Large-Scale Vehicle Trajectory Reconstruction with Camera Sensing Network

Panrong Tong
Nanyang Technological University
Singapore
tong0091@e.ntu.edu.sg

Mingqian Li*
Nanyang Technological University
Singapore
mingqian001@e.ntu.edu.sg

Mo Li
Nanyang Technological University
Singapore
limo@ntu.edu.sg

Jianqiang Huang†
Alibaba Group
China
jianqiang.hjq@alibaba-inc.com

Xiansheng Hua
Alibaba Group
China
xiansheng.hxs@alibaba-inc.com

ABSTRACT

Vehicle trajectories provide essential information to understand the urban mobility and benefit a wide range of urban applications. State-of-the-art solutions for vehicle sensing may not build accurate and complete knowledge of all vehicle trajectories. In order to fill the gap, this paper proposes VeTrac, a comprehensive system that employs widely deployed traffic cameras as a sensing network to trace vehicle movements and reconstruct their trajectories in a large scale. VeTrac fuses mobility correlation and vision-based analysis to reduce uncertainties in identifying vehicles. A graph convolution process is employed to maintain the identity consistency across different camera observations, and a self-training process is invoked when aligning with the urban road network to reconstruct vehicle trajectories with confidence. Extensive experiments with real-world data input of over 7 million vehicle snapshots from over one thousand traffic cameras demonstrate that VeTrac achieves 98% accuracy for simple expressway scenario and 89% accuracy for complex urban environment. The achieved accuracy outperforms alternative solutions by 32% for expressway scenario and by 59% for complex urban environment.

CCS CONCEPTS

• **Information systems** → **Data analytics**; *Sensor networks.*

KEYWORDS

Trajectory Reconstruction, Camera Sensing Network, Vehicle Mobility, Identity Uncertainty

ACM Reference Format:
Panrong Tong, Mingqian Li, Mo Li, Jianqiang Huang, and Xiansheng Hua. 2021. Large-Scale Vehicle Trajectory Reconstruction with Camera Sensing Network. In *The 27th Annual International Conference On Mobile Computing And Networking (ACM MobiCom '21), October 25–29, 2021, New Orleans, LA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3447993.3448617

*Also with Alibaba Group.
†Also with Nanyang Technological University.

ACM MobiCom '21, October 25–29, 2021, New Orleans, LA, USA

ACM ISBN 978-1-4503-8342-4/21/10…$15.00
https://doi.org/10.1145/3447993.3448617

1 INTRODUCTION

Knowing the moving trajectories of millions of vehicles traveling in a city provides fundamental understanding of the urban mobility, and the reconstruction of such knowledge benefits a wide range of applications in transportation and urban redevelopment, e.g., usage-based electronic toll collection, more accurate traffic analysis, intelligent traffic light control, retrospective analysis of urban planning, and many others (§2.1).

State-of-the-art solutions to trajectory reconstruction are mainly based on (i) tracking with in-vehicle devices (e.g., GPS), which however only obtains information from participating vehicles that are cooperative in reporting their GPS statuses and thus of partial knowledge, and (ii) roadway monitoring with flow sensors (e.g., loop detectors, piezoelectric sensors, and infrared sensors), which however only captures traffic flows, i.e., the number of vehicles passing by roads, thus not applicable to reconstructing individual vehicle trajectories.

In this paper, we study the use of widely deployed traffic cameras as a sensing network to observe general vehicle mobility and based on that reconstruct the complete knowledge of all general vehicles in the city with high accuracy. Compared with existing solutions, the use of the traffic camera network possesses the following advantages: (i) Traffic cameras are generally available in urban cities and require no additional deployment cost; (ii) Traffic cameras are widely deployed and provide high coverage of the city, providing an opportunity for sensing and monitoring complete vehicle traffics; (iii) Advanced image analysis techniques can be applied to extract vehicle identifiers, providing essential information for recovering their trajectories.

With the vehicle identities being discovered by traffic cameras along their moving trajectories, a straightforward solution is to maintain such identities across all roadways and reconstruct the trajectory of each vehicle by concatenating its sequence of time-stamped observations from all concerned traffic cameras. Applying such a solution in practice is challenged by the inherent incompleteness and inaccuracy of the camera sensing network. First, the traffic camera observations are spatial-temporally incomplete —

not all road intersections are covered by traffic cameras and that results in uncertainties in inferring the true moving trajectories over those uncovered areas. Such data incompleteness is made more challenging when considering camera malfunctions and data loss. Second, the vehicle identities extracted from video and image analysis may be inaccurate, and it is further complicated due to the heterogeneity among traffic cameras and their data quality — the vehicle snapshots taken from different cameras (e.g., front-view v.s. rear-view) and under different environmental conditions are of different quality in resolution, view field, focus control, and clarity. State-of-the-art vehicle identification based on license plate recognition (LPR) or visual re-identification (Re-ID) may give inaccurate results for low-quality camera data [42].

This paper proposes a comprehensive system design centered around reducing the uncertainties of vehicle identification. The proposed design considers the vehicle identities based on both their appearance in the camera data as well as their mobility dependency across locations where those camera data are obtained. Multi-dimensional similarity examination is performed with vehicle snapshots taken from different traffic cameras. Specifically, our system examines similarities on LPR text, vehicle appearance, and mobility causality across the locations of observation, and gains combined confidence to relate vehicle identities across different camera observations. A graph convolution network (GCN) model is built to cluster vehicle snapshots, and based on that maintain the identity consistency across different camera observations. With GCN, our approach benefits from its efficacy in belief propagation and is able to converge to accurate clusters of representations. Finally, our system joins the graph of GCN and the graph of urban road network, and undergoes a self-training process which incorporates the complex mobility dependency along the urban road network into the GCN, and thus further improves the accuracy of the reconstructed vehicle trajectories.

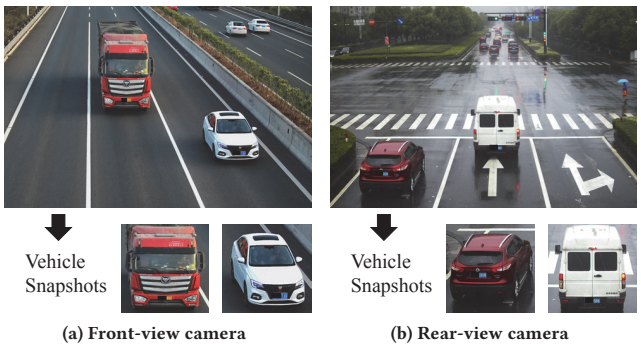
We implement the proposed approaches and build an end-to-end system — VeTrac. We extensively experiment VeTrac with real-world data collected from a network of 1342 cameras deployed in an urban region. A full day camera data consisting of over 7 million vehicle snapshots are examined, from which 1,247,835 vehicle trajectories are reconstructed. We evaluate VeTrac's performance based on several sets of manually labeled vehicle trajectories as well as synthesized ground truths from both public transits and private vehicles. The evaluation results suggest that VeTrac achieves 98% accuracy for simple expressway scenario and 89% accuracy for complex urban environment. The achieved accuracy outperforms alternative solutions by 32% for expressway scenario and by 59% for complex urban environment.

2 SENSING WITH CAMERAS

2.1 Why Vehicle Trajectories?

VeTrac reconstructs a city-wide vehicle trajectory dataset that contains general vehicles. Such a dataset provides more complete mobility information for individual, communities, as well as the city, and thus benefits various applications.

Intelligent transportation systems (ITS). Many real-world ITS are built with the knowledge of general vehicle trajectories, e.g., usage-based electronic toll collection (ETC) for highway fares, trajectory-based road pricing for congestion management, traffic light control based on prediction of driving intention, and so on.



(a) Front-view camera (b) Rear-view camera

Figure 1: Example frames of two typical traffic cameras

Traffic analysis. Traffic analysis is essential in transportation research. Instead of sticking to surveyed origin-destinations (ODs), recent studies have been using GPS-tagged vehicle trajectories (e.g., those of taxi fleets) to conduct analytical studies (e.g., OD analysis and travel time estimation [19, 34, 41]), and with that to improve the transportation efficiency and resilience [18, 38]. Obviously, a dataset with general vehicle traffics can more truthfully reflect the city mobility and support more accurate decision making.

Retrospective analysis for urban planning. Vehicle movements also reflect the operations of a city. By analyzing the observed vehicle movements, researchers are able to discover city functionalities [37], trace social footprints [7] and improve municipal services [21, 29, 30] as well as urban infrastructures [10].

2.2 Camera Sensing Network

This work employs a traffic camera network for sensing vehicles. A traffic camera is a video camera which monitors vehicles on a road. These cameras are often mounted on overhead poles and aimed at one or more lanes within their field of view (FoV). Different from speed cameras which upon triggers take still pictures of much higher resolution, traffic cameras are purposed for observation and constantly take low-resolution videos. In urban areas, traffic cameras are installed to capture either the front face or the rear face of passing vehicles. Figure 1 shows example frames from the front-view and rear-view cameras. Front-view cameras are deployed for controlling access at important locations like entrances/exits of expressways or key locations along arterial roads, where both the vehicle and its driver are monitored; Rear-view cameras are used to monitor traffic violations (e.g., running a red light), and are deployed at intersections of surface roads, where both the vehicle and traffic light signals are recorded.

