Detecting Road Intersections from Coarse-gained GPS Traces Based on Clustering

Junwei Wu[1,2] , Yunlong Zhu[1] , Tao Ku[1] and Liang Wang[1,2]
[1]Shenyang Institute of Automation, Chinese Academy of Sciences
Shenyang 110016, China
[2]University of Chinese Academy of Sciences
Beijing 100049, China
Email: {wujunwei, ylzhu, kutao, wangliang1}@sia.cn

***Abstract*—With more and more vehicles equipped with GPS tracking devices, there is increasing interest in building and updating maps using vehicular GPS traces. But commodity GPS devices have lower accuracy and lower sampling frequency, which made it more difficult to infer road network than most existing approaches that using high-precision and high-frequency GPS devices. As a key component of road network, intersection plays the role of transport hub. So, if the intersections are detected in advance, the road network can be then constructed conveniently by connecting the intersections. In this paper, we propose a novel algorithm for recognizing intersections with coarse-grained GPS traces based on data preprocessing and clustering. The algorithm first prune low quality GPS points, then find out the turning points around intersections and the converging points in the preprocessing step, and finally cluster these converging points to find out the cluster centers, i.e. the intersection positions. In addition, we introduce a simple road network construction algorithm based on the identified intersections. We evaluate our method using GPS data gathered from 2,827 taxis in Shenyang, Liaoning, China. Evaluation results demonstrate that our algorithm is able to find most of the road intersections effectively.**

***Index Terms*—GPS traces, road network, intersection, clustering**

I. INTRODUCTION

With the mature of digital map technology, the digital map services are being more and more widely used in our daily life, such as path search, vehicle navigation, and so on. One of the most important steps in the digital map production is the road network construction. Traditionally, there are two construction methods, the first is to extract road network from high-precision satellite pictures or aerial images, and the second is to create road network by road survey using specialized vehicles equipped with GPS equipment. Both of them are expensive, time-costly, and not up-to-date. An emerging alternative is to use GPS traces from dedicated GPS tracking vehicles or GPS-enabled phones. There is abundant road information hidden in these tremendous GPS data, such as the intersection position, the road direction, the road length, the road width, and the topology of road network. Moreover, these data can be conveniently gathered in real time through ubiquitous wireless networks. These make it possible to derive road map automatically or update road map in real time by using GPS data.

At present, there have been a few existing algorithms for building maps with vehicular GPS traces, e.g. clustering-based methods [1-3], machine learning-based methods [4], image processing-based methods [5-7], and a few other methods [8-11]. However, most of them only concerned about the road connectivity or topology, whereas omitted one important element of the road network—road intersection. Paper [4] is the first work that using GPS traces to find the intersections. While in this paper the intersection finding algorithm is a supervised learning algorithm, which needs labeled training data, i.e. it uses some GPS points inside the intersections as the positive samples and uses some GPS points outside the intersections as the negative samples. Besides, the training data of this algorithm is high accurate (the standard deviation is about 4.1 meters) and high-sampling-rate (once per seconds or 5 seconds). This is not the case for commodity GPS devices, which typically have low accuracy (at around tens of meters) and low sampling frequency (more than 15 seconds). So, when there is no labeled training data or the GPS data is coarse-gained, this algorithm will no longer be available. To compensate for these two weaknesses of this algorithm, we develop a new unsupervised training algorithm, which seeking intersections by clustering the ordinary GPS points.

The contributions of this paper are the following.

- We propose a data preprocessing method to improve the concentration of the vehicular turning points. It is verified by comparative experiment that this method has a remarkable effect.
- We design a clustering algorithm for intersection detection. To the best of our knowledge, this is the first attempt to detect intersections using clustering algorithm with coarse-gained GPS data.
- We have conducted empirical evaluation with a real dataset of GPS traces gathered from 2,827 taxis in Shenyang, Liaoning, China. Evaluation results show that our algorithm can find most of the intersections effectively, and exceed the X-

doi:10.4304/jcp.8.11.2959-2965

means algorithm in automatically determining the number of clusters.
- We construct a road network based on the detected intersections, which in turn are used to prune the intersections.

