SPECIAL SECTION ON ARTIFICIAL INTELLIGENCE (AI)-EMPOWERED INTELLIGENT TRANSPORTATION SYSTEMS

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 18, 2020, accepted May 25, 2020, date of publication May 27, 2020, date of current version June 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2998063

Segmented Trajectory Clustering-Based Destination Prediction in IoVs

CHAO WANG[1], (Member, IEEE), JITONG LI[1], YUNHUA HE[1], (Member, IEEE), KE XIAO[1], AND CHUNQIANG HU[2], (Member, IEEE)

[1]School of Information Science and Technology, North China University of Technology, Beijing 100144, China
[2]School of Big Data and Software Engineering, Chongqing University, Chongqing 400044, China

Corresponding author: Chunqiang Hu (hcq0394@gmail.com)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802300, in part by the National Natural Science Foundation of China under Grant 61702062, Grant 61802004, and Grant 61802005, in part by the Scientific Foundation for Yuyou Talents, in part by the Fundamental Research Funds for the Central Universities under Grant 2019CDQYRJ006, in part by the Chongqing Research Program of Basic Research and Frontier Technology under Grant cstc2018jcyjAX0334, in part by the Key Project of Technology Innovation and Application Development of Chongqing under Grant cstc2019jscx-mbdxX0044, and in part by the Overseas Returnees Innovation and Entrepreneurship Support Program of Chongqing under Grant cx2018015.

ABSTRACT Location-based services have important applications in IoVs, and especially the destination-related applications have attracted more and more attention. Due to privacy consideration or operation convenience, people hesitate to share destinations to the public. Thus, these applications need to predict the destinations of moving vehicles in order to provide better services. Some existing works on destination prediction suffer from the dataset sparsity problem or the model inaccuracy problem. To overcome these problems, a Segmented Trajectory Clustering-Based Destination Prediction mechanism is proposed in this paper. First, each original trajectory is segmented to several key sub-trajectories, with the DP-based trajectory segmentation algorithm. Then, all the sub-trajectories are clustered based on the average nearest point pair distance to reveal the common characteristics or similar tracks. Finally, a deep neural network-based model is utilized to predict destinations, according to the history trajectories. Extensive simulations are conducted for destination predictions. Simulation results show that our proposed method can predict destinations with acceptable average errors and outperform other methods in most of the cases.

INDEX TERMS Location-based service, destination prediction, trajectory segmentation, trajectory clustering, deep learning.

I. INTRODUCTION

The location-based services in the Internet of Vehicles(IoVs) have lots of attractive applications [1]–[3], such as navigation service, accident alarm service, and advertisement publish service. In the location-based services, the location information is the core part of the services, which should be well collected. With the help of the Global Positioning System(GPS) devices, the location information of vehicles can be acquired and then forwarded to the center service providers, via wireless communication techniques, including WAVE, LTE-V or cellular services [4]. The center service providers make suggestions according to their services and locations, which should be sent to the requested vehicles to make recommendations [5]. Among numerous location-based services in IoVs, the destination-related applications play important roles and provide much more convenience to people [6]. However, how to get the destination of each vehicle's trip is a significant problem that should be well solved to serve those applications.

From the perspective of service providers, it is hoped that each vehicle can public the destination of its current trip before its starting, in order to provide better services. However, due to the consideration of privacy protection [7] or operation convenience, most of the vehicle users hesitate to share their destination information. Thus, predicting the final destination of each vehicle becomes an effective way, which can help the service providers analyze vehicle behaviors. First, transportation officials can manage the road traffic better, such as traffic congestion control. Second, based on the destination prediction service, some recommendation systems such as the advertisement delivery system can be

The associate editor coordinating the review of this manuscript and approving it for publication was Chao Chen.

implemented. For example, significant changes have taken place in the field of taxi dispatch. The dispatch system should determine to notify which taxi to pick up a passenger. To achieve this purpose, the destination of each moving vehicle should be predicted. This problem has been one of the Kaggle challenges [8].

Reviewing existing works on the destination prediction, Markov and Hidden Markov Models(HMM) [9], [10] are widely used. Although these models work at a certain extent, vehicle trajectories do not always conform to the Markov process so that the prediction result is not good enough sometimes, especially when the trajectory is very long. Besides, the method of trajectory matching with historical trajectories is also proposed. However, the dataset sparsity problem is a significant obstacle to achieve an accurate prediction result [11]. In addition, some works [12], [13] predict the final destination with the method of clustering. Once all the trajectories have been clustered, by assigning each new trajectory to the clustered trajectories, the destination of each trajectory can be predicted. But, these methods are usually based on the shape of trajectories, but neglect the distance between trajectories, which may lead to worse prediction accuracy.

In this paper, we incorporate some of the above ideas and propose a segmented trajectory clustering-based destination prediction algorithm. The algorithm contains three parts, namely trajectory segmentation, sub-trajectory clustering, and destination prediction. In order to avoid the disadvantages of clustering entire trajectories, a Douglas-Peucker (DP)-based method of segmenting trajectories to common sub-trajectories is utilized. Once trajectories are segmented, some key sub-trajectories may be very similar or have common characteristics. Then we perform clustering operation on the sub-trajectories to classify the similar sub-trajectories. Here, the similarity of two sub-trajectories is defined as the average nearest point pair distance, detailed in the following section. Finally, a GRU based deep learning network is proposed, with which the destinations of moving vehicles can be predicted. Extensive simulations are done to evaluate the effectiveness of our proposed method. The result shows that our proposed method can well predict the destinations of moving vehicles with relatively smaller prediction errors, compared to three existing methods.

The main contributions of this paper are listed as follows:

1) A DP-based Trajectory Segmentation Algorithm is proposed for segmenting trajectories to several key sub-trajectories.
2) A Sub-trajectory Clustering Algorithm is defined, which can classify similar sub-trajectories to reveal common characteristics.
3) A deep learning network GRU is used to predict the traveling destination based on the segmented trajectories and clustered sub-trajectories.

The rest of this paper is organized as follows. Related works are discussed in section II. The problem is formulated in section III. Section IV elaborates on the details of destination prediction. Several extensive simulations are conducted in section V. Finally, the conclusions and discussion are stated in section VI.

II. RELATED WORK

Destination prediction is of much importance, and there have been already lots of works. Overall, there are four classes of methods, including Markov-based, history trajectory match-based, clustering-based, and deep learning-based methods.

Markov and its variant Hidden Markov Model are statistical models that have many applications, which can also be utilized in the area of destination predictions, such as [9], [10], [14]. Paper [9] uses Markov chains to predict the remaining traveling time for moving vehicles on freeways. With low-cost GPS sensors and a map database, an HMM model between traveling routes and destinations is built [10], which can predict drivers' destinations.

Another common solution of destination prediction is to derive the movement probability from historical trajectories. However, this method is based on the dataset composed of all the historical traveling trajectories, which should cover all the traveling cases. Thus, there may be a problem that no matches can be found in the dataset of historical trajectories which is called trajectory sparsity problem. Some works [11], [15], [16] have been done to solve this problem. Paper [11] proposes Sub-Trajectory Synthesis (SubSyn) algorithm on the problem of data sparsity, which decomposes historical trajectories into sub-trajectories, and connects these sub-trajectories into ''synthesized'' trajectories. The trajectory data set used for matching has greatly increased and a better predictive effect is achieved by this means. Compared to [11], paper [16] focuses on the behavior characteristics of queried trajectories. A method called MGDPre is proposed which can work better on the sparse dataset.

