Classification of Multi-Dimensional Trajectories Using Textual Approximation

Huy Xuan Do, Hung-Hsuan Huang, and Kyoji Kawagoe

***Abstract*—Classification and searching time series in the multidimensional space is a big challenge. There has been much research work on this field for many years. TAX (Textual ApproXimation) is a method for searching time series data such as stock or electrocardiogram data. The main idea behind TAX is extract a set of temporal-terms from time series data to approximate using document retrieval methods. The main problem of applying TAX to multidimensional data, such as moving object trajectory data is that how we extract temporal-terms from multidimensional data. In this paper, we propose a novel method to obtain temporal-terms by decomposing original multidimensional data into smaller segmentations, deploying a clustering method to group those segmentations into groups, and assigning a temporal-term for each group. Our research focuses on two-dimensional moving object, representing trajectories and on classification of a large set of the moving object trajectories. The experimental results confirm that our proposed method is effective in multi-multidimensional data classification.**

***Index Terms*—Trajectory clustering, classification, clustering framework.**

I. Introduction

Recently, the analysis and research on moving objects have becoming one of the most important technologies to be used in various applications and fields such as GIS, navigation systems and location based information systems. Trajectory mining technologies lead to various types of applications. For instance a commercial system in sport can track balls, players and referees in real time [1], and can analyze them to find better game strategies and to predict a future moving trajectory.

In this paper, we propose a moving object trajectory classification method which employs the idea from TAX [2].The main idea of our method is that given a set of trajectories, each trajectory is transformed into a sequence of temporal terms and the transformed trajectories are clustered using the temporal term sequence for trajectory classification. We first segment each trajectories into many sub-trajectories. Using a clustering method, we are able to automatically classify those sub-trajectories into groups, which each group represents similar moving pattern. We call those groups T-Terms. Using T-Term, each trajectory is constructed into a sequence of T-Terms, call T-doc. As T-doc has a document-like structure, we are able to classify T-docs data using one of the traditional document classification methods.

Fig. 1 gives you an example of how this textual approximation technique works with trajectory data. Fig. 1 shows two trajectories which are represented as red curves: *Tr1* and *Tr2*, each of which has its own moving pattern. Using our method, we are able to symbolize *Tr1* as: $\{A_1, A_2, A_3, A_4\}$, *Tr2* as $\{A_1, A_5, A_6\}$. Each A_i is a T-term, representing a local moving pattern. For example, A_1 representing a pattern which can be explained like: moving in the NE direction first, then rotating clockwise a bit. As indicating as this example, using T-Term, we are able to represent the original set of trajectories into a set of T-Term sequences, call T-doc $Tdoc_k = \{A_{k,1}, A_{k,2}, .., A_{k,kn}\}$. Clustering T-docs data set is much easier than clustering trajectories data, because we just need to cluster 1-dimension data, and a similarity between T-docs is much simple to calculate.

In this paper, we propose a method of classification of moving object trajectories based on textual approximation. Examples of moving object trajectories include car movements on roads, movements of balls and players in sports, and tracks of typhoons or hurricanes.

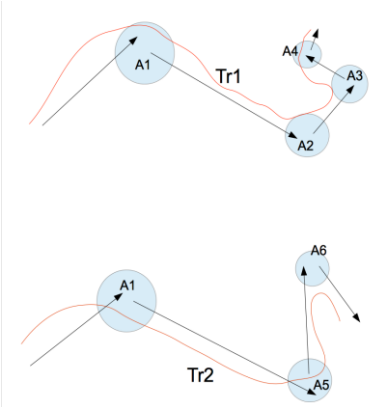


Fig. 1. Textual approximation example.

II. Related Work

There are many studies focus on trajectory pattern mining. Yi *et al*. use the traditional technique called DTW (Dynamic Time Warping) which was originally used in speech recognition, to provide robust and generic method to query similar trajectories [3]. DTW is easy to implement and compute. Another research in using DTW to create envelope queries to index trajectory was carried by Let *et al*. [4]. Agrawal *et al*. use Discrete Fourier Transform (DFT) to transform trajectories to multi-dimensional points, then using Euclidean distance to calculate similarity between points to find similar trajectories [5].