Infrastructure and data. To understand the real-world deployment of traffic cameras, we investigate an urban area in China, with 1342 traffic cameras covering 66 km^2. Figure 2 visualizes the road network within the area and the locations of the traffic cameras. As shown, the front-view cameras (denoted in red) are mainly distributed on the left of the map where locate expressways and arterial roads that lead to a local airport, whereas the rear-view

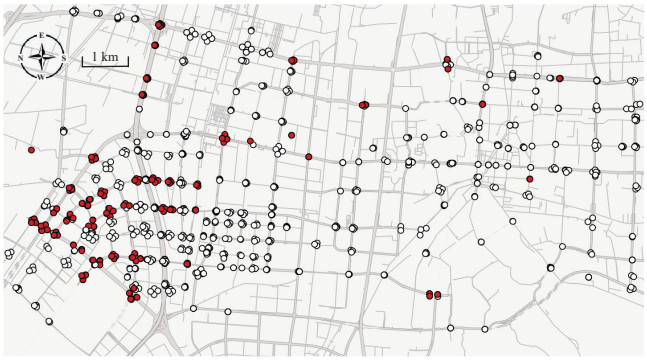


Figure 2: Spatial distribution of traffic cameras, where front-view cameras are denoted in red and rear-view cameras are denoted in white

cameras (denoted in white) scatter across road intersections over the entire area.

We perform a vision-based vehicle detection and tracking algorithm [15] over the video frames collected from the traffic cameras, and obtain snapshots each of which pictures an individual vehicle like those shown in Figure 1. Each snapshot is associated with meta information including time, location and road lane, as Table 1 illustrates[1].

On average, ~7 million vehicle snapshots are generated from the 1342 cameras every day. In this paper, we specifically report the results from the data collected on 2019-03-06, which contains 7,206,500 snapshots in total. Figure 3 presents the data statistics at different time and across cameras. Figure 3a plots the number of vehicle snapshots generated at different time of day. It shows two peak hours (i.e., 8 am and 5 p.m.), which accords with the local traffic demands. Figure 3b plots the distribution of the number of snapshots generated by the 1342 cameras. As shown, 90% of the cameras generate fewer than 10k snapshots, and the majority generate 2k–6k snapshots a day.

Data incompleteness. The snapshot data are spatial-temporally incomplete due to two main reasons. First, not all road intersections are covered by traffic cameras. The area depicted in Figure 2 contains 334 major intersections but only 190 of them (57%) are covered with cameras. Second, not all passing vehicles are properly captured as snapshots at all time. The miss is mainly attributed to camera-side issues (e.g., frame skipping, interrupted power, transmission loss, corrupted images) as well as vehicle detection failures due to low quality video input. As a result, a substantial portion of the cameras (>10%) generate fewer than 1k records as Figure 3b shows. Although traffic volume may differ across cameras, the majority of cameras provide a proper recall of over 2k records per day.

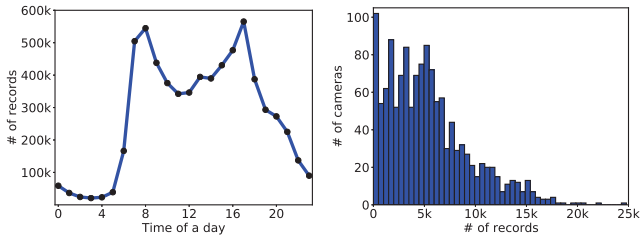
Data inaccuracy. The problem of data incompleteness is exacerbated by the inaccuracy introduced when extracting vehicle identities from the snapshots. License plate is one strong identifier for vehicles. However, the license plate image contained within each snapshot is often small and subject to conditions like blurred, dusty, obscured, incomplete or bad lighting, which introduce errors to recognized results. Figure 4a gives examples of those imperfect conditions. Visual appearance is another useful clue to identify vehicles, which however may also introduce errors due to the variance in image qualities and environment conditions, e.g., viewpoints, illumination, backgrounds. Figure 4b gives an example showing that different vehicles may have very similar appearances, while Figure 4c gives examples showing how the same vehicle captured by different cameras may look quiet different.

[1]The data used in this work come from an in-use transportation management system. The data were hosted in the data owner's private servers and can only be accessed internally under a proper supervision.

Field	Description
Camera metadata	Camera ID, location and viewpoint
Snapshot	An Image of the detected vehicle
Entering Time	Timestamp of entering the camera's FoV
Exit Time	Timestamp of leaving the camera's FoV
Lane directions*	Permissible driving directions of the lane

*Land directions might include left / right turn, U-turn, straight, and their combinations

Table 1: Information in a vehicle snapshot record



(a) Records over time **(b) Records over cameras**

Figure 3: Statistical summary of vehicle snapshots

2.3 Uncertainties in Vehicle Identification

The data incompleteness and inaccuracy cause uncertainties in vehicle identities, which hinder trajectory reconstruction. In this subsection, we analyze the cause of identity uncertainties, and quantify its impact on state-of-the-art vision based schemes targeting license plate or appearance.

License Plate Recognition (LPR). This scheme represents the common practice of obtaining vehicle trajectories from camera data. A LPR model (usually a deep CNN-based model [27]) is used to identify plate characters from each snapshot, based on which snapshots are then grouped to assemble trajectories.

The accuracy of LPR based scheme is however subject to accurate recognition of plate characters. Most existing LPR algorithm assume high-quality input, normally high-resolution images with decent lighting conditions. This is common in applications like automatic parking payment system where a front camera is dedicated to capturing plates at a short distance. Unfortunately, traffic camera snapshots are usually small and subject to a variety of imperfections as Figure 4a shows, which make LPR results error prone.

Inaccurate LPR results of the same vehicle result in high uncertainty in inferring its trajectory. Figure 5a presents a typical

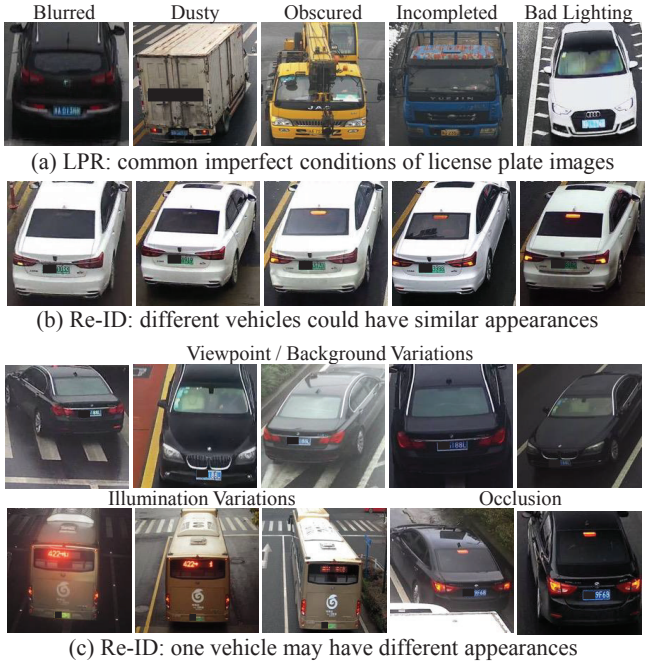


Figure 4: Cause of Identity uncertainties

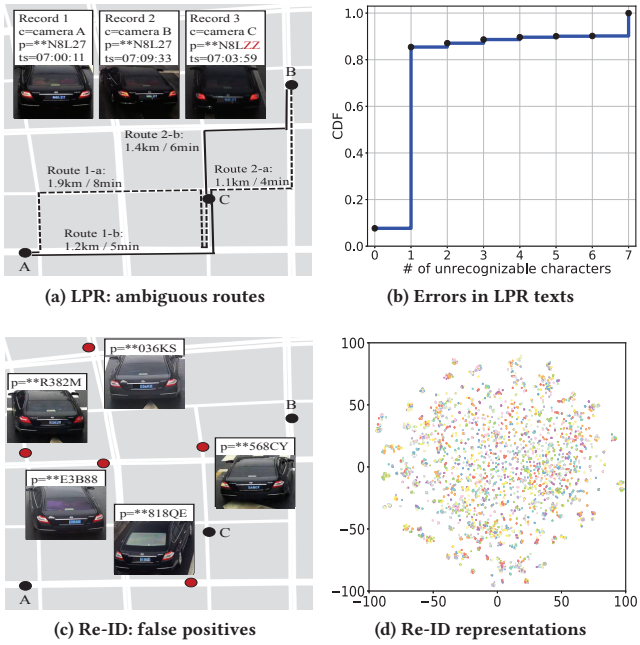


Figure 5: Impact of identity uncertainties on vehicle trajectory reconstruction

example from what we experience. Due to LPR errors, record 3 from camera C is wrongly recognized as a different identity, which leads to uncertain route candidates from camera A to camera B. Although the time and distance constraints could have helped remove such uncertainties, the efficacy has a limit, and that is further impaired when the observations become sparse due to both LPR errors as well as insufficient camera deployment.

