A Scalable Framework for Trajectory Prediction

Punit Rathore, Dheeraj Kumar, Sutharshan Rajasegarar, Marimuthu Palaniswami, *Fellow, IEEE*, and James C. Bezdek, *Life Fellow, IEEE*

***Abstract*— Trajectory prediction (TP) is of great importance for a wide range of location-based applications in intelligent transport systems, such as location-based advertising, route planning, traffic management, and early warning systems. In the last few years, the widespread use of GPS navigation systems and wireless communication technology enabled vehicles has resulted in huge volumes of trajectory data. The task of utilizing these data employing spatio-temporal techniques for TP in an efficient and accurate manner is an ongoing research problem. Existing TP approaches are limited to the short-term predictions. Moreover, they cannot handle a large volume of trajectory data for long-term prediction. To address these limitations, we propose a scalable clustering and Markov chain-based hybrid framework, called Traj-clusiVAT-based TP, for both short- and long-term TPs, which can handle a large number of overlapping trajectories in a dense road network. Traj-clusiVAT can also determine the number of clusters, which represent different movement behaviors in input trajectory data. In our experiments, we compare our proposed approach with a mixed Markov model-based scheme and a trajectory clustering, NETSCAN-based TP method for both short- and long-term TPs. We performed our experiments on two real, vehicle trajectory datasets, including a large-scale trajectory dataset consisting of 3.28 million trajectories obtained from 15 061 taxis in Singapore over a period of one month. The experimental results on two real trajectory datasets show that our proposed approach outperforms the existing approaches in terms of both short- and long-term prediction performances, based on the prediction accuracy and distance error (in km).**

***Index Terms*— Large-scale trajectory data, next location prediction, long-term trajectory prediction, scalable clustering.**

I. Introduction

With the widespread use of *Global Positioning System* (GPS) devices, smart-phones, Internet of Things [1], and wireless communication technology, it is possible to track all kinds of moving objects all over the world. The increasing prevalence of location-acquisition technologies has resulted in large volumes of spatio-temporal data, especially in the form of trajectories. These data often contain a great deal of information [2], which give rise to many location-based services (LBSs) and applications such as vehicle navigation, traffic management, and location-based recommendations. One key operation in such applications is the route prediction of moving objects.

Vehicle route prediction allows certain services to improve their quality e.g. if the route of vehicles is known in advance, *intelligent transportation systems* (ITSs) can provide route-specific traffic information to drivers such as forecasting traffic conditions and routing the driver so as to avoid traffic jams. Route prediction also enables location-based advertising, which can advertise certain products/services and special offers to the target commuters most likely to pass through business outlets and stores based on their travel trajectory.

With the advancement in artificial intelligence, various machine learning techniques have now been used for trajectory analysis. Recently, several studies have been carried out on trajectory prediction, particularly after Song *et al.* [3] demonstrated a 93% potential for predictability in user mobility, which supplied the theoretical basis for location prediction methods. These methods mainly focus on two kinds of prediction models. The first type is the short-term trajectory prediction model, which aims to predict the next-location or a few locations in the near future. These models usually rely on current location and one or two previous locations of an object to predict its next location. The second type is the long-term trajectory prediction model which focuses on location prediction at a more distant future time or on complete route prediction. These models generally rely on an available partial trajectory of a moving object to predict the complete trajectory.

In urban areas, vehicle trajectories are usually constrained to a complex road network with many parallel and perpendicular road segments and intersections, which makes their time progression very irregular. Due to the uncertainty of moving objects, most of the existing trajectory prediction methods only focus on predicting short-term partial trajectories. They have poor prediction accuracy for long-term trajectory predictions and they do not work well for estimating continuous and complete trajectories.

The sheer amount of vehicle trajectory data, if analyzed effectively, can significantly improve route prediction performance. However, it is challenging to carry out trajectory prediction from a large amount of trajectory data. The huge volumes of data to be processed precludes using machine learning based *traditional prediction* (TP) methods. Most of the existing trajectory prediction schemes cannot handle a large number of trajectories, especially when they span a large area of a road network. Existing TP methods are hybrid in nature and usually employ classical frequent sequential pattern based algorithms, Markov model-based algorithms, or clustering based algorithms to a small amount of training data.

Manuscript received June 21, 2018; revised December 1, 2018; accepted January 28, 2019. The Associate Editor for this paper was D. Kum. *(Corresponding author: Punit Rathore.)*

P. Rathore and M. Palaniswami are with the Department of Electrical and Electronic Engineering, The University of Melbourne, Parkville, VIC 3010, Australia (e-mail: prathore@student.unimelb.edu.au; palani@unimelb.edu.au).

D. Kumar is with the Lyles School of Civil Engineering, Purdue University, West Lafayette, IN 47907, USA (e-mail: kumar299@purdue.edu).

S. Rajasegarar is with the School of Information Technology, Deakin University, Burwood Campus, Melbourne, VIC 3125, Australia (e-mail: srajas@deakin.edu.au).

J. C. Bezdek is with the School of Computing and Information Systems, The University of Melbourne, Parkville, VIC 3010, Australia (e-mail: jcbezdek@gmail.com).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Digital Object Identifier 10.1109/TITS.2019.2899179

Apriori-like sequential pattern methods bear significant computational overhead in mining frequent patterns because they produce a huge number of candidate sequences and require multiple scans of the database. Markov model approaches such as higher-order Markov models incur significant overhead in terms of computation time and space for storing and learning the mobility model. Similarly, parameter learning algorithms in Hidden Markov models impose significant computational burdens for large-scale trajectory datasets. Existing trajectory clustering based location prediction approaches are not scalable due to distance matrix computations for big data and are not able to handle a large number of overlapping trajectories efficiently. Therefore, most TP methods demonstrated in the literature use synthetic or small to medium size real trajectory datasets. To the best of our knowledge, the largest real dataset was used in [4], consisting of 370 million GPS points. These authors utilized a parallel processing model, MapReduce, in their implementation to handle large datasets.

To address these challenges and overcome the drawbacks of existing TP methods, this paper presents a novel, scalable framework for both short-term and long-term TP, which is suitable for large numbers of overlapping trajectories in a dense road network, typical for major cities around the world. First, we cluster the large trajectory data using a modified version of our two-stage clusiVAT (clustering using improved Visual Assessment of Tendency) algorithm [5], [6], which we call *Traj-clusiVAT*, implemented for trajectory prediction task. The Traj-clusiVAT algorithm first extracts a smart sample using the *Maximin-Random sampling* (MMRS) scheme [7], which provides a good representation of the input cluster structure (present in the original data). Then, it uses the *improved visual assessment of cluster tendency* (iVAT) algorithm to visually determine the number of clusters (k) in input data, and subsequently, it partitions the trajectory sample into k clusters which contain different frequent movement patterns in the trajectory data. Then, the remaining non-sampled trajectories are assigned to one of k clusters using the *nearest prototype rule* (NPR). Finally, Markov chain models are constructed from the trajectories in each cluster. These models quantify the movement patterns within clusters, and subsequently, can be used for TP.

Our main contributions in this paper are as follows:

- We propose a novel, hybrid framework for short-term and long-term trajectory prediction, based on a scalable clustering algorithm and Markov chain model, which can utilize a large number of trajectories in a dense road network.
- We implement a modified version of the two-stage clusiVAT algorithm, called as Traj-clusiVAT, to cluster large numbers of trajectories in an accurate and efficient manner. In this regard,
  - We develop a new algorithm to compute a *representative trajectory* (RT) for each cluster, and subsequently, use it to assign non-sampled trajectories to one of the k clusters in the nearest prototyping phase.
  - We also develop an improved algorithm for nearest prototyping, which assigns each non-sampled trajectory to one of the k clusters, based on pattern matching and the distance of non-sampled trajectories to the RT of each cluster.
- We compare our proposed algorithm with two TP algorithms: a *mixed Markov model* (MMM) [8] and a trajectory clustering model NETSCAN [9] based algorithm. Our experiments used two real-life, taxi trajectory datasets including a large-scale taxi trajectory dataset consisting of 370 million GPS traces and 3.28 million passenger trips from 15,061 taxis during the period of one month in Singapore. To the best of our knowledge, this is the first time TP has been performed on such a large number of real-life road network trajectories.

Here is an outline of the rest of this article. Section II discusses related work and limitations. Section III formally defines the problem and presents preliminaries on techniques used in our work. Our novel, hybrid framework, based on Traj-clusiVAT and Markov models, is presented in Section IV, and its time complexity is discussed in Section V. Section VI presents our numerical experiments, including computation protocols, datasets and results, followed by conclusions in Section VII.

II. Related Work

Several studies address the problem of trajectory predictions, which includes the problem of short-term prediction such as predicting the next location, and long-term prediction such as future locations or complete route prediction. These methods mainly focus on discovering frequent patterns using various data mining methods. Many of these methods are hybrid in nature and can be broadly classified into three categories: (i) Rule-based learning based approaches (ii) Markov model-based approaches (iii) Clustering-based approaches.

A. Rule-Based Learning Based Approaches

Several rule-based methods have been used for location prediction. Morzy [10] implemented a modified version of the PrefixSpan algorithm to extract association rules from a moving object database, and used frequent pattern tree with a matching function to select the best association rule from the database of movement rules. Jeung *et al.* [11] proposed a hybrid prediction approach, which combines association rules in the form of trajectory patterns with the motion functions of an object's recent movements, to estimate future locations. Given an object's recent movement and predictive queries, the best association rule is chosen for prediction. The query processing approaches presented in [11] can only support near and distant-time predictive queries, making them unsuitable for long-term trajectory prediction. Moreover, with the huge number of trajectories, the number of association rules is also huge, which makes association-rule based algorithms impractical for large-scale mobility data.

Monreale *et al.* [12] built a decision tree that they called a T-pattern Tree, based on the frequent movement patterns extracted using a *Trajectory Pattern* algorithm, and predicted the next location of a new trajectory based on the best matching functions. However, mining of frequent trajectory patterns is computationally expensive. The method in [12] is similar

to the use of association rules as predictive rules in rule-based classifiers. Therefore, this method [12] may result in a large number of predictive rules for voluminous trajectories. Qiao *et al.* [13] proposed a TP algorithm, called PrefixTP, which examines only the prefix subsequences, and projects their corresponding postfix subsequences into projected sets. Then, for a partial trajectory, it recursively finds a postfix sequence based on the minimum support count requirement and then declares the most frequent sequential pattern as the most probable trajectory. Finding subsets of trajectory sequential patterns is a recursive mining process, which is also computationally extensive.

B. Markov Model-Based Approaches

Markov models (MMs) have been widely used to mine frequent patterns for route prediction problems. Ishikawa *et al.* [14] proposed a model to extract mobility statistics, called the Markov transition probability, which is based on a cell-based organization of target space and a Markov chain model, and employed R-tree spatial indices to compute Markov transition probabilities. Simmon *et al.* [15] presented a *Hidden Markov Model* (HMM) based probabilistic approach to predict a driver's intended route and destination through observations of the driver's habits. Asahara *et al.* [8] suggested that standard MM and HMM are not generic enough to encompass all types of movement behavior. They proposed a variant of Markov model, called the *mixed Markov-chain model* (MMM), as an intermediate model between individual and generic models, for pedestrian movement prediction.

