Mining Personal Frequent Routes via Road Corner Detection

Tianben Wang, Daqing Zhang, Xingshe Zhou, Xin Qi, Hongbo Ni, Haipeng Wang, and Gang Zhou, *Senior Member, IEEE*

***Abstract*—Frequent route is an important individual outdoor behavior pattern that many trajectory-based applications rely on. In this paper, we propose a novel framework for extracting frequent routes from personal GPS trajectories. The key idea of our design is to accurately detect road corners and utilize these new metaphors to tackle the problem of frequent route extraction. Concretely, our framework contains three phases: 1) characteristic point (CP) extraction; 2) corner detection; and 3) trajectory mapping. In the first phase, we present a linear fitting-based algorithm to extract CPs. In the second phase, we develop a multiple density level DBSCAN (density-based spatial clustering of applications with noise) algorithm to locate road corners by clustering CPs. In the third phase, we convert each trajectory into an ordered sequence of road corners and obtain all routes that have been traversed by an individual for at least *F* (frequency threshold) times. We evaluate the framework using real-world trajectory datasets of individuals for one year and the experimental results demonstrate that our framework outperforms the baseline approach by 7.8% on average in terms of precision and 21.9% in terms of recall.**

***Index Terms*—Characteristic point extraction (CPE), corner detection, frequent routes.**

I. Introduction

With the wide adoption of GPS receivers in vehicles and smartphones, huge amounts of personal GPS trajectories have been accumulated. By analyzing those GPS trajectories, we can understand each individual's mobility patterns and obtain valuable insights about her/his daily behavior. These patterns and behaviors can be further utilized to improve the quality of various trajectory-based services, such as route prediction [1]–[3], disorientation detection [4], [5], trip planning [6]–[10], and location-based recommendation [11]–[13].

Manuscript received November 30, 2014; revised February 22, 2015; accepted May 8, 2015. Date of publication July 2, 2015; date of current version March 15, 2016. This work was supported by the State Key Program of the National Natural Science of China under Grant 61332013. This paper was recommended by Associate Editor B.-F. Wu.

T. Wang, X. Zhou, H. Ni, and H. Wang are with the Department of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China (e-mail: wangtianbengx@163.com; zhouxs@nwpu.edu.cn; nihb@nwpu.edu.cn; haipeng.wang@gmail.com).

D. Zhang is with the Department of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China and Institut Mines-TELECOM/TELECOM SudParis, Evry, 91011, France (e-mail: zhang_da@telecom-sudparis.eu).

X. Qi and G. Zhou are with the Department of Computer Science, College of William and Mary, Williamsburg, VA 23187-8795 USA (e-mail: xqi@email.wm.edu; gzhou.wm@gmail.com).

Digital Object Identifier 10.1109/TSMC.2015.2444416

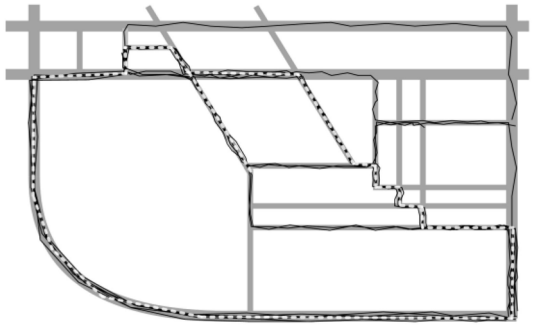


Fig. 1. Illustration of frequent routes.

Frequent route is an important individual outdoor behavior pattern that the aforementioned ubiquitous applications rely on. Fig. 1 illustrates an example of frequent routes. In the figure, gray lines denote the physical roads and black lines denote one individual's GPS trajectories. The white dotted lines highlight the frequent routes of the individual's outdoor movements. In the example, we define a route to be a frequent route only if an individual has traversed the route for a certain amount of time.

There are some existing works that attempt to extract frequent routes from personal GPS trajectories. However, extracting frequent routes from personal GPS trajectories is still a challenging problem for the following reasons.

1) GPS readings are not accurate due to hardware constraints. Inaccurate GPS readings and frequent speed changes in a trip result in irregular fluctuations in GPS trajectories. Mathematically, it is difficult to define an accurate distance function to measure trajectory similarity for irregular fluctuated trajectories. Without timing information in trajectories, this problem is even more difficult. Thus, the frequent route extraction methods based on trajectory similarity [14]–[19] cannot be applied to irregular fluctuated trajectories.
2) Physical roads are different from each other. Some direction changes are sharp, while others are smooth. Thus, it is difficult to accurately define where a road direction changes. This ambiguous feature of roads naturally propagates to GPS trajectories and disables those frequent route extraction methods based on segmenting and clustering trajectories [15], [20], since they need to partition trajectories at special GPS points where trajectory direction changes.

3) Different physical roads have different trajectory densities due to variation in visit instances collected. This fact indicates that existing frequent route extraction methods [15], [16], [18], [19] based on clustering trajectories using uniform trajectory density cannot reliably detect all the trajectory clusters with different densities.
4) The ideal route representation should be concise and close to the corresponding physical roads. However, existing frequent route extraction methods based on clustering trajectories [9], [15], [16], [18], [19] either use multiple points to represent a physical road segment between two road corners, or use simple direct connections between two hot regions to represent physical road segments.

To tackle the above challenges, we conduct preliminary analysis on a large number of personal GPS trajectories and obtain the following three observations.

1) Individuals' daily outdoor movements are constrained by physical roads.
2) Physical roads' topology information, such as road corners, is embedded in personal GPS trajectories.
3) If road corners can be detected accurately, the physical roads can be most concisely represented by connecting all the road corners sequentially.

Based on these observations, we propose a novel frequent route extraction framework that leverages corner detection techniques for making full use of physical road topology information embedded in personal GPS trajectories. Particularly, instead of defining complicated similarity metrics for clustering GPS trajectories, our method maps GPS trajectories onto physical roads with the aid of corner detection and outputs concise frequent routes in the form of physical road segments. We validate our framework on a large number of personal GPS trajectories.

Our main contributions are summarized as follows.

1) To the best of our knowledge, we are the first to leverage physical road topology information, particularly road corners and connectivity between them, for frequent route extraction.
2) We propose a characteristic point extraction (CPE) method based on linear fitting techniques. The CPE method filters out irregular fluctuations in GPS trajectories and identifies actual characteristic points (CPs) at road corners with different sharpness.
3) We design a multiple density level density-based spatial clustering of applications with noise (DBSCAN) (MDL-DBSCAN) algorithm based on an existing algorithm [21] to detect road corners by clustering CPs with different densities.
4) We define each trajectory as an ordered sequence of detected road corners rather than grid cells [28]. This trajectory representation streamlines our cluster fusing process, which maps trajectory clusters onto physical roads.
5) We evaluate the proposed framework with real-world GPS trajectories collected from more than 6800 individuals for a year. Experimental results demonstrate that our framework outperforms the trajectory clustering-based method as a baseline method in terms of both precision and recall.

The rest of this paper is organized as follows. We first review the related work in Section II. Then, we define the problem in Section III and introduce our solution in detail in Section IV. Next, we present the evaluation results in Section V. Finally, we conclude this paper in Section VI.

II. Related Work

We divide the existing work on frequent route mining into two categories: 1) trajectory clustering and 2) trajectory pattern mining.

A. Trajectory Clustering

We further divide the research in this category into two subcategories: 1) clustering whole trajectories and 2) clustering trajectory segments. Both have been widely applied to extract personal frequent routes.

1) Clustering Whole Trajectories: Based on the fact that silent durations exist in trajectories, Hung *et al.* [18], [19] proposed a time-related metric to measure the similarity between whole trajectories. With this similarity metric, they develop a graph-based trajectory-clustering algorithm. The algorithm first constructs a clue graph and then partitions it into subgraphs, where the similarity between any two vertexes (i.e., trajectories) must be high. Analogously, Nanni and Pedreschi [17] defined a similarity metric by calculating the Euclidean distance between location points of different trajectories at each time slot and utilize the density-based clustering algorithm OPTICS [22] to cluster whole trajectories. These two methods aim to extract the clusters, in which, the whole trajectories are similar not only in spatial domain but also in temporal domain. Thus, they work well to extract spatiotemporal frequent routes. However, this paper focuses on extracting frequent routes from whole trajectories only similar in spatial domain, which means the trajectories mapped into a frequent route may have quite different temporal values. Thus, the similarity metrics related to temporal information cannot be adopted in this paper.

2) Clustering Trajectory Segments: Lee *et al.* [15] proposed a partition-and-group method, which first splits trajectories into line segments and clusters line segments with a density-based clustering algorithm, DBSCAN. Li *et al.* [16] proposed an incremental clustering framework to adapt to the increase of trajectory dataset. They also partition first the trajectory into segments. Then, the segments are clustered in two phases, micro- and macroclustering. The clustering algorithm used in macroclustering is the same as [15]. Microclustering operates offline or at the background when new trajectories are added. Macroclustering is performed on the fusion result of microclustering when a query comes. These approaches use the original density-based clustering algorithms to cluster the trajectory segments and work well when the distribution of trajectories is relatively uniform. The main shortage of these algorithms is that they cannot adapt to the trajectories with different densities. Unfortunately, trajectories and trajectory segments usually have different densities in large-scale GPS trajectory datasets.

The shortages of the above methods inspire us to propose a novel approach for frequent route extraction, which should be similarity-independent and is able to adapt to trajectories with different densities.

B. Trajectory Pattern Mining

Existing trajectory pattern mining work can also be classified into two subcategories: 1) aggregating the trajectory clusters and 2) connecting the hot regions.

1) Aggregating Trajectory Clusters: As mentioned in trajectory clustering, the methods in [15] and [16] first split the trajectory into segments, and then cluster the segments. Finally, the frequent routes are represented as the representative lines of each cluster. The representative line of a cluster is generated by computing the average value of the crossing points when moving a scan line, which is perpendicular to the cluster trend vector, in the cluster. Hung *et al.* [18], [19] clustered trajectory as a whole by first employing a graph-based clustering algorithm. Then, the representative line of each trajectory cluster is computed by connecting the centers of special point clusters. Finally, all the representative lines are connected to form trajectory patterns. The frequent routes extracted by the above method are closer to the corresponding physical routes than that extracted by the methods of connecting the hot regions. Since these methods inherit the shortages of the trajectory clustering methods mentioned in Section II-A, they cannot detect all frequent routes.

