Symbolization of Mobile Object Trajectories with the Support to Motion Data Mining

Xiaoming Jin, Jianmin Wang, and Jiaguang Sun

School of Software
Tsinghua University, Beijing, 100084, China
xmjin@mail.tsinghua.edu.cn

Abstract. Extraction and representation of the events in trajectory data enable us go beyond the primitive and quantitative values and focus on the high level knowledge. On the other hand, it enables the applications of vast off the shelf methods, which was originally designed for mining event sequences, to trajectory data. In this paper, the problem of symbolizing trajectory data is addressed. We first introduce a static symbolization method, in which *typical sub-trajectories* are generated automatically based on the data. For facilitating the data mining process on streaming trajectories, we also present an incremental method, which dynamically adjusts the *typical sub-trajectories* according to the most recent data characters. The performances of our approaches were evaluated on both real data and synthetic data. Experimental results justify the effectiveness of the proposed methods and the superiority of the incremental approach.

Keywords: Motion data mining, spatial trajectory, symbolization

1 Introduction

The recent advances in geographic data collection devices and location-aware technologies have increased the production and collection of spatial trajectories of moving objects [1, 2]. Well known examples include global positioning systems (GPS), remote sensors, mobile phones, vehicle navigation systems, animal mobility trackers, and wireless Internet clients. Briefly, trajectory of a moving object is a sequence of consecutive locations of the object in a multidimensional (generally two or three dimensional) space [3]. Fig. 1 shows a simple example of 2D trajectory data.

Recently, developments of data mining techniques on trajectory data have received growing interests in both research and industry fields [4, 5, 6]. For example, sequential rules with the format “if *movement A* then *movement B*” can be discovered by searching the historical trajectories generated by a vehicle user. Then based on the rules, personalized services can be provided according to the user’s current movements in a vehicle navigation system. In another case, by analyzing the trajectories generated by all vehicles in a city, we can find some frequent patterns that reveal important passages, crossroads, highway, or other traffic facilities that are used frequently. The results of such analysis can be used for improving the management and maintenance of the traffic system.

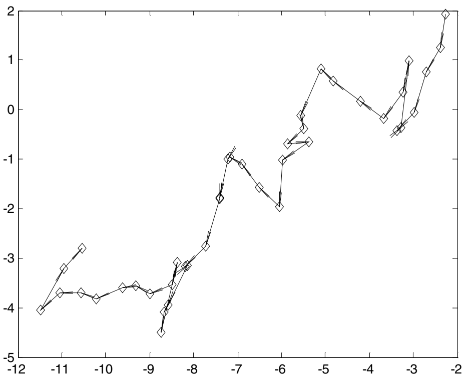

Fig. 1. An example of moving object trajectory in 2D space. Each diamond in the figure stands for a single location of the object at a certain time, and arrow represents a movement from one location to another.

In this paper, we address a novel problem, which is symbolization of trajectories. The problem is to map a trajectory into a symbolic sequence, in which each symbol represents the local movement at a time point or during a time period. For example, the trajectory of a car movement trajectory can be simply represented by a symbolic sequence in which each symbol indicates the shape of a quarter-hourly movement. Then the symbolic sequence can be examined for discovering rules such as "if a particular movement *A*, indicating a crossroads, occurs, then the car will go north straightly, say movement *B*, in the second quarter-hour."

Our research on this topic is mainly motivated by two ideas: First, there is a broad consensus that we are usually interested in high level representation of the data, rather than the primitive and quantitative values. Second, techniques for mining symbolic (event) sequences have been studies extensively in various application domains, such as click stream analysis, bio-informatics, and so on, and many sophisticated algorithms, models, and data structures were designed for handling symbolic sequences. Therefore, converting trajectory data into symbolic sequence could enable the applications of the vast off the shelf methods in mining trajectory data. Obviously, an appropriate and efficient method for extracting and representing the events in the trajectory data will achieve these goals, and whereupon, benefits the data mining tasks over trajectory data.

Symbolization is useful in mining spatial trajectory data. On the other hand, it can also help representing other data objects that seem different but are essentially with similar characters, such as features extracted from signature image and multiple attribute response curves in drug therapy.