Another approach is to find patterns in trajectories, by analysing spatial constraints of the object's lifeline in

Manuscript received October 4, 2013; revised January 23, 2014.

The authors are with the Ritsumeikan University, Japan (e-mail: doxuanhuy@gmail.com).

geographic space. Laube *et al*. proposed a geographic data mining approach to detect generic aggregation patterns such as flocking behaviour and convergence in geo-spatial lifeline data [6]. One of the approaches similar to their system is to use and find similar sub-trajectories instead of the whole. In an attempt to develop more robust measures, Lee *et al*. use some customized distance function and DBSCAN-like algorithm to cluster group of similar sub trajectories [7]. Gudmudsson *et al*. use pre-defined coarse terms called Motifs to describe patterns [8]. A motif like "first move straight then start a long curve to the right" is described. Buchin *et al*. proposed an algorithms for computing the most similar sub-trajectories under the measure of their average distance at corresponding time, assuming the two trajectories are given as two polygonal, possibly self-intersecting lines [9]. This approach was explored further by Buchin *et al*. [10], provide linear-time algorithm with better performance. Some researchers focus on searching/querying trajectories. Chen *et al.* proposed an algorithm to find the *k Best Connected Trajectories* (*k*-BCT), query for searching trajectories by multiple geographical locations [11]. Generally, the *k*-BCT query is a set of locations indicated by coordinates like {*latitude, longitude*}, with those type of queries, a nameless beach or any arbitrary place approximated by the centre location.

As we see in the existing approaching, they focus on querying trajectory, or cluster segments of trajectories data in a small preriod of time, to find region of interest. However, a generic method to cluster trajectories which have common patterns of moving object trajectories is still lacking.

III. A Method of Trajectory Classification Based on Textual Approximation

A. Motivation and Approach

To highlight the main idea of our method, Fig. 2 shows four sample trajectories *TR1*, *TR2*, *TR3*, *TR4*. It can be easily seen that *TR1* and *TR2* have very similar patterns of moving, as well as *TR3* and *TR4*. So our expect result after implement our method to classify those trajectories, is that we will have 2 groups: Group1 and Group2. Group1 includes {*TR1, TR2*}, and Group 2 includes {*TR3, TR4*}. Specifically stating, a new trajectory can be classified in one of the two groups after they are constructed to be clustered. To realize this problem, our method first extracts local features of each trajectory and represents similar features as the same word called T-Term. For example, the pattern "move down 45 degree then move up 45 degree" is represented as T-Term *A1* as shown in Fig. 2. Consequenty, these four sample trajectories are represented as a document-like structure called T-Doc as follow:

1) TR1: {A1, A2}
2) *TR2* : {*A1*, *A2*}
3) *TR3* : {*A3*}
4) *TR4* : {*A3*}

To compare similarity between T-doc data, we use Longest Common Subsequence distance (the LCS distance). Using this LCS distance, it's easy to see that in above example, *TR1* and *TR2* belong to the same cluster, as well as *TR3* and *TR4* belong to the same cluster.

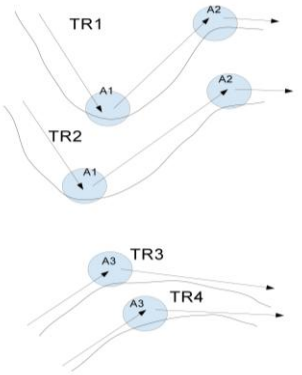


Fig. 2. Sample of textual approximation.

