IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 1, No 3, January 2013
ISSN (Print): 1694-0784 | ISSN (Online): 1694-0814
www.IJCSI.org

Mining User Similarity Using Spatial-temporal Intersection

Yimin Wang[1], Ruimin Hu[1], Wenhua Huang[1] and Jun Chen[1]

**[1] National Engineering Research Center for Multimedia Software, School of Computer, Wuhan University
Wuhan, Hubei, 430072, China**

Abstract
The booming industry of location-based services has accumulated a huge collection of users' location trajectories and also brings us opportunities and challenges to automatically discover valuable knowledge from these trajectories. In this paper, we investigate the problem of measuring the similarity between users. Such user similarity is significant to individuals, communities and businesses by helping them effectively retrieve the information. To achieve this goal, we firstly propose a storage structure to represent the user's trajectories, which not only stores the sequence of user's trajectory, but also stores regions with indexing of trajectories which pass the regions. After that, we give the similarity function between users using the spatial-temporal intersection in regions which are passed by the two users. Finally, we develop a spatial-temporal intersection algorithm to measure user similarity based on the definition and storage structure, and we illustrate the results and performance of the algorithm by extensive experiments.
Keywords: *Trajectory Analysis, User Similarity, Spatial-temporal Data Mining*

1. Introduction

Recently, the increasing pervasiveness of location-acquisition technologies, like GPS and GSM networks, are leading to the collection of large spatial-temporal trajectory data, which bring the opportunity of discovering valuable knowledge about users' movements [1]. A number of interesting applications are being developed based on the analysis of trajectories. For example, it is possible to determine migration patterns of animals by analyzing similar trajectories of them; in a city traffic system, it is helpful to locate popular routes for programming a new route by mining the trajectories of the citizens. The basis of these applications is determining the similarity among trajectories of moving object.

The trajectory of a moving object could be defined as a sequence of positions of the moving object over a period of time, see Fig. 1. We will say that A and B are more similar, because they pass more common locations. As a trajectory is a time series essentially, many sequence similarity search techniques were used to address this problem, such as Longest Common Subsequences (LCSS)[5], Dynamic Time Warping (DTW)[6], and Edit Distance on Real sequence (EDR)[7] etc. These methods address the problem of different length, noise and local shift, which often appear on comparing the similarity between trajectories. However, the complexity of the algorithm is O(n2), where n is the length of trajectory. Many existing methods were developed based on these classical methods, most of which optimize the efficiency of k-nearest neighbors search algorithm by using index structure and pruning techniques, nevertheless these optimization are restricted by the complexity of computing similarity between two trajectories.

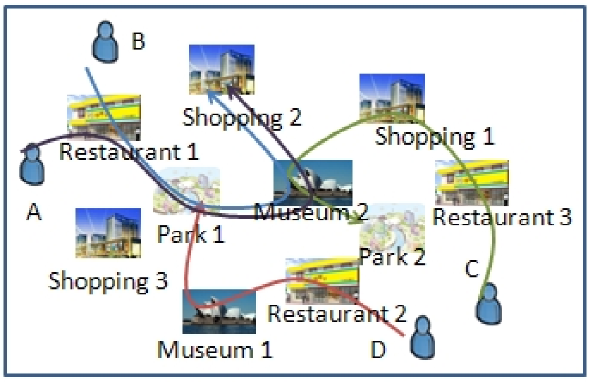


Fig. 1 Trajectories of Persons

In this paper, we proposed a new approach inspired by text retrieval methods, in which a document is seen as a set of many words and all the documents are organized as an inverted file to facilitate efficient retrieval. An inverted file is a structure that has an entry for each word in the corpus followed by a list of all the documents in which occurs. In our methods, the persons are analogy to documents and the locations are the words. Different to words which are discrete, locations are continuous. We compensate this difference by region dividing. Specially, we divide the whole activity area on which the persons move on into many small regions, and then we use the sequence of regions to approximately represent the trajectory. While computing the similarity between two persons, we use the number of regions passed by both the two persons, with the region-based representation.

The major contributions of this paper are as follows:

1. We propose Compression representation of user's trajectories and a storage structure which not only stores

the grid sequence of a trajectory, but also stores regions with indexing of trajectories which pass the regions. With the storage structure, we can easily determine whether and when a trajectory passes through a certain grid.