Gambs *et al.* [16] extended a previously proposed mobility model, named υ*-Mobility Markov Chain* (υ-MMC), to incorporate the υ previous visited locations. They showed that prediction accuracy increases with υ, but increasing υ beyond two does not compensate for the significant overhead in terms of computation and space for learning and storing the mobility model. They only considered the sequence of the significant locations, instead of all locations, to build higher order MM. Zhang *et al.* [17] proposed a group-level mobility modeling method, Gmove, which alternates between two intertwined tasks, user grouping and mobility modeling, and generates an ensemble of HMMs to characterize group-level movement.

Most of the MMs do not consider the discontinuous chain of the hidden states, and therefore, the state retention problem can drastically degrade the accuracy of location prediction system [13]. For the irregular trajectory data, the movement rules cannot be easily represented by Markov models, which may cause loss of continuous location information [13]. Moreover, the HMM approaches use the Baum-Welch algorithm for parameter learning and the Viterbi algorithm to find the most likely sequences of hidden states. These algorithms impose a significant computation burden for large-scale trajectory datasets.

Recently, deep learning techniques such as *inverse reinforcement learning* (IRL), *long short-term memory* (LSTM), and *recurrent neural networks* (RNNs) have been used for modeling vehicle trajectories [18]–[20]. However, some of these studies are still based on first-order Markov assumption to model the routing decisions for heterogeneous destinations, or they are too shallow which makes the modeling pattern varieties suffer from too few parameters.

C. Clustering Based Approaches

Some researchers have proposed trajectory clustering based route prediction methods, which partition the trajectories into several clusters representing different motion patterns based on the trajectory similarity. Various clustering approaches [21] using different methods and distance measures between trajectories have been proposed in the literature. Road network constrained trajectory clustering approaches can be classified into two broad categories. The first type uses the traditional clustering approaches such k-means and DBSCAN with specially designed distance measures [22]–[25] for trajectories. The second category of algorithms [9], [26] cluster road segment vehicle frequencies based on density and flow.

Ashbrook and Starner [27] presented a system that automatically detected the significant locations from GPS data using k-means clustering, and then incorporated these locations into an MM to predict the next location. Mathew *et al.* [28] presented a hybrid method for human mobility prediction, which first clusters location histories according to their characteristics, and then trains an HMM for each cluster. A poor prediction accuracy of 13.85% was obtained on a real, large-scale trajectory dataset using this method. Chen *et al.* [29] proposed a next-location prediction approach combining two clustering models, which cluster the objects based on the spatial locations and trajectories using a similarity metric, respectively, and trains a series of MMs with trajectories in each cluster.

Ying *et al.* [30] proposed an approach for predicting the next location based on geographic and semantic features of user trajectories. This method requires the calculation of a semantic score for each candidate path, which generally incurs additional overhead when compared with other methods. A probabilistic TP model was proposed in [31] based on two mixture models, a *Gaussian Mixture Model* (GMM) and a *Variational Gaussian Mixture Model* (VGMM), optimized using the *Expectation Maximization* (EM) algorithm. Their method requires the prior selection of the number of Gaussian components and other distribution parameters. They evaluated their method on a small dataset, which consists of only 69 trajectories. Lv *et al.* [4] proposed a spatio-temporal prediction and a next-place prediction model based on an entropy-based clustering approach and HMMs.

Traditional clustering [22]–[25] based prediction methods are not scalable for a large number of trajectories as distance matrix computation is time and space prohibitive. Most of them require the number of clusters to be known in advance, but in practice, it is often unknown, making it difficult for the user to choose the optimal number of clusters for location prediction. Furthermore, the clusters are determined by fixed rules. Some of the road network based clustering approaches [9], [26], though scalable, produce clusters having high intra-cluster variance, which span a large area of a road network.

Most of the work done in the area of trajectory prediction either use synthetic datasets [8], [10], [11], [32] or real datasets with small to medium numbers of data

points [12], [29], [33]. Most of them cannot handle big trajectory datasets. There have been several attempts to demonstrate trajectory prediction on real data having a large number of samples. For example, [13] uses a real dataset consisting of 4.9 million trajectories (790 million GPS points) as a population, but only small subsets having a maximum 30, 000 trajectories are used in their experiments. To the best of our knowledge, the largest real dataset used was in [4], consisting of 37 million GPS points. They utilized [4] the MapReduce model in their implementation to handle large datasets. In this paper, we use the GPS traces of 15, 061 taxis in Singapore over a period of one month. We extract 3.28 million passenger trajectories consisting of 370 million GPS logs from this data for trajectory prediction task. To the best of our knowledge, this is the first time trajectory prediction task has been performed on such a large number of real-world road network trajectories.

III. Preliminaries

In this section, we introduce some basic terms and definitions, which are required in the sequel.

A. Road Network and Trajectories

We represent the road network as an undirected graph

$$G_{RN} = (V, E), \tag{1}$$

comprising a set V of *intersections* or *nodes* of the road network with a set E of road *segments* or *edges*, $R_i \in E$ such that $R_i = (r_{i_a}, r_{i_b})$, where $r_{i_a}, r_{i_b} \in V$ and there exists a road between r_{i_a} and r_{i_b}. The edge R_i is given a weight equal to the length of R_i. For such a road network, we define the following:

Definition 1 (Trajectory): A ***trajectory*** *T of length l is a time ordered sequence of road segments (RS), $T = \langle R_1, R_2, \ldots, R_l \rangle$, where $R_j \in E, 1 \leq j \leq l$, and R_j and R_{j+1} are connected.*

Definition 2 (Sub-Trajectory): $T^s = \langle L_1, L_2, \ldots, L_p \rangle$ is a ***sub-trajectory*** *of sequence $T = \langle R_1, R_2, \ldots, R_l \rangle$, $p \leq l$, if there are integers $\langle i_1, i_2, \ldots i_p \rangle$ $(1 \leq i_1 < i_2 < \ldots < i_p)$, $\langle j_1, j_2, \ldots j_p \rangle$ $(1 \leq j_1, j_2 = (j_1 + 1), \ldots, j_p = (j_{p-1} + 1) \leq l)$, and $i_1 \leq j_1$, $L_{i_1} = R_{j_1}$, $L_{i_2} = R_{j_2}, \ldots, L_{i_p} = R_{j_p}$. Then T^s is called a sub-trajectory of T, denoted by $T^s \sqsubseteq T$.*

Definition 3 (Frequent Road Segment): A ***Frequent road segment*** *(FRS), R_{FRS}, in a trajectory set is a segment that contains at least $MinT$ percentage of trajectories of the set passing through the segment, otherwise, the segment is labeled as "non-FRS". The percentage $MinT$ is a tunable parameter and we call it the FRS threshold.*

Definition 4 (Partial Trajectory): A ***partial trajectory*** *T^p is a sub-trajectory of a given trajectory T if and only if their sequences start from the same segment.*

Definition 5 (Source Segment): The segment from which a trajectory T originates is called the ***Source Segment*** *(SS), R_{SS}, and the start node of T is called the* ***Source Node*** *(SN) of that trajectory. For a trajectory $T = \langle R_1, R_2, \ldots, R_l \rangle$, the road segment R_1 is R_{SS}. Node r_{1_a} is the SN, if R_2 has node r_{1_b}, else r_{1_b} is SN, where $R_1 = (r_{1_a}, r_{1_b})$, and $r_{1_a}, r_{1_b} \in V$.*

Definition 6 (Frequent Source Segment): The SS which is FRS, is called Frequent source segment (FSS), R_{FSS}.

TABLE I
Notations

Symbol	Definition
$\mathcal{T}$	The set of trajectories
T_i	The i^{th} trajectory of set $\mathcal{T}$
l_i	The length (or number of segments) of trajectory T_i
R_i	The i^{th} segment of trajectory T_i
N, n	number of trajectories in $\mathcal{T}$ and MMRS sample S, respectively
k,K	number of non-directional and directional clusters in S
$\mathcal{T}^j$	Set of trajectories in cluster j
N_j	Number of trajectories in cluster j
$\mathcal{R}^j$	Set of points (segments) in cluster j
$\mathcal{C}(\mathcal{T})$	Set of cluster of trajectories
R_{FRS}, $\mathcal{R}_{FRS}$	Frequent road segment (FRS) and the set of FRSs, respectively
R_{SS}, $\mathcal{R}_{SS}$	Source segment and the set of SSs, respectively
R_{FSS}, $\mathcal{R}_{FSS}$	Frequent source segment (FSS) and the set of FSSs, respectively
M^j	Transition probability matrix for cluster j
W^j	Transition count matrix for cluster j

Definition 7 (Problem Definition): Assume that a historical trajectory database, containing N trajectories, denoted by $\mathcal{T} = \{T_1, T_2, \ldots, T_N\}$ is given. Then, for a given partial trajectory $T^p = \langle L_1, L_2, \ldots, L_m \rangle$, the goal is to predict the future road segments L_i, where $i, m \in \mathbb{Z}$ and $i \geq m + 1$.

B. Distance Measure (trajDTW)

Most of the existing distance measures for trajectory similarity are not suitable for a large number of overlapping trajectories in a dense road network due to the use of either the number of overlapping road segments or maximum/minimum distance between trajectories in their computation. In our work, we use the Dijkstra based *dynamic time warping* (DTW) distance measure, trajDTW [34] to compute trajectory similarities which is suitable for a large number of overlapping trajectories in a dense road network. The superiority of the trajDTW over the traditionally used *dissimilarity with length* (DSL) and Hausdorff distance measures is demonstrated in [34]. The trajDTW is a normal DTW algorithm with a Dijkstra distance matrix based cost function and a window parameter w, which is set to the half of the length of shorter of two trajectories, to avoid overestimation of the actual distance. As the road network is static, the distance matrix D_{all} (of size $(|E| \times |E|)$ of all the edges E in G_{RN} can be pre-computed and stored.

C. Non-Directional trajDTW

The directionality of trajectories can result in misleading distances among them, which in turn may cause incorrect clustering results. For example, suppose there are two trajectories T_1 and T_2, which follow the same route but in opposite directions, then the distance between them considering their directions in computation will be higher than the distance computed without considering their directions. Therefore, if their movement direction is considered as part of the distance computation, T_1 and T_2 may not be grouped in the same cluster. The problem of incorrect distance measure due to the movement direction of trajectories is addressed by reversing one of them (reversing the sequence order so that the starting point becomes the ending point and vice versa), and taking the minimum distance between the first and second trajectory, and the first trajectory and second reverse trajectory. This distance

is called non-directional trajDTW [34], and is given as:

$$\text{non-directional trajDTW}(T_1, T_2) = \min(trajDTW(T_1, T_2), trajDTW(T_1, Reverse(T_2)) \quad (2)$$

D. The clusiVAT Algorithm

In our proposed framework, we modify the clusiVAT algorithm and use it for efficiently clustering large volumes of trajectory data. The clusiVAT model finds its root in the *visual assessment of cluster tendency* (VAT) [35] and *improved VAT* (iVAT) [36] algorithms. The VAT and iVAT algorithms reorder the input dissimilarity matrix D_N (for N datapoints) to D_N^* (in VAT) and D'^*_N (in iVAT), respectively, using a modified Prim's algorithm (for finding a *minimum spanning tree* (MST)) and by applying a graph theoretic distance conversion, such that the dark blocks along the diagonal of the reordered image $I(D_N^*)$ (or $I(D'^*_N)$) potentially represent different clusters. VAT and iVAT seem to work well for relatively small size datasets. However, both have space and time complexity of $O(N^2)$, which limits their usefulness for input matrix sizes of an order of 10^5 and so. To overcome this limitation, an intelligent sampling based scalable clustering algorithm, clusiVAT [5] was proposed for visual cluster tendency assessment and subsequent clustering on big data. The essential steps in clusiVAT are:

1) Maximin Random Sampling (MMRS): MMRS sampling begins by finding k' (an overestimate of k) Maximin samples (*distinguished objects*), which are furthest from each other in the input data. Then, each object in the input data is grouped with its nearest Maximin sample with NPR. This step divides the entire data into k' groups. Then, a sample S of size n (just a small fraction of input data size, N) is built by selecting a proportional number of random data points (Random sampling (RS)) from each of the k' groups. Hence the term MMRS is used for the overall process. Any value of k' which overestimates assumed true number of clusters (k) i.e., $k' \geq k$ should be a good choice [37]. For n, $n = a\ few\ hundred$ datapoints is a reasonable choice for most datasets [38].