To implement our method, we need to realize the following functions:

1) How to extract and symbolize feature points
2) How to construct and cluster T-docs data

To realize these functions, we divide our proposed method into the following phases as shown in Fig. 3:

Phase 1: T-Terms collection phase: To extract feature points and find similar features, we first use an algorithm used in polygonal curves approximation technique to remove unnecessary points (trajectory simplification). Trajectory simplification step is needed because the original data normally contains many noises, which make it very difficult to find global feature points. After simplification process, we consider the remaining points as "feature points". To represent those points, a feature point representation data is constructed from three contiguous points around the feature point called sub-trajectory. We then collect all sub-trajectories and use a clustering technique to cluster those sub-trajectories data set into groups which sub-trajectories in the same group have a similar moving pattern. Each group is considered as a T-Term. We store those T-Terms into the database to use in the next phase.

Phase 2: T-Docs classification phase: After collecting T-Terms, we then build T-Docs by approximating the original trajectory data by using T-Term database. After approximating original trajectories into T-Docs, we classify those T-Docs using a DBScan [12] like algorithm which uses LCS as a distance function

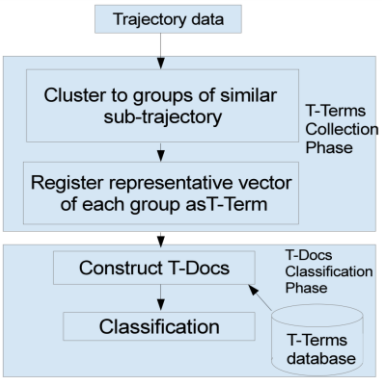


Fig. 3. Our method flow.

B. T-Terms Collection Phase

1) Trajectory simplification

Raw trajectory data normally contains a lot of noises, which leads to some unexpected result when we attempt to

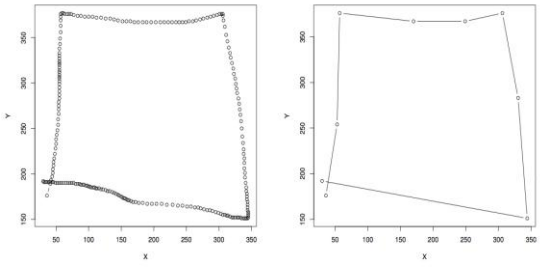
extract feature points such as falling into local-minimum or local-maximum. There are many existing methods to simplify trajectory data based on traditional techniques to approximate polygonal curves with minimum number of line segments or minimum error such as (min-# algorithm) [13], or Douglas-Peucker algorithm (DP) [14].

The min-# algorithm can produce a good result and also can simplify curves without loss too much data. However its computation cost is up to $O(N^2)$which is quite hard to apply in real-life applications. Therefore we use Douglas-Peucker algorithm to simplify trajectories. The main idea of Douglas-Peucker is, given a polyline specified by a sequence of *N* points $\langle p_1, p_2, .., p_N \rangle$ and a distance threshold δ, we can derive a new polyline with fewer points without deviating from the original polyline by at most δ. Douglas-Peucker algorithm initially constructs the line segment $\overline{p_1 p_T}$, then identifies the point p_i, the farthest point from the line. If this point perpendicular distance to the line is within δ, then the algorihm returns $\overline{p_1 p_N}$ and terminates. Otherwise, the line is splitter at p_i and the algorithm is recursively applied to the sub polylines $\langle p_1, p_2, .., p_i \rangle$ and $\langle p_i, p_{i+1}, .., p_N \rangle$.

To get the most significant δ value, the value of δ is changed as 2%, 5%, 10%, 20% value of the overall distance of the trajectory. From this experiment we found that 5% of the overall trajectory length can keep the most significant points of the original data.

$$\delta = \frac{5}{100} \times (\sum_{i=1}^{N-1} L_2(p_{i+1}, p_i))$$

$L_2(p_{i+1}, p_i)$is Euclide distance between two points p_{i+1} and p_i. Result of Douglas-Peucker is shown in Fig. 4. From this figure, it is easily seen that Douglas-Peucker algorithm is very effective in reducing noise. Douglas-Peucker is able to reduce up to over 70% points which are considered as noises.



(a). original data (b). data after processing

Fig. 4. Douglas-Peucker algorithm result sample.