2. We give three definitions of similarity function, named Maximum Co-occurrence Time (MCT): Absolute Similarity, Relative Similarity and Partial Similarity based on the compression representation. The detail description of these definitions is given in Section 2.

3. Based on the storage structure, we propose a fast k-nearest neighbor search algorithm. The detailed description of these algorithms is given in Section 3.

The rest of the paper is organized as follows. Section 2 describes the grid representation of trajectory, a storage structure and a novel similarity function. In Section 3, we give the k-nearest neighbor search algorithm. In Section 4, we evaluate the accuracy and efficiency of MCT by comparing it with other algorithms. Section 5 is a brief conclusion.

2. Related Work

Similarity search has been well studied in the context of time series and trajectory data. The simplest approach to define the similarity between two sequences is to convert sequence into a vector and then use a p-norm distance to define the similarity measure. The p-norm distance between two n-dimensional vectors $\bar{x}$ and $\bar{y}$ is defined as $L_p(\bar{x},\bar{y}) = (\sum_{i=1}^{n}(x_i - y_i)^p)^{\frac{1}{p}}$. For p=2 it is the well known Euclidean distance.

Agrawal et al. firstly proposed an approach for sequence similarity measure. Their method transforms the sequence into vector with Discrete Fourier Transformation and uses Euclidean distance to determine the similarity [2]. Chen et al. [3] took Discrete Wavelet Transformation to convert the sequence, while Cai et al. [4] transformed the sequence with Chebyshev Polynomials. However, these methods based on p-norm distance require the trajectories with the same length.Several typical similarity functions for different length sequence include Longest Common Subsequence (LCSS) [5], Dynamic Time Warping (DTW) [6], Edit Distance on Real Sequences (EDR) [7]and Edit Distance with Real Penalty (ERP) [8]. Time warping technique first has been used to match signals in speech recognition.

Berndt and Clifford[6] proposed to exploit this technique to measure the similarity of time-series data, and its basic idea is to allow ‘repeating’ some points as many times as needed to achieve the best alignment. Sakurai et al. [9] improved DTW by using an index structure with segmentation and lower bounded distance measure. Vlachos et al. [5] used LCSS to compare two trajectories with the consideration of spatial space shifting. Longest Common Subsequence is a classical distance function for two strings; another distance function is Edit Distance. The edit distance between two strings of characters is the number of operations required to transform one of them into the other. The allowed operations include ‘replace’, ‘delete’, ‘insert’ and so on. Each operation’s cost is 1. EDR and ERP are both based on Edit Distance and proposed by Chen et al. EDR directly utilize Edit Distance to measure similarity of two trajectories. Compare to EDR, ERP use the Euclidean distance as operation’s cost instead of 1[8]. These method could be used in both time series and trajectory data and the time complexity is basically O(n^2).

Lin et al. [10] use OWD to compute two trajectories similarity. They only consider the spatial shape of trajectory without the time ordering, and utilize grid to approximate represent trajectories. Grid representation is also used in this paper. Frentzos et al. [11] define a dissimilarity metric (DISSIM) for the measurement of the spatiotemporal dissimilarity between two trajectories. Pelekis et al. [12] introduce the Locality in-between Polylines (LIP) function. The idea is to calculate the area of the shape formed by two 2D polylines. This method requires that the area is finite.

Recently, Tiakas et al. [13] consider similarity search for moving object trajectories in spatial networks. They use the distance in networks instead of the Euclidean distance between two points. Chen et al. [14] study a problem of searching trajectories by locations, in which the query is a set of locations.

3. Representation and Storage of Trajectory

This section first introduces a region-based representation for user’s trajectories, and then proposes a storage structure for the region-based trajectories.

3.1 Dividing Region

For transfer continuous location information to discrete regions, the cluster method could be used for creating the regions from location information by merging the locations with a small distance in spatial [1]. In real world, the region usually dividing by manual or auto divided though the semantic function of an area, such as super market, restaurant, park and Stadium etc. For easy to explain, we use a grid-based region dividing methods in this paper, with which the whole area is divided to many grids with the same size, and a grid is seen as a region, see Fig. 2. This method was also exploited by [10]

3.2 Region-based Representation

We divide the whole activity area of moving object into disjoined small regions, and then we merge the continuous points in a region, formulated as (R_i, t_i, t_i), where R_i means the region, t_i means the time of the first point of the continuous points, and t_i means the interval between the last point and the first point.