2) Connecting Hot Regions: Cao *et al.* [23] presented a framework that splits the trajectories into segments and then detects the frequently visited spatial rectangle regions. Thus, the original trajectories can all be transformed as a sequence of frequent regions. Finally, an improved Apriori algorithm is used to find frequent patterns. Verhein and Chawla [24] studied the technique to mine spatiotemporal patterns with semantics. The sematic hot regions like stationary regions and high-traffic regions are first extracted from trajectories. Then an association rule-based method is employed to mine the spatiotemporal patterns. Mamoulis *et al.* [25] defined the hot regions as the dense clusters of location points that are split from original location sequence with respect to a static time interval. Then, each original location sequence can be transformed into an ordered sequence of hot regions. Finally, an association rule-based method is utilized to mine the frequent spatiotemporal patterns. Jeung *et al.* [26], [27] proposed a framework that utilizes the similar method to extract the hot regions as [25], and employs hidden Markov models to find the frequent movement patterns. The above methods work well to reveal the relationship between the hot regions, but the connections between hot regions do not correspond to physical roads.

In contrast to the above works, our method tackles the frequent route extraction problem by extensively utilizing physical roads' topology information, such as road corners instead of aggregating trajectory clusters or connecting hot regions.

III. Problem Statement

A GPS trajectory consists of a sequence of GPS points, containing latitude, longitude, and time stamp, generated by GPS devices. In this paper, we focus on extracting time-independent frequent routes from GPS trajectories. Thus, we ignore the temporal information in GPS trajectories and view them as time-independent trajectories. We define a GPS trajectory as follows.

Definition 1: A GPS trajectory is defined as a sequence of GPS points, i.e.,

$$T : p_1 p_2 \dots p_n$$

where p_i's are GPS points and $p_i = (x_i, y_i)$, where x_i denotes longitude and y_i denotes latitude.

Given a set of GPS trajectories of an individual, our goal is to extract frequent routes, which have been traversed frequently by the individual. Formally, the problem is defined as follows.

Problem: Given a set of trajectories $\mathrm{TS} = \{T_1, T_2, \dots, T_N\}$ traversed by an individual and a frequency threshold F, our objective is to extract the frequent routes $\mathrm{FRs} = \{S_1, S_2, \dots, S_m\}$, where S_i denotes a road segment which is contained in $T_j (j = 1, 2, \dots, N)$ and traversed at least F times by the individual.

IV. Our Proposed Framework

In this section, we present the design of our frequent route extraction framework.

A. Frequent Route Extraction Framework

Based on the observations 1) and 3) in Section I, we convert the frequent route extraction problem into a traversed road segment counting issue. By mapping personal GPS trajectories onto physical roads, we need to determine the road segments contained in each trajectory and count the number of times each road segment has been traversed. If the number of traversed time for a route is greater than a certain threshold F, then the route is identified as the frequent route for an individual.

To solve the aforementioned problem, we have to solve two subproblems: 1) how to extract roads topology information to segment a GPS trajectory and 2) how to map GPS trajectories onto road segments in physical roads. According to the observations 1) and 2) in Section I, human outdoor movements are constrained by physical roads. Thus, the GPS trajectories of personal movements inevitably contain the connected road segment of physical roads.

Road corners are identified as the metaphors to separate road segments in GPS trajectory. Physically, it is reasonable to infer that the locations, at which a number of trajectories direction change, are road corners. To utilize road corners for frequent route extraction, we can first identify the special points in each trajectory, where trajectory direction changes significantly and steadily. Then, we can locate road corners by clustering these special points. If a trajectory traverses two corners successively, the two corners form a road segment that has been traversed. By locating road corners in trajectories, we generate a minimal number of points to separate a GPS trajectory into a sequence of road segments, we then extract frequent routes as the road segments that have been traversed more than F times by an individual.

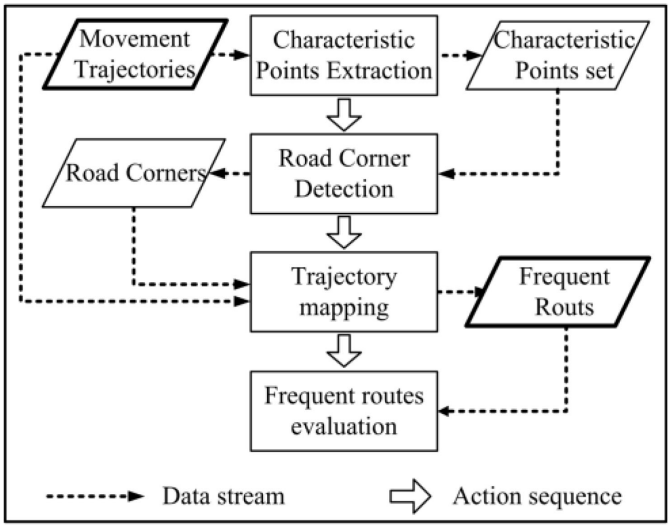


Fig. 2. Overview of our framework.

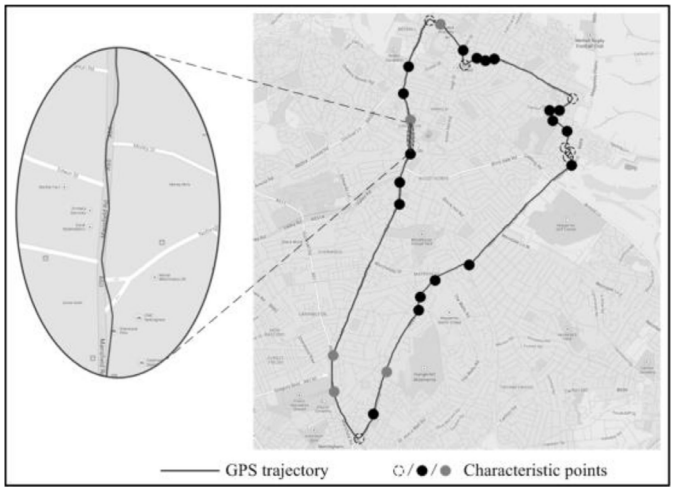


Fig. 3. Illustrative example of CP.

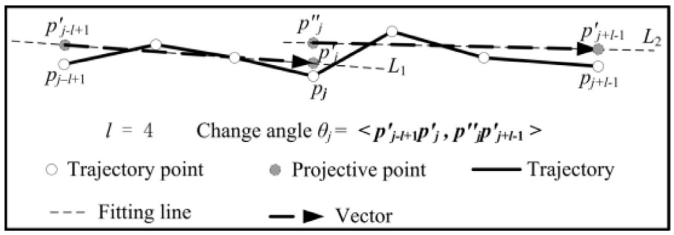


Fig. 4. Illustrative example of direction change angle detecting based on linear fitting.

With the road corners, the second subproblem of mapping trajectories into road segments in physical roads is equivalent to transforming each trajectory into an ordered sequence of road segments defined by road corners.

Based on the above analysis, we propose a novel framework to extract frequent routes. Fig. 2 illustrates the overview of our framework. Generally, the framework consists of four steps. In the first step, a CPE method is applied to extract the CPs in each trajectory. In the second step, an improved DBSCAN clustering algorithm named MDL-DBSCAN is used to locate the road corners. In the third step, all the trajectories are mapped onto physical roads and the frequent routes are extracted. Finally, the effectiveness of frequent route extraction framework is evaluated using a real-world trajectory dataset.

B. Characteristic Point Extraction

As shown in Fig. 2, given a large collection of GPS trajectories, the first task is to extract CPs of each trajectory. As mentioned previously, a CP is the GPS point where the trajectory's direction changes significantly and steadily (we will present its formal definition later). Fig. 3 illustrates an example of individual movement trajectory denoted as a gray line on the right panel. This trajectory is constrained by physical roads apparently. The CPs, marked as points, are the locations where trajectory direction changes significantly and steadily.

Given a GPS trajectory $T_i = p_1 p_2 \dots p_{ni}$, an intuitive method to extract the CPs is measuring the angle between segment $p_{j-1}p_j$ and $p_j p_{j+1}$. Unfortunately, in practice, GPS trajectory often suffers the low-sampling-rate problem, i.e., GPS devices collect data at a low and unstable frequency. What's worse is that GPS drift error cannot be ignored. Due to low-sampling-rate and drift errors, irregular fluctuations exist in GPS trajectories. An example is illustrated in the left panel in Fig. 3. If we adopt the intuitive idea directly, the irregular fluctuation in GPS trajectory will result in poor results.

To circumvent the problem, we introduce a method based on linear fitting. Fig. 4 depicts an example. To detect the direction change at GPS point p_j, we first linear fit the point sets $\{p_{j-l+1}, p_{j-l+2}, \dots, p_j\}$ and $\{p_j, p_{j+1}, \dots, p_{j+l-1}\}$ and obtain two straight lines L_1 and L_2. The referenced parameter l denotes the fitting length, i.e., the number of points the linear fitting method considers on each side of p_j. In the above example, we set $l = 4$. In fact, for real GPS trajectories, to eliminate the negative effect of irregular fluctuations effectively and keep the steady CPs, the value of l is usually set as 8–12. In the linear fitting method, suppose input point set is $A = \{p_1, p_2, \dots, p_N\}$, where $p_i = (x_i, y_i)$, $i = 1, 2, \dots, N$, and the output result $L : y = ax + b$, where a and b can be calculated using

$$\begin{cases} a = \dfrac{N\sum_{i=1}^{N} x_i y_i - \left(\sum_{i=1}^{N} x_i\right)\left(\sum_{i=1}^{N} y_i\right)}{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2} \\ b = \dfrac{\left(\sum_{i=1}^{N} x_i^2\right)\left(\sum_{i=1}^{N} y_i\right) - \left(\sum_{i=1}^{N} x_i\right)\left(\sum_{i=1}^{N} x_i y_i\right)}{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2}. \end{cases} \tag{1}$$

Second, we project the point p_{j-l+1} and p_j onto L_1. Accordingly, the projective points of p_{j-l+1} and p_j on L_1 are p'_{j-l+1} and p'_j, respectively. Likewise, the projective points of p_j and p_{j+l-1} on L_2 are p''_j and p'_{j+l-1}, respectively. Given a point $p = (x, y)$ and a line $L : y = ax + b$, the projective point $p' = (x', y')$ on L can be calculated using

$$\begin{cases} x' = (ay - ab + x)/(a^2 + 1) \\ y' = (a^2 y + ax + b)/(a^2 + 1). \end{cases} \tag{2}$$

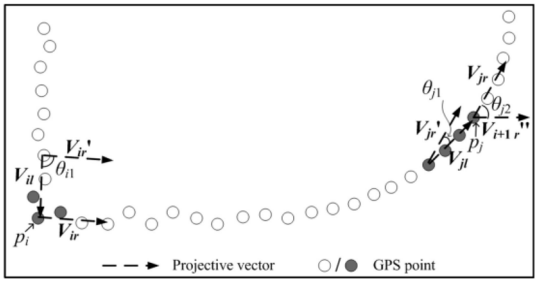


Fig. 5. Illustrative example of sharp and "smooth" direction changes coexist in a GPS trajectory simultaneously.