2) Feature point representation

After pre-processing trajectory data with Douglas-Peucker algorithm, the remaining points are considered as "feature points". For each point, a sub-trajectory of three contiguous points around itself is obtained, as shown in Fig. 5. We want each features not to depend on spatial position, so that we normalize this trajectory to its root which means that we ignore time information from the original data. Therefore, given a feature point $\langle x_i, y_i \rangle$, we will get the feature matrix of this point as below

$$\begin{pmatrix} 0 & x_i - x_{i-1} & x_{i+1} - x_{i-1} \\ 0 & y_i - y_{i-1} & y_{i+1} - y_{i-1} \end{pmatrix}$$

The above matrix is hard to collect and do clustering to segment, so we transform this matrix into a vector form

$$\langle 0, x_i - x_{i-1}, x_{i+1} - x_{i-1}, 0, y_i - y_{i-1}, y_{i+1} - y_{i-1} \rangle$$

Our research focuses on clustering trajectory by its spatial characteristic. Ignoring time *t* parameter does not affect clustering result. The original trajectory data is represented as *6*n* matrix as below (n is number of feature points)

$$\begin{pmatrix} 0 & x_2 - x_1 & x_3 - x_1 & 0 & y_2 - y_1 & y_3 - y_1 \\ 0 & x_3 - x_2 & x_4 - x_2 & 0 & y_3 - y_2 & y_4 - y_2 \\ ... & ... & ... & ... & ... & ... \\ 0 & x_n - x_{n-1} & x_{n+1} - x_{n-1} & 0 & y_n - y_{n-1} & y_{n+1} - y_{n-1} \end{pmatrix}$$

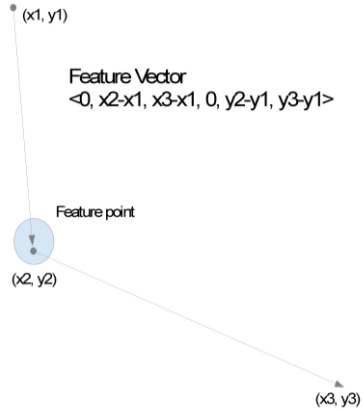


Fig. 5. Feature point representation.

3) Trajectory clustering

There are many existing clustering methods, among which, the most famous one is K-means. The K-means algorithm purpose is to minimize a criterion of the form $\sum_i d(x_i, p_{c(i)})$which $d(x_i, p_{c(i)})$is normally the Euclidean distance or in some case the Mahalanobis distance between data-point x_i and the cluster to which it belongs $p_{c(i)}$. K-means algorithm gives you a result where sub-trajectories similar in distance, are grouped into the same cluster. As this is not expected, we construct each group to contain sub-trajectories which have similar pattern of moving.One of famous extension of K-means algorithm is SKmeans algorithm (Spherical K-means) [15]. SKmeans algorithm uses cosine similarity. Using cosine similarity means that we are able to cluster sub-trajectories which have similar shape. Therefore, the proposed method employed SKmeans as sub-trajectories clustering algorithm.

The standard SKmeans problem is to minimize $\sum_i (1 - \cos(x_i, p_{c(i)}))$. Given *N* rows matrix trajectory, and *K* fixed number of centroids, the SKmeans algorithm is described in Algorithm 1. We set the stopping criteria as no more cluster assignments change. To fix number *K* of cluster, we focus on the variation of a sub-trajectory. Each of feature vector constructed from 2 vectors connect 3 contiguous points.

Therefore, there are 2 points which are able to vary in 4 main directions (west, north, south, east). Therefore, we will have up to 4×4=16 degree of freedom. Accordingly, we chose K=16 as number of centroids to cluster.

After clustering similar sub-trajectories into groups, we name each group $W_1, W_2, \ldots, W_{16}$ as T-Terms. A representative feature of each group is a concept vector obtained in SKmeans algorithm. We then store obtained concept vectors representing groups into database (T-Terms database) to use in the classification phase. Those concept vectors are periodically recalculated to update with newest data.