Besides, some works predict the destinations of moving objects by clustering. A data-driven framework – DestPre is proposed in [12]. For efficiently retrieving similar trajectories, DestPre involves an index based on the Bucket PR Quadtree and Minwise hashing. In order to predict vehicle destination, it involves a clustering method. Paper [13] clusters trajectories and models main traffic flow patterns with a mixture of 2-D Gaussian distribution. For any new trajectory, it first assigns the trajectory to a cluster which it most likely belongs to. Then it uses the characteristic of each cluster to predict the final destination.

Currently, some works use modern deep learning techniques or extra information together with trajectories to predict destinations. Different from most previous approaches, paper [17] considers more information such as the departure time, the driver id and so on to achieve this objective. It uses neural networks to predict the destination, and a fixed-length output can be obtained from a variable-length input sequence. Destination Prediction Model(DPM) is proposed in [18] to estimate a user's future destination. It combines the filter and contextual knowledge together, which

is used to increase accuracy and reduce historical and contextual knowledge mistakes. An efficient data embedding method – circular fuzzy embedding (CFE) for time-related feature pre-processing is proposed in [19], combining the ensemble learning model(ELM), and a better destination prediction accuracy can be obtained. Based on the most recent partial trajectories and additional contextual data, a Long Short-Term Memory (LSTM)-based model is proposed in [20]. A novel prediction algorithm T-CONV is proposed in [21], which treats trajectories as two-dimensional images rather than one-dimensional sequences, and higher accuracy can be obtained with convolutional neural network(CNN).

Most of the previous works can do well in some domains. However, little of work can consider the above mechanisms together to maximize the advantages. In this paper, we will consider several ideas above to propose a new destination prediction mechanism.

III. PROBLEM FORMULATION

In this paper, we focus on the problem of destination prediction in the IoVs. We first list the definitions used in this paper.

Definition 1: In an IoV, there are N vehicles, denoted as $V = \{v_1, v_2, \ldots, v_n, \ldots v_N\}$.

Each vehicle produces a sequence of locations called trajectory while traveling, which is composed of a series of space-temporal data.

Definition 2: A trajectory of vehicle v_n *can be represented as* $T_{v_n} = [l_{v_n}^{t_1}, l_{v_n}^{t_2}, \ldots, l_{v_n}^{t_j}, \ldots, l_{v_n}^{t_J}]$, *where* $l_{v_n}^{t_j} \in \mathbb{R}^2$, *is composed of the longitude and latitude and denotes the location of* v_n *at time* t_j.

The first point $l_{v_n}^{t_1}$ is the starting point and the last $l_{v_n}^{t_J}$ is the endpoint or destination in a piece of complete trajectory. We use d_{v_n} to represent the destination of the vehicle v_n, which is the same as $l_{v_n}^{t_J}$. An example of a trajectory sequence can be illustrated as the blue line in Fig. 1.

When the sample rate doesn't change, due to traffic jams or other reasons, the collected points in some areas may be dense. This phenomenon doesn't make too much sense to our problem, and it will generate too many redundant data in trajectories. Some works such as [22] and [23] have studied this problem with movement pattern mining. However, they just keep the movement pattern information rather than location information in trajectories. Therefore, in order to avoid the data redundancy problem and keep the location information in trajectories, the key or representative location information from trajectories should be extracted. The processed trajectory not only contains original location changing information, but also becomes a series of the key track segments, which is easier to be handled. Then, the key locations of a trajectory are defined as follows.

Definition 3: The key locations of trajectory T_{v_n} *can be represented as* $KL_{v_n} = [kl_1, kl_2, \ldots, kl_m, \ldots kl_M]$, *where* $kl_m = l_{v_n}^{t_j} (1 \le m \le M, t_1 \le t_j \le t_J, M \le J)$, *is the key location in the trajectory.*

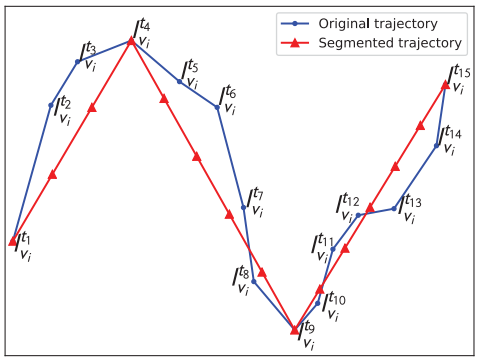

FIGURE 1. An example of trajectory segmentation.

Take the trajectory in Fig. 1 as an example, which is presented by the blue line. In this trajectory, locations $l_{v_n}^{t_1}$, $l_{v_n}^{t_4}$, $l_{v_n}^{t_9}$, and $l_{v_n}^{t_{15}}$ are the key locations. Therefore, the trajectory T_{v_n} is denoted as $[l_{v_n}^{t_1}, l_{v_n}^{t_4}, l_{v_n}^{t_9}, l_{v_n}^{t_{15}}]$, namely KL_{v_n}. The method to obtain the key locations will be detailed in section IV-A. Based on these key locations, the segments or linear representations of trajectory T_{v_n} can be defined as Definition 4.

Definition 4: The trajectory of vehicle v_n *can be represented as several segments or piecewise linear trajectories* $ST_{v_n} = [s_1, s_2, \ldots, s_m, \ldots, s_{M-1}]$, *where* $s_m = [kl_m, p_1, p_2, \ldots, p_x, kl_{m+1}]$, *is a segment called sub-trajectory from key location* $kl_m (kl_m \in KL_{v_n})$ *to* $kl_{m+1} (kl_{m+1} \in KL_{v_n})$ *with the segment evenly filled by several locations* $p_1, p_2, \ldots, p_x$.

In order to avoid information loss of trajectories, we fill points in each segment such as segment from kl_m to kl_{m+1}, and the number of filled points is the same as the original number of locations between the two key locations, kl_m and kl_{m+1}. The filled points are evenly distributed in the segment. The segmented trajectories of the original trajectory indicated by the blue line in Fig. 1 is presented in the red line. There are several points in each segmented trajectory, as shown in the red lines.

Some trajectories may have common or similar sub-trajectories. In order to find some common characteristics on trajectories, sub-trajectories are necessary to be clustered. There are two advantages to cluster sub-trajectories. First, some vehicles usually travel on similar routes, which contain some common sub-trajectories that can be clustered. Second, due to the errors of GPS devices, some sub-trajectories on the same road may be regarded as different routes, which should be corrected. A very intuitive idea to cluster is to match the points collected with GPS devices to the map to determine where the vehicle is traveling. However, this method does not always work well [24], [25], as the quality of the GPS sampling will seriously affect the map matching accuracy.

It is common that the key trajectories on the same road are always similar in location and their shapes are not of too much difference. That gives us the inspiration to find the common trajectories of the same road. The clustering method based on the trajectory shape and distance has a very good effect on solving the problem of finding similar features or attributes. We will introduce the details of clustering in section IV-B.