Finally, instead of measuring the angle between segments $p_{j-1}p_j$ and p_jp_{j+1}, we measure the angle between the left projective vector $\boldsymbol{V_{jl}} = \boldsymbol{p'_{j-l+1}p'_j}$ of p_j on line L_1 and right projective vectors $\boldsymbol{V_{jr}} = \boldsymbol{p''_jp'_{j+l-1}}$ of p_j on line L_2. This angle is called change angle at p_j. If the angle is larger than a given threshold, then we regard p_j as a candidate characteristic point (CCP). This process will run on each point p_j repeatedly until $j + l - 1$ is out of the upper bound of trajectory T, i.e., *ni.*

In Fig. 3, we can also observe that the "sharp" direction changes, denoted as hollow dotted points, the "normal" direction changes, denoted as black points, and the "slow" direction changes, denoted as gray points, coexist in a real trajectory. The CCPs in sharp and normal direction change regions will easily be detected by the linear fitting-based method, but the CCPs in slow direction change regions cannot be directly detected in most cases. The reason is explained in Fig. 5, where the angle threshold value and fitting length value are set as $\pi/4$ and 4, respectively.

As shown in Fig. 5, if we directly apply the linear fitting-based method, we can see that the change angle at p_i, θ_{i1} (to show the angle clearly, we translate V_{ir} to V'_{ir}), is larger than $\pi/4$, so p_i and the points close to p_i, such as $p_{i-1}, p_{i+1}, p_{i-2}$, and p_{i+2} can all be detected as CCPs. Without loss of generality, we assume only p_{i-1}, p_i, p_{i+1} are detected as CCPs. In contrast, the change angle θ_{j1} at p_j (likewise, we translate V_{jr} to V'_{jr}) is smaller than $\pi/4$. Thus, p_j and the points close to p_j, such as $p_{j-1}, p_{j+1}, p_{j-2}$, and p_{j+2} cannot be detected as CCPs. However, we observe that the direction change at p_j is finally close to $\pi/2$. In fact, there should be at least one CCP in a slow direction change region. The main reason for the linear fitting-based method not working when the direction changes slowly is that static angle threshold and fitting length are adopted for direction changes with different sharpness.

In order to detect the CCPs at slow direction change, we first recognize slow direction change regions (subsequences of GPS trajectory that contain slow direction change), then design special methods to extract CCPs from them.

To detect slow direction change regions, we check not only the change angle θ_1 at the current point, but also the change angle θ_2 between the right projective vector of last CCP and that of the current point. If θ_1 is larger than the angle threshold, the current point will be regarded as a CCP. Otherwise, if θ_2 is larger than the angle threshold, it indicates that there must be a slow direction change in the region from the last CCP to the current point. For instance, in Fig. 5, suppose p_j is the current point, we check the angle θ_{j1} and θ_{j2} simultaneously. Obviously, θ_{j1} is smaller than $\pi/4$ (the angle threshold) while θ_{j2} is larger than $\pi/4$, so there must be a slow direction change in the region from p_{i+1} to p_j.

After detecting a slow direction change region, we move the "current pointer" from current point to the point next to last CCP and employ the linear fitting-based method to detect CP in this region with a gradually decreased angle threshold until at least one CCP is detected.

To represent the detected trajectory direction changes in a more concise way and simplify the input of successive steps, we further sort out the CPs from CCPs, which best represent the detected trajectory direction changes. Before defining CP, we first define longest continuous CCP sequence (LCCS), which is the basis to extract CPs from CCPs.

Definition 2: Given a GPS trajectory $T = p_1p_2 \dots p_N$, and a continuous subtrajectory $S = p_sp_{s+1} \dots p_e$, where $1 \le s < e \le N$. If S satisfies:

1) the points in S are all CPPs;

2) $$\begin{cases} p_{e+1} \text{ is not a CCP} & s = 1 \text{ and } e < N \\ p_{s-1} \text{ is not a CCP} & s > 1 \text{ and } e = N \\ p_{e+1} \text{ and } p_{s+1} \text{ are not CCP} & s > 1 \text{ and } e < N \end{cases}$$

then, S is defined as an LCCS.

With the definition of LCCS, we define CP as follows.

Definition 3: Given an LCCS $S = p_sp_{s+1} \dots p_e$, if the change angle at $p_j (s \le j \le e)$ is larger than that of the rest of points in S, then p_j is identified as a CP.

An example of CP is depicted in Fig. 5. The points, $p_{i-1}, p_i, p_{i+1}, p_{j-3}, p_{j-2}, p_{j-1}$, and p_j (denoted as the gray GPS points), are detected as CCPs. According to the definition of LCCS, $S_1 = p_{i-1}p_ip_{i+1}$ and $S_2 = p_{j-3}p_{j-2}p_{j-1}p_j$ are detected as LCCS. Finally, according to the definition of CP, the points, p_i and p_{j-1}, are detected as CP (here, we assume that p_i and p_{j-1} have the largest change angle among S_1 and S_2, respectively). Note that, the start and end points of each trajectory should be added to the CP set as well.

The pseudocode of the linear fitting-based CPE method is summarized in Algorithm 1. The CPE algorithm first initializes the variable lastCPP (line 1). Then, for each point from p_{l-1} to p_{n-l}, it employs linear fitting-based method [implemented as the function isCPP($p_j, l, \theta_{\text{thrd}}$)], to detect whether the current point is a CCP. If it is true, then the candidate point is added into CCPs (lines 2–5). Otherwise, it will check whether there is a slow direction change in the sequence from p_{lastCPP} to p_j [implemented as the function isSlowChange(lastCPP, $j, l, \theta_{\text{thrd}}$)]. If the sequence of points corresponds to a slow direction change, it then extracts CCPs from this sequence (lines 6–17). After that, it extracts LCCSs from CCPs (line 18). Finally, it extracts a CP from each LCCS (lines 19–22) and adds the start and end points into the CP set (line 23).

For comparison purposes, Fig. 6 depicts experimental results of CP extraction with three different methods, minimum description length (MDL) principle-based method, linear fitting-based method and our CPE method on a single GPS trajectory, which is shown in Fig. 3. Compared to the manually labeled CPs in Fig. 3, we list the precision and recall

Algorithm 1 CPE Method

Input: trajectory $T = p_1 p_2 \dots p_j \dots p_n$; change angle threshold θ_{thrd};
; change angle decrease rate r; linear fitting length l
Output: *CP* set *CPs ;*

```
lastCPP ← l-1;
for j from l-1 to n-l do
 if isCPP(p_j, l, θ_thrd) then // check if p_j is a CPP
  Add p_j into CCPs; // CCPs denotes the set of CCP
  lastCPP ← j;
 else
  if isSlowChange(lastCPP, j, l, θ_thrd) then
   while flag do
    θ_thrd = θ_thrd*(1-r);
     for index from lastCPP + 1 to j do
      if isCPP(p_j, l, θ_thrd) then
       Add p_j into CCPs;
       lastCPP ← j;
       flag ← false;
      end for
   end while
end for
Extract LCCSs from CCPs;
for each S in LCCSs do
 get the point p whose change angle is largest among S;
 add p into CPs;
end for
add p_1 and p_n to CPs. //add the start and end point to CPs
```

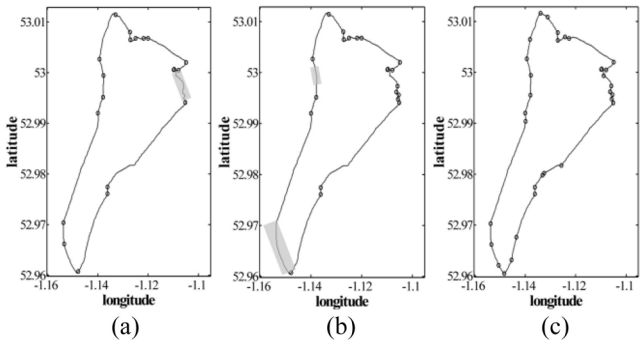


Fig. 6. Results of CP extraction by three different methods. In each subfigure, the trajectories are denoted as solid line and CPs are highlighted in gray hollow points. The light-gray rectangle shadows are used for identifying specific areas. (a) MDL principle-based method. (b) Linear fitting-based method with $\theta_{\text{thrd}} = \pi/6$, $l = 8$. (c) CPE method with $\theta_{\text{thrd}} = \pi/6$, $l = 8$.

of the three methods in Table I. We can see that although the precision of CPE method is slightly lower than the other two methods, its recall is much higher than the other two methods. The CPE method achieves almost 97% detection rate with the false alarm rate 6.1%. Compared to the manually labeled CPs in Fig. 3, we found that the MDL principle-based method is able to extract the CPs in the slow direction change regions, but it ignores the CPs in the regions where there are relatively small but continuous "*S*"- or "*Z*"-shaped direction changes [shown as the light-gray rectangle shadow in Fig. 6(a)]. The method based on linear fitting cannot find the CPs in slow direction change regions [shown as the light-gray rectangle shadow in Fig. 6(b)]. In contrast, the CPE method is superior to the above two methods, since it can not only find the CPs in slow and sharp direction change regions simultaneously, but also catch the CPs in regions that are relatively small but have continuous *S*- or *Z*-shaped direction changes.

