2024 25th IEEE International Conference on Mobile Data Management (MDM) | 979-8-3503-7455-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/MDM61037.2024.00041

Estimating mobility distributions from uncertain roadside sensor datasets

Chrysovalantis Anastasiou
Department of Computer Science
University of Southern California
Los Angeles, California, USA
canastas@usc.edu

John Krumm
Department of Computer Science
University of Southern California
Los Angeles, California, USA
jkrumm@usc.edu

Cyrus Shahabi
Department of Computer Science
University of Southern California
Los Angeles, California, USA
shahabi@usc.edu

***Abstract*—Understanding human mobility patterns is crucial for urban planning, resource allocation, and personalized recommendations. However, real-world trajectory data are rarely released publicly due to privacy concerns. At the same time, metropolitan cities are becoming equipped with various roadside sensors, such as CCTV cameras and RFIDs. Unlike trajectory data, these sensors do not uniquely identify and track vehicles, making extracting mobility patterns from their detections challenging. In this paper, we propose *VPE*, a framework that processes roadside sensor observations to estimate the probability that a vehicle visits a road segment at a certain time. At the core of *VPE*, we implement *LEM*, a novel mathematical model that calculates location transition probabilities taking into account the sensors' reliability. Lastly, we propose APD+, an algorithm that captures the uncertainty of movement between two endpoints. Our experiments show that the proposed methods achieve high accuracy while maintaining practical computation time.**

***Index Terms*—urban mobility, sensor data, uncertainty, visit probability, road networks**

I. Introduction

Understanding human movement's spatial and temporal patterns is crucial for various applications, from urban planning and resource allocation to targeted marketing and personalized recommendations. A natural way of discovering such patterns is by analyzing large datasets of real-world trajectories [1]. Unfortunately, such human mobility trajectories are rarely released publicly because of privacy concerns, and even when they are, they are sparsely sampled for various reasons (conserving battery, privacy, data compression, etc.). Much work has been done in the past to address the sparsity by reconstructing the trajectory [2]–[6]; however, two challenges remain. First, most solutions assume that a dense trajectory dataset is readily available for training the model. Second, mobility patterns tend to change over time, and hence, new trajectory datasets (even sparse ones) must be regularly obtained to keep mobility insights up-to-date.

At the same time, metropolitan cities are increasingly becoming equipped with a variety of sensors. Traditionally, these sensors are used to monitor traffic, enforce the rules of the road, charge drivers for using express lanes, weigh trucks in motion, and more. Some sensors can uniquely identify vehicles (e.g., express lane RFIDs and license plate readers). In contrast, others can only capture and extract certain features (e.g., visual features from CCTV camera feeds, number of axles from double inductive loop detectors [7]). Each sensor generates observations every time it detects an object of interest. In an ideal scenario where sensors can perfectly detect and identify vehicles, the observations can be used in place of GPS samples, and a trajectory recovery method can be applied to recover the most likely path from one sensor to the next. However, most existing methods will only provide one path and will not account for the uncertainty in which humans travel. As the distance (travel time) between two consecutive sensors increases, the number of possible ways to travel between the two locations increases exponentially. In addition, we must account for the unreliability of sensors, such as lighting conditions for CCTV cameras, and that in a typical metropolitan city, most, if not all, sensors cannot uniquely identify vehicles.

In this work, we propose a novel framework, named *VPE*, that processes uncertain data from various sensors to estimate mobility at the road network level probabilistically. This is designed to show what we can infer about a vehicle's location from static roadside sensors instead of sensors onboard the vehicle (such as GPS). The *VPE* framework, short for *Visit Probability Estimation*, estimates the probability that a vehicle visited (i.e., drove on) a road segment at a certain time by employing two key components. The first component, named *Location Estimation Model* (LEM), calculates a probability distribution for the vehicle's location at a particular step based on the location probability distribution at the previous step and the observations generated by the sensors. LEM considers each sensor's accuracy so that observations from untrustworthy sensors have less weight. The second component bridges the gap between two consecutive location probability distributions by estimating the probability with which each road segment was visited. This is achieved using a probabilistic variant of road network-based bridgelets [8]. Lastly, probabilities are aggregated across multiple time steps and vehicles to estimate overall mobility patterns. We experimentally show that *VPE* outperforms baselines in generating accurate visit probability distributions.

We summarize our contributions as follows:

- We introduce VPE, a novel framework for accurately estimating visit probabilities from uncertain sensor datasets.
- We propose a mathematical model, LEM, that leverages

sensor detections and trustworthiness to estimate the probability of the vehicle moving from one location to the next.
- We employ road network-based bridgelets to more accurately capture the uncertainty and possibilities of moving between locations.
- We conduct comprehensive experiments to show the effectiveness of our methods.

The remainder of this paper is organized as follows. Section II introduces our novel Visit Probability Estimation model in more detail. Section III explains how location probabilities are calculated from uncertain and untrustworthy sensor data. Section IV presents the results of our comprehensive experimental evaluation. Section V outlines the related work. Section VI concludes the paper.

II. Visit Probability Estimation

In this section, we introduce *VPE*, our novel visit probability estimation framework that leverages uncertain observations generated by a variety of sensors deployed along a road network.

A. Preliminary

For clarity, we introduce five definitions of fundamental elements of our description.

Definition 1 (Road Network)**.** A road network is encoded as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of nodes, with each node $v \in \mathcal{V}$ modeling an intersection, and $\mathcal{E}$ is a set of edges, with each edge $e \in \mathcal{E}$ modeling a road segment. We use the notation $e = (v, v')$ to denote that edge e is a road segment connecting the nodes v and v'.