2) Step 2. iVAT: The iVAT is applied to the small D_n (computed from $n << N$ samples) distance matrix to obtain its reordered distance matrix D'^*_n. The image $I(D'^*_n)$ usually provides an useful estimate of k, without the need to compute the very large distance matrix D_N.

3) Step 3. Clustering: Single linkage clusters are always aligned partitions in the VAT/iVAT reordered matrices. Having the estimate of k from the previous step, we cut the $k-1$ longest edges in the iVAT-built MST of D_n, resulting in k single linkage clusters. If the dataset is complex and clusters are intermixed, cutting the $k-1$ longest edges may not always be a good strategy as the outliers, which are typically furthest from normal clusters, might comprise most of the $k-1$ longest edges of the MST, resulting in misleading partitions. A useful approach in such a scenario is to manually select the dark blocks, and find the sample trajectories representing each dark block. Another useful approach [39] to obtain clusters is by cutting the MST using cut threshold magnitudes ordered by edge distances d in the MST. The cluster boundaries are defined by those indices z, which satisfy

$$d_z > \alpha \times mean(d), \quad (3)$$

where α is a parameter that controls how far two groups of data points should be from each other to be considered as separate clusters. Smaller values of α represent tighter cluster boundaries, while large values of α create loose cluster boundaries. The procedure to find an optimal value of α is described in [39].

4) Step 4. Nearest Prototyping Rule (NPR): The k-partitions of the n samples are non-iteratively extended to the remaining (non-sampled) $N-n$ objects in the dataset using the nearest prototyping rule.

The implementation and pseudocodes of the trajDTW, VAT, iVAT, and clusiVAT algorithms are well documented in the literature, and are not produced here for brevity.

E. Markov Chain Model

A *Markov chain* (MC) is the simplest form of the Markov process in which only the current state determines the probability of transitioning to the next state. Specifically, a Markov chain model is defined by the transition matrix M, which contains the transition probabilities associated with various state changes. In a road network, an MC is constructed by assigning a state to each node or road segments in the given road network. For any two adjacent road segments R_i and R_j in road network G_{RN}, the transition probability of traveling from R_i to R_j in one step is given by

$$p_{ij} = \frac{\#(R_i, R_j)}{\#(R_i)}, \quad (4)$$

where $\#(R_i, R_j)$ is the number of trajectories that contain the sequence $\{R_i, R_j\}$ and $\#(R_i)$ is the total number of trajectories that passes through R_i. For each pair of adjacent road segments in the graph network, the transition probabilities can be computed using (4), and stored as entries M_{ij} of transition probability matrix M (of size $|E| \times |E|$). We also define a transition count matrix W whose ij-th entry W_{ij} represents the number of trajectories that contain sequence $\{R_i, R_j\}$ i.e., $W_{ij} = \#(R_i, R_j)$. We utilize W in computing a representative trajectory for each cluster in our work.

IV. Proposed Framework

This section presents our proposed framework for trajectory prediction. The frequent route patterns of moving objects can be discovered by clustering their historical trajectories. In our framework, we employ a modified version of the clusiVAT algorithm that we called Traj-clusiVAT. In Traj-clusiVAT, we introduce a *representative trajectory* for each cluster to improve the performance of NPR for trajectory clustering. We also modify the NPR technique in Traj-clusiVAT to improve its performance for trajectory prediction. The Traj-clusiVAT algorithm partitions the trajectories into different groups of similar trajectories, based on the trajDTW distance measure. After clustering trajectories, we train a first-order Markov chain model for each cluster using only the trajectories contained therein. Then, these trained Markov chain models are used for trajectory prediction. The architecture of our

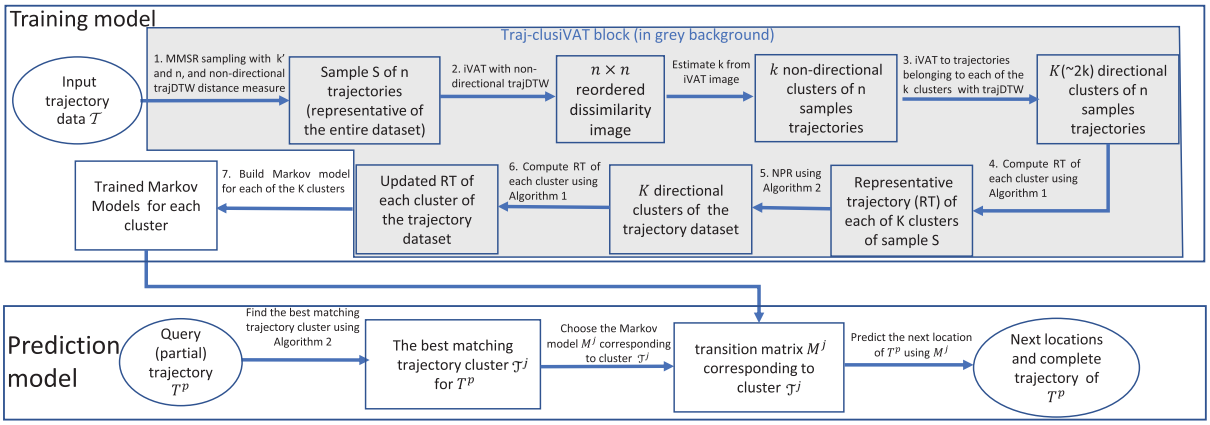


Fig. 1. The architecture of our proposed framework.

proposed framework consisting of both training and prediction models is illustrated in Fig. 1.

A. Training Model

The essential steps of our training model are: (i) MMRS sampling on input trajectory data, (ii) VAT/iVAT and clustering the trajectory sample using non-directional trajDTW to obtain k non-directional clusters (iii) VAT/iVAT and clustering the trajectories of each of the k clusters using trajDTW resulting in K (approx. $2k$) directional clusters (iv) Compute *representative trajectory* (RT) of each cluster, (v) Assign remaining non-sampled trajectories to K clusters using NPR (vi) Re-compute the RT of each cluster, and (vii) Train a first-order Markov chain model for each cluster. The first six steps constitute the Traj-clusiVAT clustering algorithm. Below, we explain each step corresponding to the steps as shown in Fig. 1.

1) MMRS Sampling on Input Trajectory Data: The first step consists of extracting a small, representative sample from the large trajectory data using MMRS sampling with non-directional trajDTW distance measure on input trajectory data $\mathcal{T}$. The aim of this step is to find the most distinguished vehicle routes in a given road network. The use of non-directional trajDTW circumvents the selection of more than one trajectory from the same route. In this way, the Maximin (first) step of MMRS ensures that MMRS samples contain the k' MM trajectories of the most distinguished vehicle routes. This divides the trajectory data $\mathcal{T}$ into k' partitions. Then, additional trajectories are randomly chosen from each of the k' partitions to generate a sample S of n trajectories. The MMRS intelligently chooses n trajectories which are almost equally distributed among the different clusters as the N trajectories in the big trajectory data, i.e., it obtains a representative sample.

2) Clustering Trajectory Samples Using Non-Directional trajDTW: The previous step provides a trajectory sample S containing n trajectories. In this step, the iVAT is applied to the distance matrix D_n returning a reordered distance matrix D'^*_n, and the cut magnitudes of the MST links, d. The visualization of D'^*_n using $I(D'^*_n)$ suggests the number of clusters k present in the dataset. The k partitions can be obtained by cutting the $k-1$ edges or by cutting the MST cut magnitudes d using cut threshold α, as mentioned in Section III-D. The non-directional trajDTW distance measure is used in this step to cluster the n trajectories in order to avoid incorrect clustering due to the movement direction of trajectories, as mentioned in Section III-C). From here on, we denote k as the number of non-directional clusters.

3) Clustering Trajectories in Each Cluster Using trajDTW: The previous step clusters the trajectories based on their path similarity computed using non-directional trajDTW, which ensures that the trajectories that are in opposite directions, but follow similar routes, are clustered together. Since Markov chain models are used in our framework to model the trajectories of each cluster, their transition probabilities may be misleading for trajectory prediction task for clusters in which the number of trajectories in opposite directions is approximately equal. To circumvent this problem, we use the trajDTW (directional) distance measure for the sample trajectories of each cluster obtained in the previous step to separate the trajectories going in opposite directions using a second application of the iVAT algorithm, which in turn, gives $K \sim 2k$ directional clusters.

4) Computing the RT of Each Cluster: In the NPR (next) step of clusiVAT, the non-sampled trajectories are assigned to one of the clusters (found in the previous step) based on their (nearest) distances from each cluster. For a fast and reliable implementation of NPR, we require a *representative trajectory* (RT) for each cluster that best describes the cluster, much like centroid-based clustering methods identify a representative "center" for each cluster. However, it is not possible to compute the centroid of trajectory clusters in a conventional way due to different lengths of trajectories in each cluster. Existing methods of calculating RT [40]–[43] in the literature either compute the mean trajectory using the average of GPS coordinates [41], [43]; or select a trajectory from each cluster which minimizes the dissimilarity between all the trajectories within the cluster [40], [42]; or pick a random trajectory [42] from each cluster, and designates it as the RT. These methods incur a large computational cost to compute an RT that minimizes the dissimilarity among all the trajectories. Additionally, RTs computed using these methods do not show all the possible variability inside a cluster [44]. The mean trajectory computed from trajectories of different lengths may be inaccurate; thus, it may not be a good representative of the cluster.