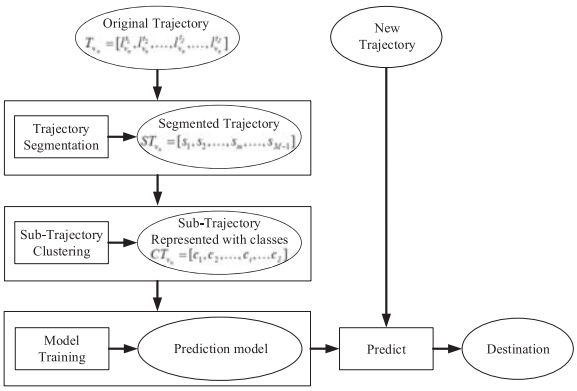

FIGURE 2. The flow chart of the destination prediction.

After all the key sub-trajectories are clustered, each of them will uniquely belong to a certain cluster.

Definition 5: The trajectory of vehicle v_n can also be represented as a sequence of clusters, denoted as $CT_{v_n} = [c_1, c_2, \ldots, c_i, \ldots, c_I]$, where $s_m \in c_i$ and c_i is a set containing numbers of sub-trajectories.

Note that if successive sub-trajectories in ST_{v_n} belong to the same cluster c_i, there may be two duplicate items in CT_{v_n}. In this case, the duplicated items should be deleted and only one is kept. This will reduce the redundant data and keep the movement patterns of vehicles.

So far, all the original trajectories recorded in longitude and latitude structure become the sequences composed of clusters after the segmentation and clustering processes. Next, our destination prediction process is to predict the final cluster in CT. As each cluster sequence follows the chronological order, the RNN network can be used to train the prediction model. In this paper, we introduce the GRU model to train the prediction model. All the clustered trajectories compose the training set and the destination cluster of each trajectory is regarded as the label. Based on different completion rates of vehicle traveling, several prediction models can be obtained, with which the final destination can be predicted. In section IV-C, we will introduce how to predict destinations in detail. At last, the overall flow chart of destination prediction is shown in Figure 2. Details of each operation in the chart are described in section IV.

IV. PROBLEM IMPLEMENTATION

A. TRAJECTORY SEGMENTATION

In order to segment trajectories, an intuitive method is to find inflection points in trajectories such as [26], [27], where vehicles turn or have a large track corner. Intuitively, an angle-based segmentation method is an effective method to find the inflection position and segment the trajectory at the position. However, if trajectories are split by this method directly, a short trajectory may be divided into many sub-trajectories, which is not feasible. Thus, we use the idea of Douglas-Peucker(DP) [28] algorithm to segment trajectories. DP is a commonly used graphic compression algorithm whose purpose is to compress a large number of redundant graphic nodes or to extract the necessary data points from the original data. In this paper, the DP algorithm is used to find the key locations namely inflection points of trajectories, and then the piecewise linear trajectories can be obtained based on the key locations.

Algorithm 1 DP Based Trajectory Segmentation Algorithm

Input:
$T_{v_i} = [l_{v_i}^{t_1}, l_{v_i}^{t_2}, \ldots\ldots, l_{v_i}^{t_J}]$: A trajectory of v_i
TH: Segmentation threshold
Output:
ST_{v_i}:The piecewise linear trajectory of v_i

$ST_{v_i} = \emptyset, num = 1$
function Segment(T_{v_i})
  for $j \leftarrow 2$ to $J-1$ **do**
    dis_j =Distance($l_{v_i}^{t_j}, l_{v_i}^{t_1}, l_{v_i}^{t_J}$)
  end for
  $d_{max} = \mathbf{max}\{dis_2, dis_3, \ldots\ldots, dis_{J-1}\}$
  if $d_{max} \leq TH$ **then**
    $s_{num} = \text{Fill}(l_{v_i}^{t_1}, l_{v_i}^{t_J}, J-2)$
    $ST_{v_i} = ST_{v_i} \cup s_{num}$
    $num++$
  else
    $index = \mathbf{arg}(d_{max})$
    $L_a = [l_{v_i}^{t_1}, l_{v_i}^{t_2}, \ldots, l_{v_i}^{t_{index}}]$
    $L_b = [l_{v_i}^{t_{index}}, \ldots, l_{v_i}^{t_J}]$
    Segment(L_a)
    Segment(L_b)
  end if
  return ST_{v_i}
end function
function Distance(p, p_s, p_e)
  dis = distance from point p to the straight line formed by p_s and p_e
  return dis
end function
function Fill(kl_s,kl_e, N)
  $\overrightarrow{Seg} = [kl_s]$
  $\overrightarrow{Piece} = (\overrightarrow{kl_e} - \overrightarrow{kl_s})/(N+1)$
  for $n \leftarrow 1$ to N **do**
    add $(kl_s + \overrightarrow{Piece} \times n)$ to Seg
  end for
  add kl_e to Seg
  return Seg
end function

The DP based Trajectory Segmentation Algorithm is proposed in Algorithm 1. The algorithm's inputs are the original trajectory sequence T_{v_i} and the segmentation threshold TH. The output is a list of piecewise linear trajectories of v_i. The workflow is sketched as follows:

1) Connect the starting point and the ending point in the trajectory sequence in a straight line, and calculate the linear equation. Next, the vertical distances from all the other points in the trajectory to the line

can be obtained(line 3 to 5). The maximum distance d_{max} among these distances can be found(line 6). The function $Distance(p, p_s, p_e)$ can be inferred as Equation 1 and Equation 2. Given two points $p_s(s_x, s_y)$ and $p_e(e_x, e_y)$, we can get a linear equation determined by them:

$$(s_y - e_y)x + (e_x - s_x)y + (s_x e_y - e_x s_y) = 0 \quad (1)$$

For any point $p(p_x, p_y)$, the vertical distance from p to the straight line connecting p_s and p_e can be calculated as:

$$d = \frac{|(s_y - e_y)x_p + (e_x - s_x)y_p + s_x e_y - e_x s_y|}{\sqrt{(s_y - e_y)^2 + (e_x - s_x)^2}} \quad (2)$$

2) Compare d_{max} with threshold TH. If $d_{max} \leq TH$, only the first point and the last point become the key locations. Then the segment starting from $l_{v_i}^{t_1}$ and ending at $l_{v_i}^{t_J}$ is filled with the function ***Fill***(line 24 - line 31). The number of filled points is the same as the original number of points between the two key locations, kl_s and kl_e(line27 - line 29). The filled points are located on the straight line uniformly connecting kl_s and kl_e. Finally, a segment is obtained and added to the list ST_{v_i}(line 8 to 9). Otherwise, find the farthest point to the line connecting the first point and the last point(line 12). This point divides the trajectory into two parts. One is the sub-trajectory from the first point $l_{v_i}^{t_1}$ to the current point and the other one is the sub-trajectory from the current point to the last point $l_{v_i}^{t_J}$. Then the two sub-trajectories call the function ***Segment*** iteratively(line 15 to 16) until d_{max} is not larger than TH. At this point, the segmentation of the entire trajectory is completed.

Here, the segmentation threshold TH is important, whose value determines the segmentation effect of trajectories. Based on our previous descriptions, the segmentation point of a trajectory should be the location where a vehicle turns. The minimum curve radius of roads is designed to measure vehicles' safety while moving at a specified driving speed [29]. As shown in Fig. 3, when a vehicle is driving at speed v, the curve radius must be greater or equal to r; otherwise, it is unsafe to passengers. That's because the road cannot provide sufficient centripetal force F to prevent moving vehicles from slipping. The radius r can be calculated with Equation 3, where μ is the lateral force coefficient, and i_h is the superelevation slope, both of which are used to measure road conditions.

$$r = \frac{v^2}{127(\mu + i_h)} \quad (3)$$

Therefore, we can let r be the segmentation threshold TH, which means a vehicle's normal turning radius will not be greater than r and the original trajectory should be segmented to several sub-trajectories with $arg(d_{max})$ if the distance d_{max} is greater than r.