TABLE I
EVALUATION OF THREE CP EXTRACTION RESULTS

Method	Precision	Recall
MDL based method	1	0.625
Linear Fitting based method	1	0.656
CPE method	0.939	0.969

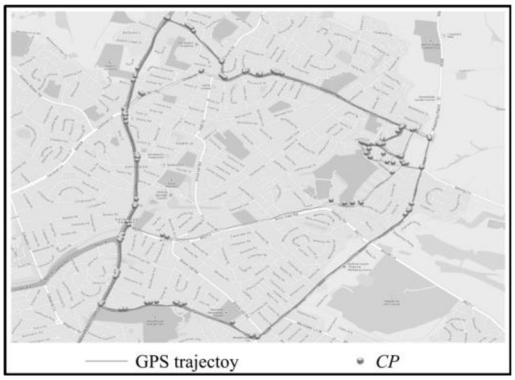


Fig. 7. CPs extraction result by CPE on a trajectory set that contains nine trajectories.

C. Road Corner Detection

In our framework, after extracting CPs, our next task is to detect road corners. Based on the observation 1) in Section I, personal outdoor movement trajectories are restricted by physical roads. Conversely, personal movement trajectories contain physical clues that reflect physical roads.

Fig. 7 shows a CP extraction result by applying the CPE algorithm on a trajectory set that contains nine trajectories. We observe that the CPs form several clusters and these clusters match the road corners perfectly. Based on this observation, we assume that the locations, which contain a number of trajectories direction changes, are road corners. Thus, in our framework we detect road corners by clustering CPs, since CPs capture direction changes.

To cluster CPs, we have the following four requirements for the clustering algorithm.

1) The algorithm should be able to identify the number of clusters automatically. It is impractical to know the number of road corners of a given trajectory set in advance. Thus, the algorithm should figure it out.
2) The algorithm should be able to find CP clusters with different shapes, since CP clusters around corners usually have different spatial shapes.
3) The algorithm should be able to eliminate the "noise" automatically. In this paper, we ignore isolated CPs and clusters containing a few CPs since they will not lead to frequent route.

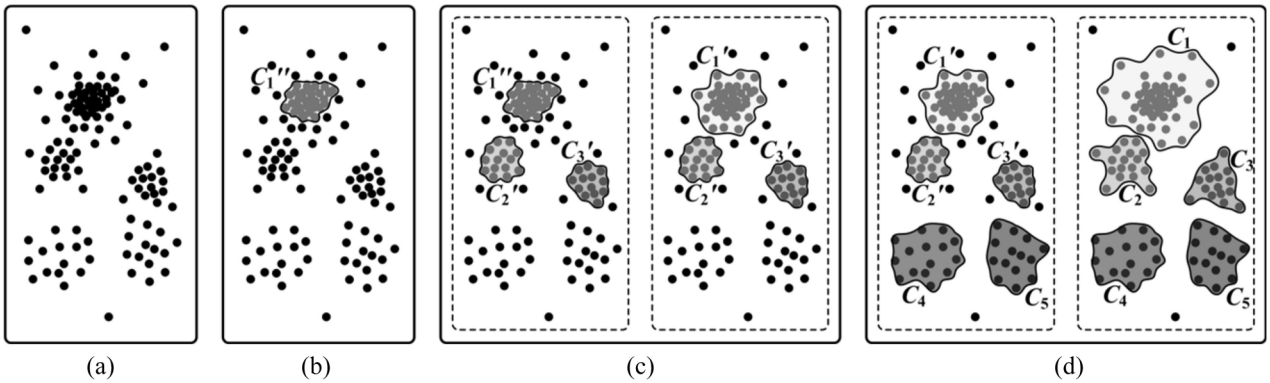


Fig. 8. Clustering process of MDL-DBSCAN. Suppose the HighestDensity: $(\text{Eps}_h, \text{MinPts}_h) = (5, 20)$, LowestDensity: $(\text{Eps}_l, \text{MinPts}_l) = (15, 8)$, DensityLevel $= 3$. (a) CPs. Clustering using (b) first density level $\text{Density}_1 = (\text{Eps}_1, \text{MinPts}_1) = (5, 20)$, (c) second density level $\text{Density}_2 = (\text{Eps}_2, \text{MinPts}_2) = (10, 14)$, and (d) third density level $\text{Density}_3 = (\text{Eps}_3, \text{MinPts}_3) = (15, 8)$.

4) The algorithm should be able to find CP clusters with different densities. Due to different quantities and distribution shapes, the CP clusters always have different densities.

Existing density-based clustering algorithms, such as DBSCAN and OPTICS, can adapt to the first three requirements. However, to the best of our knowledge, none of the clustering algorithms can adapt to the fourth requirement. In this paper, we propose MDL-DBSCAN algorithm that can meet not only the first three requirements but also the fourth requirement.

DBSCAN and OPTICS employ a unique global density threshold in the clustering process. It is the primary reason why they cannot detect the clusters with different densities. Unlike DBSCAN or OPTICS, MDL-DBSCAN clusters the CPs with multiple density thresholds. In MDL-DBSCAN, DBSCAN works as a subprocedure to cluster CPs with a given density threshold. The core techniques of MDL-DBSCAN can be summarized as "one process" and "two constraints."

1) One Process: To extract the CP clusters with different densities, MDL-DBSCAN clusters CPs on MDL iteratively. Specifically, MDL-DBSCAN first introduces three parameters, $\text{HighestDensity} = (\text{Eps}_h, \text{MinPts}_h)$, $\text{LowestDensity} = (\text{Eps}_l, \text{MinPts}_l)$, and DensityLevel to generate multiple density thresholds $\text{Density}_i = (\text{Eps}_i, \text{MinPts}_i)$ $(i = 1, 2, \ldots, \text{DensityLevel})$ by using (3). Then, it employs DBSAN to cluster CPs from the highest density level to lowest density level iteratively. As shown in Fig. 8, the CPs are clustered at three different density levels sequentially. At the first and highest density level, cluster C_1'' is extracted. At the second density level, which is lower than the first one, clusters C_2' and C_3' are extracted. Also, the cluster C_1'' generated in the first step is extended to C_1' because of lower density level. At the third and lowest density level, clusters C_4 and C_5 are extracted. Also, the clusters C_1', C_2', and C_3' generated in the previous steps are also extended to C_1, C_2, and C_3, respectively, as a result of lower density level

$$\begin{cases} \text{Eps}_i = \text{Eps}_l + (i-1)*\dfrac{\text{Eps}_h - \text{Eps}_l}{\text{DensityLevel}} \\ \text{MinPts}_i = \text{MinPts}_l + (i-1)*\dfrac{\text{MinPts}_h - \text{MinPts}_l}{\text{DensityLevel}}. \end{cases} \tag{3}$$

2) Two Constraints: In order to control DBSCAN to perform as the process shown in Fig. 8, we have to add two constraints additionally.

a) The one process may merge the clusters generated at previous density levels together when the current density level is low enough. Thus, we add the following constraint.

Constraint 1: The clusters generated at previous density levels (i.e., high-density levels) cannot be partitioned or merged into other clusters as density level decreased (i.e., lower density levels).

For example, as shown in Fig. 8(d), clusters C_1', C_2', and C_3' are extended to C_1, C_2, and C_3, respectively, and none of them is partitioned or merged into other clusters. In contrast, without this constraint, clusters C_1', C_2', and C_3' may be merged together or part of one cluster is merged into another one. Additionally, we do not forbid the extension of the clusters generated in previous density levels, since it may result in many meaningless small clusters.

b) At each iteration of MDL-DBSCAN, the unlabeled CPs that satisfy current density level may be merged into the clusters generated at previous density levels although they can independently form a new clusters at the current density level. Thus, we add the following constraint.

Constraint 2: The unlabeled point clusters that satisfy the current density level (i.e., the density level, on which, one process is clustering the CPs) should independently form a new cluster instead of merging into the previously generated clusters.

For example, in Fig. 8(c) cluster C_2' is extracted independently rather than merged into C_1'. Similarly, in Fig. 8(d) cluster C_5 is extracted independently rather than merged into C_3'. Without this constraint, cluster C_2' will be merged into cluster C_1', which finally leads to that clusters C_1 and C_2 are merged together. Similarly, cluster C_5 will be merged into C_3', which finally leads to that clusters C_3 and C_5 are merged together.

We implement the first constraint by modifying the definition of neighbors in DBSCAN as follows.

Definition 4: Given a data point p, the neighbors of p are defined as follows.

1) If p has been labeled in last clustering process, its neighbors contain:

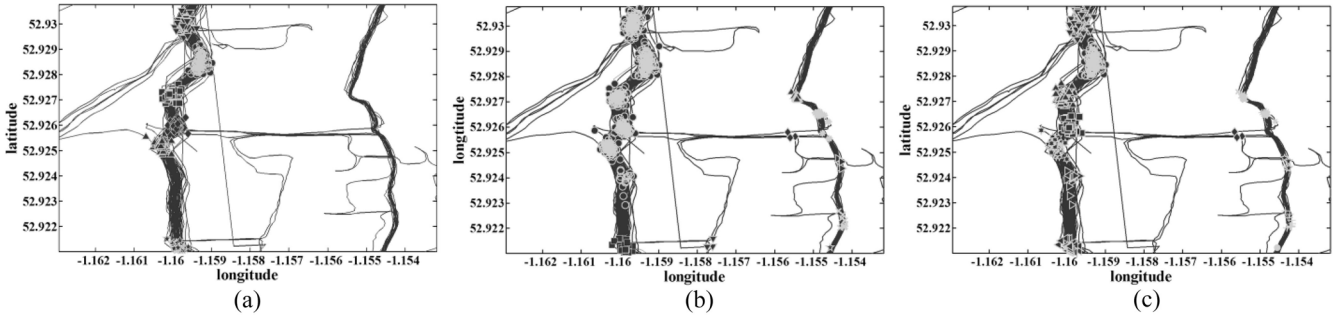


Fig. 9. CPs clustering result on a GPS dataset through DBSCAN and MDL-DBSCAN. In each subfigure, trajectories are highlighted in solid black. CPs are shown as gray points and different shapes indicate different clusters. DBSCAN executes with (a) Eps = 0.2e−3, Minpts = 20 and (b) Eps = 0.5e−3, Minpts = 4. (c) MDL-DBSCAN executes with *l*Eps = 0.5e−3, *l*Minpts = 4, *h*Eps = 0.2e−3, *h*Minpts = 20, levels = 3. Note that, here, we use coordinate distance.