II. Data Processing

A. Raw GPS Data

Our dataset are extracted from a vehicular GPS trace database maintained by China Mobile Group Liaoning Company Limited. The dataset spans from August 2010 to October. The total number of taxis in our dataset ranges from 6000 to 7000 each day. In this study, the experimental data only contains a partial region of Tiexi district of Shenyang City from 10/01 to 10/10 in 2010. We summarize the statistics of the dataset below.

GPS Data Format: Each GPS record contains the following fields which we will use in this work:

- *vehicle_id*: unique ID of the taxi in the dataset;
- *longitude*: current longitude of the taxi;
- *latitude*: current latitude of the taxi;
- *speed*: current speed of the taxi;
- *direction*: current heading direction of the taxi, the GPS device take the north direction as 0 degree (i.e. *direction* = 0), and set *direction* = 0~359 in the clockwise direction;
- *built_time*: current sampling timestamp.

Number of Records: There are totally 2,827 taxis and 5,018,715 GPS position records in the experimental data. However, some vehicles generated many erroneous data due to the failure of the GPS equipments. For example, there exist many invalid values in some GPS records such as *longitude* = 0, *speed*>200, or *direction* > 360. Thus, we need to remove these useless data for the following steps.

Sampling Interval: In our dataset, the GPS sampling frequency of the vehicles are not completely consistent, most of the sampling intervals range from 0.5 to 2 minutes.

B. Looking for the Turning Points

Our scheme is to first find out all of the intersections, and then construct the road network by connecting the intersections. But how can we find the intersections? Let us consider the characteristics of the GPS points around intersections, we find that they have two obvious characteristics, the one is that the stopping points at intersections are remarkably more than other places due to the traffic lights, the other one is that most of the vehicular turning points are located near the intersections, i.e. the heading direction of the vehicle will be changed greatly (more than 45 degree) when the vehicle makes a turn in the intersection. Probably, so long as we seek out the stopping points or the turning points, we can find the intersections by clustering or other data mining algorithms. This may be true for the case clustering turning points, but may not be true for the case clustering stopping points, because vehicles not only stop at the intersections frequently, but also stop at the gas station, the cabstand, the school gate, or the midway of streets due to the traffic jam, which will result in a lot of false intersections. Well then will the turning points produce fake intersections? Of course it will, because drivers also frequently turn the steering wheel when they enter or leave parking lots, or drive in somewhere like the park that have many curves. Fortunately, these useless turning points are easy to get rid of; we can prune them by putting some constraints on the temporal/spatial span between sequential turning points during looking for turning points, or prune them by setting some thresholds afterward.

C. Improve the Concentration

Now, we need to answer another question: do the turning points really converge at the intersections as we expected, while not disperse on the roads? Fig. 1 shows the distribution of the turning points around intersections on October 1, in which the so-called turning points are two consecutive points (i.e. a pair of turning points) before and after one car turning. To visualize the clustering characteristics, we calculated the kernel density estimate (KDE) [12, 13] of these points, as shown in Fig. 4, in which the KDE values indicate the density of these points. From these two figures we can see, the turning points are indeed mainly distributed around intersection, and the nearer from the intersection, the higher the KDE value will be, which proves that the turning points have good clustering characteristics. However, they are not enough concentrated in the center of the intersection because of the traffic light and the low sampling rate, which can be clearly seen from the zoomed region A in Fig. 1. How to improve their concentration? We know that every GPS point has its direction, that is, each point is actually one vector. Thereupon, if we find the intersection point of each pair of vectors, we hereinafter call them converging points for convenience, and use them to replace the turning point pairs; we will be able to increase the concentration of the data. As shown in Fig. 3, the intersection point p_1' of vector p_1 and p_2 is located in center of the road intersection. All the converging points of the tested turning points are illustrated in Fig. 2. By comparing the KDEs of the turning points (Fig. 4) and the converging points (Fig. 5), we can see the latter has a miracle improvement in concentration.