Figure 5b summarizes the LPR errors tested from a set of 2610 snapshots that we collect. The result shows that less than 10% of the snapshots are correctly recognized. For those incorrectly recognized plate, over 75% are with only one character wrong, while ~12% are completely unrecognizable.

Vehicle Re-Identification (Re-ID). Visual Re-ID aims at retrieving a vehicle of interest from an image database with a general model based on visual characteristics. A trained deep CNN-based model is often employed to extract features of appearance information in each vehicle snapshot. With that, all snapshots are ranked based on their feature distances to a given query image, and snapshots within a predefined threshold are retrieved to assemble trajectories.

However, using only visual features to identify vehicles are fundamentally challenging, as the examples in Figure 4b and Figure 4c suggest: (i) different vehicles may be similar in appearance, and (ii) the same vehicle observed from different camera views could have different appearances due to the variance in image qualities and environment conditions. Such discrepancy limits the identity accuracy of retrieved snapshots, especially with a large dataset of millions of snapshots. Figure 5c depicts this observation with the same vehicle. If we apply Re-ID based scheme with only visual features, five extra snapshots of other vehicle identities rank top

Accuracy	Daylight (8 a.m.)	Evening (8 p.m.)	Rain
LPR errors <=2	92%	84%	88%
Vehicle Color*	98%	89%	97%

*the accuracy of vehicle color classification based on visual features

Table 2: Impact of environment varieties

together with record 3, and the inclusion of them will severely divert the reconstructed trajectory.

We also examine the appearance features of the expressway dataset (§4.2). Figure 5d visualizes the feature vectors (each a sample point projected to a 2D space by TSNE projection) of those snapshots, where the samples of the same vehicle are denoted in the same color. It is clearly shown that samples of different vehicles are tangled together and hard to distinguish solely from their appearance features.

To mitigate the identity ambiguity, recent research efforts in visual Re-ID [2, 9, 16, 35, 42] mainly resort to learning discriminative features of vehicles or augmenting training data by generating near-duplicates and data of other views. These approaches often require high resolution input images and large amount of labeled data for training, which are not available in this study nor in many practical camera surveillance systems. In §5, we summarize recent studies in Re-ID and address its difference with trajectory reconstruction.

Environment variety. To evaluate the potential impact that environment variety has on the input data quality, we perform LPR and vehicle color classification over 3,000 snapshots independently collected in three common environment contexts, i.e., "Daylight" for reference, "Evening" for light changes and "Rain" for weather

changes. The results in Table 2 suggest consistent qualities in identity clues from vision based schemes (with slight drop of accuracies for evening and rain contexts).

3 VETRAC DESIGN

To tackle the identity uncertainties due to incomplete and inaccurate observations from traffic cameras, we propose VeTrac, which exploits the mobility dependency and embeds that in a graph convolution process in order to reduce such uncertainties. VeTrac employs a multi-dimensional similarity (MDS) block to combine estimations from different aspects and gain extra confidence on vehicle identification (§3.1). A graph convolution network (GCN) is developed to capture system-wise correlations among vehicle snapshots and achieve global optimality in assigning snapshots to different vehicle identities (§3.2). The vehicle trajectories are produced based on snapshots belonging to different vehicle identities. To further improve the accuracy, VeTrac also applies an iterative self-training process to incorporate the trajectory-level mobility within the road networks into the GCN (§3.3).

3.1 Multi-Dimensional Similarity (MDS)

For any pair of vehicle snapshots, MDS block derives a more robust identity similarity measure with fused identity estimations from LPR texts, vehicle appearance and mobility causality. We detail the three similarity estimators as follows.

LPR similarity. Figure 5b suggests that over 90% of the LPR results contain errors with the traffic camera snapshots. The figure however also suggests that ~80% of those only contain 1–2 unrecognized characters, which means a substantial portion of the LPR text can provide information on vehicle identities. In order to use that information, VeTrac employs a neural network model to quantify similarity between any pair of LPR texts.

Conventional LPR models assume fixed-length inputs and adopt a CNN-based multi-headed network architecture, where a CNN-based backbone network (e.g., VGGNet [27]) first extracts a global feature vector from the license plate image and uses a number of heads (i.e., fully-connected layers) to decode the feature vector into a fix-length text sequence. Each of the head is trained to classify one character in the license plate image into one of many predefined characters. When a mismatch of plate length occurs (which is common with the traffic camera snapshots under the imperfect conditions as exemplified in Figure 4), the model wrongly segments the feature vector and thus introduces uncontrollable uncertainties to LPR results.

Instead of the all-or-none strategy, VeTrac recognizes readable characters by convolutional recurrent neural network (CRNN) [26]. CRNN is chosen because it can support variable-length recognition with slight modifications to three types of its layers: (i) Convolution layers, which extract a feature sequence from the input license plate image. Different from conventional LPR models, the convolution layers obtain a sequence of feature vectors, each of which corresponds to a region smaller than a character. (ii) Recurrent layers, which predict a label distribution for each feature vector. The recurrent layers consist of multiple bi-directional LSTM cells, each trained to predict a label distribution based on the information of its own element as well as the information from adjacent cells.

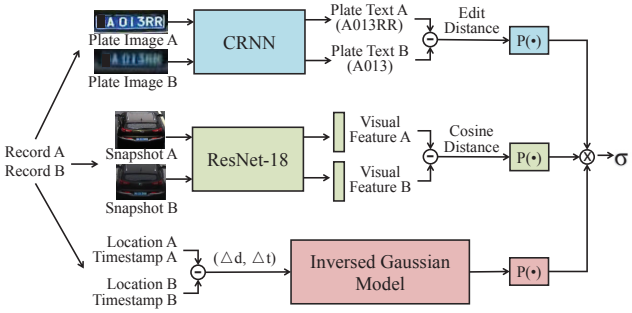


Figure 6: Multi-dimensional Similarity (MDS) block

The unrecognizable characters on the plate can be skipped in the model. (iii) Transcription layers, which convert the prediction of each LSTM cell to a text of readable plate characters.

With the CRNN-based LPR model, VeTrac is able to handle license plates that contain unrecognizable characters or have varied lengths. The similarity between any two LPR texts A and B can thus be modeled by:

$$P_{plate}(A,B) = 1 - 0.08 * ed(A,B) * (ed(A,B)+1) \quad (1)$$

where $ed(A,B)$ refers to the edit distance of plate texts.

Appearance similarity. VeTrac adopts a multi-head CNN module to extract high-level visual features containing information of the vehicle color, type and make from the snapshots. Different from previous works that exploited sophisticated features (§5), we focus on the most intrinsic three attributes for their general availability and relative robustness across snapshots of various resolutions, viewpoints and lighting conditions as offered by the traffic camera network.

The CNN module employs a ResNet-18 network as the backbone to extract a 512-dimension feature vector from each snapshot. The feature vector is then independently fed to three heads, each of which is a fully-connected layer trained to classify one attribute of the following:

- The color head classifies each snapshot into one of the 10 colors, i.e., white, black, grey, blue, red, yellow, green, brown, purple and pink.
- The type head classify each snapshot into one of the 9 types, i.e., sedan, SUV, bus, minibus, taxi, van, MPV, truck, minitruck.
- The make head classifies each snapshot into one of the 96 brands, e.g., Audi, Honda, Nissan, Toyota, Volkswagen.

The appearance similarity of any two snapshots can thus be quantified by the cosine distance of their feature vectors:

$$P_{app}(A,B) = 1 - \frac{\sum_{i=1}^{512} A_i B_i}{\sqrt{\sum_{i=1}^{512} A_i^2}\sqrt{\sum_{i=1}^{512} B_i^2}} \quad (2)$$

where A and B are the corresponding feature vectors.

Mobility similarity. VeTrac models the similarity between two snapshots taken at different times and locations by the probability that a vehicle can travel from one to the other within the time

constraint. The lane direction (see Table 1) is specially considered to build more accurate mobility transitions.

VeTrac first extracts direct neighbors of each camera by considering the road connectivity and camera lane information. With that, a camera transition graph is built where each vertex represents a camera and each edge denotes a direct link between neighboring cameras (which corresponds to the road segment connecting the two cameras). Two attributes are specified for each edge: (i) The transition time of each edge is estimated by the map API [1] provided by a popular vehicle navigation service provider in the country. (ii) The lane direction of each edge denotes the driving action (i.e., right turn, left turn, straight, or U-turn) it takes to transit from the camera to its neighbor on the graph.