Algorithm 2 CPs Clustering Algorithm

Input: *CP* list *CPl*; *HighestDensity*; *LowestDensity* ; *DensityLevel*
Output: *CPs* with cluster label*;*

1: *startID* ← 0;
2: **for** *j* from 1 to *DensityLevel* **do**
3: Compute the density threshold $Density_j = (Eps_j, MinPts_j)$; /*handling the unlabeled points preferentially */
4: Move the unlabeled *CPs* to the beginning of the *CPl*; /*using the *DBSCAN*, the *neighbors* definition of which has been */ /*modified according to Definition 4, to cluster the *CPs* in *CPl */
5: *startID* ← *DBSCAN*(*CPl*, Eps_j, $MinPts_j$, *startID* + 1); /* let all the *CPs* can be visited again */
6: Change the *VisitedFlag* of each *CP* to *false*;
7: **end for**

a) the points whose cid are same as p.cid;
b) the unlabeled points that satisfy the condition

$$\text{dist}(p, q) < \text{Eps}$$

where q denotes any one of the unlabeled points. cid denotes the cluster ID. The function dist (p, q) is used to measure the distance between point p and q. Eps denotes the radius of ε-neighborhood of p.

2) If p is unlabeled, its neighbors are the unlabeled points that satisfy the condition

$$\text{dist}(p, q) < \text{Eps}$$

where q denotes any one of the unlabeled points.

We implement the second constraint by always handling the unlabeled points preferentially when clustering on each density level.

The pseudocode of MDL-DBSCAN is shown in Algorithm 2. In step 1, MDL-DBSCAN initializes the cluster ID (lines 1). In step 2, MDL-DBSCAN calculates a new density threshold by using (3) (lines 3). In step 3, to comply with Constraint 2 (i.e., ensure unlabeled points have higher priority to be visited), MDL-DBSCAN moves the unlabeled CPs to the beginning of the CP list (line 4). In step 4, to comply with Constraint 1, MDL-DBSCAN clusters the CPs by employing DBSCAN, in which the neighbors of a given point has been redefined (see Definition 4) (line 5). In step 5, to enable all CPs can be visited again in the next iteration, the VisitedFlag of each CP is set as false (lines 6). The steps from 2 to 5 are performed repeatedly until the CPs have been handled at all the density levels.

To demonstrate that MDL-DBSCAN is able to find clusters with different densities, we conduct a CPs clustering experiment on a small dataset that contains 106 movement trajectories by using DBSCAN and MDL-DBSCAN, respectively. The experimental results are illustrated in Fig. 9. We use different shapes to denote different clusters and the CPs in the same cluster have the same shape.

As shown in Fig. 9(a), to identify the clusters in the left part by employing DBSCAN, the density threshold must be set as Eps = $0.2e^{-3}$, Minpts = 20. With this threshold, the clusters in the left part are identified but the clusters in the right part are regarded as noise. It is because that the densities of the clusters in left part are far lower than Eps = $0.2e^{-3}$, Minpts = 20. As shown in Fig. 9(b), to identify the clusters in the right part by employing DBSCAN, the density threshold must be set as Eps = $0.5e^{-3}$, Minpts = 4. With this threshold, the clusters in the right part are identified but the clusters in the left part are merged together. To summarize, by employing DBSCAN to cluster the CPs, we cannot find a density threshold that can simultaneously identify the clusters with different densities. However, as shown in Fig. 9(c), the clusters with different densities are all identified by employing MDL-DBSCAN.

Finally, the center of each CP cluster is extracted as road corner. In summary, the unique characteristic of identifying clusters with different densities enables MDL-DBSCAN to detect almost all involved road corners. What's more, MDL-DBSCAN has the potential of being extended to work on other types of datasets with different cluster densities.

D. Trajectory Mapping

In our framework, the next task is to map each trajectory onto physical roads and construct connectivity matrix (CM) among all involved corners. Then, we can get frequent routes by retrieving CM with a given frequency threshold.

1) Trajectory Mapping:

Definition 5: Given a CPs cluster $\{\text{CP}_1, \text{CP}_2, \ldots, \text{CP}_m\}$, $\text{CP}_i = (x_i, y_i)$, $i = 1, 2, \ldots, m$. The center of this cluster, $C(x, y)$, is denoted as

$$\begin{cases} x = \dfrac{1}{m} \sum_{i=1}^{m} x_i \\ y = \dfrac{1}{m} \sum_{i=1}^{m} y_i. \end{cases} \tag{4}$$

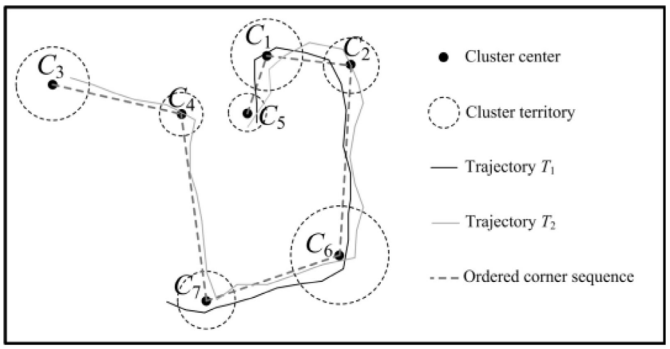


Fig. 10. Example of trajectory mapping.

Definition 6: Given a CPs cluster $\{\mathrm{CP}_1, \mathrm{CP}_2, \ldots, \mathrm{CP}_m\}$, $\mathrm{CP}_i = (x_i, y_i)$, $i = 1, 2, \ldots, m$. The radius, R, of this cluster, is denoted as

$$R = \max\{D_1, D_2, \ldots, D_m\} \tag{5}$$

where the D_i, $i = 1, 2, \ldots, m$, is denoted as

$$D_i = \mathrm{dist}(\mathrm{CP}_i, C) \tag{6}$$

where the function $\mathrm{dist}(\mathrm{CP}_i, C)$ is used for measuring the distance between points CP_i and C (center of this cluster).

Definition 7: Cluster territory is defined as a circle whose center and radius are defined as the cluster center (Definition 5) and the cluster radius (Definition 6), respectively.

For each trajectory, the trajectory mapping procedure checks it from its starting point to the end point. If it traverses one cluster territory, the center of this cluster will be added into the ordered sequence of this trajectory. Since each cluster center corresponds to a road corner, the trajectory mapping procedure actually transforms each original trajectory into an ordered sequence of road corners. From the ordered sequences, we can obtain the connectivity information among involved road corners and the number of trajectories mapped onto each physical road segment.

An example of trajectory mapping is illustrated in Fig. 10. Our trajectory mapping method maps trajectory T_1 (highlighted in black solid line) as the ordered corner sequence $C_3C_4C_7C_6C_2C_1C_5$ and trajectory T_2 (depicted as the gray solid line) as the ordered corner sequence $C_7C_6C_2C_1C_5$.

2) Connectivity Matrix Construction: Given the ordered corner sequence of each trajectory, we construct CM to formally represent the connectivity between road corners and record the number of trajectories mapped onto each road segment. In an $N \times N$ **CM**, where N is the number of detected road corners, the element $\mathbf{CM}(i, j)$ $(i, j = 1, 2, \ldots, N)$ denotes the number of trajectories whose ordered corner sequences contain C_iC_j. At the beginning, **CM** is set to an $N \times N$ zero matrix. Then, given an ordered corner sequence S of a trajectory, we increase the matrix element $\mathbf{CM}(i, j)$ and $\mathbf{CM}(j, i)$ by 1 when the corner sequence C_iC_j occurs in S. Note that, in our framework, the connection between two corners is undirected.

For example, as shown in Fig. 10, there are seven corners $C_1, C_2, \ldots, C_7$ and two trajectories T_1 and T_2. The ordered corner sequences of T_1 and T_2 are $C_3C_4C_7C_6C_2C_1C_5$ and $C_7C_6C_2C_1C_5$, respectively. As shown in Fig. 11, we initialize the **CM** as the left zero matrix. After T_1 has been processed, **CM** is updated to the middle matrix. The elements $\mathrm{CM}(C_3, C_4)$, $\mathrm{CM}(C_4, C_3)$, $\mathrm{CM}(C_4, C_7)$, $\mathrm{CM}(C_7, C_4)$, $\mathrm{CM}(C_7, C_6)$, $\mathrm{CM}(C_6, C_7)$, $\mathrm{CM}(C_6, C_2)$, $\mathrm{CM}(C_2, C_6)$, $\mathrm{CM}(C_2, C_1)$, $\mathrm{CM}(C_1, C_2)$, $\mathrm{CM}(C_1, C_5)$, and $\mathrm{CM}(C_5, C_1)$ are increased by 1. After T_2 has been processed, CM is updated to the right matrix. The elements $\mathrm{CM}(C_7, C_6)$, $\mathrm{CM}(C_6, C_7)$, $\mathrm{CM}(C_6, C_2)$, $\mathrm{CM}(C_2, C_6)$, $\mathrm{CM}(C_2, C_1)$, $\mathrm{CM}(C_1, C_2)$, $\mathrm{CM}(C_1, C_5)$, and $\mathrm{CM}(C_5, C_1)$ are increased by 1.

$$\begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \end{array}\!\!\begin{array}{c} C_1C_2C_3C_4C_5C_6C_7 \\ \begin{pmatrix} 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \end{pmatrix} \end{array}\quad \text{(a)}$$

$$\begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \end{array}\!\!\begin{array}{c} C_1C_2C_3C_4C_5C_6C_7 \\ \begin{pmatrix} 0&1&0&0&1&0&0 \\ 1&0&0&0&0&1&0 \\ 0&0&0&1&0&0&0 \\ 0&0&1&0&0&0&1 \\ 1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&1 \\ 0&0&0&1&0&1&0 \end{pmatrix} \end{array}\quad \text{(b)}$$

$$\begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \end{array}\!\!\begin{array}{c} C_1C_2C_3C_4C_5C_6C_7 \\ \begin{pmatrix} 0&2&0&0&2&0&0 \\ 2&0&0&0&0&2&0 \\ 0&0&0&1&0&0&0 \\ 0&0&1&0&0&0&1 \\ 2&0&0&0&0&0&0 \\ 0&2&0&0&0&0&2 \\ 0&0&0&1&0&2&0 \end{pmatrix} \end{array}\quad \text{(c)}$$

Fig. 11. CM updating process. (a) Initialization of CM. (b) Updated result after T_1 has been processed. (c) Updated result after T_2 has been processed.

