_{ix}, p_{jy}-t)$ denote point the distance between p_{jy} and p_{ix} after k-unit S_{jn} is translated by *t*. $dist(p_{ix}, p_{jy})$ denote Euclidean distance between the points p_{ix}, p_{jy}, and if $dist(p_{ix}, p_{jy}) > \omega$, p_{ix} must be not similar to p_{jy}. λ is local threshold, and the distance between two points is ∞ if their Euclidean distance after translating *t* is more than λ.

According to above analysis, we can compare not only the shape of trajectory segments but also inherent moving pattern. Improved Minimum Hausdorff Distance under Translation is more suited to measure the similarity between the trajectories. It not only eliminates common deviation through translating trajectory, but also takes moving pattern into account through measure distance by point-to point.

Because *k*-unit is trajectory segment with less point number (*k*), comparing trajectory segments must start with dividing trajectory segments into k-units. Based on this method, our solution determines, whether two trajectory segments are similar or not, by measuring their k-units as following *Definition*.

***Definition* 5**: Given two *k*-units S_{im}, S_{jn}, if $d(S_{im}, S_{jn})$ is less than local similar threshold θ, S_{im}, S_{jn} are Local Similarity, denotes S_{jn} ɛ̃ $LS_{(k,\theta)}(S_{im})$.

***Definition* 6**: Given two trajectory segments S_u , S_v, $S_{uv}^{+}=\{p \mid p \in S_{im}, S_{jn} \in LS_{(k,\theta)}(S_{im})\}$. If the following quotation is satisfied, S_u , S_v are Global Similarity:

$$|S_{uv}^{+}| \ge \zeta * |S_u|$$

where, |.| denotes points numbers in the set, and ζ is Global Similarity threshold, provided by the user in advance.

```
Input: Trajectory set $T = \{T_1, T_2, T_3 \ldots T_l\}$
Output: Clusters set $Set_c = \{C_1 \ldots C_{numclus}\}$
01: for each $T_i$ do
02:   $S_{im}$=PartitioningTrajectory($T_i$);
03:   add $S_{im}$ into set $S$;
04: $Set_c$ = ClusterTrajectory($T$);
06: return $Set_c$
```

Figure 2. Pseudo Code of *TraClustMHD*

3. The Framework of TraClustMHD

In this section, we focus on ***TraClustMHD*** framework. The algorithm could be divided into three phases. Firstly, each trajectory could be divided into trajectory segments by characteristic point; secondly, according to above similarity definitions, the similarity of each two trajectory segments is evaluated, in here, an optimized method is introduced to find out candidate similar *k*-units; finally, traditional clustering algorithm is employed to search the clusters of trajectory segments. Figure 2 shows its pseudo code.

3.1. Partitioned by character point

In order to partition the trajectory efficiently, characteristic point [7] is employed. Character point is the point in which moving objects changes their speed and direction significantly. Figure 3 shows sub- segment T_i which is formed eight points (p_1, p_2, … and p_8). Obviously, p_1, p_4, p_5, p_6, and p_8 are character points. Figure 4 shows pseudo code of partitioning trajectory. For each trajectory, the first step is taking start point and end point into Character Point Set Set_{cp}, the second step is that each point would be measure whether its changes in the direction or speed is more than given threshold, if it is true, this point is regarded as character point, and added into Character Point Set.

According to above algorithms, trajectory T_i in Figure 3 will be partitioned by character points, and formed as {(p_1, p_2, p_3), (p_4, p_5), (p_5, p_6), (p_6, p_7, p_8)}.

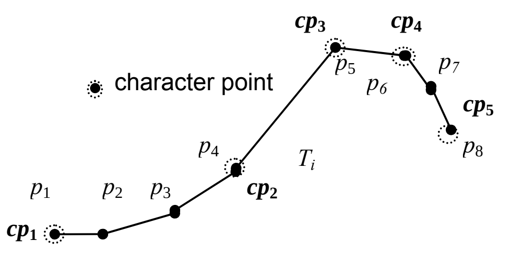


Figure 3. The Example of Character Point

```
Input: $T_i$={ $p_1$, $p_2$, …, $p_{ni}$}, MinDir, MinVelocity
Output: $Set_{cp}$
PartitioningTrajectory($T_i$)
01: Add $p_1$, $p_{ni}$ into the $Set_{cp}$;
02: for each $p_j$ ($p_j \in T_i$, $1<j<n_i$) DO
03:   Directory= computeDirectory($p_{j-1}$, $p_j$, $p_{j+1}$)
04:   Velocity = computeVelocity($p_{j-1}$, $p_j$, $p_{j+1}$)
05:   If (Directory > MinDir)
          or (Velocity > MinVelcity)
06:   Add $p_j$ into the set $Set_{cp}$;
```