Algorithm 1 Computing the RT of Each Cluster

Input: $\mathcal{T}^j$- set of trajectories in cluster j, N_j- number of trajectories in cluster j, $\mathcal{R}^j$- set of road segments in cluster j, $\mathcal{C}(\mathcal{T}) = \{\mathcal{T}^1, \ldots, \mathcal{T}^K\}$- set of cluster of trajectories, $MinT$- FRS threshold

1: **for** each cluster $\mathcal{T}^i \in \mathcal{C}(\mathcal{T})$ **do**
2: Compute transition count matrix W^i for cluster $\mathcal{T}^i$
3: Compute FRSs, $\mathcal{R}^i_{FRS}$, from $\mathcal{R}^i$, $\mathcal{R}^i_{FRS} = \{R_j \in \mathcal{R}^i\}_{\#(R_j) \geq MinT \times N_i}$
4: Compute FSSs, $\mathcal{R}^i_{FSS}$ from $\mathcal{R}^i_{SS} = \{R_j\}_{R_j \in \mathcal{R}^i_{SS} \in \mathcal{R}^i_{FRS}}$
5: **for** each FSS $R^i_{FSS} \in \mathcal{R}^i_{FSS}$ **do**
6: Assign R^i_{FSS} as current road segment, $R_{current} = R^i_{FSS}$
7: Initialize an imaginary trajectory IT with $R_{current}$, $IT^i(R^i_{FSS}) = \{R_{current}\}$
8: $Count\_score(IT^i(R^i_{FSS})) = 0$
9: **while** each RS of $IT^i(R^i_{FSS}) \in \mathcal{R}^i_{FRS}$ **do**
10: Compute next RS, $R_{next} = \arg\max_{R_j \in \mathcal{R}^i} \{W^i_{current,j}\}$
11: **if** $R_{next} \in \mathcal{R}^i_{FRS}$ **then**
12: Append R_{next} to existing $IT^i(R^i_{FSS})$
13: $R_{current} = R_{next}$
14: $Count\_score(IT^i(R^i_{FSS}))$ += $W^i_{current,next}$
15: **else**
16: **break;**
17: **end if**
18: **end while**
19: **end for**
20: Select $IT^i(R^i_{FSS})$ with the highest $Count\_score(IT^i(R^i_{FSS}))$ from all $|\mathcal{R}^i_{FSS}|$ ITs of $\mathcal{T}^i$, and assign it as RT for cluster $\mathcal{T}^i$
21: **end for**

Output: $RT(\mathcal{T}^i)$- RT for each cluster $\mathcal{T}^i \in \mathcal{C}(\mathcal{T})$

Our scheme generates an *imaginary trajectory* (IT) (it may not belong to any of the trajectories in the cluster) as an RT for each cluster that describes the major movement patterns of the trajectories belonging to that cluster. The pseudocode of our proposed method to compute RT for each cluster is shown in Algorithm 1. Below, we explain our RT computing algorithm.

First, we compute the transition count matrix W^i for each cluster $\mathcal{T}^i$ using the trajectories in that cluster (line 2). Then, for each cluster $\mathcal{T}^i$, we compute the set of *frequent road segments* (FRSs) $\mathcal{R}^i_{FRS}$ using the $MinT$ threshold (line 3). The road segments in cluster $\mathcal{T}^i$ which contains at least $MinT\%$ of the total trajectories in that cluster are assigned to $\mathcal{R}^i_{FRS}$. Then, a set of *frequent source segments* (FSSs) $\mathcal{R}^i_{FSS}$ is identified (line 4). A source segment R^i_{SS} is a FSS, R^i_{FSS}, if at least $MinT\%$ of total trajectories in the cluster originate from R^i_{SS} i.e, $R^i_{FSS} \in \mathcal{R}^i_{SS}$, $R^i_{FSS} \in \mathcal{R}^i_{FRS}$. Then for each FSS $R^i_{FSS} \in \mathcal{R}^i_{FSS}$ (line 5), an imaginary trajectory $IT^i(R^i_{FSS})$ is initialized with R^i_{FSS} assigning it as current segment $R_{current}$ (lines $6-7$). In lines $9-17$, we compute the next RS R_{next} based on the highest transition count from current RS $R_{current}$ using transition count matrix W^i (refer to Section III-E) . If $R_{next} \in \mathcal{R}^i_{FRS}$, then R_{next} is added to current $IT^i(R^i_{FSS})$, and assigned as $R_{current}$ to compute new R_{next}. The steps in lines $9-18$ are repeated until R_{next} is non-FRS, *which means an imaginary trajectory is an ordered sequence of only frequent road segments in that cluster*. A total of $|\mathcal{R}^i_{FSS}|$ imaginary trajectories will be generated for each cluster $\mathcal{T}^i$, corresponding to each $R^i_{FSS} \in \mathcal{R}^i_{FSS}$. We define a variable $Count\_score$ (line 8) for each imaginary trajectory $IT^i(R^i_{FSS})$, $R^i_{FSS} \in \mathcal{R}^i_{FSS}$, which is the sum of the total transition counts of each RS $\in IT^i(R^i_{FSS})$ in cluster $\mathcal{T}^i$. Among all $|\mathcal{R}^i_{FSS}|$ ITs, the one which has the highest $Count\_score$ will be assigned as $RT(\mathcal{T}^i)$ of cluster $\mathcal{T}^i$ (line 20). As the $RT(\mathcal{T}^i)$ is the sequence of FRS with highest $Count\_score$, it contains major movement behavior or patterns of the trajectories belonging to the cluster $\mathcal{T}^i$.

Algorithm 1 does not require the computation of dissimilarity among all trajectories in that cluster to compute RT, which is computationally expensive for large size clusters. In contrast, Algorithm 1 is a novel algorithm to compute RT based on the transition count matrix of each cluster.

5) Assigning Non-Sampled Trajectories to Identified K Clusters Using NPR: The previous step gives representative trajectory $RT(\mathcal{T}^i)$ for each cluster $\mathcal{T}^i$. In this step, $N-n$ non-sampled trajectories are assigned to one of the K directional clusters based on the NPR. The NPR method in clusiVAT uses the trajDTW (directional) distance measure to assign non-sampled trajectories to one of the K clusters based on their nearest distance from (clustered) sample trajectories. However, trajDTW distance of a non-sampled trajectory to cluster RTs may not be an appropriate measure for the NPR step due to its dependency on the *length* of trajectories, as explained by the following example.

Suppose T_a is a non-sampled trajectory in $\mathcal{T}$, and $RT(\mathcal{T}^i)$ and $RT(\mathcal{T}^j)$ are the RT of cluster $\mathcal{T}^i$ and $\mathcal{T}^j$, respectively. Let T_a be a sub-trajectory of $RT(\mathcal{T}^i)$ i.e., T^a is fully contained in $RT(\mathcal{T}^i)$. Since the trajDTW distance relies on a warping window size parameter w, the $trajDTW(T_a, RT(\mathcal{T}^i))$ not only depends on the coordinates of RSs of both trajectories, but it also depends on the length of T_a and $RT(\mathcal{T}^i)$. Moreover, $trajDTW(T_a, RT(\mathcal{T}^i))$ also varies depending on the position of T_a in $RT(\mathcal{T}^i)$ due to window parameter. Therefore, even if $T_a \sqsubseteq RT(\mathcal{T}^i)$ and $T_a \not\sqsubseteq RT(\mathcal{T}^j)$, T_a may be incorrectly assigned to cluster $\mathcal{T}^j$ instead of $\mathcal{T}^i$ if $trajDTW(T_a, RT(\mathcal{T}^i)) \geq trajDTW(T_a, RT(\mathcal{T}^j))$. Here is such an example from T-Drive data. Suppose $T_1 = \langle 70, 75, 90, 89, 88\rangle$ is a non-sample trajectory, and $RT(\mathcal{T}^1) = \langle 16, 18, 68, 70, 75, 90, 89, 88\rangle$ and $RT(\mathcal{T}^2) = \langle 68, 70, 75, 91, 92\rangle$ are RTs of two clusters, where each trajectory is represented by a sequence of road segments' IDs of Beijing road network (Refer to Section VI-A). The trajDTW distances are: $trajDTW(T_1, RT(\mathcal{T}^1)) = 0.3482$ and $trajDTW(T_1, RT(\mathcal{T}^2)) = 0.2767$. Therefore, although T_1 is a sub-trajectory of $RT(\mathcal{T}^1)$, it will be assigned to cluster $\mathcal{T}^2$ based on nearest trajDTW distance. Such assignments of non-sampled trajectories to (incorrect) cluster may include outlier trajectories or road segments in that cluster, which may adversely affect Markov chain modeling, and consequently, degrade the performance of trajectory prediction.

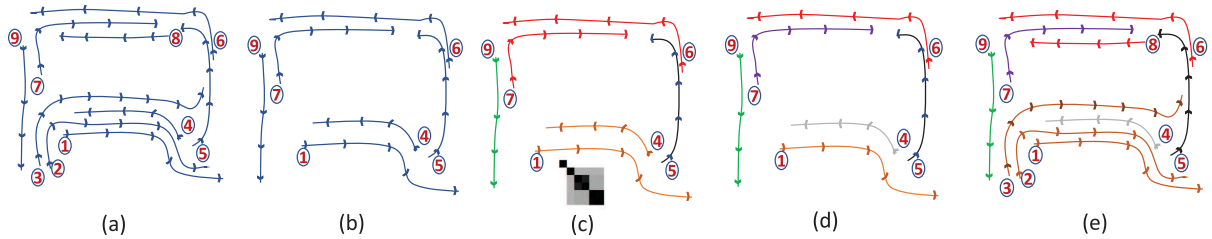


Fig. 2. A simple illustration of Traj-clusiVAT for trajectory clustering.

Algorithm 2 Hybrid NPR Method

Input: T_q - query trajectory, M^j - transition probability matrix for cluster j, $RT(\mathcal{T}^j)$- representative trajectory for cluster j

Compute the path probability $P^i(T_q)$ of query trajectory in each cluster $\mathcal{T}^i$ using M^i and Eq. 5.
if any$(P^i(T_q) > 0$ **then** ▷ if T_q is present in any cluster $\mathcal{T}^i$
  Select the cluster c with the highest $P^i(T_q)$ i.e., $c = \arg\max_{\mathcal{T}^i \in \mathcal{C}(\mathcal{T})} \{P^i(T_q)\}$
else
  Compute the trajDTW distance of T_q from $RT(\mathcal{T}^i)$, $y^i = trajDTW(T_q, RT(\mathcal{T}^i))$, for each cluster $\mathcal{T}^i \in \mathcal{C}(\mathcal{T})$
  Select the cluster c with the minimum y^i i.e., $c = \arg\min_{\mathcal{T}^i \in \mathcal{C}(\mathcal{T})} \{y^i\}$
end if
Assign the T_q with cluster c (or $\mathcal{T}^c$).

Output: cluster label for T_q

To address above issue, we propose a *hybrid* NPR strategy based on the path probability and trajDTW distance measure. *Hybrid* NPR is similar to clusiVAT NPR except for those non-sampled trajectories, which are sub-trajectory of any of the clusters' trajectories. The pseudocode of our hybrid NPR method is shown in Algorithm 2. For a query trajectory $T^q = \{R_1, R_2, \ldots, R_l\}$, we first compute the path probability $P^i(T^q)$ for each cluster $\mathcal{T}^i$, which is defined as

$$P^i(T^q) = \prod_{j=1}^{l} p_{j(j+1)} \Leftrightarrow \prod_{j=1}^{l} M^i_{j(j+1)}. \tag{5}$$

$P^i(T^q) > 0$ means that sequence T^q appears at least once in cluster $\mathcal{T}^i$, whereas $P^i(T^q) = 0$ means that sequence T^q is not present in cluster $\mathcal{T}^i$. If the sequence T^q is present in any cluster $\mathcal{T}^i$ i.e., any$(P^i(T^q)) > 0$, then T^q is assigned to the cluster with the highest path probability. If the sequence T^q is not present in all clusters $\mathcal{T}^i$ i.e., all$(P^i(T^q)) = 0$, then T^q is assigned to the cluster based on its (minimum) trajDTW distance from RTs. All non-sampled trajectories in $\mathcal{T}$ are assigned to one of the K clusters using Algorithm 2.

6) Recompute the RT of Each Cluster After NPR: The assignment of all non-sampled trajectories to one of the K clusters in the NPR step updates each cluster with new trajectories. Therefore, a representative trajectory is recomputed for each updated cluster using Algorithm 1.