Finally, after all the trajectories have been processed, we can obtain frequent routes that a user has traversed for at least F times by retrieving **CM** using F as the given frequency threshold.

E. Frequent Route Evaluation Methods

We evaluate the extracted frequent routes from two aspects. First, we measure the closeness from them to the corresponding physical roads. Second, by using the manually labeled frequent routes as ground truth, we analyze their precision and recall.

1) Closeness to Physical Roads: How close the extracted frequent routes are to the physical roads is an important indicator to evaluate our framework. The ideal method of measuring the distance between physical roads and extracted frequent routes should compute the average point-to-line distance from the sample points of physical roads to the corresponding extracted frequent routes. However, the ideal method is infeasible since the numerical information of the involved roads is not available from electronic map applications such as Google Maps, and we can only obtain maps in the form of pictures. Therefore, we employ an approximate method, which measures the average point-to-line distance from the sample points of GPS trajectories to the corresponding frequent routes.

Given a trajectory T, our framework first maps it to an ordered corner sequence. We ignore the trajectory directly when its ordered corner sequence is empty. An empty ordered corner sequence indicates that the trajectory is isolated from the majority and will not result in frequent route. Second, our framework checks whether each segment in ordered corner sequence is contained in the extracted frequent routes or not. For example, in Fig. 12, the mapping result of trajectory T is $C_6C_2C_1C_5C_4C_3$ and the segments C_2C_1, C_1C_5, C_5C_4, and C_4C_3 are detected as frequent route segments (C_6C_2, which is highlighted as the thick black dotted segment, is not a frequent route segment). Third, for each frequent segment, we construct a straight line, named scan line, which is perpendicular to the

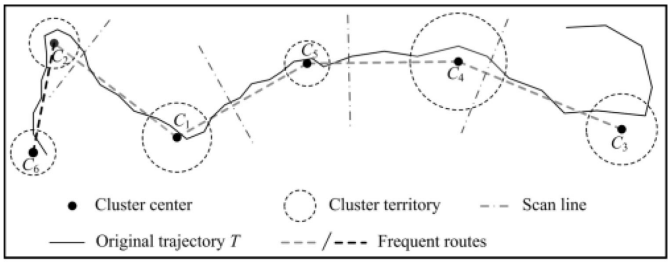


Fig. 12. Example of calculating the average distance from a trajectory to the corresponding frequent route.

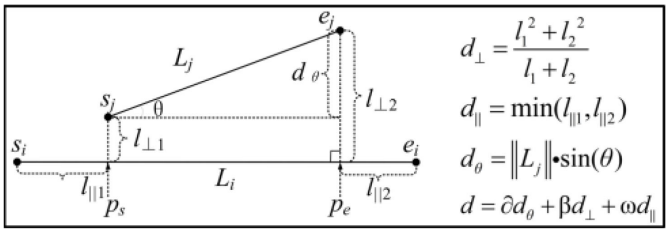


Fig. 13. Abstract distance function [15].

current frequent route segments (shown as the dashed-dotted lines in Fig. 12). For each frequent route segment, we move its scan line and calculate the Euclidean distance from each GPS point of T to the intersection point on frequent route segment.

Suppose: 1) the ordered corner sequence of trajectory T contains m frequent road segments and 2) the GPS points traversed by the scan line of the ith ($i = 1, 2, \ldots, m$) frequent segment are denoted as p_{ij} ($j = 1, 2, \ldots, n_i$, n_i is the total number of GPS points) and the corresponding intersection is denoted as $\text{foot}_\perp(p_{ij})$. Then, the closeness $C(T)$ from a trajectory T to the corresponding frequent route can be calculated as

$$C(T) = \sum_{i=1}^{m} \sum_{j=1}^{ni} \text{Dist}\big(p_{ij}, \text{foot}_\perp\big(p_{ij}\big)\big) \Big/ \sum_{i=1}^{m} n_i \,. \tag{7}$$

For the input of our framework $\text{TS} = \{T_1, T_2, \ldots, T_n\}$, the total closeness TC(TS) is calculated as

$$\text{TC(TS)} = \frac{1}{n} \sum_{t=1}^{n} C(T_t). \tag{8}$$

TC(TS) is finally transformed into the distance in meters with the aid of Google Map distance conversion tool.

2) Precision and Recall: To determine whether the extracted frequent routes are effective, we compare the extracted frequent routes with the manually labeled frequent routes. For quantitative description purposes, we utilize precision and recall, which are popularly used in information retrieval, to characterize the extracted frequent routes.

Due to the fact that the extracted frequent routes and the manually labeled frequent routes are impossible to completely overlap, we use an abstract distance function [15], as shown in Fig. 13, to measure the similarity (note that, here the similarity is just for evaluating the effectiveness of our framework).

Suppose the extracted frequent routes $\text{eFR} = \{\text{eS}_1, \text{eS}_2, \ldots, \text{eS}_{\text{Me}}\}$, where $\text{eS}_j (j = 1, 2, \ldots, \text{Me})$ denotes an extracted frequent route segment; the manually labeled frequent routes $\text{mFR} = \{\text{mS}_1, \text{mS}_2, \ldots, \text{mS}_{\text{Mm}}\}$, where $\text{mS}_i, (i = 1, 2, \ldots, \text{Mm})$ denotes the manually labeled frequent route segment. Given an eS_j, we search the mS_t that satisfies the condition

$$d\big(\text{eS}_j, \text{mS}_t\big) < d\big(\text{eS}_j, \text{mS}_i\big), i = 1, 2, \ldots, \text{Mm}, i \neq t$$

if $d(\text{eS}_j, \text{mS}_t)$ is smaller than the given distance threshold and the distance threshold is small enough, it is reasonable to conclude that eS_j matches with mS_t (extensive experiments demonstrate that when the threshold value is set as 0.003, the confidence coefficient is larger than 0.98). Suppose there are $\text{Amount}_{\text{mS\_matched}}$ eSs are matched with eSs, we calculate the values of precision (P) and and recall (R) using

$$\begin{cases} P = \dfrac{\text{Amount}_{\text{mS\_matched}}}{\text{Me}} \\ R = \dfrac{\text{Amount}_{\text{mS\_matched}}}{\text{Mm}}. \end{cases} \tag{9}$$

V. Experimental Evaluation

In this section, we conduct experiments to evaluate our framework following the process stated in Section IV-E.

A. Experimental Setup

In the experiments, we use a real-world individual daily outdoor movement GPS dataset, which is collected from more than 6800 individuals who lived in more than ten countries within one year (from January 1, 2012 to December 31, 2012). In this dataset, each individual collects his or her own GPS data by a smartphone equipped with the software named "Runkeeper" (http://runkeeper.com/). The GPS sampling rate is set to one record per 4 s. For the purpose of simplicity and visualization, we restrict our interest to the areas that contain most of the trajectories of individual.

The frequent threshold F should be properly set depending on the size of the dataset of individual. In this experiment, we set it as

$$F = \max\{6, [0.02n]\} \tag{10}$$

where n denotes the size of a dataset.

To provide a quantitative evaluation, we pick up four trajectory groups, which contain trajectories from different numbers of individuals and have no individual overlap. For each trajectory dataset of an individual, in all the four groups, we ask three volunteers to manually label whether a road segment (a line segment between two adjacent road corners) is a frequent route or not corresponding to the frequent threshold. In particular, if one volunteer thinks the road segment is a frequent route, it is labeled as frequent. The datasets are summarized in Table II, and the first two individuals' trajectories in group G_1 are illustrated in Fig. 14.

For comparison purposes, we use the line segment clustering-based method as a baseline, whose basic idea is to cluster the trajectory segments according to the similarity between trajectory segments. The evaluation criteria we used