Therefore, the user's trajectory can be compressed represented as:

$$C(o) = \{(R_0, t_0, t_0), (R_1, t_1, t_1), \ldots , (R_m, t_m, t_m)\}, t_i + t_i < t_{i+1}, 0 \le i \le m \quad (1)$$

Further, (t_i, t_i) is denoted by T_i for short, that is $T_i = (t_i, t_i) = [t_i, t_i + t_i]$, and then (R_i, t_i, t_i) is denoted as (R_i, T_i), and we have $t_i = |T_i|$.

The diagram of compressed representation is shown as Fig 2. We don't restrict the way of dividing the activity area, but use the grid with the same size in this paper merely.

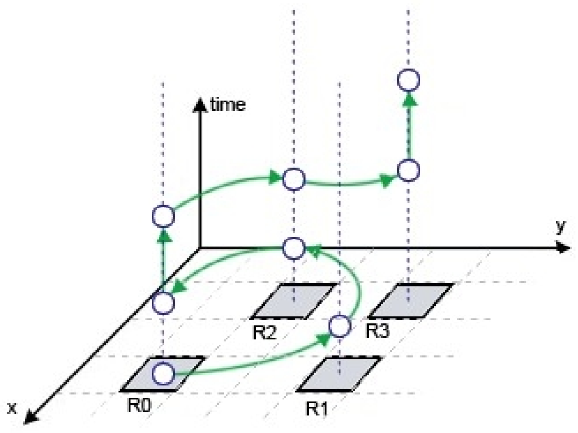


Fig. 2 compressed representation

3.3 Storage Structure

Storage structure is divided into two parts, shown in Fig 3. The first part is a users' trajectories table for storing the users' trajectories which store index of regions which are passed by the users. The second part stores a set of regions, in which indexing of trajectories and the intervals of users passing the region are stored. For example, as shown in the Fig 3, User3 passes three regions, R3, R5 and R7; and R7 is passed by three users, User1, User3 and User5, while User5 passes R7 at three interval [t1,t2], [t3,t4], [t5,t6].