Figure 4. Pseudo Code of *PartitionTrajectory*

3.2. Optimized Searching Method by Feature Matrix

According to our solution, each k-units need to be compared to all k-units from others trajectories. Obviously, its comparing cost is very expensive. How to optimize this searching process is another important problem in our framework. Our solution is finding out all candidate local similar *k*-units pairs by sort of similar of trajectory feature. In order to describe clearly, Distance Feature Matrix are introduced firstly. Distance Feature Matrix consists of point pairs which come from two trajectories respectively.

Definition 7: Suppose The Target S_i = {p_{i1}, ..., p_{im}} and The Comparing S_{jn} = {p_{j1}, ..., p_{jm}}, the distance between each point pairs is:

$$M=\begin{Bmatrix} (p_{i1},p_{j1}) & \cdots & (p_{im},p_{j1}) \\ \cdots & \ldots & \cdots \\ (p_{i1},p_{jn}) & \cdots & (p_{im},p_{jn}) \end{Bmatrix}$$

Definition 8: Diagonal Serial No.(*DSNo.*) is the difference between the row and the column of matrix, denotes $DSNo(p_{ie}, p_{jf}) = e\text{-}f$, and (p_{ie}, p_{jf}) is one matrix elements.

***Definition** 9*: *k*-diagonal Neighboring (*k-DN* for short) consists of *k* matrix elements only if the following two conditions are satisfied:

(1) $DSNo(p_{ie1}, p_{jf1}) = \ldots = DSNo(p_{iek}, p_{jfk})$;

(2) $e_2 - e_1 = \ldots = e_k - e_{k-1} = 1$.

Obviously, *k-DN* includes two *k*-units from two trajectory segments. Therefore, according to Definition 3, two k-units couldn't be similar only if one element (p_{ie1}, p_{jf1}) satisfied that the distance between its points is more than neighbour threshold ω. Based on this idea, if we orders matrix elements by (*DSNo*, *RowID*). If continuous *k* elements all satisfy ***Definition* 3**, corresponding k-units in this k-DN is candidate similar k-unit pair.

Therefore, our solution is as follows (Pseudo code as Figure 5): Firstly, for each point p_{im} of each trajectory T_i, $N_\omega(p_{im})$ is finding out by sort of R-Tree, and the element (p_{im}, p_{jn}) would be added into candidate matrix set (Candidate Set in the short), which elements order by (*i*, *DSNo*, *m*), only if p_{im}, and p_{jn} satisfy $p_{jn} \in N_\omega(p_{im})$, $dist(p_{im}, p_{jn}) < \omega$. Secondly, top matrix elements (p_{im}, p_{jn}) would be removed from Candidate Set one by one. If continuous *k* elements consists *k-DN*, the distance between its k-unit pair could be checked to determine whether two k-units is local similar. If it is satisfied, all points of target k-unit would insert into local similar point set S^+, in which all points with similar segment are included. After all pairs of trajectory segment are checked, local similar degree of the trajectory segment could be taken out. Finally, after local similar degree of all trajectory segments is calculated, DBSCAN, which is traditional clustering algorithm, is introduced into mine trajectory segments clusters.

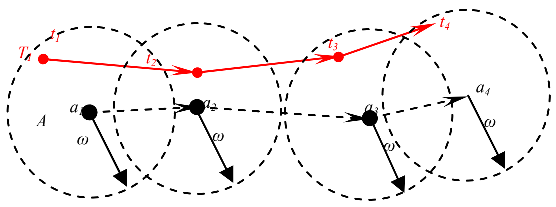


Figure 5. the Example of Distance Feature Matrix

Example 2: given A={a_1,a_2,a_3,a_4} is **The Target** in Definitoin 1(Figure 5), and T_1={t_1,…,t_4} is **The Comparing.** Distance Feature Matrix between A and T_1 is as follows.

$$M=\begin{Bmatrix} (a_1,t_1) & (a_1,t_2) & (a_1,t_3) & (a_1,t_4) \\ (a_2,t_1) & (a_2,t_2) & (a_2,t_3) & (a_2,t_4) \\ (a_3,t_1) & (a_3,t_2) & (a_3,t_3) & (a_3,t_4) \\ (a_4,t_1) & (a_4,t_2) & (a_4,t_3) & (a_4,t_4) \\ (a_5,t_1) & (a_5,t_2) & (a_5,t_3) & (a_5,t_4) \end{Bmatrix}$$