With the camera transition graph, possible routes and their estimated travel time can be enumerated. The mobility model between each pair of cameras can be learned and represented by an inverse Gaussian distribution $IG(\mu, \lambda)$ [13]. Thus, the mobility similarity between snapshots can be modeled by:

$$P_{mob}(t|\mu,\lambda) = \left[\frac{\lambda}{2\pi t^3}\right]^{1/2} \exp\left\{-\frac{\lambda(t-\mu)^2}{2\mu^2 t}\right\} \tag{3}$$

where t is the difference in timestamps of two snapshots, and μ and λ are learned parameters of inverse Gaussian distribution $IG(\mu, \lambda)$, which models the travel time from the road segments of one camera to that of the other.

MDS score. For any pair of snapshots, VeTrac employs an MDS block (as depicted in Figure 6) to parallelly estimate their similarities on all three aspects, and uses the product to describe the probability of the two snapshots belonging to the same vehicle trajectory:

$$\sigma_{A,B} = P_{plate}(A,B) * P_{app}(A,B) * P_{mob}(A,B) \tag{4}$$

3.2 Learning Vehicle Identities with Graph

MDS derives a more robust identity similarity measure for any pair of snapshots. The next key problem to trajectory reconstruction is how to accurately assign vehicle identities to snapshots for assembling trajectories.

Query-based approaches have been conventionally applied to solve the identity assignment problem, where all snapshots are ranked based on their distances to a given query and those within a predefined threshold are considered as being of the same identity. However, most query-based approaches provide no guarantee that all top images are of the same identity and the best threshold may differ from query to query. As a result, query-based approaches fail non-negligible number of hard cases with high intra-trajectory diversity or high inter-trajectory similarity.

VeTrac addresses this problem by modeling complex correlations among many snapshots with a graph structure. Instead of considering only pair-wise similarities (i.e., the query to other snapshots), a graph can model system-wise relationships (i.e., among all the snapshots) to achieve global optimality in assigning vehicle identities to different snapshots. The hard cases that fail pair-wise similarity assessment may be inferred from correlation of their similarity assessments with other strongly connected peers. Errors introduced in MDS estimation may also be mitigated through a graph convolution process.

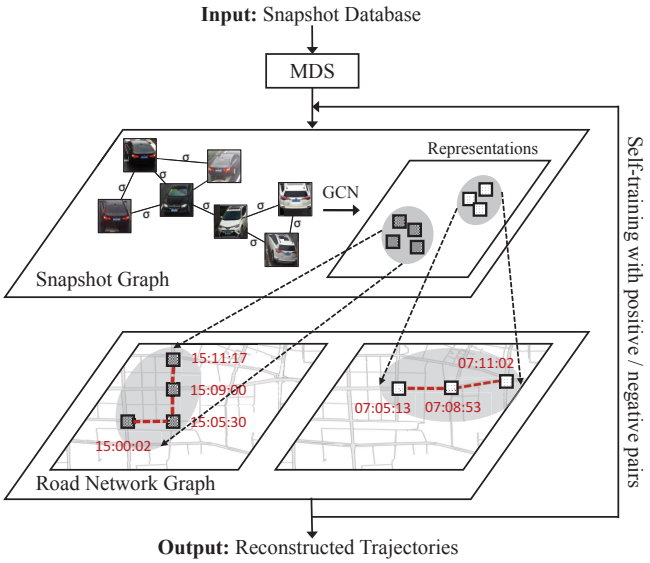


Figure 7: VeTrac overview

Figure 7 gives an overview of the VeTrac's workflow. The MDS scores are taken as input to a graph convolution network (GCN) containing all vehicle snapshots. After a graph convolution process, a new identity representation is learned for each snapshot and snapshots can then be clustered into individual vehicle trajectories based on the representations of all snapshots. In the following, we detail the GCN-based learning process in three steps.

Snapshot graph construction. With the full dataset of vehicle snapshots, VeTrac builds a undirected graph $G_{snapshot} = (V, E)$, where a vertex set V corresponds to all snapshots and an edge set E denotes the multi-dimensional similarity (MDS) between two vertices. Theoretically, $G_{snapshot}$ can be a complete graph. Such a graph however will introduce quadratic complexity, i.e., $O(n^2)$ where n is the number of vertices, to the convolution process, and thus not scale to our graph containing millions of vertices. In practice, VeTrac reduces the number of edges by removing those of large spatio-temporal spans. We define two vertices to be *k-hop connected* if the two snapshots are taken from cameras within k-hop distance in the road network. Similarly, we define two vertices to be *T-minute valid* if the time difference between their timestamps is below T minutes. An edge is generated only between those *k-hop connected* and *T-minute valid* vertices. In VeTrac's implementation, k and T are empirically set to 3 and 10, and that helps to reduce the edges from ~52 trillion to ~2 billion, where the average vertex degree is 307. After calculating MDS scores as edge weights and removing edges of zero weight, the average vertex degree becomes 15.

Learning identity representations. Graph convolution is performed on the snapshot graph, where each vertex keeps updating itself to a suitable representation based on the knowledge it gains from its neighboring vertices.

Figure 8 depicts the developed GCN structure and the learning process. In the input layer, for each vertex $v_i \in V$, a representation

vector x_i is initialized to represent its identity. To start with, the one-hot encoding technique can be used to assign each vertex a unique representation, e.g., a 1-by-n vector with its i-th element to be 1 and the rest to be 0 for x_i, where $n = |V|$. The initial representation matrix X^0 can thus be constructed by combining representation vectors of all vertices, as indicated by Equation 5.

$$X^0 = I = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{5}$$

In practice, we leverage noisy LPR results to initialize representations to speed up the convolution process. The rationale is that, the more a certain LPR text is observed across different cameras, the more likely that LPR text is a correct one. So for a snapshot with an LPR text that appears more than three times in the entire dataset we consider the LPR text credible and thus label all involved snapshot vertices by the same representation vector.

$$A = \begin{bmatrix} 1 & \sigma_{1,2} & \cdots & \sigma_{1,n} \\ \sigma_{2,1} & 1 & \cdots & \sigma_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n,1} & \sigma_{n,2} & \cdots & 1 \end{bmatrix} \tag{6}$$

An adjacency matrix A is also constructed with MDS scores as edge weights, as indicated by Equation 6. Each element in A represents the probability of the corresponding two vertices belonging to the same trajectory. We add self-loops to the snapshot graph by setting the diagonal as 1. Since the graph only contains a significantly reduced edge set, the adjacency matrix A is a sparse matrix with most elements equal 0. The sparsity of A ensures the computational efficiency when performing graph convolution.

In the graph convolution layers, the representation of each vertex is iteratively updated by aggregating its own representation and the representations of its neighbors. The aggregation is a normalized averaging with regard to edge weights. The propagation rule of graph convolution at each iteration can be expressed in matrix multiplication, i.e., $X^{i+1} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X^i$, where $D = diag(d_1, d_2, ..., d_n)$ is the degree matrix of A and $d_i = \sum_j a_{ij}$ is the degree of vertex v_i.

From representations to identities. A two-layer GCN (Figure 8) is used to learn new identity representations. The two graph convolution layers complete the aggregation of updated identity information — in the first layer each vertex updates its representation with knowledge from its neighbors and in the second layer the updated representations are aggregated across the snapshot graph. In such a way, the vertices converge to proper representations and the complex correlations in the graph are encoded into those representations. The snapshot representations of the same trajectories tend to agree while those of different trajectories tend to differ. Therefore, we can cluster the snapshots into individual trajectories by identifying snapshots of similar representations. Since the learned representation vectors are sparse, VeTrac first employs the random projection [4] to reduce the dimensionality to 512 and then performs HDBSCAN ($min\_cluster\_size = 5$) [5] to cluster vertices with the cosine distance. The random projection and HDBSCAN

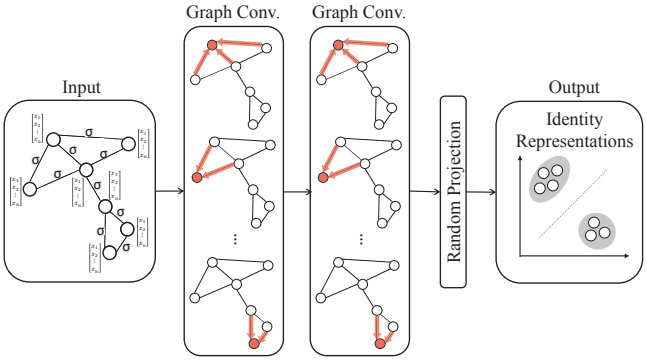


Figure 8: Identity learning with the two-layer GCN

are chosen for their accuracy and efficiency in processing sparse vectors.

3.3 Self-Training with Road Network Graph

The GCN learns graph wide correlations and mitigates identity uncertainties from inaccurate pair-wise similarity estimations. As a result, clusters of same vehicle identities are identified. For each cluster of snapshots, VeTrac is able to generate a maximum likelihood trajectory that connects them based on their timestamps.