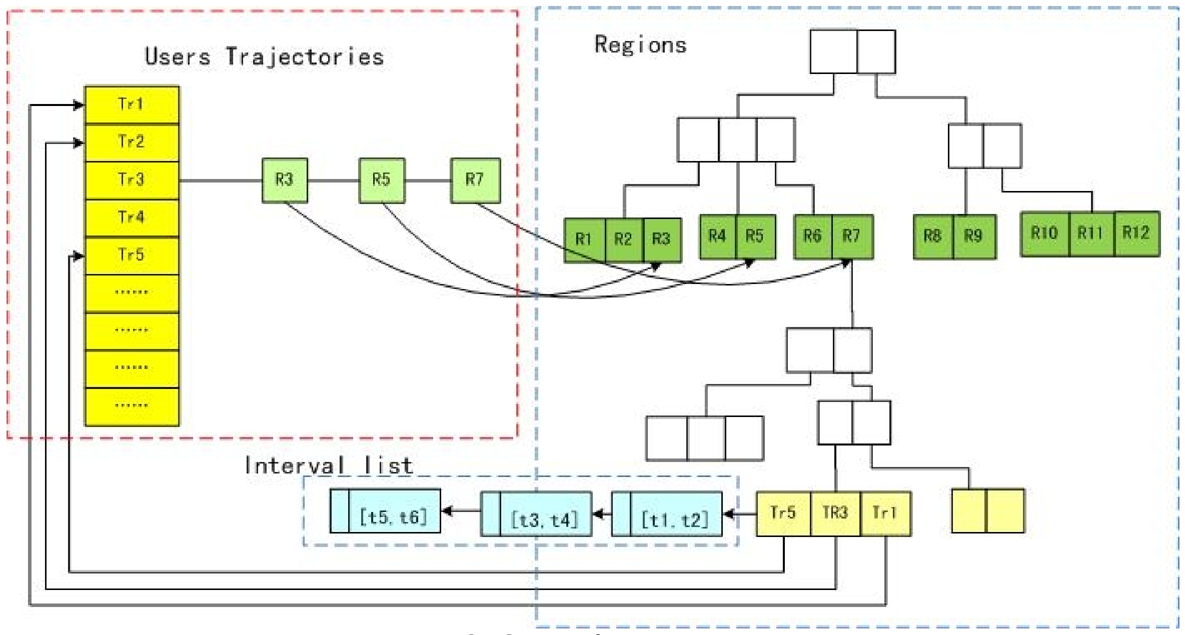


Fig. 3 The diagram for storage structure

4. Similarity Function and KNN Search

This section first gives three similarity functions between trajectories and then provides an algorithm for KNN search with the storage structure proposed in Section 3.

4.1 Similarity Function

Before defining the similarity function, we first introduce a variable to measure the user's activity scope in a region with several operations.

Definition 1 : The user's activity scope in the region R_i is defined as $V(R_i,t_i,t_i)$, and $V(R_i,t_i,t_i)=S(R_i)\times t_i$, where $S(R_i)$ represents the area of the region R_i .

Intersection operation:

$$V((R_i,t_i,t_i)\cap(R_j,t_j,t_j))=\begin{cases}0 & R_i\neq R_j\\ V(R_i,T_i\cap T_j)=S(R_i)\times\left|T_i\cap T_j\right| & R_i=R_j\end{cases}\qquad(2)$$

Union operation:

$$V((R_i,t_i,t_i)\cup(R_j,t_j,t_j))=\begin{cases}V(R_i,t_i,t_i)+V(R_j,t_j,t_j) & R_i\neq R_j\\ V(R_i,T_i\cup T_j)=S(R_i)\times\left|T_i\cup T_j\right| & R_i=R_j\end{cases}\qquad(3)$$

Based on the above operations, we define three similarity functions: Absolute Similarity, Relative Similarity and Partial Similarity.

Definition 2: we define the Absolute Similarity as the absolute value of intersection of the user's activity scope. The formula is as follows.

$$aSim(o_i,o_j)=V(C(o_i)\cap C(o_j))\qquad(4)$$

Definition 3: we define the Relative Similarity as the ratio of the intersection and union of the user's activity scope. The formula is as follows:

$$rSim(o_i,o_j)=\frac{V(C(o_i)\cap C(o_j))}{V(C(o_i)\cup C(o_j))}\qquad(5)$$

From the definition of Relative Similarity, we can easily obtain that the similarity between two trajectories is between 0 and 1.

Definition 4: we define the Partial Similarity as the ratio of the intersection and one user's activity scope. The formula is as follows:

$$pSim_{o_i}(o_j)=\frac{V(C(o_i)\cap C(o_j))}{V(C(o_i))}\qquad(6)$$

Obviously, *Absolute Similarity* and *Relative Similarity* function is symmetrical, where *Partial Similarity* is not, and *Relative Similarity* can be seen as Normalized form of *Absolute Similarity*. In some applications, *Partial Similarity* is valuable. This paper focuses on *Absolute Similarity*.

4.2 Similarity Search Problems

K-nearest neighbors search problem can be described as follows:

Given a set of Users, S={O_1,O_2,……, O_n}, a query User Q, and a positive integer k, find the k Users in S to form a new set S', meeting for any User O_i in S' and any User O_j in S-S' , Sim(O_i,Q)>=Sim(O_j,Q).

4.3 KNN-search Algorithm

This subsection gives the k-nearest neighbor search algorithm, see Algorithm 1. Inputs include target user, the number of the nearest neighbor and user set with compressed trajectories. And the output is the k nearest users. Lines 1-2 are used to initialize memory space used for storing similarity values, Lines 4-10 are used to sum up the co-occurrence time of user at each region to calculate the similarity of users. The "Com_Time" in Line 6 is a function to calculate intersection of two interval lists.

```
Algorithm 1 MCT_KNN(Q,K,Os)
input: Q—Target User
       K—The number of the nearest neighbor
       Os—user set with compressed trajectory
Output: the user set of K nearest neighbor
1.  Initialize V to store the trajectories similarity
2.  initialize Vk to store the k most similar trajectories
3.  convert Q into compressed representation
4.  For each Ri in Q do
5.    For each Oi in Ri do
6.      V(Oi) + = Com_Time(Tqi,Toi)
7.  update Vk using V(Oi)
8.    End for
9.  End for
10. Return Vk
```