The follow takes Example 2 as the example to describe the processing of algorithms. In the first step, R-Tree is employed to search all neighbour points of each point. In the Example 2, all neighbour points of {a_1,a_2,a_3,a_4} is $N_\omega(a_1)$={t_1,t_{21},t_{31}}, $N_\omega(a_2)$={t_2,t_{22},t_{32}}, $N_\omega(a_3)$= {t_3,t_{23}}, $N_\omega(a_4)$={t_4,t_{24},t_{34}}. If the element, which distance is more than ω, is set to zero, Feature Matrix between A and T_1 is translated to:

$$M=\begin{Bmatrix} (a_1,t_1) & 0 & 0 & 0 \\ 0 & (a_2,t_2) & 0 & 0 \\ 0 & 0 & (a_3,t_3) & 0 \\ 0 & 0 & 0 & (a_4,t_4) \end{Bmatrix}$$

According above distance feature matrix, we could find out all candidate similar k-unit pairs easily. If k is set to 3, we could get two candidate similar k-units pairs {(a_1, t_1), (a_2, t_2), (a_3, t_3)}, { (a_2, t_2), (a_3, t_3), (a_4, t_4)} in Example 2.

3.3. TraClustMHD Alogrithm

In this section, we present how to use the local features to improve the efficiency of the proposed approach. We first assume that all points are indexed by R-Tree. Fig. 6 shows the pseudo code of the optimized outlying trajectory detection. As mentioned previously, this algorithm consists of two phases: pruning and refining. For each trajectory, the function of ClusteringTrajectory is used to achieve the pruning and refining.

First, for each target segment $S_{i,}$ all $N_\lambda(p_{ix})$ are discovered by searching R-Tree and then form $N\lambda(T_i)$, which is sorted by (*tid*, *DSNo.*). This process is achieved by the function of *QueryingRTree*.

Second, while the ordered stack *stackNeigh* is not empty, the following steps are executed:

(1) The entry *fEn* is obtained from the top of the ordered stack *stackNeigh*, and this recursive operation is skipped when all trajectories are processed.

(2) It is checked whether *fEn* and *lastEn* are *k-DN* adjacent (*lastEn* is the latest top of stack). One of the following operations works:

i) If the target trajectory of *fEn* is different from that of *lastEn*, which implys *fEn* is obtained from new $N_\lambda(T_j)$ (*j*<>*i*), *LSArray* is cleared, C*andLSSet* is clear, and curEn is inserted into CandLSSet.

ii) if *fEn* and *lastEn* belongs to same trajectory and aren't *k-DN* adjacent, C*andLSSet* is clear, and curEn is inserted into CandLSSet, and curDSNo is set to DSNo of curEn;

iii)if *fEn* and *lastEn* are *k-DN* adjacent: (*a*) *fEn* is inserted before the last element of *LSArray*, which is a point pair array with *k*-length (the function *addEntry* is called); (*b*) if the number of point pairs in the *LSArray* is *k* and the dissimilarity between its *k*-segments is less ζ (the function *KMatch* is called), the points p_{ix} ($p_{ix} \in T_i$) in *LSArray* are inserted into the point set *Result_Array*.

(3) *lastEn* is replaced by *fEn*.

Finally, traditional clustering analysis (DBSCAN in here) is employed to mine all segment clusters.

```
Input:  A set of trajectories $T = \{ T_1 \ldots T_n \}$, parameters $\zeta$, $k$, $\omega$, $\lambda$, MinDirectory,
MinVelocity.
Output: A set of clusters $Set_c = \{C_1 \ldots C_{numclus}\}$
ClusteringTrajectory()
01. For each sub-segment $S_i \in S$ DO
02. Result_Array= MatchTrajectory ($S_i$, $i$, $\zeta$, $k$, $\omega$, $\lambda$)
03. $Set_c$ =Cluster(Result_Array)
MatchTrajectory ($S$, $I$, $\theta$, $p$, $k$, $\omega$)
01: stackNeigh = QueryRTree ($S_i$, $\omega$);
02:   while (stackNeigh isn't NULL)
03:     curEn = stackNeigh. RevTop ();
04:     If (curEn. traj != cur_traj){
05:           cur_traj = curEn. traj;
06:           CandLSSet=NULL; LSArray=NULL;
07:           CandLSSet.addEntry(curEn);
07:     else
08:        If (curEn.DSNo!=curDSNo) or (curEn.pos-1 == curPos)
09:           CandLSSet=NULL; curDSNo = curEn.DSNo;
11:           CandLSSet.addEntry(curEn);
10:        else
11:           If ((CandLSSet.size())> k-1)
12:              isMatch=KMatch(curEn, k, $\lambda$));
13:              If (isMatch)
14:                 LSArray.addEntry(curEn,  CandLSSet);
15:                 If LSArray.Size()$\geq$ $\zeta$*$S_i$.Size();
16:                   Result_Array[i][j]=1;
17:                 Else Result_Array[i][j]=0;
18:         lastEn =curEn;
18   return (Result_Array);
```