The reconstructed trajectories, though greatly improved from those query-based approaches, may still contain errors in its precision (snapshots of other vehicle contained in the trajectory) and recall (not all snapshots of the trajectory are retrieved). We perform an error analysis on the GCN results of the expressway dataset (§4.2) and observe that over 70% of the generated clusters still contain errors mainly due to two problems: (i) 14% of them are incomplete clusters which contain only snapshots from the same vehicles but do not have full recall, and (ii) 85% of them are imprecise clusters which contain snapshots from more than one vehicle. As we introduce more graph convolution layers in the GCN for thorough identity propagation, the former source of errors may further go down while the latter persists. That means the inaccuracy in identity clustering stems from the difficulty in separating negative pairs of snapshots rather than aggregating positive pairs of snapshots in the clusters.

VeTrac inspects the trajectory-level mobility within the road network to address the issue. As depicted in Figure 7, an iterative self-training process is introduced, which detects positive and negative pairs from the reconstructed trajectories and feeds the information back to the GCN to stop identity propagation between negative pairs. The self-training process enhances the GCN by instilling trajectory-level mobility into the representation learning process. Three steps are involved in building the self-training loop.

Build the road network graph. VeTrac builds a directed road graph $G_{road} = (V, E)$ from the road networks, where a vertex set V includes all road segments each represented as a vertex and an edge set E denotes the linkage and physical directions between road segments. The information about each road segment (i.e., coordinates of its two ends, name, length, category, accessibility for vehicles) is extracted from OpenStreetMap, which is then used to derive the linkage between adjacent road segments. The vehicle

transition time on each edge is obtained from the third-party vehicle navigation service [1]. Each snapshot in $G_{snapshot}$ can be mapped to its corresponding vertex in G_{road}. The transition time from one snapshot to another along the road network can thus be estimated via shortest path search on G_{road}.

Detect positive and negative pairs. For a specific snapshot, its similar snapshots that have a small cosine distance can be retrieved. The retrieved snapshots are spatio-temporally connected to the query snapshot as ensured by the mobility branch of MDS (snapshot-level), but together they may fail to form a feasible trajectory (trajectory-level). By modeling trajectory-level mobility, VeTrac examines whether those snapshots constitute one or more feasible trajectories. Specifically, the snapshots are first mapped into the G_{road}. With their timestamps and lane directions (see Table 1), possible trajectory candidates are enumerated on G_{road}. The feasibility of a trajectory is then calculated based on transition probabilities regrading time and distance between every two consecutive snapshots. The transition probability is modeled by an inverse Gaussian, as indicated by Equation 3. The overall feasibility of a trajectory is derived as the product of all consecutive transition probabilities and regularized by the total number of concerned snapshots. An empirical threshold in the range of 0.65–0.85 can be set to judge the feasibility (0.8 is set in VeTrac's implementation, though the performance is insensitive to the exact value). Based on that, VeTrac labels all pairs of snapshots on $G_{snapshot}$. A pair of snapshots from a feasible trajectory is considered as a positive pair, and that from an infeasible trajectory or different trajectories is considered as a negative pair.

Iterative graph convolution with positive and negative pairs. With positive and negative pairs, VeTrac adjusts the adjacency matrix A in GCN. For each positive vertex pair, an edge is added if there exists none in the original $G_{snapshot}$, and the MDS block is invoked to derive its weight. For each negative vertex pair, the edge is removed if it exists in $G_{snapshot}$. After the adjustment, VeTrac reruns the graph convolution process and updates the identity representations. The self-training process is iterative and terminates when all newly obtained pairs become positive. The self-training process converges after 9 rounds on our dataset.

After the iterative graph convolution and self-training process, VeTrac obtains clean and confident identity representations, and with that reconstructs feasible vehicle trajectories.

4 EVALUATION

In this section, we describe the experimental setups (§4.1), compare VeTrac performance with state-of-the-art alternatives under various scenarios (§4.2) and analyze VeTrac by its components (§4.3).

4.1 Experimental Setups

Data and Implementation. We use the data described in §2.2, which contain 7,206,500 snapshots and make up ~150 GB. We implement VeTrac on a server equipped with an Intel Xeon E5-2682 2.50GHz CPU and an NVIDIA RTX 2080Ti GPU (32 GB RAM). The MDS block is trained with 10,000 labeled snapshots collected from similar scenes. VeTrac takes approximately 3 hours to reconstruct vehicle trajectories from one million snapshots, and takes 16 hours in total to process all the data. The processing time can be significantly reduced by paralleling the self-training module.

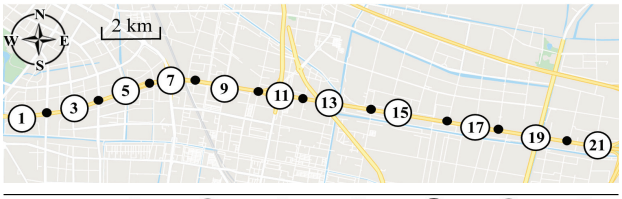
Overall statistics. In total, VeTrac reconstructs 1,247,835 trajectories, 239,405 of which took place during the morning rush hours (i.e., 7–9 a.m.). For roads, about 3% of the roads are traveled more frequently by at least 10% of the trajectories, while most of the roads (88%) are traveled by at least one thousand trajectories. For trajectory distance, those trajectories usually pass 8–12 intersections with an average distance of 3.8 kilometers. Most of the trajectories (about 90%) are within a distance of 10 kilometers. For trajectory duration, the average is 16.9 minutes. 40% of the trajectories are within a duration of 10 minutes, while 20% of the trajectories have a duration longer than an hour.

Alternative schemes to compare. To the best of our knowledge, none of existing works can directly reconstruct trajectories of general vehicles from camera snapshots. We thus borrow key ideas of three different principles and implement them for reconstructing the vehicle trajectories:

- **License plate recognition (LPR) [27].** As described in §2.3, this scheme represents common practice of obtaining vehicle trajectories from camera data. We implement the scheme by employing VGG-16 to recognize license plates, ordering snapshots of the same plate according to timestamps and forming trajectories by linking consecutive snapshots with the shortest path on the road network.
- **Vehicle re-identification (Re-ID) [17].** As described in §2.3, this scheme represents visual appearance based re-identification methods. Considering the high false-positive rate of using only visual features, we adopt a progressive vehicle Re-ID pipeline as suggested by [17] due to its state-of-the-art performance when multimodal information is available. Specifically, we first employ a coarse filtering with visual features to select candidates, which are then refined by removing snapshots with large plate difference (i.e., 3 or more characters) or infeasible spatio-temporal relations. A most likely trajectory is formed by linking consecutive candidate snapshots with the shortest path on the road network.
- **History based route inference system (HRIS) [40].** This scheme represents mobility-based methods that reduce uncertainties of low-sampling-rate trajectories. Instead of looking into camera data, this scheme assumes that trajectories with the same OD usually have similar route choices, and thus rebuilds uncertain parts of a trajectory by referring to other vehicles with the same OD. We implement this scheme by taking the results of LPR as input and assign the most popular routes sharing the same ODs as final trajectories.

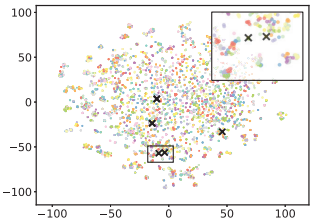
Performance metrics. We define four metrics:

- *acc* measures the accuracy of reconstructed trajectories, and is defined as the fraction of correctly reconstructed trajectories among all the trajectories.
- p measures the precision of a vehicle cluster, and is defined as the fraction of true positives within a cluster.
- r measures the recall of a vehicle cluster, and is defined as the fraction of the total number of snapshots that are actually clustered into a cluster.

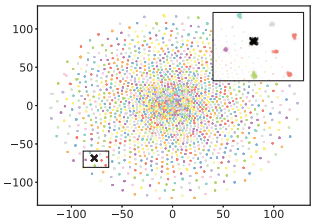


Interchange	5	6	7	8	9	10	11
Ground Truth							
VeTrac	✓	✓	✓	✓	✓	✓	✓
LPR	✓	✓		✓		✓	
Re-ID	✓		✓	✓	✓		
HRIS	✓			✓			

(a) Comparison of reconstructed trajectories



(b) Visual representation



(c) VeTrac representation

Figure 9: An example. (a) Ground truth of a trajectory passing 7 interchanges; (b) The snapshots clustered in the representation space of visual features; (c) The snapshots clustered in the representation space of VeTrac. The black crosses in (b) and (c) indicate 7 snapshots of the same vehicle in the representation spaces.

- D_{geo} quantifies the errors in reconstructed trajectories. We represent geolocations in trajectories with longitude and latitude coordinates, and employ the dynamic time warping algorithm [24] to derive D_{geo}. A smaller D_{geo} means two trajectories are geographically closer to each other and $D_{geo} = 0$ when they are identical.