Algorithm 1. Skmeans with trajectory data.
Data: Trajectory matrix
Result: Cluster vectors, and vector assignment information (which cluster the vector belong)

1. Initialize clustering. Start with some initial partitioning of the vector, namely $\{\pi_j^{(0)}\}_{j=1}^{N}$. Let $\{c_j^{(0)}\}_{j=1}^{N}$ be the concept vectors of the associated partitioning. Set iteration count t
2. $t \longleftarrow 0$
3. **while** Stopping criterion is not met **do**
4. **for** Each row p_i of trajectory matrix **do**
5. Compute $p_i^T * c_l^{(t)}$ for all $l = 1,2,\ldots,K$;
6. From all $p_i^T * c_l^{(t)}$ computed above, find j = argmax($p_i^T * c_l^{(t)}$) (break ties arbitrary if x_i has largest cosine similarity with more than one concept vector);
7. Induce new partitioning $\pi_j^{(t+1)} = \{p_i : j = \arg\max(p_i^T * c_l^{(t)})\}, 1 \le j \le K$
8. Update concept vectors: $s_j = \sum_{x_i \in \pi_j} x_i$, $c_j^{(t+1)} = \frac{s_j}{\| s_j \|}, 1 \le j \le K$
9. **end**
10. $t \longleftarrow t+1$
11. **end**

C. T-Docs Classification Phase

Using T-Terms, we are able to construct each original trajectory data into a sequence of T-term (W_i), called T-Doc To classify T-Docs data, we carried out an DBScan type clustering algorithm with LCS (Longest common subsequence) as distance function. We need to redefine original "Reachable" definition as in Definition 1.

Definition 1. two T-Docs L_1 and L_2 is density reachable if LCS distance between L_1 and L_2 is greater than

$$\frac{\min(length(L_1), length(L_2))}{2}$$

We proposed an algorithm which employ original idea from DB-Scan algorithm. Our proposed algorithm main idea is that, given a T-Doc as a seed, we then find all around T-Docs which are reachable from the seed, then assign those T-Docs to the cluster which is initialized by the seed. The algorithm is highlighted at Algorithm 2.

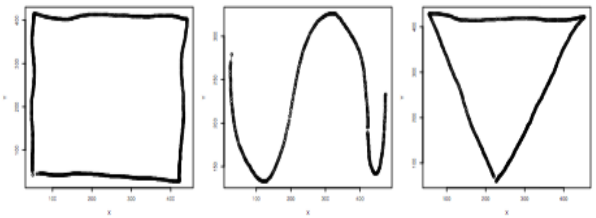
IV. Evaluation

We conducted some experiments to evaluate our method with an artificial trajectory test data set. We used WACOM BAMBOO Pen table to generate by drawing three types of main patterns: triangle, rectangle and wave. This trajectory data set contains the similar characteristics as real-life trajectories which contain noises and variations of sub-trajectories. We asked a group of 10 people to draw

Algorithm 2. Clustering T-doc algorithm.
Data: T-docs data
Result: Cluster of T-dcs

1. **for** Each unvisited T-doc T **do**
2. **if** T not belongs to any cluster **then**
3. init cluster C_T with T;
4. $Seed \longleftarrow T$;
5. $type\_of(C_T) \longleftarrow type\_of(Seed)$;
6. **for** Each unvisited T-Doc T' except Seed **do**
7. **if** Reachable(T', Seed) **then**
8. add T' to C_T
9. **end**
10. **end**
11. **else**
12. Next;
13. **end**
14. **end**

Those three patterns with both right hand and left hand. Then, the test dataset are composed of 60 trajectories. Fig.6 shows some examples of our test data set.



(a). Regtangle pattern (b). Wave pattern (c). Rriangle pattern

Fig. 6. Sample input data.