Generally, symbolization can be viewed as an explaining process that classifies (or approximates) each individual "atomic" sub-trajectory into a *typical sub-trajectory* movements, e.g. "go north", "circumambulate", which is retrieved or defined beforehand. A simple and intuitive solution for generating the *typical sub-trajectory* is to choose it manually based on the domain expert's analysis and explanation. And then the "atomic" sub-trajectory at each time is represented by a simple nearest neighbor query through the given sets of *typical sub-trajectory*. Such ideas had been embedded into some ad hoc problem definitions, as well as data mining methods, implicitly or explicitly. However, it is usually extremely expensive to find and understand all candidate sub-trajectories in many real applications, whereupon it is very difficult, if not impossible to generate *typical sub-trajectories* by the above manual approach.

Another important aspect of mobile object trajectories is that the data are usually born with streaming property in many application domains. That is, the trajectories are frequently appended in the end over time. Since many factors that impact a moving object might be time varying, the patterns of the object movements are usually time varying, whereupon the *typical sub-trajectory* should also be adjusted correspondingly. In this case, it is usually difficult to apply a static approach from a practical point of view, where *typical sub-trajectories* are generated based on a snapshot of the trajectory, because it may fail to give a good representation of the new data. Alternatively, the *typical sub-trajectories* can be re-generated concurrently with each update. However, such re-generation will need all the data scanned, whereupon the time complexity might be extremely poor.

The above issues challenge the research on symbolizing trajectory data. Therefore, it is by no means trivial to consider this novel problem and develop effective and efficient methods for it. In this paper, the problem of symbolizing trajectory data is addressed. We first propose a symbolization method, in which *typical sub-trajectory* is generated automatically based on the data. For facilitating the data mining process on streaming trajectories, we also present an incremental symbolization method, which dynamically adjusts the *typical sub-trajectories* according to the most recent data characters, without scanning the whole date set. The performance of our approach is evaluated on both real trajectory data and synthetic data. Experimental results justify the effectiveness of the proposed methods and the superiority of the incremental approach.

2 Related Work

Data management and data mining on trajectory data had been studied in many applications [1, 2, 3, 4, 5]. Data mining applications can be found in [5, 6]. The most fundamental works in these contexts is on similarity measuring and indexing, e.g. [7, 8, 9, 10]. Related methods were developed mainly by extending the exist ones, e.g. Euclidean distance, Dynamic Time Warping (DTW), Longest Common Subsequence (LCSS), and multi-dimensional indexes. These works did not consider the symbolization process, but many ideas may serve as subroutines in the approach proposed in this paper.

In some previous work, unsupervised learning techniques were applied on the trajectories or similar data objects. For example, [11] presented an approach to clustering the experiences of an autonomous agent. Such methods were designed for purposes fundamentally different from ours, and were not further developed to fit the general data mining tasks.

Symbolization on time series data, as an important preprocessing subroutine, had been extensively studied for various data mining tasks [12,13,14], such as rule discovery, frequent pattern mining, prediction, and query by content, etc. A shape definition language was proposed in [15]. In [16], the time series was symbolized using cluster method for discovering rules with the format "if event A occurs, then event B occurs within time T." This method was then used in many applications that focus on mining time series [13,14]. [17] Claims that the method in [16] is meaningless if the step of sliding window is set to be 1. Actually, this problem could be solved by increasing the step of sliding windows. The above works deal with one-dimensional time series and the further extension for mining trajectory data have not been well considered. Our approach can be viewed as an expansion of the method introduced in [16] to facilitate the data mining process on trajectory data.

3 Problem Descriptions

A trajectory of a moving object is a sequence of consecutive locations in a multidimensional (generally two or three dimensional) space. In this paper, we only address the trajectories in 2D for clearness. A trajectory is denoted as: T=T(1),…,T(N) and $T(n)=(Tx(n), Ty(n))$ stands for the location of the moving object at time n. $|T|=N$ denotes the length of T. The projection of T in x-axis and that in y-axis are represented as $Tx = Tx(1),\ldots, Tx(N)$ and $Ty = Ty(1),\ldots, Ty(N)$ respectively. The sub-trajectory $T(m), T(m+1),\ldots,T(n)$ is denoted by $T[m,n]$.