5. Experimental Evaluation

In this Section, we present experimental results to evaluate our techniques, Maximum Co-occurrence Time (MCT) by comparing it with well known method: LCSS, DTW and EDR. For experimental purposes, we used the synthetic datasets which are generated by Network generator—a moving object dataset generator developed by Brinkhoff [15]. This trajectory generator is also used by [10]. Our experiments were run on a PC AMD Athlon at 2.09GHz with 1.87G RAM and 160G hard disk.

5.1 Accuracy Evaluation

Accuracy is one of the most important aspects of similarity function. In this work, we use an objective evaluation method recently exploited by [5-7,10] to evaluate the accuracy of our techniques. The idea is to use a k-nearest neighbor classifier on labeled data to evaluate the efficacy of the distance measure used. We used the trajectories generator create 1000 trajectories with 500 time steps and 20 Categories. We continue to adjust the size of e and select the optimal value for LCSS and EDR, and set the parameter d for LCSS to be ∞ to get the optimal accuracy. For our method, we set the size of grid to be 2 times of e for LCSS. The ratio of train set is 20%. In order to avoid possible random, we repeated the experiments 20 times and took the average.

Fig. 4 shows the results. Our method-MCG has nearly classification accuracy with LCSS, while has higher accuracy than DTW and EDR.

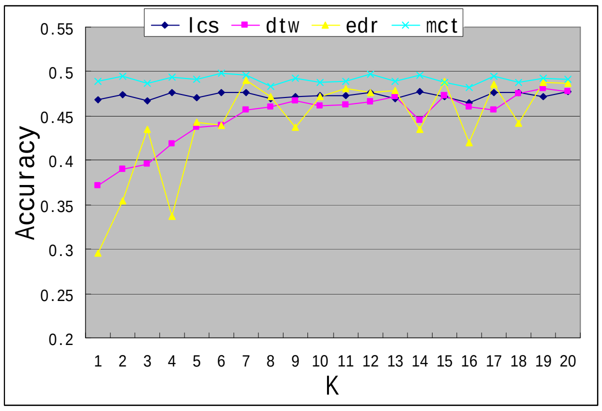


Fig. 4 Classify accuracy with different train ratio

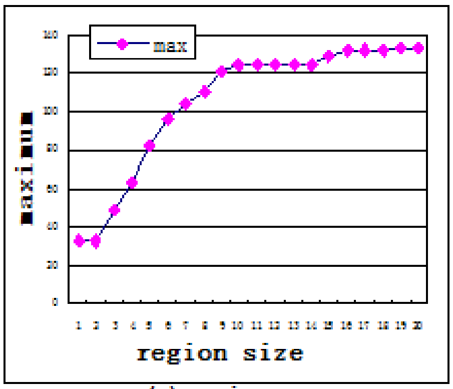
5.2 The number of Trajectories in grid

By algorithm 3, the key of KNN search efficiency is the number of trajectories in grid. It is easy to know, the number is relative to the size of grid cells and dataset. In this section, we analyze the impact of the two factors.

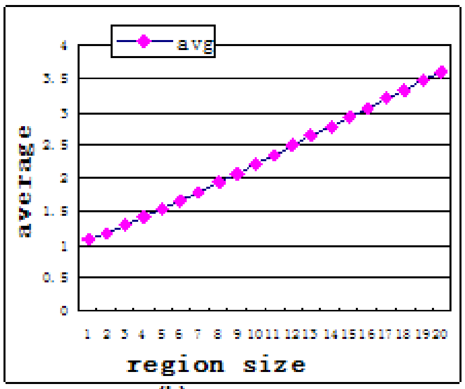
5.2.1 Impact of grid size

In this subsection, we do experiments on Trucks dataset, changing the size of grid cell from 1 to 20, and recording the maximum and average of the number of trajectories in grid cells. Fig. 5(a) shows changes of maximum value. When the size is smaller than 9, maximum value grows as a linear almost, while when the size is greater than 9, the value is steady. Fig. 5(b) shows average value grows as a linear as size of grid is increase.