4.2 Performance Evaluation

We assess the quality of reconstructed paths by emulating two test scenarios of VeTrac:

- **Expressway: complete ground truth.** We form a expressway dataset from the data collected along a local airport expressway. The expressway contains 21 interchanges, where vehicles can either enter the expressway or exit via ramps. We collect the morning-rush-hour data (i.e., 7–9 a.m.) from the 54 cameras deployed along the expressway and obtain 8945 snapshots. Since the expressway has a simple road topology and relatively isolated traffic, we are able to manually label the true identities of all obtained snapshots, which contain 1746 vehicles.

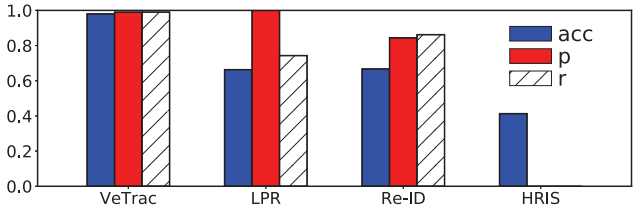
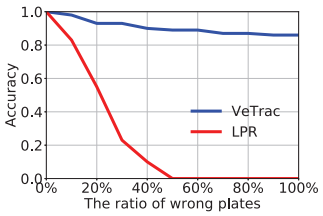
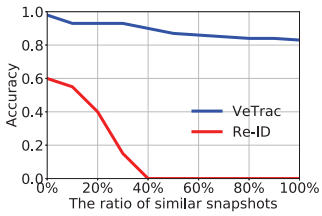


Figure 10: Expressway: end-to-end performance



(a) Wrong LPR texts



(b) Similar appearances

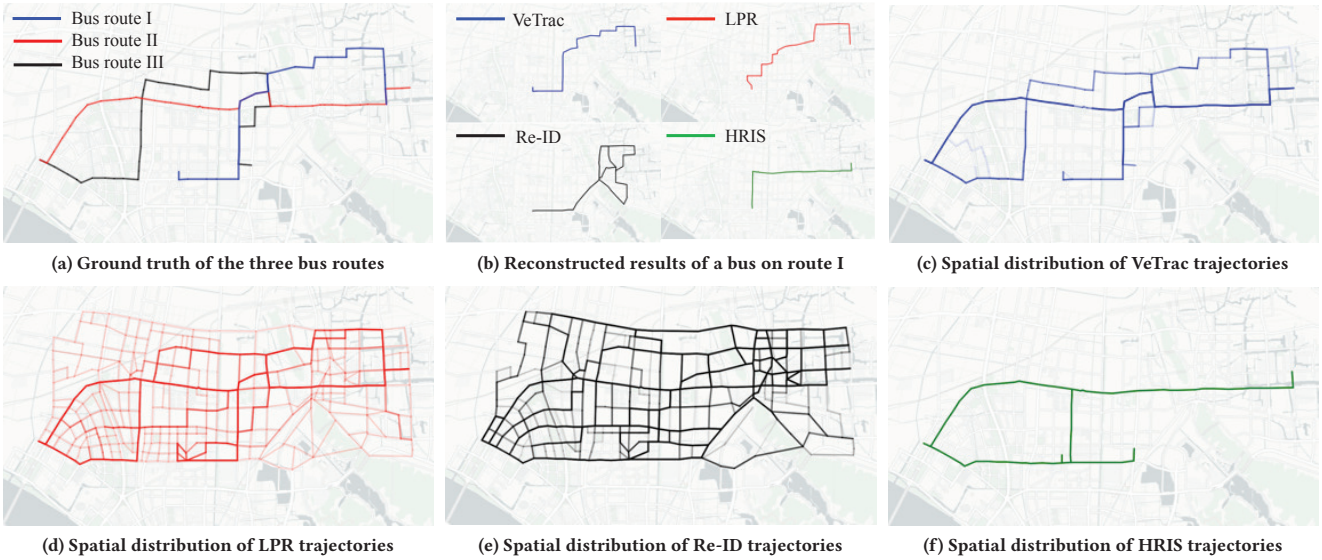
Figure 11: Robustness evaluation

- **Urban: large-scale and complex road network.** We form a urban dataset by collecting morning-rush-hour data (i.e., 7–9 a.m.) from the original dataset, which consists of 1,098,467 snapshots obtained from 1342 cameras. We do not have full ground truths, so we rely on manually labeled representative subsets to evaluate the performance.

Expressway. Figure9a visualizes the expressway and locations of the 21 interchanges (i.e., from ① to ㉑). With the complete ground truths of 1746 vehicles, we observe that vehicles' ODs evenly distributed across the 21 interchanges. Most of the vehicles (more than 80%) passed 4 or more interchanges before they exited the expressway.

Figure 9a inspects a representative trajectory that enters the expressway from interchange ⑤, passes 7 interchanges and exits from interchange ⑪. The ground truth of the trajectory with the actual snapshots captured by the cameras at those interchanges are presented. With the snapshot captured in interchange ⑤ as the query, the remaining rows of Figure 9a indicate snapshots that are clustered as the same identity by VeTrac and alternative schemes, respectively. The results show that VeTrac successfully clusters all 7 snapshots into one, even though some license plate are wrongly recognized (i.e., at interchange ⑦, ⑨ and ⑪) and some snapshots are visually different from the rest (i.e., at interchange ⑥, ⑩ and ⑪). As comparison, LPR is vulnerable to plate errors and Re-ID fails to recall some of the snapshots with different viewpoints due to visual differences. HRIS cannot rebuild the correct trajectory either, due to the incapability of extracting unobvious mobility dependency from the historical data.

Figure 9b depicts the representation space of visual features and Figure 9c depicts that of VeTrac, by TSNE projection. Each point indicates a snapshot and those of the same vehicle identity are marked with the same color. The 7 snapshots from the trajectory are highlighted in the two representation spaces. It is clear to see that they are separated in the representation space of visual features



(a) Ground truth of the three bus routes (b) Reconstructed results of a bus on route I (c) Spatial distribution of VeTrac trajectories

(d) Spatial distribution of LPR trajectories (e) Spatial distribution of Re-ID trajectories (f) Spatial distribution of HRIS trajectories

Figure 12: Result trajectories for the public transits

but are clustered very close in VeTrac representation space. In the representation space learned by VeTrac, snapshots of the same identity but different appearances are located more closely (average intra-class distance is 1% of that in the representation space of visual features), while snapshots of different identities but similar appearances are better separated (on average, VeTrac introduces 0.3 false positives to recall all related snapshots while visual features introduce 387.5 false positives). The results suggest high-quality representations learned by VeTrac, which contribute to the high clustering accuracy.

Figure 10 summarizes the end-to-end performance of all methods in terms of the average accuracy, precision and recall. Overall, VeTrac achieves 98% accuracy, which outperforms all alternatives by at least 32%. For precision and recall, VeTrac achieves 99% recall at the cost of merely 1% decrease in precision. For LPR, although all the recalled snapshots are correct, it cannot handle snapshots with wrongly recognized plates, which results in only 74.3% recall. Re-ID achieves a slight improvement on the recall (86.2%), but at a cost of largely reduced precision (84.4%). HRIS gives the lowest accuracy (41.3%) because the route choices between similar ODs are not always the same.

To evaluate the robustness of VeTrac, we perform controlled experiments by synthesizing two groups of datasets from the expressway data: (i) the first group of datasets contain 11 datasets, where the ratio of records with wrongly recognized plates ranges from 0% to 100%. Figure 11a shows a slight decrease in the accuracy of VeTrac as the ratio increases, while the accuracy of LPR drops steeply when the ratio is larger than 50%. (ii) the second group of datasets also contain 11 datasets, where the ratio of records with similar appearances ranges from 0% to 100%. Figure 11b shows a slight decrease in the accuracy of VeTrac as the ratio increases, while the accuracy of Re-ID drops steeply when the ratio is larger than 40%.

Urban. The urban dataset contains huge number of snapshots and their trajectories. Due to the difficulty in labeling the entire dataset of such scale, we focus on labeling two representative subsets: *public transits* and *private vehicles*. For *public transits*, we identify 125 trajectories on 3 bus routes by cross-referring the bus routes released by the bus operator and snapshots captured by the camera sensing network. For *private vehicles*, we randomly select 300 snapshots, manually label their identities by checking all snapshots within proximity and then identified a most likely trajectory for each snapshot with regard to the road network. We eventually identified 2801 snapshots that belong to the 300 trajectories.