As introduced in section 1, symbolization is a process to represent the object's behaviors at each individual time point. Then it is intuitive to first divide the trajectory to extract the sub-trajectories sequentially at various time points, and then to symbolize each extracted sub-trajectory individually by comparing it with a group of *typical sub-trajectories*. A *typical sub-trajectory* is a "prevalent" sub-trajectory that represents a typical form of sub-trajectory movements. Here we use the intuitive sliding window approach: Given trajectory T, window sub-trajectories of T are contiguous sub-trajectories $W_n=T[s(n),e(n)]$ or $(Wx_n, Wy_n)=(Tx[s(n),e(n)], Ty[s(n),e(n)])$ extracted sequentially by sliding the window through the trajectory, where $s(n)=nk\text{-}k+1$ and $e(n)=nk\text{-}k+w\text{-}1$ stand for the starting point and ending point of the n-th window respectively, n denotes the order number of the windows, parameter w controls the size of each window, and k controls the offset of positions of two consecutive windows.

Based on the above notions, the problem of symbolizing trajectory can be defined as follows: Given a trajectory T and all its window sub-trajectory W_n, symbolization is to convert T into a temporal sequence $S=(S(1),\ldots,S(M))$, which is a ordered list of symbols where each symbol $S(n)$ comes from a predefined alphabet Σ and represents the movements in the n-th sub-trajectory W_n.

4 Symbolization Approach

The key idea of our symbolizing approach is to cluster all the sub-trajectories, and then each sub-trajectory is represented by the identifier of the cluster that contains it. The overall approach is formally illustrated as follows:

1) Extract all sub-trajectories W_n in trajectory S (W_n is defined in section 3).
2) Normalize each sub-trajectory W_n to W_n'.
3) Cluster all W_n' into sets $C_1, \ldots, C_H$, that is, $W_n' \in C_{j(n)}$.
4) For each cluster C_h, a unique symbol a_h from Σ is inducted.

5) The symbol sequence S is obtained by looking for each W_n' the cluster $C_{j(n)}$, and using the corresponding symbol $a_{j(n)}$ to represent the sub-trajectory at that point, i.e. $S = a_{j(1)}, a_{j(2)}, \ldots, a_{j(M)}, M=\lfloor N/k \rfloor$.

Though the overall strategy seems similar with that for time series data, the symbolization process for spatial trajectory is different in several important aspects, e.g. normalization process, similarity measurement, and cluster process.

The normalization step is applied for precise representation by removing the impacts of the absolute location value and the scaling factors in both dimensions. For time series $X=(X(1),\ldots,X(N))$, normalization can be easily done by $X'=(X-\mathrm{E}(X))/\mathrm{D}(X)$ where $\mathrm{E}(X)$ is the mean of X and $\mathrm{D}(X)$ is the standard deviation of X. However, this problem is somewhat less straightforward for trajectory data, because the scale of different dimension may differ. For example, given a trajectory T, it may be unadvisable to simply use the above strategy on Tx and Ty respectively, since this process zooms the movements in both dimensions into exactly the same scale, whereupon the shape of the original trajectory may be demolished. We normalize a sub-trajectory $W_n= (Wx_n, Wy_n)$, as follows:

$$Wx'_n = \left(Wx_n - \mathrm{E}(Wx_n)\right) / \max\left(\mathrm{D}(Wx_n), \mathrm{D}(Wy_n)\right)$$
$$Wy'_n = \left(Wy_n - \mathrm{E}(Wy_n)\right) / \max\left(\mathrm{D}(Wx_n), \mathrm{D}(Wy_n)\right)$$

This normalizing process zooms a sub-trajectory into a cell with unit size without changing its shape. The difference of the above two normalization processes is demonstrated in Fig. 2.

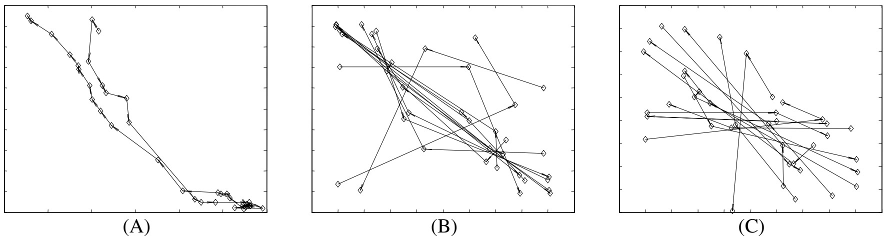

Fig. 2. Example trajectory (A) and the normalized sub-trajectories with the original shapes modified (B), and normalized sub-trajectories with the shape preserved (C).