Figure 1. Vehicular turning points.

Figure 2. The converging points.

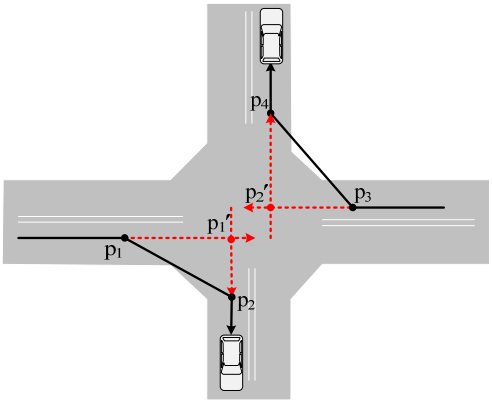

Figure 3. Intersection points of the vector.

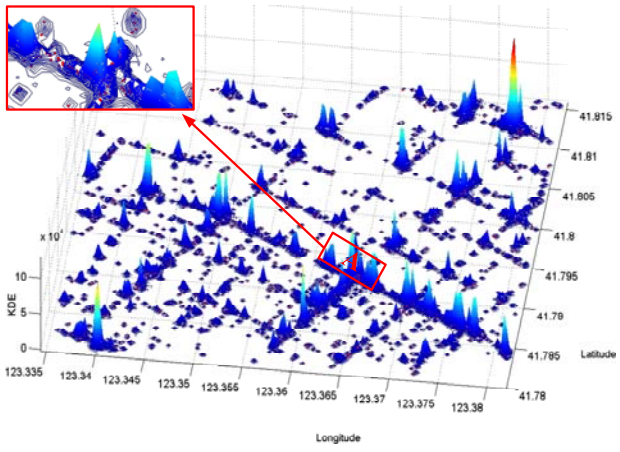

Figure 4. Kernel density estimations of the turning points.

III. CLUSTERING

A. Clustering Characteristis

There are many available clustering methods such as partitioning methods, hierarchical methods, density-based methods, gird-based methods, and model-based methods. Which kind of algorithm is able to meet our needs? Before answering this question, we should consider the clustering characteristics of the turning points and the

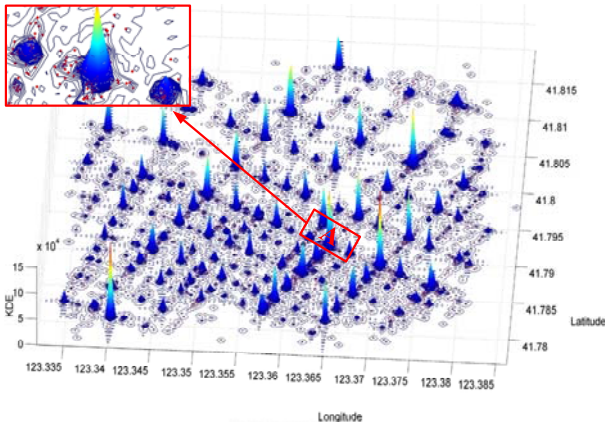

Figure 5. Kernel density estimations of the converging points.

converging points, i.e. the constraints for choosing algorithm.

- Firstly, we have no prior knowledge about the probability distribution of the data model and the exact number *K* of the clusters (i.e. the number of intersections), hence the desired algorithm must be able to automatically determine the value of *K*;
- Secondly, the shape of the clustering should be round or oval rather than arbitrary shape, since the turning points are distributed around the intersections;
- Thirdly, the size of the cluster should not be too big, because too big cluster will cover more than one intersection.
- Finally, the distance between any two cluster centers should not be too small, because any two intersections are rarely close to each other too much in reality.