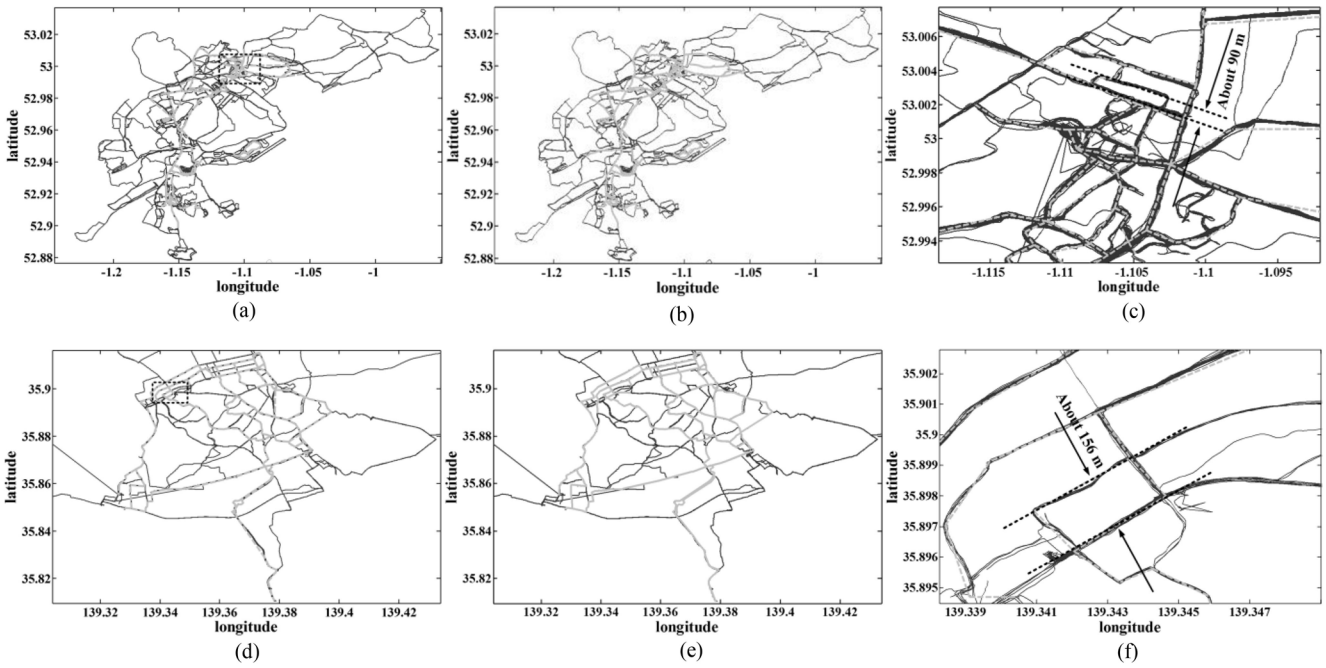


Fig. 14. Visualization of results on two individual's trajectory datasets in G_1. In each subfigure, original trajectories are highlighted in black solid. (a) Extracted frequent routes from the trajectory dataset of the first individual, where extracted frequent routes are highlighted in gray dotted. (b) Manually labeled frequent routes on the trajectory dataset of the first individual, where manually labeled frequent routes are highlighted in gray solid. (c) Enlarged result of black dotted rectangle region in (a). (d) Extracted frequent routes from the trajectory dataset of the second individual, where extracted frequent routes are highlighted in gray dotted. (e) Manually labeled frequent routes on the trajectory dataset of the second individual, where manually labeled frequent routes are highlighted in gray solid. (f) Enlarged result of black dotted rectangle region in (d).

TABLE II
DATASETS USED IN OUR EXPERIMENTS

Group	Number of individuals	Lower limit	Average
G_1	5	300	468
G_2	10	250	349
G_3	15	200	287
G_4	20	150	232

Each individual in different Group has more than *Lower limit* number of trajectories. The average number of trajectories of each individual in each group is *Average*.

TABLE III
AVERAGE CLOSENESS TO REAL ROADS COMPARISON

	G_1	G_2	G_3	G_4
Our method	8.54m	11.87m	13.82m	8.71m
Line	9.19m	12.88m	14.75m	10.83m
vsd	125.34m	133.06m	137.94m	124.87m

for each group include average closeness to physical roads (introduced in Section IV-E), average precision and recall. Obviously, a good frequent route extraction method should not only low average closeness to physical roads, but also high average precision and recall.

The evaluations are conducted on a server with Intel Xeon E5405 and 8 GB RAM.

B. Experimental Results

We first show the results of first two individuals' trajectories of G_1 in Fig. 14, where the manually labeled frequent routes and the frequent routes extracted by our framework are depicted.

From Fig. 14, we observe that the extracted frequent routes are very similar to the manually labeled frequent routes. This observation indicates that our framework is effective in extracting frequent routes. Moreover, to illustrate that the framework performance on average closeness to physical roads is good enough, we enlarge the region where the roads are visually densest and measure the visually smallest distance between physical roads by Google Map distance measuring tool [shown in Fig. 14(c) and (f)]. Table III compares the average closeness to physical roads of our framework to that of the baseline method and the corresponding average visually smallest distance between physical roads (for simplicity, we denote it as vsd). From the table, we observe that our framework achieves smaller average closeness to physical roads for each group than that of the baseline method. It is because that the baseline method always ignores the compact *Z*- or *S*-shape trajectory direction changes (shown in Fig. 6), but our framework does not. In addition, our framework extracts frequent routes-based on detecting road corners, which are very close to physical roads. Thus, our method achieves smaller average closeness to the physical roads. Additionally, we observe that the average closeness to physical roads of our and baseline methods are both far less than the visually smallest distance between physical roads. It indicates that the extracted frequent routes have sufficient resolution to be utilized by trajectory-based applications.

Tables IV and V compare the average precision and recall of our framework to those of the baseline method. From the tables, we observe that our framework achieves relatively

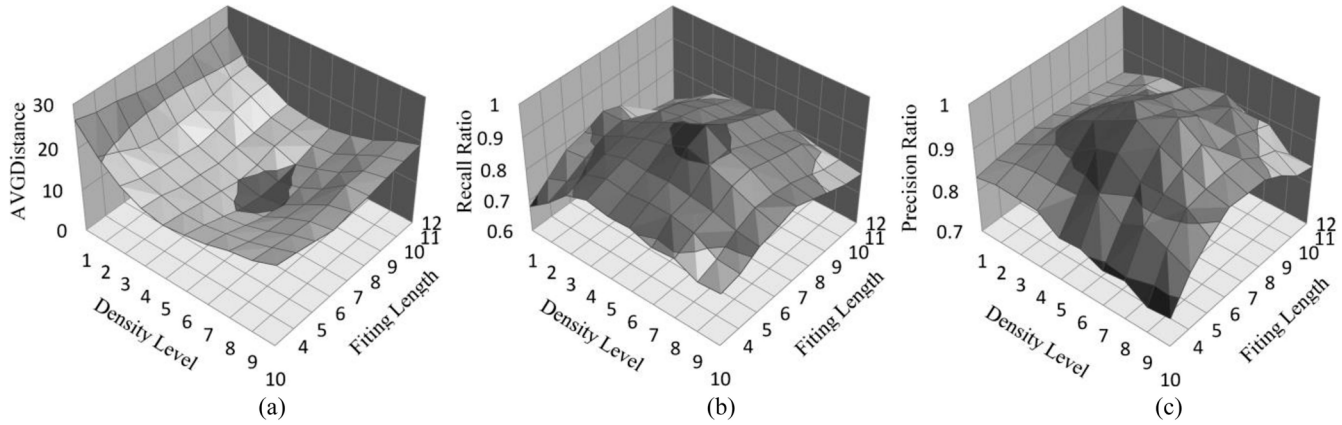


Fig. 15. Parameter effect on frequent routes extraction. The effect of two most important parameters: density level and fitting length is presented. (a) Average distance to real roads as these two parameters is varied. (b) Precision ratio as these two parameters is varied. (c) Recall ratio as these two parameters is varied.

TABLE IV
AVERAGE PRECISION COMPARISON

	G_1	G_2	G_3	G_4
Our method	0.964	0.975	0.979	0.982
Line	0.917	0.906	0.919	0.876

TABLE V
AVERAGE RECALL COMPARISON

	G_1	G_2	G_3	G_4
Our method	0.915	0.921	0.936	0.923
Line	0.814	0.774	0.738	0.712

higher average precision (> 0.96 on all groups) and recall (> 0.91 on all groups), while the baseline method achieves relatively lower average precision (< 0.92 on all groups) and recall (< 0.82 on all groups). Our framework outperforms the baseline approach by 7.8% in terms of average precision and 21.9% in terms of average recall. It is because that the baseline method uses density-based clustering algorithm with a unique global density threshold. The algorithm cannot detect the trajectory clusters with different densities simultaneously, so it fails to detect some frequent routes. In contrast, the MDL-DBSCAN algorithm in our framework is able to detect CP clusters with different densities and then detect all the involved road corners traversed more than F times by an individual. So it will decrease the possibility of missing frequent routes.

Additionally, we have conducted the same experiments with different parameter settings. Fig. 15 illustrates the results from the first individual's 498 trajectories in G_1. F is set according to formula (10). The minimum closeness to real roads is achieved at $l = 8$, levels $= 6$; the maximum precision is achieved at $l = 7$, levels $= 6$ or $l = 8$, levels $= 7$; the maximum recall is achieved at $l = 8$, levels $= 6$. Since the three performance metrics, closeness to physical roads, precision and recall cannot be simultaneously optimized at a certain combination of l and levels, we suggest ($l = 8$, levels $= 6$) to be a good tradeoff setting for our framework on the dataset. Generally speaking, the best performance is achieved at $l = [6, 9]$, levels $= [4, 7]$ for all groups.

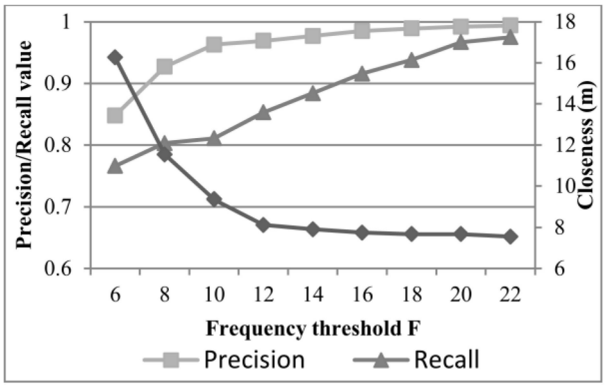


Fig. 16. Frequent routes extraction performance as different frequency threshold F is varied.

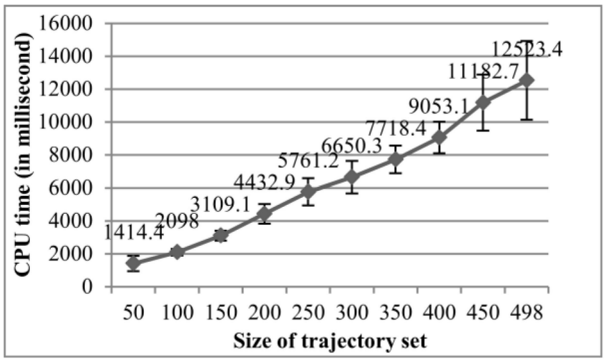


Fig. 17. Processing time as data scale is varied.

In Section V-A, although we suggest that the frequency threshold F is set according to formula (10), a user can set F to different values. Fig. 16 shows the frequent routes extraction performance on the first individual's trajectories in G_1 as frequency threshold F is varied. We can see that the precision and recall gradually approach 1, while the closeness to real road tends to stabilize at about 7.6 m as F is varied.