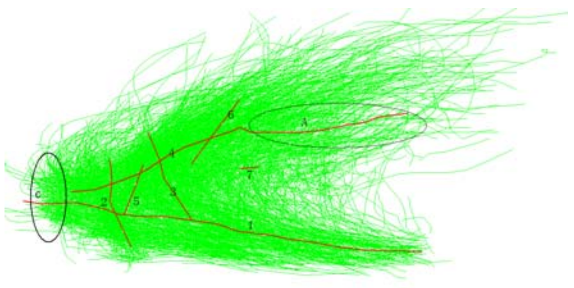
Figure 6. the Example of Distance Feature Matrix

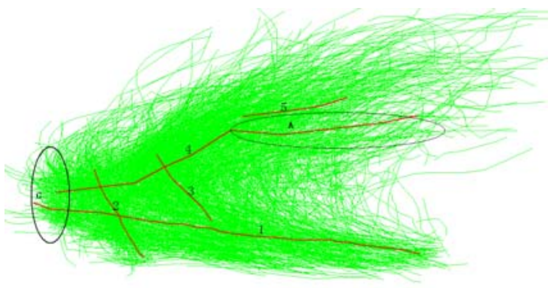
4. Experimental Analysis
4.1. Experimental Environment

We implement relative algorithms and *TraClustMHD* in Visual C++, on the XP OS and execute all experiments on a notebook with Centrino 2 2.1G CPU and 2G main memory. The experimental dataset is from the wealth of hurricane information including charts on the track of the storm plus a text based table of tracking information. We select hurricane data of Atlantic hurricanes from 1850 to 2010, which has 1294 trajectories with 30368 points. All points in trajectory data are imported into binary file according to its trajectory ID, and then indexed by R*-Tree. And clustering algorithm is DBSCAN.

4.2. Experimental Analysis

Based on ensuring best performance through adjusting each parameter, we make a comparison in Trajectory Clustering through Line Hausdorff Distance [8], Hausdorff Distance [16], and our framework. Figure 7 shows comparing results in them. From the graphs, we could find that all algorithms shows moving pattern of hurricane. However, there are some bugs in the algorithms based on Line HD and HD because based on absolute distance and global comparing. Both of them could find out the clusters in Region A and C, because this Region own enough trajectories. But in our framework, the clusters in Region A and C are disappear. The reason is that moving direction is taken into account in our framework.


(a) Result of using Line Hausdorff Distance


(b) Result of using Hausdorff Distance

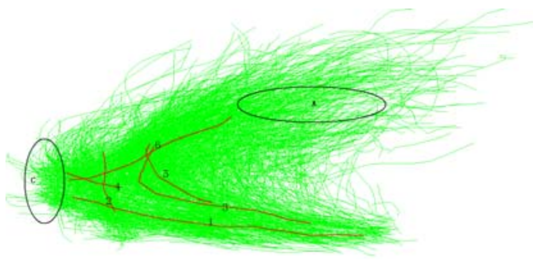

(c) Result of using TraClustMHD

Figure 7. Comparing Results in Three Algorithm

Figure 8 shows that the comparison of CPU Time in three algorithms. From the graph, TraClustMHD owns higher efficiency than others. And more trajectory number is, more superiority TraClustMHD owns. The reason is that R-Tree are employed to eliminate computation between irrespective points.