B. X-means

In view of these issues, we should choose the partition-based clustering algorithm [14-16]. As we all known, K-means algorithm [17] is used widely for its simplicity, and works well for finding spherical-shaped clusters. However, it has some shortcomings. For example, the number *K* of clusters must be supplied in advance as the parameter, and there is no guideline to follow for choosing the value of *K*; it always finds poor optima when given a fixed *K* empirically; and it is slow in the iteration. X-means algorithm [18] improves these inadequacies of the K-means algorithm. X-means does not require a pre-specified *K* in the beginning of algorithm, only need a specified range of *K* values [K_{min}, K_{max}], then it will automatically find *K* by optimizing a criterion such as Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC). In addition, X-means can significantly accelerate the K-means by using a *kd*-tree as an information keeper of the statistics of subsets in clustering. It appears that X-means is able to help us with finding intersections. Unfortunately, X-means can only solve the first two problems in above section, while was helpless to do anything on the latter two issues, i.e. the ultimate clusters in X-means are not

always consistent with reality. This will be proved in section Ⅳ.

C. Our Algorithm

Now, we present an improved algorithm based on X-means in order to meet the latter two requirements. It consists of four operations depicted as follows:

1. **Improve parameters**
2. **Improve structure**
3. If $K > K_{max}$ or there is no cluster to be split during the search, Then goto 4; Else, goto 1.
4. **Merge clusters**
5. **Prune clusters**

The **Improve parameters** operation is same as *X*-means: running conventional *K*-means to convergence.

In *X*-means, the **Improve structure** operation splits clusters according to the BIC score of the parent clusters and their offspring. Such a splitting strategy has a very rigorous theory foundation, and works well in most cases. However, it cannot obtain good results for our GPS data, because it always over split the cluster. For example, the turning points usually distribute around the intersection, but not concentrate in the center of the intersection (as shown in region *A* of the Fig. 1). In this case, these points will be split into more than one cluster according to the *X*-means's BIC criterion, which does not agree with the real story. So we need modify the heuristic criterions of cluster splitting so as to assign the points that belong to one intersection into one cluster. To this end, we designed two new splitting criterions.

First of all, we calculate the radius r_i of the parent cluster, the distance d_i between the two children's centroids, the parent's BIC bp_i, and the children's BICs bc_i, then we split the parent cluster when one of the following two conditions is true:

1. $r_i > R_{up}$ and $d_i > D_{up}$
2. $R_{low} < r_i < R_{up}$, $D_{low} < d_i < D_{up}$, and $bp_i < bc_i$

Where R_{low} and R_{up} are the lower and upper bounds of r_i respectively, D_{low} and D_{up} are the lower and upper bounds of d_i respectively. With the first rule, we can ensure that any cluster will not cover more than one intersection, and the distance between any two intersections will not be too close. Nevertheless, only such a rule cannot split all clusters very well. If the values of R_{low} and R_{up} are set too high, some relatively small clusters that still cover multiple intersections will not be split, while if the values are set too small, some clusters that only cover one intersection will be split into multiple subsets incorrectly. Thus, we split big cluster with rule1, and split small cluster using rule 2. Let us take an example for this, assuming that C_1, C_2, and C_3 are three clusters with the radius 110m, 120m, and 150m respectively, wherein C_1 contains 2 intersections, C_2 contains 1 intersection, and C_2 contains 2 intersections. If we use rule 1 to split cluster with the threshold R_{up}=140, then only C_3 will be split; if we set R_{up}=100, C_2 will be split incorrectly. When we use both the two rules, in which R_{low}=100 and R_{up}=140, both C_1 and C_3 will be split correctly, and C_1 will no longer be split under the guarantee of the condition $bp_i < bc_i$. The roles of parameters D_{low} and D_{up} are similar to those of R_{low} and R_{up}. With them, we can ensure that the two cluster centroids after split will not be too closed.