Fig. 17 demonstrates how the processing time changes with respect to the size of trajectory set, where l is set to 8 and levels is set to 6. From the figure, we observe that the processing time approximately linearly increases as the trajectory set size increases. This result indicates that, as the trajectory

set grows, the computation time of our framework will not increase dramatically.

VI. Conclusion

Frequent routes are an important context for trajectory-based applications. In this paper, we propose a novel framework to extract frequent routes from personal GPS trajectories. In our framework, we first propose a CPE method to extract CPs, which characterize the physical roads. Second, we propose the MDL-DBSCAN clustering algorithm to locate road corners. Instead of using a unique global density threshold, MDL-DBSCAN uses several density thresholds to cluster the CPs at MDL. Third, we propose a method to map all the trajectories on physical roads and detect frequent routes. Finally, we evaluate our framework on real personal GPS trajectory datasets. The results demonstrate that our framework outperforms the baseline approach by 7.8% in terms of average precision and 21.9% in terms of average recall.

In the future, we plan to extend this paper in two directions. First, we attempt to exploit other information embedded in GPS trajectories such as time stamps for frequent route extraction. Second, we intend to develop practical applications, such as disorientation detection and movement trend prediction, leveraging the techniques developed in this framework.

Acknowledgment

The authors would like to thank the L. Bai, X. Zheng, and D. Lu for their assistance in the experiments.

References

[1] L. Chen, M. Lv, Q. Ye, G. Chen, and J. Woodward, "A personal route prediction system based on trajectory data mining," *Inf. Sci.*, vol. 181, no. 7, pp. 1264–1284, Apr. 2011.

[2] L. Chen, M. Lv, Q. Ye, and G. Chen, "A system for destination and future route prediction based on trajectory mining," *Pervas. Mobile Comput.*, vol. 6, no. 6, pp. 657–676, Dec. 2010.

[3] C. Anagnostopoulos and S. Hadjiefthymiades, "Intelligent trajectory classification for improved movement prediction," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 10, pp. 1301–1314, Apr. 2014.

[4] Q. Lin *et al.*, "Disorientation detection by mining GPS trajectories for cognitively-impaired elders," *Pervas. Mobile Comput.*, vol. 19, pp. 71–85, May 2015.

[5] Q. Lin, D. Zhang, X. Huang, H. Ni, and X. Zhou, "Detecting wandering behavior based on GPS traces for elders with dementia," in *Proc. ICARCV*, Guangzhou, China, 2012, pp. 672–677.

[6] E. Lu, C. Chen, and V. Tseng, "Personalized trip recommendation with multiple constraints by mining user check-in behaviors," in *Proc. 20th ACM SIGSPATIAL GIS*, Redondo Beach, CA, USA, 2012, pp. 209–218.

[7] Z. Chen, H. Shen, and X. Zhou, "Discovering popular routes from trajectories," in *Proc. IEEE ICDE*, Hanover, Germany, 2011, pp. 900–911.

[8] C. Chen, D. Zhang, N. Li, and Z. Zhou, "B-Planner: Planning bidirectional night bus routes using large-scale taxi GPS traces," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 4, pp. 1451–1465, Aug. 2014.

[9] J. Zhou, C. L. P. Chen, L. Chen, and W. Zhao, "A user-customizable urban traffic information collection method based on wireless sensor networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1119–1128, Sep. 2013.

[10] C. L. P. Chen, J. Zhou, and W. Zhao, "A real-time vehicle navigation algorithm in sensor network environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1657–1666, Dec. 2012.

[11] V. Zheng, Y. Zheng, X. Xie, and Q. Yang, "Towards mobile intelligence: Learning from GPS history data for collaborative recommendation," *Artif. Intell.*, vol. 184, pp. 17–37, Jun. 2012.

[12] H. Yoon, Y. Zheng, X. Xie, and W. Woo, "Smart itinerary recommendation based on user-generated GPS trajectories," in *Proc. UIC*, Xi'an, China, 2010, pp. 19–34.

[13] Y. Zheng and X. Xie, "Learning travel recommendations from user-generated GPS traces," *ACM Trans. Intell. Syst. Technol.*, vol. 1, no. 1, pp. 1–29, Jan. 2011.

[14] P. Kalnis, N. Mamoulis, and S. Bakiras, "On discovering moving clusters in spatio-temporal data," in *Proc. SSTD*, Angra dos Reis, Brazil, 2005, pp. 364–381.

[15] J. Lee, J. Han, and K. Whang, "Trajectory clustering: A partition-and-group framework," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Beijing, China, 2007, pp. 593–604.

[16] Z. Li, J. Lee, X. Li, and J. Han, "Incremental clustering for trajectories," in *Proc. DASFAA*, Tsukuba, Japan, 2010, pp. 32–46.

[17] M. Nanni and D. Pedreschi, "Time-focused clustering of trajectories of moving objects," *J. Intell. Inf. Syst.*, vol. 27, no. 3, pp. 267–289, Nov. 2006.

[18] C. Hung, W. Peng, and W. Lee, "Clustering and aggregating clues of trajectories for mining trajectory patterns and routes," *VLDB J.*, vol. 24, no. 2, pp. 1–24, Nov. 2011.

[19] C. Hung, L. Wei, and W. Peng, "Clustering clues of trajectories for discovering frequent movement behaviors," in *Behavior Computing*. London, U.K.: Springer, 2012, pp. 179–196.

[20] J.-G. Lee, J. Han, X. Li, and H. Gonzalez, "TraClass: Trajectory classification using hierarchical region-based and trajectory-based clustering," *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 1081–1094, Aug. 2008.

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

[22] M. Ankerst, M. Breunig, H. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," in *Proc. SIGMOD*, Philadelphia, PA, USA, 1999, pp. 49–60.

[23] H. Cao, N. Mamoulis, and D. W. Cheung, "Mining frequent spatiotemporal sequential patterns," in *Proc. ICDM*, Houston, TX, USA, 2005, pp. 82–89.

[24] F. Verhein and S. Chawla, "Mining spatio-temporal association rules, sources, sinks, stationary regions and thoroughfares in object mobility databases," in *Proc. DASFAA*, Singapore, 2006, pp. 187–201.

[25] N. Mamoulis *et al.*, "Mining, indexing, and querying historical spatiotemporal data," in *Proc. SIGKDD*, Seattle, WA, USA, 2004, pp. 236–245.

[26] H. Jeung, H. T. Shen, and X. Zhou, "Mining trajectory patterns using hidden Markov models," in *Proc. DaWaK*, Regensburg, Germany, 2010, pp. 470–480.

[27] H. Jeung, Q. Liu, H. T. Shen, and X. Zhou, "A hybrid prediction model for moving objects," in *Proc. ICDE*, Cancún, Mexico, 2008, pp. 70–79.

[28] J. Biagioni, T. Gerlich, T. Merrifield, and J. Eriksson, "Easytracker: Automatic transit tracking, mapping, and arrival time prediction using smartphones," in *Proc. SenSys*, Seattle, WA, USA, 2011, pp. 68–81.

Tianben Wang received the B.S. degree in computer science from Northwest A&F University, Yangling District, China, in 2011, and the M.S. degree in computer application technology from Northwestern Polytechnical University, Xi'an, China, in 2013, where he is currently pursuing the Ph.D. degree.

His current research interests include ubiquitous computing and intelligent elder assisting technology.

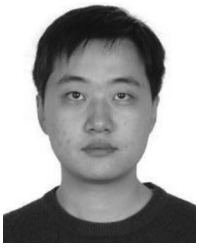
Daqing Zhang received the Ph.D. degree from the University of Rome "La Sapienza", Rome, Italy, and the University of L'Aquila, L'Aquila, Italy, in 1996.

He is currently a Professor with the School of Computer Science, Northwestern Polytechnical University, Xi'an, China. He is also a Professor at Institut Mines-TELECOM/TELECOM SudParis, Evry, France. He has published over 180 referred journal and conference papers, all his research has been motivated by practical applications in digital cities, mobile social networks, and elderly care. His current research interests include large-scale data mining, urban computing, context aware computing, and ambient assistive living.

Dr. Zhang was a recipient of the Best Paper Runner Up Award from Mobiquitous in 2011, the Best Paper Award from the IEEE International Conference on Ubiquitous Intelligence and Computing in 2012, and the winner of the Ten Years CoMoRea Impact Paper Award from the IEEE International Conference on Pervasive Computing and Communications in 2013. He is an Associate Editor of four journals, including *ACM Transactions on Intelligent Systems and Technology*. He has been a frequent Invited Speaker in various international events on ubiquitous computing.

Xingshe Zhou received the M.S. degree from Northwestern Polytechnical University, Xi'an, China, in 1984.

He is a Professor with the School of Computer Science, Northwestern Polytechnical University, and the Director with the Shaanxi Key Laboratory of Embedded System Technology, Xi'an. His current research interests include embedded computing and pervasive computing.



Xin Qi received the B.Sc. degree in computer science from Nanjing University, Nanjing, China, in 2007, and the M.Eng. degree in pattern recognition and intelligent systems from the National Key Laboratory of Pattern Recognition at Institute of Automation, Chinese Academy of Science, Beijing, China, in 2010. He is currently pursuing the Ph.D. degree with the Department of Computer Science, College of William and Mary, Williamsburg, VA, USA.

His current research interests include ubiquitous computing and mobile systems.

Hongbo Ni received the Ph.D. degree from Northwestern Polytechnical University, Xi'an, China, in 2009.

He is an Associate Professor with the School of Computer Science, Northwestern Polytechnical University. His current research interests include pervasive computing, embedded computing, and system.

Haipeng Wang received the Ph.D. degree from Northwestern Polytechnical University, Xi'an, China, in 2007.

He is an Associate Professor with the School of Computer Science, Northwestern Polytechnical University. His current research interests include human–computer interaction, machine learning, and health informatics.

Gang Zhou (SM'14) received the Ph.D. degree from the University of Virginia, Charlottesville, VA, USA, in 2007.

He is an Associate Professor with the Department of Computer Science, College of William and Mary, Williamsburg, VA, USA. He is a Senior Member of IEEE and also a Senior Member of ACM. His current research interests include ubiquitous computing, mobile computing, body sensor networks, and wireless networks.