In some applications, the information of interest is the relative movements that are irrelevant to its absolute directions, e.g. "go left" or "forward", rather than "go north" or "go along a meridian 45° counterclockwise from east." On such occasions, a rotating transform need to be applied based on the starting direction, i.e. each movement (represented by two consecutive locations) in a sub-trajectory is rotated to its relative direction to the last movement in the preceding sub-trajectory.

Clustering is the process of grouping a set of objects into classes of similar objects. Our symbolization approach has no constraint on the clustering algorithm, any common distance based clustering method could be used here, e.g. greedy method, recursive k-means, agglomerative and divisive hierarchical clustering, BIRCH, CUBE, Chameleon, etc. We use greedy method as the cluster subroutine in presenting our method and experiments, because the time complexity is sound, it can be easily implemented, and it does a good job in supporting the incremental symbolization process. For each W_n, greedy method first finds the cluster center q such that the distance between W_n and q is minimal. If the distance is less than a predefined threshold d_{max}, W_n is added to the cluster whose center is q and the cluster center of q is regenerated as the point-wise average of all the sub-trajectory contained in it, otherwise a new cluster with center W_n is created.

In the cluster process, Euclidian distance is used as the distance measurement. Given two trajectories A=(Ax(1),Ay(1)),…, (Ax(|A|),Ay(|A|)) and B=(Bx(1),By(1)),…, (Bx(|B|),By(|B|)) with the same length M, the Euclidian distance between A and B is defined as:

$$\mathrm{D}(A,B)=\sqrt{\sum_{i=1}^{M}\left((Ax(i)-Bx(i))^2+(Ay(i)-By(i))^2\right)}$$

Note that, other more sophisticated similarity measurements can also be applied in our method, such as *Dynamic Time Warping* (DTW) or *Longest Common Subsequence* (LCSS). DTW is defined as:

$$\mathrm{DTW}(A,B)=\mathrm{Lp}\big((Ax(|A|),Ay(|A|)),(Bx(|B|),By(|B|))\big)+\min\begin{Bmatrix}\mathrm{DTW}(\mathrm{Head}(A),B),\\ \mathrm{DTW}(A,\mathrm{Head}(B)),\\ \mathrm{DTW}(\mathrm{Head}(A),\mathrm{Head}(B))\end{Bmatrix}$$

where Lp stands for p-norm, i.e. Lp(V1,V2) =($\sum_k$|V1(K)- V2(K)|^p)$^{1/p}$ for vector V1 and V2, and Head (A) = (Ax(1),Ay(1)),…, (Ax(|A|-1),Ay(|A|-1)). LCSS is defined as follows: Given parameters δ and 0<ε<1,

$$\mathrm{LCSS}_{\delta,\varepsilon}(A,B)=\begin{cases}0 & A=\phi\vee B=\phi\\ 1+\mathrm{LCSS}_{\delta,\varepsilon}(\mathrm{Head}(A),\mathrm{Head}(B)) & |Ax(|A|)-Bx(|B|)|<\varepsilon\wedge\\ & |Ay(|A|)-By(|B|)|<\varepsilon\wedge\\ & ||A|-|B||<\delta\\ \max(\mathrm{LCSS}_{\delta,\varepsilon}(\mathrm{Head}(A),B),\mathrm{LCSS}_{\delta,\varepsilon}(A,\mathrm{Head}(B))) & \text{otherwise}\end{cases}$$

Another important problem is the symbol mapping model, i.e. Σ. The symbol set is usually defined by a group of simple symbols, e.g. the lower case alphabet a…z. Since our symbolization approach generates the *typical sub-trajectories* automatically, the meaning of each symbol in Σ cannot be explained or defined a priori. However, after the whole trajectory has been symbolized, the meaning of symbol $a_h\in\Sigma$ can be extracted by manual reviews of the corresponding cluster center of C_h. Therefore, if meaningful representations are more favorable, a simple post-processing procedure can be applied by introducing a new alphabet Σ' that is generated by first replacing each symbol a_h in Σ by a meaningful one based on the manual explanation of the cluster C_h, and then rewriting the resulting symbolic sequence based on Σ'.

5 Incremental Symbolization Approach

Streaming trajectory is a trajectory with new data items generated and appended in the end frequently. On such occasions, the incremental symbolizing process can be formalized as follows: Whenever a trajectory T is updated to TU (TU=T(1),…, T(N),U(1),…,U(K) is the direct connection of T=T(1),…,T(N) and U=U(1),…,U(K)), update the representing symbol sequence from S_T to S_TS_U, where S_T, S_U correspond to original trajectory T and the update U respectively. In this process, both *typical sub-trajectories* and symbol mapping model need to be dynamically updated with the data collections. Therefore, the static symbolization approaches are not applicable for streaming trajectories from a practical point of view as introduced in section 1.