Our algorithm start to excute with a initial value $K = K_{min}$, and then perform the **Improve parameters** and the **Improve structure** operation iteratively, until K is greater than a upper bound K_{max} or there is no cluster to be split in **Improve structure** operation.

Because the initial cluster centers are selected randomly, we cannot guarantee that all the final clusters meet the constraint condition 4 in section *B*. So we need the **Merge clusters** operation to merge all too closed clusters.

Finally, the **Prune clusters** operation gets rid of all the fake intersections, i.e. the clusters that contains too few points (e.g. 1~3 points).

IV. Experiment

In this section, we make two comparisons to demonstrate the effectiveness of our methods. Because we have no real intersection positions, we can only give a few visual demonstrations of the experimental results by superimposing the clustering results on the satellite road map of Google Earth.

A. Comparisons

First of all, let us compare the clustering results of the turning points and the converging points that both using our algorithm. In Fig. 6 and 7, the red dots represent the centroids of clusters. From them we can see our algorithm can achieve good results on both the two datasets. But, our algorithm cannot identify some small intersections correctly when clustering the turning points, as shown in region B of Fig. 6; this is because some intersections are too closed so that their turning points are mingled together, and are viewed as one cluster by our algorithm. When clustering the converging points, our algorithm can recognize all the intersections that cannot be recognized in turning points, as shown in region B of Fig. 7. This comparison proves that the converging points can reflect the location of the intersections better than the turning points.

In this experiment, the parameters of our algorithm are

Figure 6. Clustering results of our algorithm in the turning points.

Figure 7. Clustering results of our algorithm in the converging points.

set to: K_{min}=50, K_{max}=300, R_{low} =115, R_{up}=175, D_{low}=115, D_{up} =175.

To prove the effectiveness of the presented algorithm, we make a comparison between it and X-means using the converging points. Table 1 show the number of clusters N_c of them that correspond to different K_{min}. We can clearly see that *X*-means almost has nothing to do with the cluster splitting, while our algorithm get stability at around 191 when K_{min} <100. Fig. 8 illustrates the cluster centroids of X-means, showing that X-means always over split the big clusters that cover the too busy intersection. Here, we set K_{min}=165 for *X*-means.

By comparison, we find that *X*-means not only cannot find the correct location of the intersection, but also cannot find the appropriate intersection number, while our algorithm is just the opposite.

B. Parameter Sensitivity

The proposed algorithm includes several tuning parameters: R_{low}, R_{up}, D_{low}, D_{up}. We conduct a sensitivity analysis to see how they impact the performance. Some of the results of this analysis are discussed below.

Figure 8. Clustering results of *X*-means in the converging points.

TABLE III.
COMPARISON OF X-MEANS AND OUR ALGORITHM IN FINDING THE VALUE OF *K*.

K_{min}	N_c (*X*-means)	N_c (Proposed)
2	2	191
10	10	191
30	30	190
50	55	190
70	82	192
100	110	196
130	130	202
150	150	207
170	190	214

Table 2 shows the changes of N_c when tune the value of R_{low}, R_{up}, during the test we simply set D_{low}= R_{low} and D_{up}= R_{up} for convenience. As can be seen from this table, when given a R_{up}, the value of N_c almost remain unchanged as the increase of R_{low}, and when given a R_{low}, N_c linearly decrease with the increase of R_{up}. In other words, the number of cluster is sensitive to R_{up}, while inert to R_{low}.

TABLE I.
AFFECTIONS OF DIFFERENT R_{LOW} AND R_{UP} ON CLUSTER NUMBER *K*

R_{low}	R_{up}			
	160	165	170	175
110	207	199	191	183
115	207	197	190	183
120	206	197	192	182
125	207	198	189	181

Still, we have another problem, that is, which parameters are optimal? Because of the lack of benchmark intersections, we can only conduct an empirical analysis by comparing the clustering results and Google Earth. Through observation we find that when 110 < R_{low} < 120 and 165 < R_{up} < 175, the clustering results is the best.