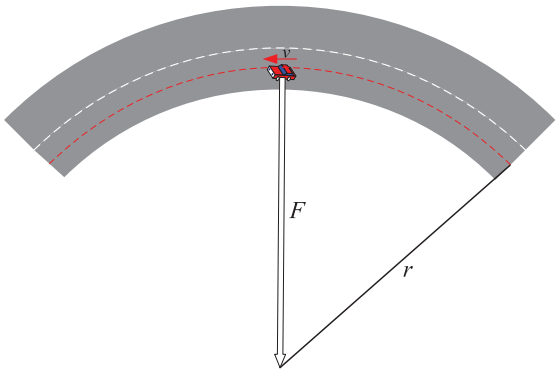

FIGURE 3. The minimum turning radius.

B. SUB-TRAJECTORY CLUSTERING

Many clustering algorithms have been proposed and it is necessary to choose a suitable clustering algorithm in order to ensure an acceptable clustering result. Some existing clustering algorithms need to fix the number of clusters before clustering, such as K-means and Gaussian Mixture Model (GMM). However, these algorithms can not be used in our case, where the final number of clusters cannot be fixed in advance. Another class of clustering algorithms are not necessary to fix the number of clusters in advance such as the Density-Based Spatial Clustering of Applications with Noise(DBSCAN). The clustering process of DBSCAN is to constantly add neighbors according to distance information among items without considering the shape of each item itself. Therefore, it is unsuitable for our scenario. Besides, Affinity Propagation(AP) is a new clustering algorithm proposed in Science Magazine in 2007, but its time complexity is $O(N^3)$, which is not fit for the case of clustering millions of trajectories.

Based on the above observation, the Single-pass Clustering Algorithm, which is fit for processing large streaming or amounts of text data [30], is utilized in our paper to cluster millions of trajectories. The idea of the algorithm is sketched as follows:

At first, let the first item in the dataset be a new class. Next, traverse the other items in the dataset to compare the similarities between these items and the existing classes. The item with the similarity less than a threshold with an existing class will be added to the class. Otherwise, if the similarities to all the existing classes are greater than the threshold, the data will be treated as a new class. Based on this idea, we design our sub-trajectory clustering algorithm and the first task is to define the measurement of similarity on sub-trajectories. We define the Average Nearest Point Pair Distance for measuring the similarity of two sub-trajectories.

Definition 6: The Average Nearest Point Pair Distance – ANPPD of two sub-trajectories s_i and s_j, denoted as $ANPPD_{(s_i,s_j)}$, is referred to the average distance over each point in s_i to the nearest point in s_j, where $|s_i| \leq |s_j|$.

The distance in the definition can be Euclidean distance or Manhattan distance, and here we choose the Euclidean distance. $|s_i|$ means the length of the sub-trajectory s_i.

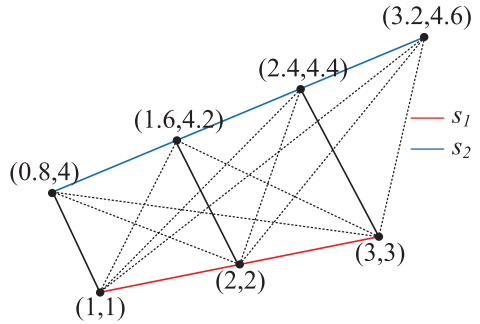

FIGURE 4. An illustrative example of the average nearest point pair distance.

Besides, when $|s_i| \neq |s_j|$, according to the definition of *ANPPD*, it is defined that *ANPPD* is calculated from the short sub-trajectory to the long one. An example of the calculating process is illustrated in Fig. 4. The nearest point pairs are connected with the solid lines. The Euclidean distances from (1, 1) to (0.8, 4), (2, 2) to (1.6, 4.2), (3, 3) to (2.4, 4.4) is about 3.01, 2.24 and 1.52 respectively. Thus $ANPPD_{(s_1,s_2)} = (3.01 + 2.24 + 1.52)/3 = 2.257$.

The purpose of clustering is to find sub-trajectories in the same road or in the same direction with a small distance, so that their *ANPPDs* should be very small. The threshold of *ANPPD* is defined for tuning the effect of clustering, which means if *ANPPD* of two sub-trajectories is less than the threshold, they should be in the same cluster. However, when two sub-trajectories are perpendicular to each other or the angle between the two sub-trajectories is relatively large, and one sub-trajectories may have a few points with the other one containing many points, their *ANPPD* may also satisfy the condition. Therefore, we have to consider the angle between sub-trajectories while clustering. Then a single pass-based Sub-Trajectory Clustering Algorithm is proposed as Algorithm 2.

The input parameters are the sub-trajectory set S, threshold on angle Θ and *ANPPD* Φ between two sub-trajectories. The clustering result C is the output. The main idea of algorithm 2 is sketched as follows.

At first, the first item in S is added to the cluster set C as the first cluster to initialize all parameters(line 1 to line 2). Next, iterate over the other sub-trajectories to calculate the *ANPPDs* and angles to existing clusters(line 4 to line 8). Based on the above, the minimal *ANPPD* in *DIS* can be found, which means that the current sub-trajectory is closest to the cluster with the minimal *ANPPD*(line 9 to line 10). If the *ANPPN* is less than the threshold Φ and the angle is less than Θ, the sub-trajectory can be added to the cluster and the center line of the cluster should be recalculated(line 11 to line 13); Otherwise, it should be created as a new cluster and added to the cluster set(line 14 to line 16).

To calculate the cluster center, we first determine the vector direction of the cluster center. We define all the sub-trajectories in a cluster c with sub-trajectories $s_1, s_2, s_3, \ldots, s_n$. Then each sub-trajectory can be denoted as a vector, namely $\{\overrightarrow{V_{s_1}}, \overrightarrow{V_{s_2}}, \overrightarrow{V_{s_3}}, \ldots, \overrightarrow{V_{s_n}}\}$ respectively. Each vector is from the starting point to the endpoint of each sub-trajectory. We define that the point with the smaller abscissa value of the sub-trajectory is the starting point. The vector direction of the cluster center can be derived from Equation 4:

$$\overrightarrow{V_c} = \frac{\overrightarrow{V_{s_1}} + \overrightarrow{V_{s_2}} + \overrightarrow{V_{s_3}} + \cdots + \overrightarrow{V_{s_n}}}{|V_c|} \tag{4}$$

Once the direction of the cluster center has been fixed, the two endpoints on the center line need to be fixed. The starting point of the center vector is the center of all the start points of sub-trajectories in this cluster. The endpoint of the center vector is obtained by projecting the center of all the terminals of sub-trajectories to the center vector. In order to avoid a large number of redundant points on the cluster center, we fill the center line uniformly between the start point and the endpoint. The number of filled points is the average number of points in each sub-trajectory in the cluster. An example of calculating the cluster center is shown in Fig. 5.