Figure 12 visualizes the reconstructed trajectories over the *public transit* data of 3 bus routes. We identify 38 trajectories on bus route I, 61 trajectories on bus route II and 26 trajectories on bus route III. Figure 12a presents the three bus routes denoted in different colors. Figure 12b inspects one specific bus trajectory (which belongs to bus route I) and visualizes the reconstructed trajectories by VeTrac and alternative schemes. According to the figure, VeTrac successfully reconstructs the correct trajectory. The trajectory reconstructed by LPR (denoted in red) deviates from the ground truth in several parts of the trajectory due to the loss of snapshots that contain errors. The granularity of the reconstructed trajectories is thus limited. The trajectory reconstructed by Re-ID (denoted in black) is significantly deviated from the ground truth due to misclassified vehicle snapshots. Many false positives (which belong to other vehicles) lead to wrong detours and thus limit the precision of the reconstructed trajectory. The trajectory reconstructed by HRIS (denoted in green) is a very different path when compared with the ground truth. That is because most vehicles sharing the same OD choose different routes from the bus, which results in the mobility dependency different from the operation of the bus route. Figure 12c-f further depict the spatial distributions of all the 125 reconstructed bus trajectories by VeTrac and alternative schemes. We see that the majority of the trajectories reconstructed by VeTrac are correct,

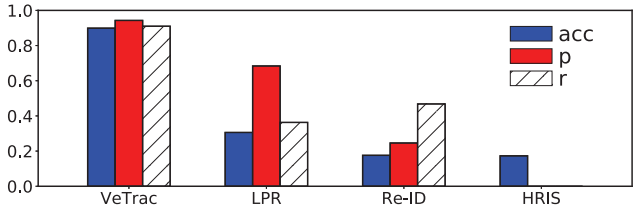
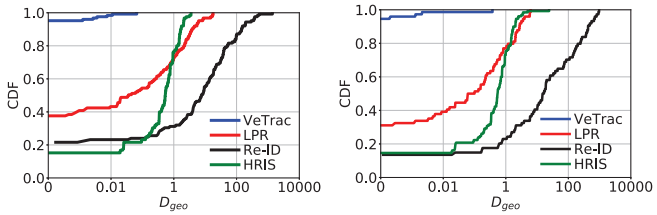
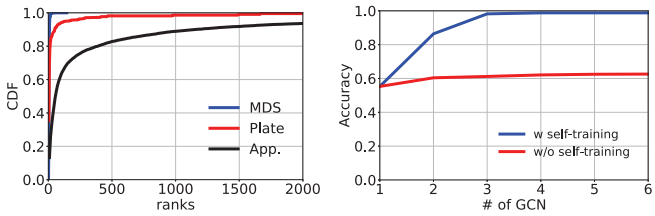


Figure 13: Urban: end-to-end performance



(a) Public transits **(b) Private vehicles**

Figure 14: Errors of the reconstructed trajectories



(a) MDS block **(b) Self-training**

Figure 15: Performance of VeTrac components

	Expressway	Urban
Graph Construction	1.3m	2.5m
GCN	2.1m	5.5m
Self-training*	13m (26s)	2.75h (5.1m)
Overall*	0.3h (3.8m)	2.9h (13.1m)

*the computation time in parentheses are estimated by projecting to 32 CPUs.

Table 3: Computation time of VeTrac

and they converge to the ground truth routes shown in Figure 12a. Substantive noises exist in trajectories by LPR and Re-ID due to the identity uncertainties as analyzed in §2.3. The trajectories by HRIS converge to routes of high inaccuracy.

We perform an overall evaluation on the urban dataset with the ground truths of both the 125 *public transit* trajectories and the 300 *private vehicle* trajectories. Figure 13 compares the average accuracy, precision and recall achieved by VeTrac and other schemes. Overall, VeTrac is able to correctly reconstruct 89% of the trajectories, which outperforms other alternatives by at least 59%. For precision and recall, VeTrac produces trajectories with 94.3% precision and 91.0% recall, which significantly outperforms other alternatives as well. The accuracy of LPR (30.6%) is limited due to the errors in LPR texts. Compared with the results of the expressway data (Figure 10), the precision of LPR drops to 68.5% because more vehicle snapshots have similar LPR texts when the total number of vehicles grow to such a scale. Re-ID scheme is also affected by such a scale effect as more snapshots are visually similar in vehicle appearance. The precision of Re-ID drops to 36.3% and the recall drops to 46.5%. HRIS gives an accuracy of 17.3%, which is comparable to Re-ID.

To quantify the errors of inaccurately reconstructed trajectories, we further derive the geographic differences D_{geo} between reconstructed trajectories and their corresponding ground truths. Figure 14 plots the cumulative distributions of D_{geo} for both datasets, where D_{geo} is presented in log scale. For both *public transits* and *private vehicles*, the majority ($\sim$ 90%) of the D_{geo} of VeTrac are 0, which means that most of the trajectories by VeTrac are identical to their ground truths. For the remaining 10%, the max D_{geo} of VeTrac is 0.3 for public transits and 0.65 for private vehicles, which is lower than about 40%, 70% and 85% of that of LPR, HRIS and Re-ID. On average, VeTrac's D_{geo} is 4.6x, 5.4x and 16x better than that of HRIS, LPR and Re-ID respectively. Small D_{geo} suggests that even when the reconstructed trajectories by VeTrac are not 100% correct, they are close to the ground truths and are representative for many analytical applications.

4.3 VeTrac by Components

We demonstrate the efficacy of key components in VeTrac by controlled experiments with the expressway data.

The MDS. We investigate the efficacy of the proposed MDS block by examining its performance in ranking snapshots belonging to the same identity. Given a query snapshot, robust similarity features should consistently rank the snapshots of the same identity to the top. We query the dataset by each of its snapshots and extract the ranks of snapshots of the same identities. Figure 15a presents the cumulative distribution of the ranks by using the license plate text, appearance features and MDS. As the figure suggests, 10% and 40% of the snapshots are poorly ranked by their license plate texts or appearance features due to the identity uncertainties. MDS greatly reduces the uncertainties by fusing similarity from multiple dimensions and thus 98% of MDS ranks are within the top 20.

Self-training GCN. Figure 15b demonstrates the efficacy of the proposed self-training. As indicated in the figure, without the self-training, the accuracy of graph convolution improves slowly. After reaching 62%, the accuracy stops improving even when more graph convolution layers are added. That is because adding more graph convolution layers may only help to address incomplete clusters. However, when there is adequate depth of graph convolutions, the imprecise clusters are the major problems. The proposed self-training process effectively handles imprecise clusters and helps to improve the accuracy to $\sim$ 98%.

Computation efficiency. Table 3 presents the computation time of VeTrac by its components. As shown, the snapshot construction and GCN are computation efficient because we ensure the sparsity in the generated snapshot graph. The most time-consuming component is the self-training process, which require fetching vertices with similar representation for a large number of vertices. Currently,

self-training process is implemented by sequentially computing pair-wise cosine distance with all other vertices. Such a process however can be easily paralleled by applying multiple CPUs and optimized by better neighborhood indexing (i.e., less amount of cosine distances needed to be evaluated for each vertex). We derive the projected computation time with 32 CPUs which can be reduced to 3.8 minutes and 13.1 minutes for the expressway and urban dataset respectively.

5 RELATED WORKS

Mobility based trajectory recovery. This problem has been studied previously in two settings, either data of low sampling rate or data with large localization errors. For low-sampled data, extra information include historical data[3], personal habits [36], road networks and reference trajectories [31, 40] have been used to infer uncertain pieces of original trajectory. For noisy data, the key rationale is to make use of location sequence information and environment constraints (rather than individual observations). As such, methods like Kalman filter[6] and Hidden Markov Model [22] has been exploited to perform map matching. However, existing solutions assume high level of confidence with the observed identities in mobility traces, and thus cannot be applied to address low-confidence observations like the vehicle snapshots that VeTrac utilizes. Their ability to distinguish objects with similar mobility patterns is also limited. Recently, new sensing technologies have been applied to sense trajectories, e.g., GPS [8], Wi-Fi signals [14, 23], inertial sensors [25, 28], visible lights [12], steerable cameras[11, 39], vehicular network [32, 33]. To the best of our knowledge, no existing approaches have been tested for recovering complete traffics at a scale close to VeTrac does.

Vision based vehicle Re-ID. Most of the visual Re-ID works focus on designing a general model solely based on visual characteristics. In view of this, recent advances fall into three main categories: (i) mining discriminative visual features, e.g., semantic features like the number of doors/seats [16], local features like windshield stickers, decoration marks and tissue boxes [9], and region features of predefined vehicle landmarks [35]. The extraction of those complex features requires high resolution input images and large amount of labeled data for training, which are not available in this study nor in many practical camera surveillance systems; (ii) augmenting training data, e.g., selecting hard cases into triplets to train a deep metric network [2], and generating features of other views with GAN-based architecture [42]. Training above networks however again requires a great number of labeled data. Selecting and labeling adequate number of hard cases or multi-view data to represent the diversity of real-world vehicle snapshots is challenging with large-scale data as studied in this paper; and (iii) re-ranking with other information, e.g., with a progressive pipeline [17], with spatio-temporal regularization [35], and with kinematic information [20]. VeTrac differs from this thread of works by specifically embedding system-wise correlation and trajectory-level mobility into the process of trajectory generation.