The key idea of our dynamic symbolizing approach is that instead of using the static cluster centers as the *typical sub-trajectories*, we dynamically maintain the cluster information, and then the symbol sequence of new sub-trajectories is generated based on the up-to-date version of *typical sub-trajectories*.

Assume that, the method introduced in section 4 has already been applied on an initial trajectory, and the resulting cluster centers are saved as the initial *typical sub-trajectories*. Then, whenever the trajectory is updated, the cluster information is updated by introducing the new sub-trajectories and removing the "old" sub-trajectories with the generation time t such that $t<t_{now}-t_{max}$ where t_{now} denotes the current time and t_{max} is a predefined threshold *maximal time range*. Here it is assumed that the *typical sub-trajectories* will evolve with the updating process, whereupon the sub-trajectories that are generated too long ago will have minor effect on representing the current sub-trajectories.

The detailed method is illustrated as follows: First, extract all sub-trajectories W_n' contained in clusters and with the generation time $t<t_{now}-t_{max}$. Remove each W_n' from the corresponding clusters $C_{j(n)}$, and the cluster center $q_{j(n)}$ is re-computed as the point-wise average of all the sub-trajectories remained in $C_{j(n)}$. If $C_{j(n)}$ becomes empty after the deletion, remove it. Then extract and normalize all sub-trajectories contained in U, let the resulting sub-trajectories be W_m'. For each W_m', add it to a cluster $C_{j(m)}$ and generate symbol $a_{j(m)}$ for W_m'. The method used in this step is same as that introduced in section 4.

When a trajectory T is updated to TU, only the sub-trajectories in update part U or the "old" ones need to be considered. Then the total number of affected sub-trajectories is roughly O($|U|+|U|$). Each new sub-trajectory can be inserted into the proper cluster in O(H) time where H is the number of clusters that is depended on predefined threshold d_{max}. And the update of *typical sub-trajectories* can be simply done by a weighted sum of the original *typical sub-trajectories* and the affected sub-trajectories. Therefore, the overall time complexity of the incremental symbolization approach is O($H|U|+|U|$). This time complexity, in our opinion, can meet the requirements of real applications.

6 Experimental Results

In this section, we present an empirical study of the proposed methods. The objectives of this study are: 1) to evaluate the effectiveness of the proposed method, and 2) to compare the performance of incremental symbolization method with the static one. In the experiments, two sets of data were used:

Real Data: The real data were a combination trajectory of a group of animal movements collected by satellite tracking. The whole trajectory consists of 238 locations (i.e. data points), each of which indicates the longitude and latitude of the observed object at a time point.

Synthetic Data: The real data is relatively small. To evaluate our approach on varying size data, we used synthetic data that were generated as $T(n)=L(n)+Rg$, where L was a trajectory that was generated based on the real data introduced above, but with varying size, each $L(n)$ was randomly selected around the relative position in the real trajectory. g was Gaussian noise with zero mean and unit variance. And parameter R controlled the noise-to-signal ratio.

The performance of a symbolization approach was evaluated by the average distance between original sub-trajectories and the corresponding *typical sub-trajectories*. That is, a representation that is more similar to the original sub-trajectory is a better symbolization result. Given the trajectory T, symbolization results is $S=a_{j(1)}, a_{j(2)}, \dots, a_{j(M)}$, then the performance of this symbolization process was evaluated as:

$$P(S)=\frac{1}{|S|}\sum_{m=1}^{|S|} D\left(C_{j(m)}, W'_m\right)$$

where D is Euclidian distance, W_n' stands for the normalized sub-trajectories of T, and C stands for the cluster centers (detailed definitions can be found in section 4). It should be noted that the less the measuring result, the better the performance.