Once the clustering result is obtained, each sub-trajectory can be categorized. When a new sub-trajectory needs to be categorized, the *ANPPD* value to each cluster should be calculated. Then the cluster with the minimum value is chosen as the category. Once the process is finished, the original trajectory data set becomes a set of sequences of clusters.

Algorithm 2 Sub-Trajectory Clustering Algorithm

Input:
$S = \{s_1, s_2, \ldots, s_k, \ldots, s_K\}$: Sub-trajectory set
Φ: Threshold of *ANPPD* between two sub-trajectories
Θ: Threshold of the angle between two sub-trajectories
Output:
C: Clustering set of S

1: $C = \emptyset, c_1 = \{s_1\}$
2: $C = C \cup \{c_1\}$ // Initialization
3: **for** $k \leftarrow 2$ to K **do**
4: $\quad DIS = \emptyset$
5: $\quad$ **for** $i \leftarrow 1$ to $|\mathbf{C}|$ **do**
6: $\quad\quad dis_i = ANPPD_{(s_k,c_i)}$
7: $\quad\quad angle_i =$ Angle between s_k and c_i
8: $\quad$ **end for**
9: $\quad$ **DIS** $= \{dis_1, dis_2, \ldots, dis_i, \ldots\}$
10: $\quad index =$ **arg**$(min(DIS))$
11: $\quad$ **if** $dis_{index} < \Phi$ and $angle_{index} < \Theta$ **then**
12: $\quad\quad c_{index} = c_{index} \cup \{s_k\}$
13: $\quad\quad$ Recalculate the center of c_{index}
14: $\quad$ **else**
15: $\quad\quad c_{|C|+1} = \{s_k\}$
16: $\quad\quad C = C \cup \{c_{|C|+1}\}$
17: $\quad$ **end if**
18: **end for**
19: **return** C

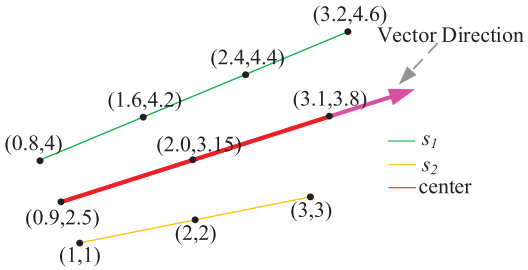

FIGURE 5. An example of calculating the cluster center.

C. DESTINATION PREDICTION

Neural network models such as CNN and recurrent neural network(RNN) have shown great advantages on sequence predicting problem. Deep learning-based predictions are needed for large scale prediction problems. However, the problem of gradient explosion or disappearance usually appears in the neural network models. Although changing some parameters appropriately can make some improvements, this problem can't be solved absolutely as they are in short-term memory. Some new solutions are proposed to overcome that problem, such as the LSTM model and the Gate Recurrent Unit(GRU) model. Compared to LSTM, GRU has fewer parameters and is easier to converge, with almost the same effect as LSTM. Therefore, in this paper, we choose the GRU network as the prediction model.

About the training process of deep learning networks, the selection of features and labels is essential. All the trajectories compose the training set and each trajectory can be represented as a sequence of clusters, shown as definition 5. The items except the last one in trajectories are regarded as the feature sequence, and the last one is the label while training models. Take $CT_{v_i} = \{c_1, c_2, \ldots, c_{I-1}, c_I\}$ as an example. $\{c_1, c_2, \ldots, c_{I-1}\}$ is the feature sequence and c_I is the label, which means the cluster of the destination. The feature set and label set can be obtained after performing the feature and label extraction on all the trajectories. Then the GRU network can be trained based on the training set, after which, the destination prediction models are also obtained. Referring to the prediction process, for a new trajectory, we first perform the trajectory segmentation operation as stated in IV-A. Next, classify the output sub-trajectories to the corresponding clusters and form a sequence of clusters. The cluster of the destination can be predicted based on the sequence and the prediction model.

However, the prediction process is not complete, as with the above steps only the area of the destination, not the final location point, can be obtained. We define that the final location is located in the segment of the cluster center. Besides, due to vehicles moving closer to destinations most of the time, we select the closest location in the segment to the last known location of the current trajectory as the final destination location. In this way, we can roughly estimate the movement destination of each vehicle.

V. SIMULATIONS

A. DATA SET DESCRIPTION

In this paper, the taxi trajectories of Porto city in Portugal are used as the experiment data, which are also public for the dataset of Kaggle challenge at "ECML/PKDD 15: Taxi Trajectory Prediction (I)" section [31]. There are 1710670 trajectories in the original dataset, and the sampling rate of the dataset is four times per minute. Then, the trajectories with the number of sampling locations between 12 and 600(namely, the traveling time lasts for 3 minutes to 1.5 hours) are chosen. Besides, the trajectories whose longitudes are outside of the range $[-8.65, -8.57]$, or whose latitudes are outside of the range [41.10, 41.20], will be filtered out. At last, 30000 trajectories among them are selected for our simulation. A snapshot of the dataset is shown in Fig. 6.

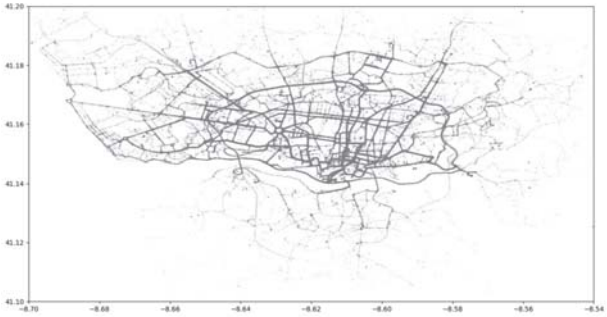

FIGURE 6. A snapshot of the Porto dataset.

B. TRAJECTORY SEGMENTATION

As shown in previous descriptions in IV-A, the segmentation threshold should be fixed first. Generally, the lateral force coefficient μ is about 0.06 and the superrelevation slope i_h is about 0.08 of an ordinary highway. According to Equation 3, the turning radius is related to the vehicle speed. As we all know, the speed range is usually between 0 and 120km/h. If the vehicle speed is always fixed to 120km/h, then many corners on roads may be ignored, so that some inflection points cannot be pointed out. Otherwise, if the vehicle speed is set very small, then most of the points are regarded as inflection points. Therefore, we set the vehicle speed to the average value 60km/h in this paper. Then the threshold is determined. The trajectories can be segmented based on the threshold, and several segmentation examples are shown in Fig. 7. The blue lines and the red lines represent the original trajectories and the segmented trajectories respectively. Here, we can see that the inflection points can be pointed out and each sub-trajectory is uniformly filled.

C. CLUSTERING

According to the inputs of Algorithm 2, all the sub-trajectories segmented from Algorithm 1 are selected as inputs. In order to choose an optimal degree Θ, we try the value 15°, 30°, and 45°. Similarly, we try the value $40m$, $80m$, $120m$, $160m$, and $200m$ to select the optimal value of Φ.