7) Train Markov Chain Model: For each of the K clusters, we build a first-order Markov chain model using the trajectories of that cluster. Specifically, we compute the transition probability matrix M^i for each cluster $\mathcal{T}^c$.

For a basic understanding of Traj-clusiVAT algorithm, we graphically explain its steps on a small trajectory data $\mathcal{T}$, as shown in Fig 2. An input trajectory data $\mathcal{T}$ containing $N = 9$ trajectories is shown in Fig 2 (a). The MMRS sampling on $\mathcal{T}$ with non-directional trajDTW in the first step returns a MMRS sample S containing $n = 6$ sample trajectories $\{1, 4, 5, 6, 7, 9\}$, which are well-distributed in sample S, as shown in Fig 2 (b). In the next step, iVAT is applied to S using the non-directional trajDTW distance measure, which clusters the trajectories based on the path similarity irrespective of their movement directions. The iVAT image in Fig 2 (c) shows four dark blocks along its diagonal, which indicates four clusters in sample S. Having an estimate of $k = 4$, sample S is partitioned into four (non-directional) clusters $\{\{1, 4\}, \{5\}, \{6, 7\}, \{9\}\}$, as shown with four different colors in Fig 2 (c). Then, the trajectories in each cluster going in opposite directions are separated using the iVAT with the trajDTW distance measure, which gives $K = 6$ directional clusters $\{\{1\}, \{4\}, \{5\}, \{6\}, \{7\}, \{9\}\}$, each cluster is shown with a different color in Fig 2 (d). Since there is only one trajectory in each cluster in this case, they are the RTs for corresponding clusters. In the next step, non-sampled trajectories $\{2, 3, 8\}$ are assigned to one of the 6 clusters using NPR (Algorithm 2), which partitions the complete data into 6 clusters $\{\{1, 2, 3\}, \{4\}, \{5\}, \{6, 8\}, \{7\}, \{9\}\}$. Trajectory 4 is in different cluster than $\{1, 2, 3\}$ due to opposite direction. Then, a Markov chain model is trained for each cluster using the trajectories of that cluster.

B. Prediction Model

For a given partial trajectory $T^p = \langle L_1, L_2, \ldots, L_m \rangle$, we first estimate the best matching representative cluster $\mathcal{T}^c$ using our hybrid NPR approach, and then choose the corresponding Markov model of the cluster to predict the next locations L_i, $i \geq m + 1$. Using the cluster $\mathcal{T}^c$, the location L_{m+1} that the object will arrive at next is given by

$$L_{m+1} = \arg\max_{L_j \in \mathcal{R}^c} \{p_{mj}\} \Leftrightarrow \arg\max_{L_j \in \mathcal{R}^c} \{M^c_{mj}\} \tag{6}$$

The T^p is updated with the next predicted location L_{m+1}. Then, the updated T^p is used to estimate the best matching cluster and the corresponding MM is used to predict the next location. The complete trajectory is predicted by computing next locations in a sequential manner using these steps.

V. Time Complexity

In this section, we discuss the time complexity of our proposed Traj-clusiVAT based TP approach. The first step in Traj-clusiVAT is the selection of k' distinguished trajectories which are at maximum distance from each other. This step has the time complexity of $O(k'N)$, where k' is a user-defined parameter for an overestimate of the number of clusters in the input trajectory data and is usually chosen to be (inessentially) large (usually 50 to 200). The next step is to randomly select n trajectories from k' NPR groups to get a sample S. The computation of distance matrix D_n and VAT on a sample S has a time complexity of $O(n^2)$. Usually $n << N$, so the computation of D_n and VAT on S is pretty fast and takes just a small fraction of the total run time of Traj-clusiVAT. The trajDTW distance measure uses Dijkstra's shortest path distance in the standard DTW algorithm. Its best, average case complexity with binary heaps is $O(|E| + |V| \log |V|)$ [45]. For two trajectories of length l_1 and l_2, standard DTW has time complexity of $O(l_1 l_2)$. Remark- There are approximate algorithms such as FastDTW [46] which have a linear time complexity in the average length of trajectories, however, we have not used this implementation in our experiments. The NPR step in Traj-clusiVAT has complexity of $O(n(N-n))$. The computation of RTs has linear time complexity in K. The construction of a Markov model for each cluster is a simple and fast process, which has $O(K)$ time complexity and $O(|E|^2)$ space complexity.

VI. Experiments

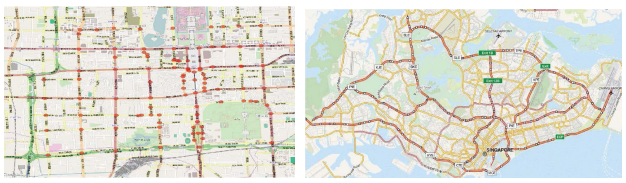
In this section, we conduct an extensive experimental study on two real-life, vehicle trajectory datasets to evaluate the performance of our proposed framework. We first describe the datasets, their preprocessing, evaluation metrics and computational protocols adopted in our empirical study, and then present the experimental results.

A. Datasets

We performed our experiments on two real trajectory datasets.

1) T-Drive Taxi Trajectory Data [47]: This trajectory dataset is obtained from the T-Drive project which contains one-week trajectories of 10,357 taxis during the period of Feb. 2 to Feb 8, 2008 within Beijing, China. The total number of points is about 15 million and the total distance of the trajectories is 9 million kilometers. In our experiment, we have taken a subset of this dataset, which contains trajectories from a road network in the center of Beijing city, as shown in Fig. 3(a). This road network consists of 100 nodes and 141 road segments (edges). The average sampling interval is 177 seconds with an average distance of about 623 meters, which is quite large for a city traffic environment as the length of many road segments is smaller than the average sampling distance.

2) Singapore Taxi Trajectory Data: This dataset consists of the trajectories of more than 15,000 taxis collected over a duration of 1 month from a road network in Singapore City, as shown in Fig. 3(b). This dataset is very dense as it consists of more than 370 million GPS logs. The general format of each data point is as follows: {Time Stamp, Taxi Registration, Latitude, Longitude, Speed, Status}. The Status field contains information about the occupation state of Taxi, such as *FREE* and *POB* (Passenger on Board). In order to extract each individual taxi's trip from the raw data, we detect the following sequence: starting from *FREE* to *POB* and ending from *POB* to *FREE*, using the trip extraction framework presented in [48]. This road network consists of 1641 nodes and 2941 edges, with an average edge length of 350m.



(a) (b)

Fig. 3. Road networks used in our trajectory prediction experiments. (a) T-Drive: Road network in the center of Beijing. (b) Singapore road network.

TABLE II
Training and Test Set Description

	T-Drive Taxi	Singapore Taxi
Training Set	35,501	1,955,573
Test Set	7,904	1,303,717
Total trajectories	43,405	3,259,290

Data Pre-Processing

To obtain the trajectories as a sequence of road segments, we use the popular open source map matching tool GraphHopper [49], which provides an implementation of the approach presented in [50].

After pre-processing, we have $N = 43,405$ trajectories in the T-Drive data whose lengths lie in the range of 5 to 200 road segments and have an average of 14 road segments, and $N = 3,259,290$ (3.26 million) trajectories in the Singapore data whose lengths lie in the range of 10 to 250 road segments and have an average of 22 road segments. To prepare training and test sets for both datasets, we first divided the trajectories into two sets based on the day of week viz., weekdays and weekends, during which the trip is being made. For the one-week T-Drive data, we considered trajectories during first 4 weekdays (Monday to Thursday) and first weekend day (Saturday) as the training set, and trajectories during the remaining days (Friday and Sunday) of that week as the test set. For the one-month Singapore data, we considered 60% trajectories randomly as training set and remaining 40% as the test set, for both weekdays and weekend data. The size of training and test sets for both trajectory datasets is shown in Table II. We split each trajectory in a test set into two halves. The first half is used as a partial trajectory (or query trajectory) for predicting its future locations and the second half is used as ground truth to validate our predictions. The distribution of predicted trajectories (second half) in the T-Drive and Singapore test sets is shown in Fig. 4.

B. Evaluation Metrics

In our experiments, we assess the performance of our framework for next location prediction (also known as one-

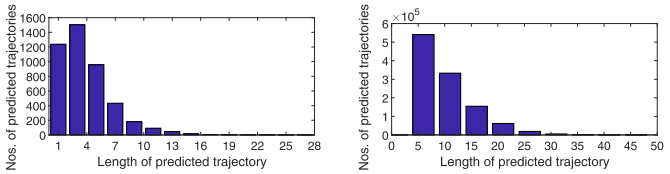


Fig. 4. Trajectory distribution of predicted trajectories based on their lengths. (a) T-Drive Taxi. (b) Singapore Taxi.

step prediction) and long-route prediction using following evaluation metrics:

1) Prediction Accuracy (PA): The PA is the ratio of correctly predicted locations to the total possible number of predicted locations for each trajectory. Given a predicted trajectory sequence $T_{pred} = \{L_1, L_2, \ldots, L_m\}$ and a true (actual) trajectory sequence $T_{true} = \{R_1, R_2, \ldots, R_m\}$, the prediction accuracy is defined as

$$PA = \frac{1}{|T_{pred}|} \sum_{j=1}^{m} H(L_j, R_j), \tag{7}$$

where $H(L_j, R_j)$ is 1 if $L_j = R_j$, else 0. The average prediction accuracy is the average of PA for all predicted trajectories in test set $\mathcal{T}^{test}$.

2) Prediction Rate (PR): The PR is the number of trajectories that are correctly predicted over the total number of trajectories in test set. It is defined as

$$PR = \frac{1}{|\mathcal{T}^{test}|} \sum_{j=1}^{|\mathcal{T}^{tr}|} H(T_{pred_j}, T_{true_j}), \tag{8}$$

where $H(T_{pred_j}, T_{true_j})$ is 1 if $T_{pred_j} = T_{true_j}$, else it is 0.

3) Distance Error (DE): Another important performance metric of the long-term prediction system is the capability of continuous route prediction. The *distance error* is defined as the average spatial (*Haversine*) distance between predicted and actual routes. Given a route sequence T_{pred} and T_{true}, the distance error between them is given as

$$DE(T_{pred}, T_{true}) = \frac{1}{|T_{pred}|} \sum_{j=1}^{m} D_H(L_j, R_j), \tag{9}$$

where $D_H(L_j, R_j)$ is the Haversine [51] distance between two locations (road segments).

4) One-Step Accuracy (OA): This is the ratio of correctly predicted next locations to the total predicted next locations for all trajectories in test set.

5) One-Step Distance Error (ODE): The ODE defined as the average distance error for one-step (or next location) prediction.

C. Comparison Methods

Among the plethora of MM and clustering based TP methods available in the literature, we implemented these two approaches for comparison.

1) Mixed Markov model (MMM) based TP [8]: MMM was proposed as an intermediate model between standard MM and HMM which can encompass all types of movement behavior present in an input trajectory data. It first clusters the trajectories into groups using the EM algorithm, and then builds an MM for each group, which is subsequently used for prediction. This approach was tested on synthetic and real datasets in [8], which showed 74.1% accuracy for MMM, in comparison to $16.9 - 45.6\%$ for MM and $2.4 - 4.2\%$ for HMM.
2) NETSCAN-based TP: The well-known density-based algorithm DBSCAN and its variants [32], [52]–[54] have been used extensively as a trajectory clustering method for location prediction [11]. However, they are not suitable for a large number of trajectories as computation of the distance matrix is time intensive. Kharrat *et al.* [9] proposed a trajectory clustering relative of DBSCAN, called NETSCAN which first finds dense road segments based on the moving object counts, merges them to form dense paths on the road network, and then assigns sub-trajectories to the dense paths based on a measure of similarity. This method requires two user-defined parameters: a density threshold – the minimal required density for transition, and a similarity threshold- the maximum density difference between neighboring road segments. We implement NETSCAN to cluster trajectories into dense road segments, then built an MM for each cluster, and subsequently used them for TP.