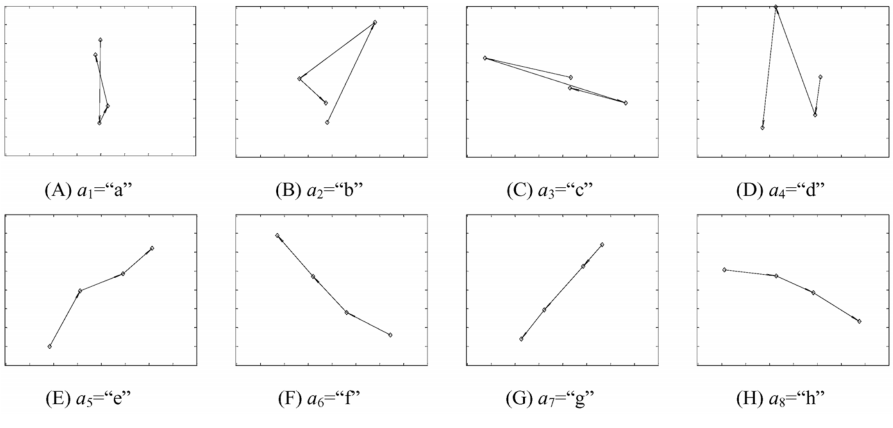

Fig. 3. *Typical sub-trajectories* and corresponding symbol representations, generated by static symbolization approach, on real data set.

Fig. 3 shows the experimental results on real data, which includes all the resulting *typical sub-trajectories* generated by the static approach. In the experiments, the width *w* and the step *k* of the sliding window were set to be 4 and 3 respectively, because this setting (k=w-1) can help fully avoiding the correlation between consecutive sub-trajectories without losing any information. The results indicated that there were 8 main typical movements in the trajectory, in which (E) - (H) represented straight motions roughly, (A), (B) represented round-motions in certain directions, and (C), (D) represented zigzag motions in two directions respectively.

By representing each sub-trajectory with a symbol indicating the corresponding *typical sub-trajectory*, the symbolization process gave us a concise and meaningful representation of the trajectory data, whereupon a data mining problem could be easily solved by simply applying an off the shelf symbol-sequence-oriented data mining approach. Recall from section 1 that such approaches can be easily found in many application domains. Since each symbol in the symbolic sequence represents the object's behaviors at an individual time point, the emphasis of this data mining process is on the high level representation of the data, rather than the quantitive values. In addition, the borrowed data mining method can be applied directly without any modification required. For example, we could apply a pattern mining method to discover frequent patterns with the form "a period of round-motions was always followed by a direct motion to the north-east in two hours."

In another group of experiments, synthetic data with various sizes and various amounts of noise were generated respectively for performance comparisons, and then both static symbolization approach and incremental one were used on the generated data alternatively. During these experiments, the window width and window step were also set to be 4 and 3 respectively. Since the static approach need all the data involved for re-computation, whereas in the incremental approach only the updated sub-trajectories need to be examined, the efficiency issue for the two approaches is quite obvious. Therefore, here we only give the results on effectiveness comparisons by considering the performance measure introduced above.

Fig. 4 shows the performances of the two approaches on trajectories with various lengths. By visual analysis, we could find that the incremental approach outperformed the static approach. This is partly because the static approach grouped many sub-trajectories into relatively incompact clusters, represented by out-dated *typical sub-trajectories*.

Fig. 5 shows the performances of the two approaches on trajectories with various amounts of noise involved, generated by varying the parameter *R*. This group of results also justifies the superiority of the incremental approach over the static one. In addition, the improvement in performance of the incremental approach decreased with the increase of the noise. Finally, the performances of the two approaches tended to be same. This is because the potential movement patterns implied by the original real data were influenced by the noise. When the amplitude of the noise was set to be large enough, the noise flooded the original real data, whereupon the generated data became a completely random one. For fully random data, both the static and the incremental approaches will perform like a meaningless random selection.

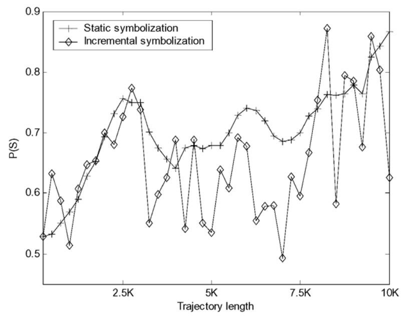

Fig. 4. Comparisons between the static and incremental symbolization approaches on trajectories with various lengths.

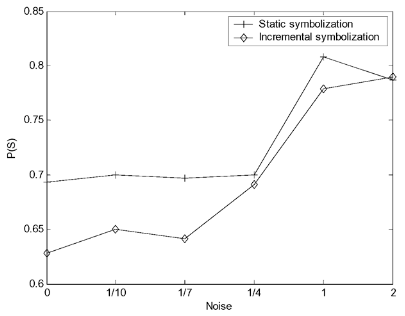

Fig. 5. Comparisons between the static and incremental symbolization approaches on trajectories with various amounts of noises.