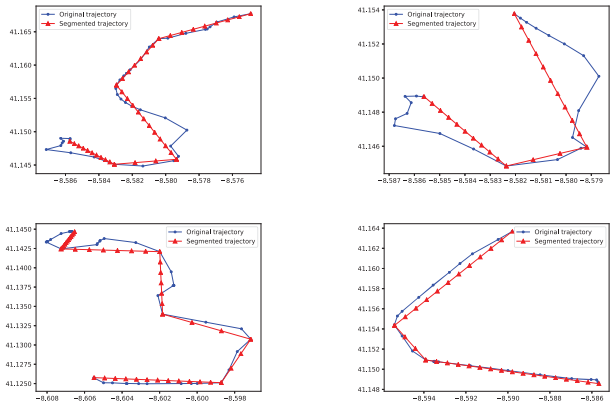

FIGURE 7. Several examples of trajectory segmentations.

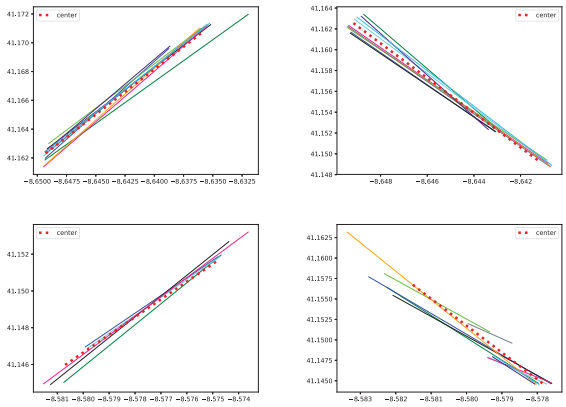

FIGURE 8. Several examples of clusters.

TABLE 1. Performance evaluation for different Φ and Θ values.

Θ \ Φ		40	80	120	160	200
15°	*Dis*	22.48	29.09	30.64	30.97	31.34
30°		23.13	33.60	38.03	39.03	39.46
45°		23.17	34.95	41.72	44.25	45.62
15°	*Num*	4867	3294	3074	3024	2983
30°		4691	2553	1999	1927	1846
45°		4658	2340	1639	1439	1393

There are two objectives to measure the performance of the two thresholds. First, the sum of *ANPPD* distances between each element of a cluster and its cluster center is calculated, which is represented by *Dis*. The smaller *Dis* is, the better the performance is. Second, the total number of clusters *Num* is another metric, which should be minimized and means that a cluster can include more sub-trajectories. The values of the two objectives for different Φ and Θ are shown in Table 1.

In order to select the optimal threshold Θ and Φ, a multi-objective optimization method is used, whose formula is shown as equation 5 and 6.

$$\eta_{Dis} = \frac{f_{\max}(Dis) - f_{Cur}(Dis)}{f_{\max}(Dis) - f_{\min}(Dis)} \tag{5}$$

$$\eta_{Num} = \frac{f_{\max}(Num) - f_{Cur}(Num)}{f_{\max}(Num) - f_{\min}(Num)} \tag{6}$$

in which $f_{\min}(X)$ and $f_{\max}(X)$ are the minimum and maximum values of X respectively, and $f_{Cur}(X)$ is the current value of X. Here, the value of η_X denotes how close $f_{Cur}(X)$ is to the best value.

$$\eta = \eta_{Dis} \times \eta_{Num} \tag{7}$$

According to Equation 7, we choose the pair of Θ and Φ with the largest η value. Finally, when the angle Θ is set to 30° and the threshold Φ of *ANPPD* is set to 80m, the optimal value can be obtained. Several clustering examples are shown in Fig. 8, where the clustering center is indicated in red lines and the others are the sub-trajectories in the same cluster.

D. DESTINATION PREDICTION

Mean Haversine Distance is used to evaluate the final prediction accuracy in this paper, which is commonly used in navigation by measuring distances between two points on a sphere based on their latitudes and longitudes. It is also used as the evaluation metric in Kaggle competition. The Harvesine Distance between the two locations can be computed as follows:

$$\alpha = sin^2(\frac{\phi_1 - \phi_2}{2}) + cos(\phi_1)cos(\phi_2)sin^2(\frac{\lambda_1 - \lambda_2}{2}) \tag{8}$$

$$d = 2 \cdot r \cdot atan(\sqrt{\frac{\alpha}{1-\alpha}}) \tag{9}$$

where ϕ is the latitude and λ is the longitude in Equation 8, d is the distance between two points in Equation 9, and r is the sphere's radius whose value is Earth's radius in our case (i.e. 6371 kilometers).

In order to tune the hyperparameters of the GRU network to improve prediction accuracy, several simulation works have been done by changing the number of layers and neurons of each layer in the GRU network under different cases of trajectory completion rate. The average MSEs of different models are shown in Table 2, where *CR* is the ratio between the traveling distance and the total length of the trajectory. L is the number of layers in the GRU network and N is the number of neurons in each layer of the GRU network. We set the number of layers from 1 to 5 respectively and the number of neurons 16, 32, 64, 128 and 256 respectively. Due to space limitations, the data of layer 4 - 5 is not listed in table 2. The activation function and loss function are set to 'tanh' and 'mse' respectively in our simulation. Dropout function and early stopping function are used to prevent overfitting, where the value of 'dropout' is 0.2 and the value of 'patience' in early stopping function is 5. The simulation results are shown in table 2 and we compare the MSE of each model.

It's obvious that with the trajectory completion rate increasing, the overall average MSE declines. It is found that when the trajectory completion rate is 30%, the average MSE is minimum with 16 neurons in each layer. Referring to the other trajectory completion rates, the average MSE is

TABLE 2. The average MSEs of different models under different completion rates ($\times 10^{-5}$).

CR	N / L	16	32	64	128	256	Ave
30%	1	19.488	20.769	19.225	19.714	20.931	**20.026**
	2	19.573	19.566	19.889	19.974	21.668	20.134
	3	22.407	22.255	23.201	22.112	26.082	23.211
40%	1	19.361	17.521	17.239	17.493	18.014	17.926
	2	17.494	17.229	16.751	17.409	18.669	**17.510**
	3	20.593	21.932	17.919	18.131	19.629	19.641
50%	1	13.721	13.287	13.64	12.781	14.286	**13.543**
	2	14.757	14.198	13.429	13.925	15.928	14.447
	3	15.926	14.110	14.093	14.416	17.224	15.154
60%	1	11.754	10.326	9.875	10.921	9.908	**10.557**
	2	11.571	10.747	10.230	11.101	11.239	10.978
	3	12.314	11.023	10.443	11.798	13.303	11.776
70%	1	8.157	7.210	6.928	7.509	7.924	**7.546**
	2	10.968	7.993	7.363	8.049	8.265	8.528
	3	10.392	8.659	8.327	8.393	9.750	9.104
80%	1	6.130	5.187	4.798	5.434	5.434	**5.397**
	2	6.420	6.507	5.307	5.445	6.628	6.061
	3	8.147	6.065	6.551	6.087	8.054	6.981
90%	1	4.374	3.506	3.512	3.248	3.500	**3.628**
	2	5.564	4.276	3.955	3.524	4.321	4.328
	3	5.250	4.793	4.306	4.390	5.293	4.807

TABLE 3. Mean prediction errors of models with different batch sizes according to trajectory completion rate(km).