Our proposed method and the baseline methods discussed above are also comparable in terms of prediction time (which will be discussed shortly). They all require a short prediction time and satisfy the requirement of real-time prediction.

D. Computation Protocols

All algorithms were coded in MATLAB on a PC with the following configuration; OS: Windows 7 (64 bit); processor: Intel Core $i7-4770$ @3.40GHz; RAM: 16GB. We denote the comparison approaches of [8] as MMM, of [9] as NETSCAN, and our Traj-clusiVAT based TP approach as Traj-clusiVAT. All three algorithms were applied to T-Drive data. The MMM method requires the computation and storage of an intermediate matrix of size $|E| \times |E| \times N$, which is very large for Singapore data (due to large $|E|$ and N), so we cannot apply MMM to the Singapore data. However, we have compared it with NETSCAN-TP and Traj-clusiVAT-TP on a subset obtained from a smaller part of the Singapore road network. The number of mixed models of MMM was determined using 10-fold cross-validation. The NETSCAN parameter, density threshold and similarity threshold, were chosen to get as many dense paths (with at least six road segments) as the number of clusters we get using the Traj-clusiVAT algorithm, for a fair comparison. The parameters for Traj-clusiVAT were chosen as follows: $k' = 150$, $n = 500$, and $\alpha = 0.05$ for the T-drive data, and $k' = 300$, $n = 1000$, and $\alpha = 0.06$ for the Singapore data, and $MinT = 30\%$ for both data. It is worth noting that, unlike other clustering algorithms, the clusiVAT algorithm is relatively insensitive to the choice of k and n [5]. Moreover, we study the effect of α on Traj-clusiVAT performance in our experiments.

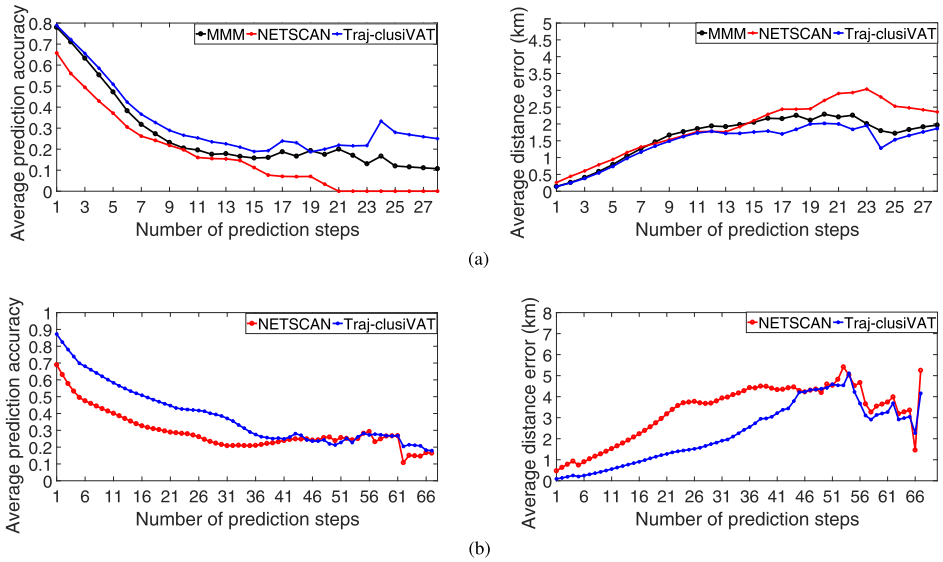


Fig. 5. Average prediction accuracy and average distance error comparison by prediction steps. (a) T-Drive Dataset. (b) Singapore Taxi Dataset.

E. Comparison of MMM, NETSCAN, and Traj-clusiVAT for Long-Term Predictions

Long-term prediction, also known as continuous route prediction, is a challenging and ongoing research problem in TP. In this experiment, we compare the performance of the MMM, NETSCAN, and Traj-clusiVAT-based prediction approaches for m-step predictions. Specifically, this refers to predicting the next m locations for a given partial trajectory. Fig. 5 shows the average prediction accuracy (left panels) and average distance error (right panels) of all three algorithms for increasing prediction steps. The graphs in Fig. 5 support these observations:

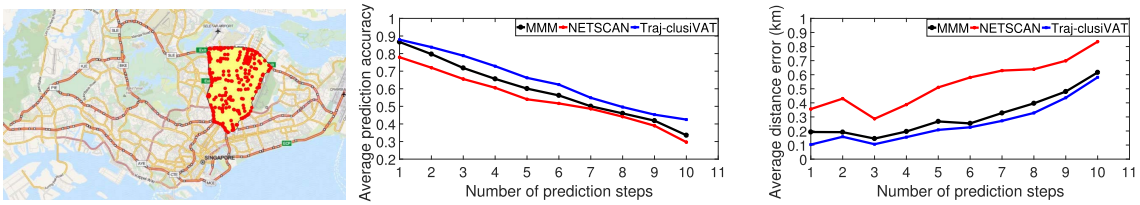
(i) First, the Traj-clusiVAT outperforms the MMM and NETSCAN-based TP approaches based on the average PA and DE for the T-Drive data, as shown in Fig. 5(a). The higher the number of prediction steps, the larger the gap between Traj-clusiVAT and other two approaches. This means that the Traj-clusiVAT performs better not only for short-term predictions but it performs even better than the other two approaches for long-term predictions. This is probably because Maximin sampling in Traj-clusiVAT finds the trajectories which are furthest from each other. As the trajDTW distance measure yields higher distances for longer trajectories, Maximin sampling tends to pick longer trajectories in its output sample which form separate clusters in subsequent steps. The Markov models trained on these clusters after the NPR step contain all movement behaviors similar to those longer trajectory patterns. Therefore, if a query trajectory pattern is not available in any cluster, which is frequent for longer query patterns, then it is assigned to a cluster based on its nearest distance from all cluster RTs. This will assign longer query trajectories to any of the clusters containing longer trajectory patterns, and subsequently, corresponding MMs trained on these clusters contribute towards better predictions for longer query trajectories during the prediction phase. On the other hand, the longer movement rules cannot be easily represented by Markov-based models, especially for irregular trajectory data, due to uncertainty in movement behaviors of vehicles in a complex road network. As there are only a few prediction trajectories available for the T-Drive test set whose lengths are greater than 16 as shown in Fig. 4 (a), the performance of all approaches cannot be considered conclusive for longer prediction steps ($m > 16$) based on their performance on the T-drive data.

(ii) Fig. 5 (b) shows that the Traj-clusiVAT model also performs better than the NETSCAN-based method based on the average PA and DE values for the Singapore data. The gap between the NETSCAN and Traj-clusiVAT plots increases until 31th prediction step and then reduces with longer prediction steps. This may be because the trajectory clusters obtained by NETSCAN are usually spread over the entire road network [34], which results in longer dense paths. Therefore, its performance becomes competitive with Traj-clusiVAT for longer prediction lengths compared to its short-term prediction performance.

(iii) It can be observed that difference in performance of the NETSCAN method and the proposed method is less for short-term and more for long-term prediction for T-drive data, whereas it is opposite for the Singapore data. This is because the T-drive subset contains many parallel and perpendicular road segments and intersections that span only a small road network (Beijing city center), whereas, Singapore data contains relatively longer and straight (fewer intersections) trajectories (compared to T-Drive subset) that span entire Singapore city road network (significantly bigger than Beijing city center road network). Therefore, incorrect predictions cause relatively smaller distance error for T-Drive data as compared to distance error for Singapore taxi dataset. NETSCAN's clusters for Singapore dataset are ordered sequences of only a few but longer and straight dense paths, therefore, it modeled long trajectories better for Singapore dataset, and in turn, performed better for long-term prediction as compared to the T-Drive dataset.

(iv) The performance of all three approaches deteriorates as the prediction step increases. This may be because the number of frequent trajectory patterns obtained is small for long-term predictions, which do not contain enough information to forecast future locations.[1]

[1] And the other reason, as Niels Bohr said, is that "it is very hard to predict, especially the future"



(a) (b) (c)

Fig. 6. Singapore Taxi subset: (a) A small part (highlighted) of the Singapore road network; (b) Average prediction accuracy and (c) Average distance error comparison by prediction steps.

TABLE III
LONG-TERM PREDICTION: COMPARISON OF MMM, NETSCAN AND TRAJ-CLUSIVAT

	T-Drive Data		
	Average PA	Average DE (km)	PR (%)
MMM	0.55	0.68	39.9
NETSCAN	0.41	0.87	24.3
Traj-clusiVAT	**0.62**	**0.58**	**49.8**
	Singapore taxi data		
NETSCAN	0.34	1.41	5.1
Traj-clusiVAT	**0.59**	**0.60**	**24.8**
	Singapore taxi subset		
MMM	0.59	0.30	32
NETSCAN	0.54	0.53	19
Traj-clusiVAT	**0.64**	**0.25**	**46**

TABLE IV
NEXT LOCATION PREDICTION: COMPARISON OF MMM, NETSCAN AND TRAJ-CLUSIVAT

	T-Drive		Singapore Taxi	
	OA	ODE (km)	OA	ODE (km)
MMM	0.77	0.24	-	-
NETSCAN	0.67	0.54	0.62	0.29
Traj-clusiVAT	**0.80**	**0.23**	**0.86**	**0.05**

In our experiments, we find that most of the clusters contain frequent trajectory patterns whose lengths are less than six or seven. Only a few clusters contain frequent trajectory patterns whose lengths are longer than seven steps. This finding conforms with the real-world situation, where a driver usually predicts only next few locations.

The average long-term prediction performance of all three approaches is summarized in Table III. The best performance is shown in bold for both datasets. Traj-clusiVAT achieves the highest PA, 0.62 and 0.59 and the lowest DE, 0.58km and 0.60km, for the T-Drive and Singapore taxi datasets, respectively. The MMM-based prediction approach is the second best method for T-Drive in terms of all three evaluation metrics. Traj-clusiVAT achieves prediction rates of 49.8% and 24.8% for the T-Drive and Singapore trajectory datasets, respectively. In other words, Traj-clusiVAT is able to predict complete trips for around 50% of the trajectories in T-Drive, and for around 25% of the trajectories in Singapore data. In contrast, MMM predicts about 40% of the total trajectories correctly for the T-Drive dataset. Although, NETSCAN performance improved for longer predictions due to longer dense paths, it only predicted about 5% of the total trajectories correctly. Overall, Traj-clusiVAT based prediction approach outperforms both MMM and NETSCAN-based prediction approaches based on all three evaluation metrics.

As we could not apply MMM to the Singapore data due to its high computation and space complexity, we applied it to a subset of a small part of the Singapore road network, as shown in Fig. 6 (a). This sub-graph consists of 238 nodes, 417 edges, and 828, 870 trajectories. Similar to our previous experiments, we considered 60% of the trajectories (497, 320) randomly as training data and the remaining 40% (331, 550) as the test set, for both weekdays and weekend data. Although we could reduce the computational time and space requirements by considering a smaller road network, it still has high complexity due to large N (497, 320). We could not run MMM with more than 10, 000 trajectories as it started to slow down our PC significantly and gave "Out of Memory" error message. Therefore, we used 10, 000 randomly selected trajectories as a training set for MMM-based TP to avoid associated computational and storage overload. We used the same subset for NETSCAN-TP and Traj-clusiVAT-TP for training in this experiment.