7 Conclusions

In this paper, the problem of symbolizing trajectory data is addressed. We first introduce a static symbolization method, in which *typical sub-trajectories* are generated automatically based on the data. For facilitating the data mining process on streaming trajectories, we also present an incremental symbolization method, which dynamically adjusts the *typical sub-trajectories* to fit the up-to-date trajectory characters.

The performances of our approaches were evaluated on both real data and synthetic data. Experimental results justify the effectiveness of the proposed methods and show that the incremental approach outperformed the static one on trajectories with various lengths and various amounts of noise.

In future, we intend to generalize our symbolization approaches by introducing other similarity measurements and cluster models that are more sophisticated.

Acknowledgements

The work was partly supported by the NSFC 60373011 and the 973 Program 2002CB312000. We thank the anonymous reviewers for the valuable comments.

References

1. N. Priyantha, A. Miu, H. Balakrishnan, S. Teller. The cricket compass for context-aware mobile applications. In MOBICOM2001 Conference Proceedings, pages 1–14, 2001.
2. G. Chen, D. Kotz. Categorizing binary topological relations between regions, lines, and points in geographic databases. Technical Report TR2000-381. Dept. of Computer Science, Dartmouth College, 2000.
3. M. Vlachos, G. Kollios, Dimitrios Gunopulos. Discovering Similar Multidimensional Trajectories. In Proc. of the 18th International Conference on Data Engineering (ICDE'02). San Jose, California, 2002.
4. Y. Yanagisawa, J. Akahani, T. Satoh. Shape-based Similarity Query for Trajectory of Mobile Objects. In Proc. of the 4th International Conference on Mobile Data Management, pages 63 - 77. 2003.

5. C. S. Smyth. Mining mobile trajectories. H. J. Miller and J. Han (eds.) Geographic Data Mining and Knowledge Discovery, London: Taylor and Francis, 337-361. 2001.
6. G. Kollios, S. Sclaroff, M. Betke. Motion Mining: Discovering Spatio-Temporal Patterns in Databases of Human Motion. Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD 2001, Santa Barbara, CA, May 2001.
7. P. K. Agarwal, L. Arge, J. Erickson. Indexing moving points. In Proc. of the 19th ACM Symp. on Principles of Database Systems (PODS), pages 175–186, 2000.
8. S. Saltenis, C. Jensen, S. Leutenegger, and M. A. Lopez. Indexing the Positions of Continuously Moving Objects. In Proceedings of the ACM SIGMOD, pages 331–342, May 2000.
9. D. Pfoser, C. Jensen, and Y. Theodoridis. Novel Approaches in Query Processing for Moving Objects. In Proceedings of VLDB, Cairo Egypt, Sept. 2000.
10. M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, E. Keogh. Indexing Multi-Dimensional Time-Series with Support for Multiple Distance Measures. In proc. of the 9th International Conference on Knowledge Discovery and Data Mining (KDD 2003). 2003.
11. T. Oates, M. Schmill, P. Cohen. A Method for Clustering the Experiences of a MobileRobot that Accords with Human Judgments. In Proc. of AAAI 2000.
12. Y. Zhu, D. Shasha. Fast approaches to simple problems in financial time series streams, Workshop on management and processing of data streams. 2003.
13. Z. Yao, L. Gao, X. S. Wang: Using triangle inequality to efficiently process continuous queries on high-dimensional streaming time series. In proc. of SSDBM 2003. 2003.
14. X. Jin, Y. Lu, C. Shi. Distribution discovery: local analysis of temporal rules. In proc. of the 6th Pacific-Asia Conf. on knowledge discovery and data mining (PAKDD 2002). 2002.
15. R. Agrawal, G. Psaila, E. Wimmers, M. Zaot. Querying shapes of histories. In proc. of the 21st international conference on very large database (VLDB'95). 1995.
16. G. Das, K. Lin, H. Mannila, G. Renganathan, P. Smyth. Rule discovery from time series. In proc. of the 4th International Conference on Knowledge Discovery and Data Mining (KDD 1998). 1998.
17. E. Keogh, J. Lin, W. Truppel. Clustering of time series subsequences is meaningless. In proc. of ICDM 2003. 2003.