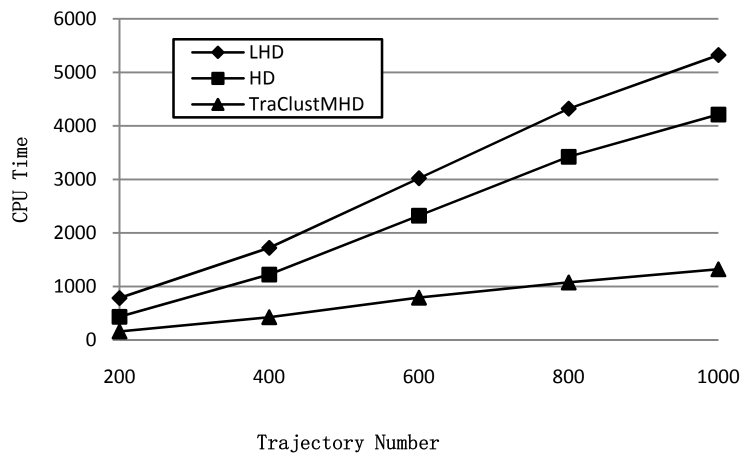


Figure 8. Comparing result in CPU time

5. Conclusion

Along with more researchers focusing on trajectory clustering, this paper proposes a trajectory clustering framework based on local relative distance. This framework not only uses local relative distance to measure similarity more correct, but also improves the performance through indexed by R-Tree. Experimental Results show that our framework has more efficiency and effective than other algorithms.

Acknowledgement

This work was supported by Zhejiang Provincial Education Department scientific research project of under grant No. Y201119588, and Natural Science Foundation of Ningbo under grant No. 2009A610090, No. 2010A610106, and No. 2011A610175.

References

[1] YF Li, JW Han, J Yang. Clustering Moving Objects. *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Seattle. USA, 2004; 617-622.

[2] Ester M, Kriegel, HP, Sander J, and Xu, X. *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*. Proc. 2nd Int'l Conf. on Knowledge Discovery and Data Mining, Portland, Oregon. 1996; 226-231.

[3] S Gaffney and P Smyth. *Trajectory clustering with mixture of regression models*. Proceedings *of* the 5th International Conference on Knowledge Discovery and Data Mining (KDD'99), San Diego, 1999; 63–72.

[4] D Chudova, S Gaffney, E Mjolsness and P Smyth. *Translation invariant mixture models for curve clustering*. Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (KDD'03), Washington. 2003; 79-88.

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

[6] Hwang JR, Kang HY, Li KJ. *Spatio-temporal Similarity analysis between trajectories on road networks*. Proceeding of 24th international conference on Perspectives in Conceptual Modeling (ER'05).Klagenfurt, Austria. 2005; 280-289.

[7] J Lee, J Han, and Kyu-Young Whang. *Trajectory clustering: A partition-and-group framework*. Proc. 2007 ACM SIGMOD Int'l Conf. on Management of Data, Beijing China. 2007. 593-604.

[8] Christian S Jensen, Dan Lin, Beng Chin Ooi. *Continuous Clustering of Moving Objects*. Knowledge and Data Engineering. 2007; 19(9): 1161-1174.

[9] Yingyi Bu, Lei Chen , Ada Wai-Chee , Fu Dawei Liu. E*fficient Anomaly Monitoring Over Moving Object Trajectory Streams*. *KDD'09,* Paris, France. 2009: 159-168.

[10] Zhenhui Li, Jae-Gil Lee, Xiaolei Li and Jiawei Han. *Incremental Clustering for Trajectories*. Proceedings of the 15th international conference on Database System for Advance Applications. Tsukuba Japan. 2010: 32-46.

[11] Zhen hui, Jiawei Han, Ming Ji. et al. *MoveMine: Mining Moving Object data for discovery of aninal movement patterns*. ACM Transactions on Intelligent System and Technology (TIST). 2010. 2(4): 37:1-37:32

[12] Lou Jian-guang, Liu Qi-feng, Tan Tie-niu, et al. *Semantic interpretation of object activities in a surveillance system*. Proceedings of the 16 the International Conference on Pattern Recognition (ICPR'02), Washington. 2002; 777-780.

[13] Junejo IN, Javed O, Shah M. *Multi feature path modeling for video surveillance*. 17th International Conference on the Pattern Recognition (ICPR'04), Washington. 2004; 716-719.

[14] Khalid S, Naftel A. *Evaluation of matching metrics for trajectory-based indexing and ret rieval of video clips*. Proceedings of the Seventh IEEE Workshops on Application of Computer Vision (WACV/ MO2TION' 05), Washington. 2005; 242-249.

[15] Qu Lin, Zhou Fan, Chen Yao-wu. *Trajectory classification based on Hausdorff distance for visual surveillance system*. 2009; 39(6): 288-299.