This is consistent to our conscious. By increasing the size of grid, two trajectories which were in different grids originally, may be in the same grid now, so the maximum and average value may increase. To reduce the computation, it should be reduce grid size, but too small grid cells may lead to greater errors and more Space consumption. Therefore, we should set the grid size dependent on the application and the dataset.



(a) maximum



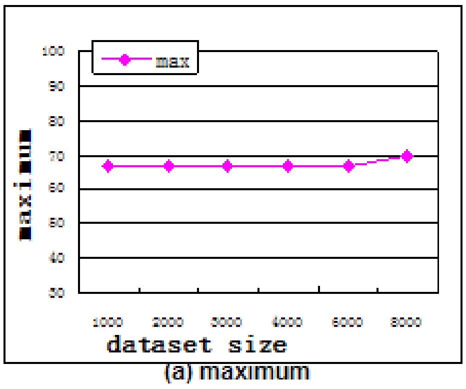
(b) average

Fig. 5 Impact of grid size to the number of trajectories in one grid

5.2.2 Impact of dataset size

In this section, we use Network generator to create datasets with different size, concluding 1000, 2000, 3000, 4000, 6000, and 8000.

Fig. 6 shows that the maximum and average value is almost steady, when the size of datasets increases. The reason is that new trajectories increase the number of grid cells, thus maximum and average value change little. Further, the average number of trajectories in one region is much smaller than the total number of trajectories set. This property makes our methods significantly faster than other methods with matching trajectory directly. In a sense, we use a hash technology when we use an inverted file.



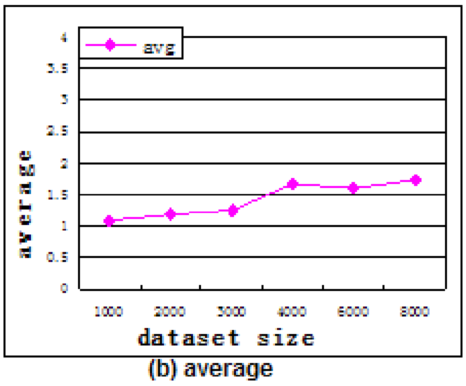


Fig. 6 Impact of dataset size to the number of trajectories in one grid

5.3 Efficiency Evaluation

Efficiency is the other important aspect. We use Network generator to create datasets with different size, concluding 1000, 2000, 3000, 4000, 6000, and 8000.

Fig. 7 presents the results. The time of DTW is similar to EDR, and is longer than LCS. Our method is much faster than other methods: LCS, DTW and EDR, even not in an order of magnitude. There are two factors to get the fast speed for our method. First, the number of trajectories in grid cells is much smaller than the number of trajectories; and second, we only compute the grid cells passed by the target users, which is also much smaller than the number of grids, instead of comparing them with all users.

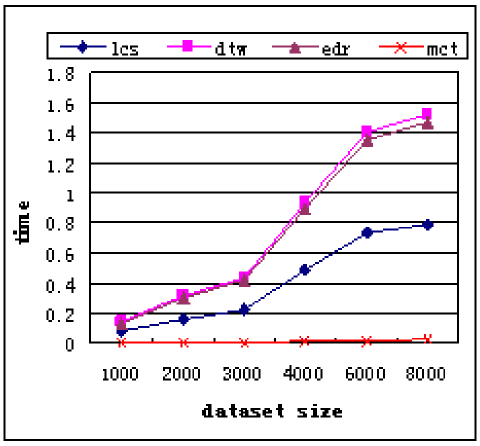


Fig. 7: Efficiency of KNN-search

6. Conclusion

In this paper, we present efficient techniques to compute the similarity between users by mining the historical trajectories. We first divide the space into small regions and compressed represent user's trajectories. Then, we presented a storage structure which not only stores the sequence of user's trajectory, but also stores regions with indexing of trajectories which pass the region. Based on compressed representation, we defined three similarity functions between users. At last, we give the k-nearest neighbor search algorithm and evaluation the accuracy and efficiency. Our experiments indicate that our similarity function has good accuracy comparing with LCSS, DTW and EDR and our algorithm is significantly faster than other algorithms.