CR \ Batch	16	32	64	100	200
30%	1.6002	1.6088	**1.5979**	1.7467	1.7664
40%	1.4693	**1.4289**	1.4655	1.7909	1.7295
50%	1.0955	**1.0729**	1.0945	1.6063	1.6561
60%	**1.0408**	1.0723	1.0661	1.2142	1.2556
70%	**0.8759**	0.8852	0.8911	1.3345	1.3353
80%	0.6760	**0.6450**	0.7190	0.8605	1.0227
90%	0.3907	0.4791	**0.3851**	0.5687	0.4872

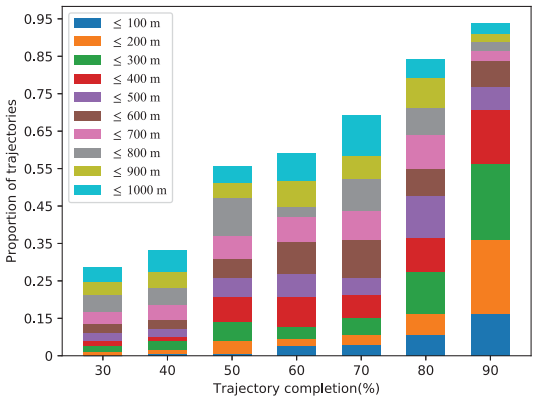

FIGURE 9. Distribution of distance between predicted and true destination according to trajectory completion.

minimum when the number of neurons in each layer is 64. Furthermore, the 2-layer model performs best with completion rate 40% and the 1-layer model performs best in other completion rates. Therefore, in the following simulations, we select 2-layer GRU model with 64 neurons in each layer for 40% completion rate and l-layer model with 16 neurons for 30% completion rate; for other completion rates, we choose 1-layer model with 64 neurons in the layer as the simulation parameter.

The batch size of training also affects the prediction accuracy of models. Thus the models with different batch sizes are trained so that the best prediction models with different trajectory completion rates can be found. The mean prediction errors of those models under different batch sizes and completion rates are shown in table 3. The smallest prediction errors are shown in bold texts in the table, and the corresponding models will serve as our final prediction models. Although the batch size affects the prediction accuracy, the rule that with the trajectory completion rate increasing the mean prediction error declines, doesn't change. Moreover, it can be found the mean prediction errors are quite small. Especially, when the completion rate is 90%, the minimum error is less than 390m.

The distributions of distances between the predicted destinations and the true destinations with different trajectory completion rates are shown in Fig. 9. As the trajectory completion rate increases, the proportion of destination prediction error lower than 1km also increases. In addition, the proportion of prediction accuracy lower than 100m is increasing too, which is similar to other prediction accuracy levels. When the completion rate is 90%, the proportion of trajectories with the prediction error less than 1km can almost reach 95%, which proves the effectiveness of our proposed method.

In order to better illustrate the effectiveness of our proposed method, we compare our method with the other three methods [19]–[21] that do well in the trajectory prediction area. An ensemble learning model(ELM) based on support vector regression(SVR) and deep learning is proposed in [19], and we use 'ELM' to represent the method. A novel prediction algorithm T-CONV is proposed in [21] to predict the destination of partial trajectories. In [20], the destination is predicted based on LSTM model and we use 'LSTM' to represent it. The comparison result is shown in Fig. 10, where we can see that the method we proposed performs best among the three models under most of trajectory completion rates.

VI. CONCLUSIONS AND DISCUSSION

A segmented trajectory clustering-based destination prediction method is proposed in this paper, including trajectory segmentation, sub-trajectory clustering and destination prediction process. In the trajectory segmentation, a DP-based

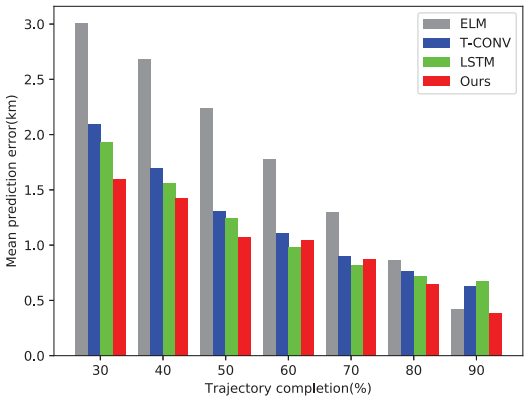

FIGURE 10. Mean prediction errors of different models under different trajectory completion rates.

trajectory segmentation algorithm is used, with which trajectories are split to sequences of key sub-trajectories. Second, we propose a sub-trajectory clustering algorithm to classify sub-trajectories, which reveals similar characteristics among sub-trajectories. Here, the average nearest point pair distance is defined to measure the similarity of any two key sub-trajectories while clustering. At last, a neural network GRU is applied to predict the final destination. Extensive simulations have been done on Porto dataset to tune the hyper-parameters and train the neural network model. Simulation results show that our proposed method can predict the final destination with acceptable errors and it can outperform the other methods most of the cases.

In this paper, the number of clusters is still high, which may affect the prediction accuracy in the GRU network. In order to lower down the number of clusters, other good clustering algorithms can also be tried while clustering the key sub-trajectories, such as AP. However, its efficiency is very low with so many sub-trajectories. Our future work is to improve its efficiency and decrease the number of clusters while utilizing the algorithm. Additionally, the proposed method is only tested on the Porto dataset, which is limited. In the future, we will test it on other datasets to check the robustness, such as vehicle traveling dataset in Cologne and taxi traveling Dataset in Beijing.

REFERENCES

[1] H. Huang, G. Gartner, J. M. Krisp, M. Raubal, and N. Van de Weghe, "Location based services: Ongoing evolution and research agenda," *J. Location Based Services*, vol. 12, no. 2, pp. 63–93, Apr. 2018.

[2] Y. Pu, C. Hu, S. Deng, and A. Alrawais, "R^2PEDS: A recoverable and revocable privacy-preserving edge data sharing scheme," *IEEE Internet Things J.*, to be published.

[3] S. Guo, C. Chen, J. Wang, Y. Liu, X. Ke, Z. Yu, D. Zhang, and D.-M. Chiu, "ROD-revenue: Seeking strategies analysis and revenue prediction in Ride-on-demand service using multi-source urban data," *IEEE Trans. Mobile Comput.*, early access, Jun. 10, 2019, doi: 10.1109/TMC.2019.2921959.

[4] S. Bhanot and J. Mueller, "Tracking building locations of fixed wireless devices," U.S. Patent 10 264 423, Apr. 16, 2019.

[5] C. Chen, D. Zhang, X. Ma, B. Guo, L. Wang, Y. Wang, and E. Sha, "CROWDDELIVER: Planning city-wide package delivery paths leveraging the crowd of taxis," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 6, pp. 1478–1496, Jun. 2017.

[6] C. Wang, J. Li, Y. He, K. Xiao, and H. Zhang, "Destination prediction-based scheduling algorithms for message delivery in IoVs," *IEEE Access*, vol. 8, pp. 14965–14976, 2020.

[7] Y. Pu, T. Xiang, C. Hu, A. Alrawais, and H. Yan, "An efficient blockchain-based privacy preserving scheme for vehicular social networks," *Inf. Sci. J.*, p. 1, 2020.

[8] Kaggle. (Apr. 2015). *Kaggle Data Set ECML/PKDD 15: Taxi Trajectory Prediction (1)*. [Online]. Available: https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/overview

[9] D. Tian, C. Liu, Y. Wang, G. Zhang, and H. Xia, "A freeway travel time predicting method based on IoV," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, Dec. 2015, pp. 1–5.