In this paper, we study the problem of trajectory reconstruction, which differs from the problem of Re-ID in two ways. First, Re-ID is purposed at ranking images of similar identities to the query image, but not properly assigning vehicle identities to connect them. Simple threshold-based solution is suboptimal because there is no guarantee on identity assignment and the best threshold is hard to determine. Second, Re-ID is highly query oriented while trajectory reconstruction is identity oriented. As a result, Re-ID techniques focus on pair-wise relationships but often neglect system-wise relationships, which however is critical to trajectory reconstruction. Therefore, directly applying any of existing Re-ID solutions cannot solve the problem of trajectory reconstruction.

6 CONCLUSIONS

In this paper, we present a vehicle trajectory reconstruction system that uses widely deployed traffic cameras as a sensing network and reconstructs the trajectories of general traffics. Large scale experiments with city-wide camera sensing data suggest high accuracy and efficiency of VeTrac. We believe VeTrac design provides insights that can be generalized to other mobility analysis of similar nature.

7 ACKNOWLEDGMENTS

This research is supported, in part, by NRF Singapore under its grant SDSC-2019-001, Alibaba Group through Alibaba Innovative Research (AIR) Program and Alibaba-NTU Singapore Joint Research Institute (JRI), and Singapore MOE Tier 1 grant RG18/20. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of funding agencies.

REFERENCES

[1] A navigation service by Gaode Maps. https://lbs.amap.com/.
[2] Y. Bai, Y. Lou, F. Gao, S. Wang, Y. Wu, and L.-Y. Duan. Group-sensitive triplet embedding for vehicle reidentification. *IEEE Transactions on Multimedia*, 20(9):2385–2399, 2018.
[3] P. Banerjee, S. Ranu, and S. Raghavan. Inferring uncertain trajectories from partial observations. In *IEEE ICDM*, pages 30–39, 2014.
[4] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *ACM SIGKDD*, pages 245–250, 2001.
[5] R. J. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *Springer PAKDD*, pages 160–172, 2013.
[6] M. E. El Najjar and P. Bonnifait. A road-matching method for precise vehicle localization using belief theory and kalman filtering. *Springer Autonomous Robots*, 19(2):173–191, 2005.
[7] Q. Gao, G. Trajcevski, F. Zhou, K. Zhang, T. Zhong, and F. Zhang. Trajectory-based social circle inference. In *ACM SIGSPATIAL*, pages 369–378, 2018.
[8] H. Hassanieh, F. Adib, D. Katabi, and P. Indyk. Faster gps via the sparse fourier transform. In *ACM MobiCom*, pages 353–364, 2012.
[9] B. He, J. Li, Y. Zhao, and Y. Tian. Part-regularized near-duplicate vehicle re-identification. In *IEEE CVPR*, pages 3997–4005, 2019.
[10] S. He, F. Bastani, S. Abbar, M. Alizadeh, H. Balakrishnan, S. Chawla, and S. Madden. Roadrunner: improving the precision of road network inference from gps trajectories. In *ACM SIGSPATIAL*, pages 3–12, 2018.
[11] S. Jain, V. Nguyen, M. Gruteser, and P. Bahl. Panoptes: Servicing multiple applications simultaneously using steerable cameras. In *ACM/IEEE IPSN*, pages 119–130, 2017.
[12] L. Li, P. Xie, and J. Wang. Rainbowlight: Towards low cost ambient light positioning with mobile phones. In *ACM MobiCom*, pages 445–457, 2018.
[13] M. Li, A. Ahmed, and A. J. Smola. Inferring movement trajectories from gps snippets. In *ACM WSDM*, pages 325–334, 2015.
[14] H. Liu, Y. Gan, J. Yang, S. Sidhom, Y. Wang, Y. Chen, and F. Ye. Push the limit of wifi based localization for smartphones. In *ACM MobiCom*, pages 305–316, 2012.
[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *Springer ECCV*, pages 21–37, 2016.
[16] X. Liu, W. Liu, T. Mei, and H. Ma. A deep learning-based approach to progressive vehicle re-identification for urban surveillance. In *Springer ECCV*, 2016.
[17] X. Liu, W. Liu, T. Mei, and H. Ma. Provid: Progressive and multimodal vehicle reidentification for large-scale urban surveillance. *IEEE Transactions on Multimedia*, 20(3):645–658, 2017.
[18] Z. Liu, Z. Li, M. Li, W. Xing, and D. Lu. Mining road network correlation for traffic estimation via compressive sensing. *IEEE TITS*, 17(7):1880–1893, 2016.

[19] Z. Liu, Z. Li, K. Wu, and M. Li. Urban traffic prediction from mobility data using deep learning. *IEEE Network*, 32(4):40–46, 2018.
[20] B. C. Matei, H. S. Sawhney, and S. Samarasekera. Vehicle tracking across nonoverlapping cameras using joint kinematic and appearance features. In *IEEE CVPR*, pages 3465–3472, 2011.
[21] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe. Parknet: drive-by sensing of road-side parking statistics. In *ACM MobiSys*, pages 123–136, 2010.
[22] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. In *ACM SIGSPATIAL*, pages 336–343, 2009.
[23] B. Qi, L. Kang, and S. Banerjee. A vehicle-based edge computing platform for transit and human mobility analytics. In *ACM/IEEE SEC*, pages 1–14, 2017.
[24] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *IOS Press IDA*, 11(5):561–580, 2007.
[25] S. Shen, M. Gowda, and R. Roy Choudhury. Closing the gaps in inertial motion tracking. In *ACM MobiCom*, pages 429–444, 2018.
[26] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE T PATTERN ANAL*, 39(11):2298–2304, 2016.
[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014.
[28] F. Tahmasbi, Y. Wang, Y. Chen, and M. Gruteser. Your phone tells us the truth: Driver identification using smartphone on one turn. In *ACM MobiCom*, pages 762–764, 2018.
[29] P. Tong, W. Du, M. Li, J. Huang, W. Wang, and Z. Qin. Last-mile school shuttle planning with crowdsensed student trajectories. *IEEE TITS*, 2019.
[30] G. Wang, X. Chen, F. Zhang, Y. Wang, and D. Zhang. Experience: Understanding long-term evolving patterns of shared electric vehicle networks. In *ACM MobiCom*, pages 1–12, 2019.
[31] J. Wang, N. Wu, X. Lu, X. Zhao, and K. Feng. Deep trajectory recovery with fine-grained calibration using kalman filter. *IEEE TKDE*, 2019.
[32] X. Wang, L. Fu, Y. Zhang, X. Gan, and X. Wang. Vdnet: an infrastructure-less uav-assisted sparse vanet system with vehicle location prediction. *Wiley WCMC*, 16(17):2991–3003, 2016.
[33] X. Wang, J. Zhang, X. Tian, X. Gan, Y. Guan, and X. Wang. Crowdsensing-based consensus incident report for road traffic acquisition. *IEEE TITS*, 19(8):2536–2547, 2017.
[34] Y. Wang, Y. Zheng, and Y. Xue. Travel time estimation of a path using sparse trajectories. In *ACM SIGKDD*, pages 25–34, 2014.
[35] Z. Wang, L. Tang, X. Liu, Z. Yao, S. Yi, J. Shao, J. Yan, S. Wang, H. Li, and X. Wang. Orientation invariant feature embedding and spatial temporal regularization for vehicle re-identification. In *IEEE ICCV*, pages 379–387, 2017.
[36] Y. Yang, X. Xie, Z. Fang, F. Zhang, Y. Wang, and D. Zhang. Vemo: Enabling transparent vehicular mobility modeling at individual levels with full penetration. In *ACM MobiCom*, pages 1–16, 2019.
[37] N. J. Yuan, Y. Zheng, X. Xie, Y. Wang, K. Zheng, and H. Xiong. Discovering urban functional zones using latent activity trajectories. *IEEE TKDE*, 2014.
[38] D. Zhang, Y. Li, F. Zhang, M. Lu, Y. Liu, and T. He. coride: carpool service with a win-win fare model for large-scale taxicab networks. In *ACM SenSys*, pages 1–14, 2013.
[39] T. Zhang, A. Chowdhery, P. Bahl, K. Jamieson, and S. Banerjee. The design and implementation of a wireless video surveillance system. In *ACM MobiCom*, pages 426–438, 2015.
[40] K. Zheng, Y. Zheng, X. Xie, and X. Zhou. Reducing uncertainty of low-sampling-rate trajectories. In *IEEE ICDE*, pages 1144–1155, 2012.
[41] P. Zhou, Y. Zheng, and M. Li. How long to wait? predicting bus arrival time with mobile phone based participatory sensing. In *ACM MobiSys*, pages 379–392, 2012.
[42] Y. Zhou and L. Shao. Aware attentive multi-view inference for vehicle re-identification. In *IEEE CVPR*, pages 6489–6498, 2018.