Acknowledgments

Acknowledgments The research was supported by the major national science and technology special projects (2010ZX03004-003-03, 2010ZX03004-001-03), the National Basic Research Program of China (973 Program) (2009CB320906), the National Natural Science Foundation of China (60970160, 61070080, 61003184, 61172173, 61170023, 61231015).

References

[1] Zheng, Y., et al. , Recommending friends and locations based on individual location history. ACM Trans. Web, 2011,5(1), pp5-42.
[2] Agrawal, R., C. Faloutsos and A.N. Swami. Efficient Similarity Search In Sequence Databases. in FODO '93. 1993. London, UK: Springer-Verlag.

[3] Efficient Time Series Matching by Wavelets. in ICDE '99. 1999. Washington, DC, USA: IEEE Computer Society.
[4] Cai, Y. and R. Ng. Indexing spatio-temporal trajectories with Chebyshev polynomials. in SIGMOD '04. 2004. New York, NY, USA: ACM.
[5] Vlachos, M., D. Gunopoulos and G. Kollios, Discovering Similar Multidimensional Trajectories, in 18th International Conference on Data Engineering (ICDE'02). 2002: Los Alamitos, CA, USA. p. 0673.
[6] Berndt, D.J. and J. Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. 1994.
[7] Chen, L., et al. Robust and fast similarity search for moving object trajectories. in Proceedings of the 2005 ACM SIGMOD international conference on Management of data. 2005. Baltimore, Maryland: ACM.
[8] Chen, L. and R. Ng. On the marriage of Lp-norms and edit distance. in VLDB '04. 2004: VLDB Endowment.
[9] Sakurai, Y., M. Yoshikawa and C. Faloutsos. FTW: fast similarity search under the time warping distance. in Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. 2005. Baltimore, Maryland: ACM.
[10] Lin, B. and J. Su, One Way Distance: For Shape Based Similarity Search of Moving Object Trajectories. GeoInformatica, 2008. 12(2): p. 117-142.
[11] Frentzos, E., K. Gratsias and Y. Theodoridis. Index-based Most Similar Trajectory Search. in Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on. 2007.
[12] Pelekis, N., et al. Similarity Search in Trajectory Databases. in Proceedings of the 14th International Symposium on Temporal Representation and Reasoning. 2007: IEEE Computer Society.
[13] Tiakas, E., et al., Searching for similar trajectories in spatial networks. 2009. 82(5): p. 772-788.
[14] Chen, Z., et al. Searching trajectories by locations: an efficiency study. in SIGMOD '10. 2010. New York, NY, USA: ACM.
[15] Brinkhoff, T. Generating Traffic Data, IEEE Data Eng. Bull., 2003, 26(2), pp. 19-25.

Yimin Wang received the B.S. degree in computer school of Wuhan University, Wuhan, China, in 2008. He is currently working as a doctor at National Engineering Research Center for Multimedia Software, Wuhan, China. His research interests include public safety, multimedia content analysis, machine learning, and Data Mining.

Ruimin Hu received the B.S and M.S degrees from Nanjing University of Posts and Telecommunications, Nanjing, China, in 1984 and in 1990, and Ph.D degree in Communication and Electronic System from Huazhong University of Science and Technology, Wuhan, China, in 1994. Dr. Hu is the director of National Engineering Research Center for Multimedia Software, Wuhan University and Key Laboratory of Multimedia Network Communication Engineering in Hubei province. He is Executive Chairman of the Audio Video coding Standard (AVS) workgroup of China in Audio Section. He has published two books and over 100 scientific papers. His research interests include audio and video coding and decoding, video surveillance and multimedia data processing.

Wenhua Huang received the B.S. degree in computer school of Wuhan University, Wuhan, China, in 2010. He is currently working as a Graduate at National Engineering Research Center for Multimedia Software, Wuhan, China. His research interests include public safety and multimedia content analysis.

Jun Chen received the Ph.D degree in computer school of Wuhan University, Wuhan, China. He is currently a professor at National Engineering Research Center for Multimedia Software, Wuhan, China. His research interests include public safety and multimedia content analysis