We applied our proposed method to classify this set of trajectories into groups that represent different patterns of trajectories. Fig. 8, Fig. 9 and Fig. 10 shows three examples of obtained clusters. The red bold line curve shows representative of three clusters each of which corresponds to one T-term. Although there were some minor noises but it can be clearly seen that each T-Term represents a similar pattern of sub-trajectories. Using T-Term databases, we can build a T-Doc from original trajectories. For example, a wave pattern input data shown in Fig. 6(b) was represented in T-Doc form as $\langle W_{16}, W_{16}, W_{13}, W_3, W_2, W_4, W_{16}, W_{15}, W_8, W_3, W_7 \rangle$ as shown in Fig. 7.

We carried out an classification algorithm with Algorithm 2 and obtained the result of 7 clusters. Two clusters represent the triangle pattern, two clusters represent the rectangle pattern, and three clusters represent the wave pattern. A T-Doc can be correctly clustered as far as it is similar to one of these three patterns by our method. To evaluate, we calculate the average precision as bellow:

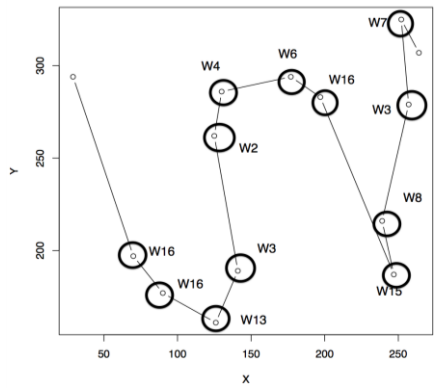


Fig. 7. Textual Approximation of wave pattern.

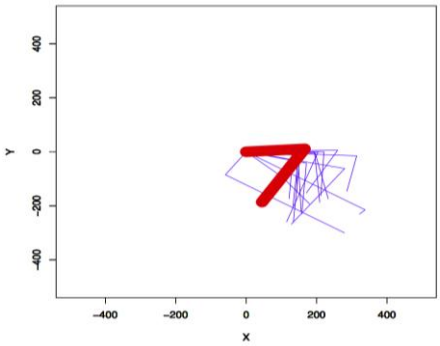


Fig. 8. Sub-trajactories cluster result 1.

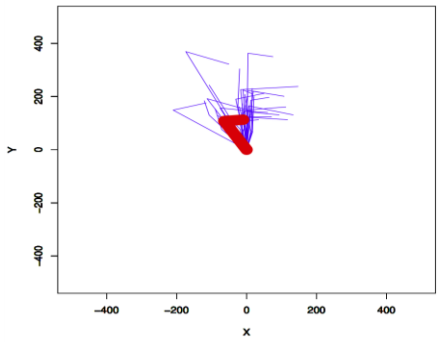


Fig. 9. Sub-trajactories cluster result 2.

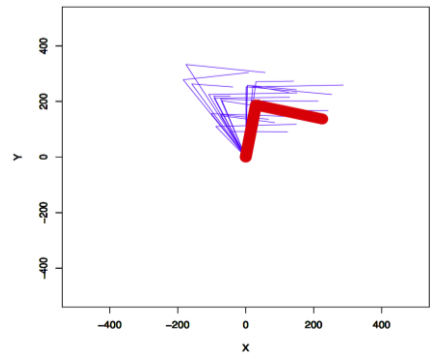


Fig. 10. Sub-trajactories cluster result 3.

$$precision = \frac{correct\_set}{total\_set} \times 100\%$$

correct_set is the number of correct clustered T-Docs, and *total_set* is the number of total data set. The result is accurate with overall 84% precision. We intend to do more detailed evaluation, by comparing our method with other existing methods.

V. Conclusion

In this paper, we propose a new generic textual approximation method for moving object trajectories. Our method is accurate from our preliminary experiment. Remaining issues include: (1) application development based on our method to show its effectiveness in real-life data comparing with existing methods and (2) derivation of some constraints in it.

Acknowledgment

This work is partially supported by KAKENHI \#24300039 and also by MEXT-Supported Program for the Strategic Research Foundation at Private Universities, 2013-2017.F.