C. Finding Roads and Pruning Intersections

With the intersections, we can easily construct roads network by connect the intersections, which, in turn, can be used for refining the intersections' number.

Our road finding algorithm is relatively simple and very effective. In reality, each trip may pass through a number of intersections, and the intersections that have connections will have a busy traffic. So that we can count the traffic flow between the intersections, and then connect the intersections that have heavy traffic. However, this may cause some mistaken connections due to the coarse-gained GPS samples. For example, assume intersection A is connected to intersection C through intersection B, and the length of AB and BC is comparatively short, as illustrated in Fig. 9, when some high-speed cars pass through A, B, and C, may be only two GPS points P_1, P_2 fall into the intersections A and C respectively. This will cause our algorithm to connect A to C directly, which is not the case in ground truth. To prune such mistakes, for every two connected intersections, we find their shortest indirect path and their

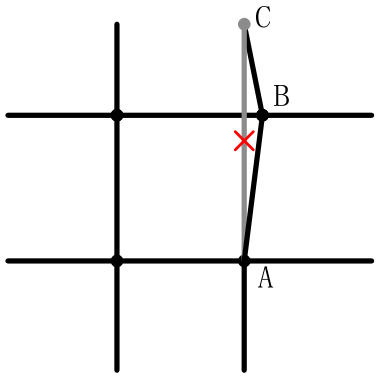

Figure 9. Inaccurate road.

Figure 10. Intersections and roads.

direct path (e.g. A-B-C and A-C). If the former has significantly larger traffic than the latter or the former is almost the same length as the latter, then we remove the latter. The final road network is illustrated in Fig. 10. After roads construction, we can next prune the fake intersections that have less than three connections.

D. Explanation of the Experimental Results

Through the above experiments, we can find that our algorithm can get the optimal when clustering the converging points with the conditions $110 < R_{low} < 120$ and $165 < R_{up} < 175$. However, the experimental results do not seem to be perfect. By carefully observing Fig. 7, we can see that some small intersections were not recognized. Perhaps, this is because these roads have not yet opened at that time or too few vehicles pass through these intersections. Also, the precisions of some detected intersections are not enough, which deviate from the actual positions. This is because we lack the benchmark positions of the intersections, so we cannot train the best parameter values. Moreover, we find that the accuracy of the identified intersections near the boundary area is lower than that within the region; this is because the turning points near the edge of the region are incomplete.

V. CONCLUSIONS

In this paper we present a data preprocessing method and a new cluster method for finding road intersections from the vehicular Coarse-gained GPS traces. Proved by experiments, our data preprocessing method can notably enhance the concentration of the GPS points, and our cluster algorithm can find most of the intersections effectively. However, they still have several problems that need further improvement. For example, the found intersection locations are not highly precise, and the method of finding converging points may lose its effectiveness in curves, because the vehicular heading direction will keep changing during the car moving in the curves.

So the future work should improve the precision and reduce the false positive rate of the intersection locations. One possible solution is to use the kernel density estimate method to determine the number of intersections, and improve the accuracy of the intersection locations.

ACKNOWLEDGMENT

This work was supported by NSFC (Grant No. 51205389, 61003208, 61174164, and 61105067), the S&T Program of Shenyang (Grant No. F11-264-1-08), and the National Key Technology R&D Program (Grant No. 2012BAF10B11 and 2012BAF10B06). We would like to thank all the members of the Sense Network research team in Key Laboratory of Information Services and Intelligent Control Technology of Shenyang Institute of Automation, Chinese Academy of Sciences.

REFERENCES

[1] S. Worrall and E. Nebot, "Automated process for generating digitised maps through gps data compression," presented at the Australasian Conference on Robotics and Automation, 2007.

[2] S. Schroedl, *et al.*, "Mining GPS traces for map refinement," *Data mining and knowledge Discovery*, vol. 9, pp. 59-87, 2004.