Figs. 6 (b) (c) show the average prediction accuracy and average distance error, respectively, for all three algorithms. Traj-clusiVAT-TP method outperforms the MMM-based and NETSCAN-based TP approaches for this subset of the Singapore dataset, based on the average PA and DE. Overall, Traj-clusiVAT-TP achieves the highest average PA (0.64), lowest average DE (0.25km) and highest PR (46%) among all three methods for Singapore taxi subset, as shown in Table III.

F. Next Location Predictions

In this experiment, we compare Traj-clusiVAT to the other two comparison approaches for predicting next locations. Given a taxi's current location, the next location prediction is to forecast the next location where the taxi may go. Table IV shows the one-step accuracy (OA) and one-step distance error (ODE) on the T-Drive and Singapore trajectory datasets. The Traj-clusiVAT-based approach predicts next location with more than 80% accuracy and with distance error of less than a quarter of km for both T-Drive and Singapore data. The long-term prediction performance (Table III) of NETSCAN and Traj-clusiVAT is better for T-Drive than for the Singapore data. Conversely, the next location prediction performance of both approaches is better for the Singapore data than the T-Drive data. This may be because Singapore data contains a large number of longer trajectories that span entire Singapore city, whereas T-drive contains partial trajectories belonging to small part of the entire road network, hence modeling is not that efficient for T-drive data. In summary, Traj-clusiVAT outperforms both MMM and NETSCAN for next location prediction.

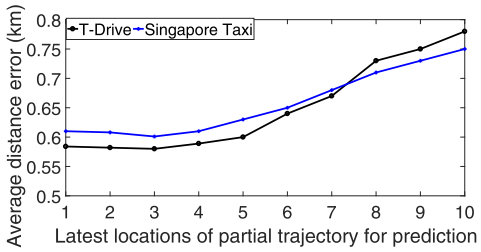


Fig. 7. Average DE vs latest locations of partial trajectory used to select best cluster in the hybrid NPR step.

G. Effect of Latest Locations of Partial Trajectory for Prediction

In the prediction step of Traj-clusiVAT, a partial trajectory $T^p = \{R_1, R_2, \ldots, R_l\}$ is assigned to one of the K clusters using our hybrid NPR approach. For a T^p, the best cluster is chosen based on either its path probability $P^i(T^p)$ in each cluster or its trajDTW distance from each cluster (if T^p is not fully contained in any cluster). The length of known partial trajectory T^p increases after each next location prediction as T^p is updated with a predicted location after each prediction, and subsequently, the updated T^p is used for next location prediction, and so on.

We conduct an experiment in which instead of using full known partial trajectory T^p, we use only the latest movement steps or latest subsequence of T^p until prediction to choose the best matching cluster in the hybrid NPR step. In this regard, we choose a different number of latest locations of known partial trajectories until prediction and investigate the effect on the performance for trajectory prediction.

Fig 7 shows the average distance error for a different number of latest locations of known partial trajectories until prediction for the T-drive and Singapore data. It can be inferred from the figure that the best performance is achieved when only the latest two or three locations of partial trajectory are used to find the best matching cluster. The average distance error increases if more than three latest locations are used to find the best cluster in the hybrid NPR step. This is because as the length of T^p increases, its path probability in each cluster decreases, which means that the chance of sequence T^p being fully contained in any cluster decreases. Moreover, if T^p is not fully contained in any cluster representative trajectory, its distance from all the clusters increases with increasing length. This may result in wrong cluster assignment, which in turn, may degrade Traj-clusiVAT's prediction performance.

H. Effect of Cut Threshold α

In this experiment, we study the effect of cut threshold α. The parameter α in Traj-clusiVAT controls how far two groups of data points should be from each other to be considered as different clusters. Figure 8 shows the average DE and the number of clusters K for different values of α for the T-Drive and Singapore data. The lower the cut threshold, the tighter the cluster boundaries, and hence, the higher the number of clusters. As the number of clusters K increases, the average DE reduces. This is primarily because the higher K corresponds to a larger number of unique frequent patterns, which

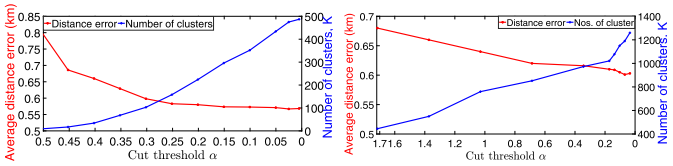


Fig. 8. Effect of cut threshold α. (a) T-Drive. (b) Singapore trajecory data.

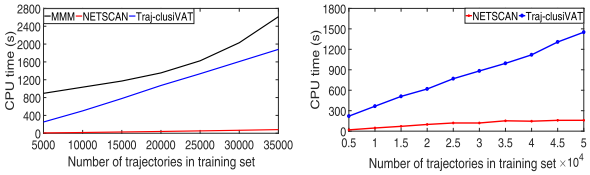


Fig. 9. Training time comparison. (a) T-Drive. (b) Singapore.

TABLE V
PREDICTION TIME IN SECONDS FOR ALL THREE ALGORITHMS

	MMM	NETSCAN	Traj-clusiVAT
T-Drive Taxi	0.0014s	0.0011s	0.0012s
Singapore Taxi	-	0.063s	0.066s

improves the prediction performance. Figure 8 shows that the Traj-clusiVAT performance improves with lower cut threshold α or with the higher number of clusters. However, with a large K, more MM needs to be trained, and hence, system complexity increases. Moreover, Traj-clusiVAT performance does not improve significantly below a certain value of α for either dataset. The procedure to find an optimal value of α is described in [39].

I. Time Performance Analysis

The training time of all three algorithms on different-size training sets is shown in Fig. 9. The CPU-time for MMM increases most with the training data size because the computation of an intermediate matrix of size $|E| \times |E| \times N$ incurs high computational overhead and space complexity for large N. On the other hand, NETSCAN incurs the lowest computation time among all three methods. This is because it just computes dense paths based on the movement counts and density threshold, and assigns all trajectories to these dense paths based on similarity. Although it takes less time for training, it suffers from lower prediction accuracy. Traj-clusiVAT scales almost linearly in the number of trajectories, which make it scalable for big trajectory datasets.

Prediction-time is also an important criterion in real-time trajectory prediction. The average prediction time for all three approaches is presented in Table V. We can see that all three approaches take similar times to forecast each trajectory for the T-drive dataset. The response time is less than 1.5ms for T-drive, which suggests that all three approaches satisfy the requirement for real-time prediction. The average prediction time is higher for the Singapore dataset due to a large number of clusters identified by both NETSCAN and Traj-clusiVAT algorithms, but at ~ 0.06 seconds, it is negligible in terms of real-time prediction utility.

VII. Conclusions

Most existing TP approaches are not suitable for large volumes of overlapping trajectories in a dense road network. This article presents a novel, scalable, hybrid architecture for short-term and long-term trajectory prediction, which can handle a large number of trajectories from a large-scale dense road network. The proposed framework is based on a scalable clustering approach, called Traj-clusiVAT, which is a modified version of clusiVAT for trajectory prediction. In particular, Traj-clusiVAT develops a novel algorithm to compute a representative trajectory for each cluster. We also presented a new, hybrid nearest prototyping approach for accurate trajectory assignment to (one of) the clusters identified in previous steps of Traj-clusiVAT. Finally, we also propose a hybrid prediction framework based on hybrid NPR which can assign a query trajectory to best-matching cluster in a robust way to improve prediction performance.

We demonstrated the superiority of our proposed approach by comparing it with mixed Markov model and NETSCAN based TP approaches on two real trajectory datasets, including a large-scale trajectory dataset containing 3.28 million trajectories of passenger trips obtained from 15, 061 taxis within Singapore over a period of one month. Our experimental results on both trajectory datasets show that Traj-clusiVAT based TP approach outperforms the other two approaches based on the prediction accuracy and distance error for short-term and long-term prediction for these two datasets. Our experimental results also suggest that Traj-clusiVAT satisfies the requirement for real-time predictions. Our next effort will focus on online training in Traj-clusiVAT using incremental/decremental VAT approaches [55] to update clusters in real-time. We also intend to include additional factors such as speed, time, and user information in our prediction system to improve its prediction performance.

Acknowledgement

The authors would like to thank the Institute for Infocomm Research (I2R), A*STAR Singapore for providing them with the Singapore taxi GPS log dataset.

References

[1] P. Rathore, A. S. Rao, S. Rajasegarar, E. Vanz, J. Gubbi, and M. Palaniswami, "Real-time urban microclimate analysis using Internet of Things," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 500–511, Apr. 2018.

[2] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding mobility based on GPS data," in *Proc. 10th Int. Conf. Ubiquitous Comput.*, 2008, pp. 312–321.

[3] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[4] Q. Lv, Y. Qiao, N. Ansari, J. Liu, and J. Yang, "Big data driven hidden Markov model based individual mobility prediction at points of interest," *IEEE Trans. Veh. Tech.*, vol. 66, no. 6, pp. 5204–5216, Jun. 2017.

[5] D. Kumar, J. C. Bezdek, M. Palaniswami, S. Rajasegarar, C. Leckie, and T. C. Havens, "A hybrid approach to clustering in big data," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2372–2385, Oct. 2016.

[6] D. Kumar, H. Wu, S. Rajasegarar, C. Leckie, S. Krishnaswamy, and M. Palaniswami, "Fast and scalable big data trajectory clustering for understanding urban mobility," *IEEE Trans. Intell. Transp. Syst.*, no. 19, no. 11, pp. 3709–3722, Nov. 2018.

[7] J. C. Bezdek, *A Primer on Cluster Analysis: 4 Basic Methods that (Usually) Work*, vol. 1, 1st ed. Design Publ. Inc., 2017.

[8] A. Asahara, K. Maruyama, A. Sato, and K. Seto, "Pedestrian-movement prediction based on mixed Markov-chain model," in *Proc. 19th ACM SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst.*, 2011, pp. 25–33.

[9] A. Kharrat, I. S. Popa, K. Zeitouni, and S. Faiz, "Clustering algorithm for network constraint trajectories," in *Headway Spatial Data Handling*. Berlin, Germany: Springer, 2008, pp. 631–647.

[10] M. Morzy, "Mining frequent trajectories of moving objects for location prediction," in *Proc. Int. Workshop Mach. Learn. Data Mining Pattern Recognit.* Berlin, Germany: Springer-Verlag, 2007, pp. 667–680.

[11] H. Jeung, Q. Liu, H. T. Shen, and X. Zhou, "A hybrid prediction model for moving objects," in *Proc. IEEE 24th Int. Conf. Data Eng. (ICDE)*, Apr. 2008, pp. 70–79.

[12] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti, "WhereNext: A location predictor on trajectory pattern mining," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 637–646.

[13] S. Qiao, N. Han, J. Wang, R.-H. Li, L. A. Gutierrez, and X. Wu, "Predicting long-term trajectories of connected vehicles via the prefix-projection technique," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 7, pp. 2305–2315, Jul. 2017.