References

[1] Y. Zhang *et al*., "Discovering tactics in broadcast sports video with trajectories," in *Proc. ICIMCS '09*, New York, USA, 2009, pp. 170–173.

[2] A. A. Maruf *et al.*, "Time series classification method based on longest common subsequence and textual approximation," in *Proc. ICDIM*, 2012, pp. 130–137.

[3] B. K. Yi, H. V. Jagadish, and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," in *Proc. the Fourteenth International Conference on Data Engineering*, Washington D.C., USA, 1998, pp. 201–208.

[4] C. Lei *et al*., "Robust and fast similarity search for moving object trajectories," in *Proc. SIGMOD '05*, New York, USA, 2005, pp. 491–502.

[5] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient similarity search in sequence databases," in *Proc. the 4th International Conference on Foundations of Data Organization and Algorithms*, London, UK, 1993, pp. 69–84.

[6] P. Laube, M. V. Kreveld, and S. Imfeld, "Finding remo detecting relative motion patterns in geospatial lifelines," presented at 11th Int. Symp. on Spatial Data Handling, 2004.

[7] J. G. Lee *et al.*, "Trajectory clustering: a partition-and-group frame-work," in *Proc. SIGMOD '07*, New York, USA, 2007, pp. 593–604.

[8] G. Joachim *et al*., "Of motifs and goals: mining trajectory data," in *Proc. SIGSPATIAL '12*, New York, USA, 2012, pp. 129–138.

[9] M. V. Kreveld and J. Luo, "The definition and computation of trajectory and subtrajectory similarity," in *Proc. the 15th Annual ACM International Symposium on Advances in Geographic Information Systems*, New York, USA, 2007, pp. 44:1–44:4.

[10] K. Buchin, M. Buchin, M. V. Kreveld, and J. Luo, "Finding long and similar parts of trajectories," in *Proc. the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, New York, USA, 2009, pp. 296–305.

[11] Z. B. Chen, H. T. Shen, X. F. Zhou, Y. Zheng, and X. Xie, "Searching trajectories by locations: an efficiency study," in *Proc. the 2010 ACM SIGMOD International Conference on Management of Data*, New York, USA, 2010, pp. 255–266.

[12] M. Ester, H. P. Kriegel, S. Jrg, and X. W. Xu, "A density- based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD 1996*, Portland, USA,1996, pp. 226–231.

[13] D. Z. Chen *et al*., "Space-efficient algorithms for approximating polygonal curves in two dimensional space," in *Proc. COCOON '98*, London, UK, 1998, pp. 45– 54.

[14] D. D. H. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *The Canadian Cartographer*, vol. 10, no. 2, pp. 112–122, 1973.

[15] I. Dhillon and D. Modha. *Concept Decompositions for Large Sparse Text Data Using Clustering*, 2001.

Huy Xuan Do was born in HaNoi, Viet Nam, on September 1st, 1988. He is a second year master student at Ritsumeikan University now. He received BSc. in computer science from Ritsumeikan University in 2011. His research interests are time series data mining algorithm, text mining algorithm.

Hung-Hsuan Huang received BSc. in computer science from National Chen-Chi University, Taiwan in 1998 and MSc. from National Taiwan University, Taiwan. He received his PhD from the Kyoto University in 2009. Currently he is a professor at the Ritsumeikan University, Japan. He has research interest in embodied conversational agent and virtual 3D space. He is a member of JSAI, IPSJ, TAAI, HIS, ACM and IEICE.

Kyoji Kawagoe received B.Eng. and M.Eng in electronic engineering from Osaka University in 1975 and 1977, respectively. He also received Ph.D from Tsukuba University in 1992. He joined Ritsumeikan University in 1997, while he had worked for NEC Corporation since 1977. He is currently a full professor of Collage of Information Science and Engineering, Ritsumeikan University. His research interests include multimedia data engineering, ubiquitous computing and network related software research and development. He is a member of IEEE, ACM, ACM SIGMOD, Database Society of Japan, IEICE, and IPSJ.