[10] R. Simmons, B. Browning, Y. Zhang, and V. Sadekar, "Learning to predict driver route and destination intent," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Sep. 2006, pp. 127–132.

[11] A. Y. Xue, J. Qi, X. Xie, R. Zhang, J. Huang, and Y. Li, "Solving the data sparsity problem in destination prediction," *VLDB J.*, vol. 24, no. 2, pp. 219–243, Apr. 2015.

[12] M. Xu, D. Wang, and J. Li, "DESTPRE: A data-driven approach to destination prediction for taxi rides," in *Proc. ACM Int. Joint Conf. Pervas. Ubiquitous Comput.*, Sep. 2016, pp. 729–739.

[13] P. C. Besse, B. Guillouet, J.-M. Loubes, and F. Royer, "Destination prediction by trajectory distribution-based model," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 8, pp. 2470–2481, Aug. 2018.

[14] J. A. Alvarez-Garcia, J. A. Ortega, L. Gonzalez-Abril, and F. Velasco, "Trip destination prediction based on past GPS log using a hidden Markov model," *Expert Syst. Appl.*, vol. 37, no. 12, pp. 8166–8171, Dec. 2010.

[15] X. Li, M. Li, Y.-J. Gong, X.-L. Zhang, and J. Yin, "T-DesP: Destination prediction based on big trajectory data," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 8, pp. 2344–2354, Aug. 2016.

[16] L. Wang, Z. Yu, B. Guo, T. Ku, and F. Yi, "Moving destination prediction using sparse dataset: A mobility gradient descent approach," *ACM Trans. Knowl. Discovery Data*, vol. 11, no. 3, pp. 1–33, 2017.

[17] A. de Brébisson, É. Simon, A. Auvolat, P. Vincent, and Y. Bengio, "Artificial neural networks applied to taxi destination prediction," 2015, *arXiv:1508.00021*. [Online]. Available: http://arxiv.org/abs/1508.00021

[18] A. Nadembega, T. Taleb, and A. Hafid, "A destination prediction model based on historical data, contextual knowledge and spatial conceptual maps," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2012, pp. 1416–1420.

[19] X. Zhang, Z. Zhao, Y. Zheng, and J. Li, "Prediction of taxi destinations using a novel data embedding method and ensemble learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 1, pp. 68–78, Jan. 2020.

[20] P. Ebel, I. Emre Göl, C. Lingenfelder, and A. Vogelsang, "Destination prediction based on partial trajectory data," 2020, *arXiv:2004.07473*. [Online]. Available: http://arxiv.org/abs/2004.07473

[21] J. Lv, Q. Sun, Q. Li, and L. Moreira-Matias, "Multi-scale and multi-scope convolutional neural networks for destination prediction of trajectories," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–12, 2019.

[22] S. Sharma, D. Puthal, S. Tazeen, M. Prasad, and A. Y. Zomaya, "MSGR: A mode-switched grid-based sustainable routing protocol for wireless sensor networks," *IEEE Access*, vol. 5, pp. 19864–19875, 2017.

[23] D.-W. Choi, J. Pei, and T. Heinis, "Efficient mining of regional movement patterns in semantic trajectories," *Proc. VLDB Endowment*, vol. 10, no. 13, pp. 2073–2084, Sep. 2017.

[24] Y.-L. Hsueh, H.-C. Chen, and W.-J. Huang, "A hidden Markov model-based map-matching approach for low-sampling-rate GPS trajectories," in *Proc. IEEE 7th Int. Symp. Cloud Service Comput. (SC2)*, Nov. 2017, pp. 271–274.

[25] Y.-L. Hsueh and H.-C. Chen, "Map matching for low-sampling-rate GPS trajectories by exploring real-time moving directions," *Inf. Sci.*, vols. 433–434, pp. 55–69, Apr. 2018.

[26] C. Chen, Y. Ding, Z. Wang, J. Zhao, B. Guo, and D. Zhang, "VTracer: When online vehicle trajectory compression meets mobile edge computing," *IEEE Syst. J.*, early access, Aug. 28, 2019, doi: 10.1109/JSYST.2019.2935458.

[27] C. Chen, Y. Ding, X. Xie, S. Zhang, Z. Wang, and L. Feng, "TrajCompressor: An online Map-matching-based trajectory compression framework leveraging vehicle heading direction and change," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 5, pp. 2012–2028, May 2020.

[28] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica, Int. J. Geographic Inf. Geovisualization*, vol. 10, no. 2, pp. 112–122, Dec. 1973.

[29] J. Luo, D. Zhang, and T. Guo, "Analysis for influence of large vehicle's lateral stability on ultimate horizontal curve radius of highway design," *China J. Highway Transp.*, vol. 23, pp. 42–46, Dec. 2010.
[30] L. Fang, D. Longlong, J. Zhiying, and L. Shunzi, "Single-pass clustering algorithm based on storm," *J. Phys., Conf. Ser.*, vol. 806, Feb. 2017, Art. no. 012017.
[31] Kaggle. (Apr. 2015). *Kaggle Data Set ECML/PKDD 15: Taxi Trajectory Prediction (1)*. [Online]. Available: https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/data

CHAO WANG (Member, IEEE) received the Ph.D. degree from the Beijing Institute of Technology.

He was a Visiting Scholar with The George Washington University, from 2012 to 2014. He is currently a Faculty Member of the School of Information Science and Technology, North China University of Technology. He has published several research articles in refereed international conferences and premier journals. His research interests include security and privacy in cyber-physical systems and the Internet of Vehicles.

JITONG LI received the bachelor's degree in information and computation science from the Chongqing University of Posts and Telecommunications, Chongqing, in 2018. He is currently pursuing the master's degree in wireless networks and the Internet of Things with the North China University of Technology.

YUNHUA HE (Member, IEEE) received the Ph.D. degree in computer science from Xidian University, Xi'an, China, in 2016. He has been an Assistant Professor with the North China University of Technology, China, since 2016. He has published 16 research articles in refereed international conferences and premier journals. His research interests include security and privacy in cyber-physical systems, bitcoin-based incentive mechanisms, and security and privacy in vehicle ad hoc networks. He received the Best Paper Award from the WASA 2017 Conference.

KE XIAO received the Ph.D. degree in circuits and systems from the Beijing University of Posts and Telecommunications, Beijing, China, in 2008. He has been a Professor with the North China University of Technology, China, since 2018. His research interests include wireless communications, the Internet of Things, and embedded systems. He is a member of the IEEE Communications Society and the IEEE VTS Society. He serves as a Reviewer for the IEEE COMMUNICATIONS LETTERS, the *IEEE Communications Magazine*, and the IEEE INTERNET OF THINGS (IoT) JOURNAL.

CHUNQIANG HU (Member, IEEE) received the B.S. degree in computer science and technology from Southwest University, Chongqing, China, in 2006, the M.S. and Ph.D. degrees in computer science and technology from Chongqing University, in 2009 and 2013, respectively, and the second Ph.D. degree in computer science from The George Washington University, Washington, DC, USA, in 2016. He was a Visiting Scholar with The George Washington University, from 2011 to 2012. He is currently a Faculty Member of the School of Big Data and Software Engineering, Chongqing University. His research interests include privacy-aware computing, big data security and privacy, wireless and mobile security, applied cryptography, and algorithm design and analysis. He was honored with the Hundred-Talent Program by Chongqing University.