[3] X. Liu, *et al.*, "Road Recognition using Coarse-grained Vehicular Traces," HP Labs, HP Labs2012.

[4] A. Fathi and J. Krumm, "Detecting road intersections from gps traces," *Geographic Information Science*, pp. 56-69, 2010.

[5] W. Shi, *et al.*, "Automatic generation of road network map from massive GPS, vehicle trajectories," presented at the Intelligent Transportation Systems, 2009. ITSC '09. 12th International IEEE Conference on, 2009.

[6] C. Chen and Y. Cheng, "Roads Digital Map Generation with Multi-track GPS Data," presented at the Proceedings of the 2008 International Workshop on Education Technology and Training \& 2008 International Workshop on Geoscience and Remote Sensing - Volume 01, 2008.

[7] J. J. Davics, *et al.*, "Scalable, distributed, real-time map generation," *Pervasive Computing, IEEE*, vol. 5, pp. 47-54, 2006.

[8] M. Ahmed and C. Wenk. (2012). *Constructing Street-Maps from GPS Trajectories*. Available: http://www.cs.utsa.edu/~carola/FWCG11-StreetMapConstruction.pdf

[9] L. Cao and J. Krumm, "From GPS traces to a routable road map," presented at the Proceedings of the 17th ACM

SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, Washington, 2009.
[10] R. Bruntrup, *et al.*, "Incremental map generation with GPS traces," presented at the Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE, 2005.
[11] Y. Chen and J. Krumm, "Probabilistic modeling of traffic lanes from GPS traces," presented at the Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, San Jose, California, 2010.
[12] B. W. Silverman, *Density estimation for statistics and data analysis* vol. 26: Chapman & Hall/CRC, 1986.
[13] D. W. Scott, *Multivariate density estimation: Theory, Practice, and Visualization* vol. 1: Wiley, 1992.
[14] D. Lei, *et al.*, "Automatic PAM Clustering Algorithm for Outlier Detection," *Journal of Software,* vol. 7, pp. 1045-1051, 2012.
[15] X. Li, "A New Text Clustering Algorithm Based on Improved K_means," *Journal of Software,* vol. 7, pp. 95-101, 2012.
[16] G. Lin, et al., "A Novel Clustering Algorithm Based on Graph Matching," Journal of Software, vol. 8, pp. 1035-1041, 2013.
[17] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1967, p. 14.
[18] D. Pelleg and A. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," in *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 727-734.

Junwei Wu was born in Henan province, P. R. China, in 1982. He received the B.S. degree in Computer Science and Technology from Henan University of Science and Technology, China, in 2007. He is a PhD student at the of the Lab. of the Information Service and Intelligent Control, Shenyang Institute of Automation of the Chinese Academy of Sciences. His research interests are in sense network, mobile data mining, and intelligent transportation.

Yunlong Zhu is the director of the Lab. of the Information Service and Intelligent Control, Shenyang Institute of Automation of the Chinese Academy of Sciences. He received his Ph.D. in 2005 from the Chinese Academy of Sciences, China. He has research interests in various aspects of Enterprise Information Management but he has ongoing interests in Artifical Intelligence, Machine Learning, and related areas. Prof Zhu's research has led to a dozen professional a publication in these areas includes the biography here.

Tao Ku was born in Shanxi province, P. R. China, in 1979. He received B.S. in Computer Science and Technology from Shanxi University of Science and Technology, China, in 2003, and received Ph.D. degrees in electromechanical engineering from Shenyang Institute of Automation of the Chinese Academy of Sciences, China, in 2009. He is an Associate Researcher in Shenyang Institute of Automation of the Chinese Academy of Sciences. His research interests are in sensor network, reality mining, intelligent information processing, and printed electronics equipment and technology.

Liang Wang received M.S. degree in automatic control from Northeast University, Shenyang, China, in 2009. He is currently pursuing the Ph.D. degree at Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China. His current research interests include data mining and decision support systems.