[14] Y. Ishikawa, Y. Tsukamoto, and H. Kitagawa, "Extracting mobility statistics from indexed spatio-temporal datasets," in *Proc. STDBM*, 2004, pp. 9–16.

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

[16] S. Gambs, M.-O. Killijian, and M. N. del Prado Cortez, "Next place prediction using mobility Markov chains," in *Proc. ACM 1st Workshop Meas., Privacy, Mobility*, 2012, p. 3.

[17] C. Zhang, K. Zhang, Q. Yuan, L. Zhang, T. Hanratty, and J. Han, "GMove: Group-level mobility modeling using geo-tagged social media," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1305–1314.

[18] F. Altché and A. de La Fortelle, "An LSTM network for highway trajectory prediction," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2017, pp. 353–359.

[19] H. Wu, Z. Chen, W. Sun, B. Zheng, and W. Wang, "Modeling trajectories with recurrent neural networks," in *Proc. IJCAI*, 2017, pp. 3083–3090.

[20] J. Bock, T. Beemelmanns, M. Klösges, and J. Kotte, "Self-learning trajectory prediction with recurrent neural networks at intelligent intersections," in *Proc. VEHITS*, 2017, pp. 346–351.

[21] G. Yuan, P. Sun, J. Zhao, D. Li, and C. Wang, "A review of moving object trajectory clustering algorithms," *Artif. Intell. Rev.*, vol. 47, no. 1, pp. 123–144, 2017.

[22] J.-I. Won, S.-W. Kim, J.-H. Baek, and J. Lee, "Trajectory clustering in road network environment," in *Proc. IEEE Symp. Comput. Intell. Data Mining*, Mar./Apr. 2009, pp. 299–305.

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

[24] Y. Wang, Q. Han, and H. Pan, "A clustering scheme for trajectories in road networks," in *Proc. Adv. Technol. Teach. 3rd Int. Conf. Teach. Comput. Sci. (WTCS)*. Berlin, Germany: Springer, 2012, pp. 11–18.

[25] G.-P. Roh and S.-W. Hwang, "NNCluster: An efficient clustering algorithm for road network trajectories," in *Proc. Int. Conf. Database Syst. Adv. Appl.* Berlin, Germany: Springer-Verlag, 2010, pp. 47–61.

[26] B. Han, L. Liu, and E. Omiecinski, "Road-network aware trajectory clustering: Integrating locality, flow, and density," *IEEE Trans. Mobile Comput.*, vol. 14, no. 2, pp. 416–429, Feb. 2015.

[27] D. Ashbrook and T. Starner, "Using GPS to learn significant locations and predict movement across multiple users," *Pers. Ubiquitous Comput.*, vol. 7, no. 5, pp. 275–286, 2003.

[28] W. Mathew, R. Raposo, and B. Martins, "Predicting future locations with hidden Markov models," in *Proc. ACM Conf. Ubiquitous Comput.*, 2012, pp. 911–918.

[29] M. Chen, Y. Liu, and X. Yu, "Predicting next locations with object clustering and trajectory clustering," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining*. Cham, Switzerland: Springer, 2015, pp. 344–356.

[30] J. J.-C. Ying, W.-C. Lee, T.-C. Weng, and V. S. Tseng, "Semantic trajectory mining for location prediction," in *Proc. 19th ACM SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst.*, 2011, pp. 34–43.

[31] J. Wiest, M. Höffken, U. Kreßel, and K. Dietmayer, "Probabilistic trajectory prediction with Gaussian mixture models," in *Proc. Intell. Vehicles Symp. (IV)*, Jun. 2012, pp. 141–146.

[32] P.-R. Lei, T.-J. Shen, W.-C. Peng, and I.-J. Su, "Exploring spatial-temporal trajectory model for location prediction," in *Proc. IEEE 12th Int. Conf. Mobile Data Manage. (MDM)*, vol. 1, Jun. 2011, pp. 58–67.
[33] L. Chen, M. Lv, and G. Chen, "A system for destination and future route prediction based on trajectory mining," *Pervasive Mobile Comput.*, vol. 6, no. 6, pp. 657–676, 2010.
[34] D. Kumar, S. Rajasegarar, M. Palaniswami, X. Wang, and C. Leckie, "A scalable framework for clustering vehicle trajectories in a dense road network," in *Proc. 4th Int. Workshop Urban Comput. (UrbComp), Held Conjunct 21th ACM SIGKDD*, 2015, pp. 1–9.
[35] J. C. Bezdek and R. J. Hathaway, "VAT: A tool for visual assessment of (cluster) tendency," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 3, 2002, pp. 2225–2230.
[36] T. C. Havens and J. C. Bezdek, "An efficient formulation of the improved visual assessment of cluster tendency (iVAT) algorithm," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 5, pp. 813–822, May 2012.
[37] R. J. Hathaway, J. C. Bezdek, and J. M. Huband, "Scalable visual assessment of cluster tendency for large data sets," *Pattern Recognit.*, vol. 39, no. 7, pp. 1315–1324, 2006.
[38] P. Rathore, J. C. Bezdek, D. Kumar, S. Rajasegarar, and M. Palaniswami, "Approximate cluster heat maps of large high-dimensional data," in *Proc. 24th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2018, pp. 195–200.
[39] D. Kumar, J. C. Bezdek, S. Rajasegarar, C. Leckie, and M. Palaniswami, "A visual-numeric approach to clustering and anomaly detection for trajectory data," *Vis. Comput.*, vol. 33, no. 3, pp. 265–281, Mar. 2017.
[40] G. Yavaş, D. Katsaros, Ö. Ulusoy, and Y. Manolopoulos, "A data mining approach for location prediction in mobile environments," *Data Knowl. Eng.*, vol. 54, no. 2, pp. 121–146, 2005.
[41] J.-G. Lee, J. Han, X. Li, and H. Gonzalez, "TraClass: Trajectory classification using hierarchical region-based and trajectory-based clustering," *Proc. VLDB Endowment*, vol. 1, no. 1, pp. 1081–1094, 2008.
[42] E. H.-C. Lu, V. S. Tseng, and P. S. Yu, "Mining cluster-based temporal mobile sequential patterns in location-based service environments," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 6, pp. 914–927, Jun. 2011.
[43] C. Sung, D. Feldman, and D. Rus, "Trajectory clustering for motion prediction," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2012, pp. 1547–1552.
[44] N. Ferreira, J. T. Klosowski, C. E. Scheidegger, and C. T. Silva, "Vector field k-means: Clustering trajectories by fitting multiple vector fields," *Comput. Graph. Forum*, vol. 32, no. 3pt2, pp. 201–210, 2013.
[45] M. Barbehenn, "A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices," *IEEE Trans. Comput.*, vol. 47, no. 2, p. 263, Feb. 1998.
[46] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, 2007.
[47] J. Yuan *et al.*, "T-drive: Driving directions based on taxi trajectories," in *Proc. ACM 18th SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst.*, 2010, pp. 99–108.
[48] Y. Lu, S. Xiang, and W. Wu, "Taxi queue, passenger queue or no queue?" in *Proc. 18th Int. Conf. Extending Database Technol. (EDBT)*, Brussels, Belgium, 2015, pp. 593–604.
[49] GraphHopper. (2017). *Map-Matching*. [Online]. Available: http://www.unhabitat.org/pmss/listItemDetails.aspx?publicationID=3387
[50] P. Newson and J. Krumm, "Hidden Markov map matching through noise and sparseness," in *Proc. 17th ACM SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst.*, 2009, pp. 336–343.
[51] 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. 2017.
[52] Q. Huang, "Mining online footprints to predict user's next location," *Int. J. Geograph. Inf. Sci.*, vol. 31, no. 3, pp. 523–541, 2017.
[53] S. Qiao, N. Han, W. Zhu, and L. A. Gutierrez, "TraPlan: An effective three-in-one trajectory-prediction model in transportation networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 3, pp. 1188–1198, Jun. 2015.
[54] H. Jeung, H. T. Shen, and X. Zhou, "Mining trajectory patterns using hidden Markov models," in *Proc. Int. Conf. Data Warehousing Knowl. Discovery*. Springer, 2007, pp. 470–480.
[55] D. Kumar *et al.*, "Adaptive cluster tendency visualization and anomaly detection for streaming data," *ACM Trans. Knowl. Discovery Data (TKDD)*, vol. 11, no. 2, p. 24, 2016.

Punit Rathore received the M.Tech. degree in electrical engineering (instrumentation) from IIT Kharagpur, India, in 2011, and the Ph.D. degree from the Department of Electrical and Electronics Engineering, The University of Melbourne, Melbourne, Australia, in 2019. He was a Researcher with Tata Steel, Ltd., India, from 2011 to 2014. His research interests include big data clustering, incremental clustering, spatio-temporal analytics, the Internet of Things, machine learning, and pattern recognition.

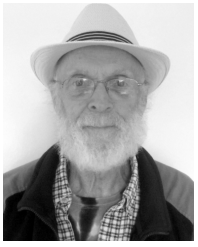
Dheeraj Kumar received the B.Tech. and M.Tech. dual degree in electrical engineering from IIT Kanpur, India, in 2010, and the Ph.D. degree from the Department of Electrical and Electronic Engineering, The University of Melbourne, Melbourne, VIC, Australia, in 2017.

He is currently a Post-Doctoral Research Fellow with the Lyles School of Engineering, Purdue University, USA. His current research interests include big data clustering, incremental clustering, spatio-temporal estimations, the Internet of Things, and machine learning.

Sutharshan Rajasegarar received the Ph.D. degree from The University of Melbourne, Melbourne, VIC, Australia, in 2009. He was a Research Fellow with the Department of Electrical and Electronic Engineering, The University of Melbourne, and a Researcher in machine learning with the National ICT Australia. He is currently a Lecturer with the School of Information Technology, Deakin University, Geelong, Australia. His current research interests include anomaly/outlier detection, distributed machine learning, spatio-temporal estimations, pattern recognition, the Internet of Things, and wireless communication.

Marimuthu Palaniswami (F'12) received the M.E. degree in electrical, electronic and control engineering (EECE) from the Indian Institute of Science, Bengaluru, India, the M.Eng.Sc. degree in EECE from The University of Melbourne, Melbourne, VIC, Australia, and the Ph.D. degree from The University of Newcastle, NSW, Australia.

He is currently a Professor with The University of Melbourne. He is representing Australia as a core partner in EU FP7 projects, such as SENSEI, SmartSantander, the Internet of Things Initiative, and SocIoTal. He has been funded by several Australian Research Council and industry grants (over 40 million) to conduct research in sensor network, the Internet of Things (IoT), health, environmental, and machine learning areas. His current research interests include SVMs, sensors and sensor networks, the IoT, machine learning, pattern recognition, and signal processing and control.



James C. Bezdek (LF'10) received the Ph.D. degree in applied math from Cornell University in 1973. His research interests include optimization, pattern recognition, clustering in very large data, coclustering, and visual clustering. He is a Life Fellow of the IFSA. He was a recipient of the IEEE 3rd Millennium, the IEEE CIS Fuzzy Systems Pioneer, the IEEE Frank Rosenblatt TFA, and the Kempe de Feret IPMU Awards. He is also the Founding Editor of the *International Journal of Approximate Reasoning* and the IEEE TRANSACTIONS ON FUZZY SYSTEMS. He is the past President of the North American Fuzzy Information Processing Society, the International Fuzzy Systems Association, and the IEEE Computational Intelligence Society as the NNC. He retired in 2007.