Definition 2 (Location)**.** A location l is denoted by the tuple (e, r), where e is an edge and $r \in [0, 1]$ is a fraction indicating how far along the length of the edge the location is found. We use $\mathcal{L}$ to denote the set of all locations.

Definition 3 (Visits)**.** An edge e is considered visited by a vehicle if it fully or partially traverses the edge. A location $l = (e, r)$ is considered visited by a vehicle if it traverses at least an r fraction of the edge e, i.e., if the vehicle passes by the location.

Definition 4 (Sensor)**.** A sensor is defined by the tuple $s = (l, \epsilon_{tp}, \epsilon_{fp})$, where l is the location of the sensor, and ϵ_{tp} and ϵ_{fp} are the estimated true positive and false positive rates of the sensor, respectively. We denote the set of all sensors by $\mathcal{S}$. Additionally, we denote the set of locations that a sensor s can detect a vehicle using $\mathcal{L}_s$, where $|\mathcal{L}_s| > 0$.

Definition 5 (Sensor Observation)**.** An observation is generated by a sensor s whenever it detects a vehicle v in any location $l \in \mathcal{L}_s$. We denote the observation using the tuple $\rho = (s, \tau, p)$, where s is the sensor that generates the observation, τ is the timestamp of the detection, and p its detection probability. We denote the set of all sensor observations with $\mathcal{D}$.

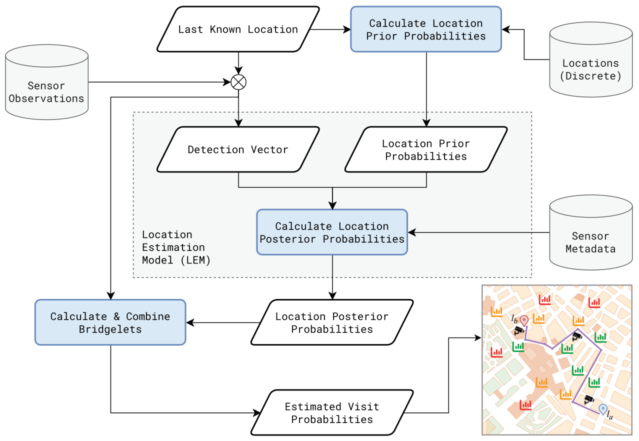

Last Known Location
Calculate Location Prior Probabilities
Locations (Discrete)
Sensor Observations
Detection Vector
Location Prior Probabilities
Location Estimation Model (LEM)
Calculate Location Posterior Probabilities
Sensor Metadata
Calculate & Combine Bridgelets
Location Posterior Probabilities
Estimated Visit Probabilities

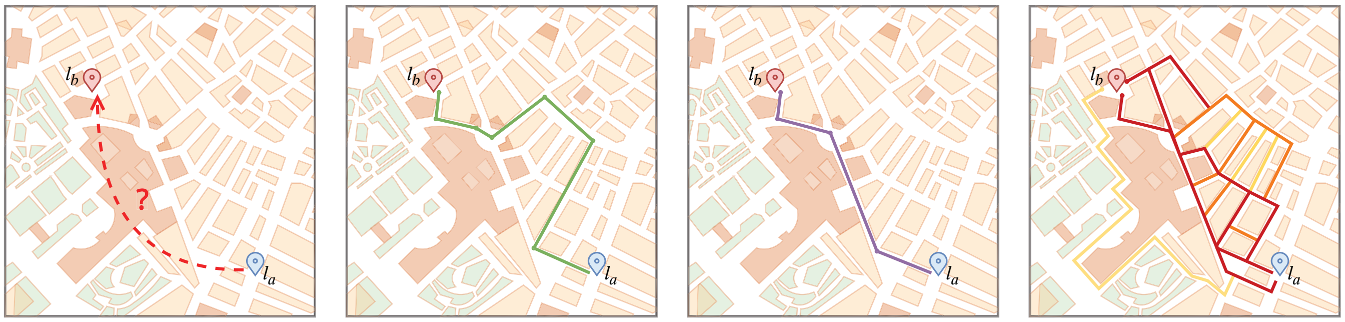
Fig. 1. The architecture and data flow of the VPE framework.

B. Framework Overview

We overview our framework here and give more details in subsequent sections. Our proposed framework processes sensor observations to estimate the probability that a given vehicle visited, i.e., traversed, a road segment. Figure 1 shows the components of *VPE* and how they interconnect. VPE takes as input a set of sensors $\mathcal{S}$, their observations $\mathcal{D}$, and a set of locations $\mathcal{L}$ at which the sensors detect vehicles. The output of VPE is a visit probability distribution over the road segments at each time step and for each vehicle. At first, sensor observations are processed to generate the detection vector for a given vehicle for the current time step, where each sensor accounts for one element of the detection vector. Then, the location of the vehicle at the previous time step (referred to as the last known location) is used to make an initial estimation of the probability that the vehicle is at any of the locations in the next time step (prior probabilities). Subsequently, the sensor's characteristics and trustworthiness are used to refine these probabilities and account for errors in detection, i.e., false positives and negatives. A visit probability distribution is calculated between the last known location and each next location that has non-zero probability. VPE produces its final output by combining these individual visit probability distributions. We explain how location uncertainty can be introduced into the framework in Section II-D by replacing the last known location with a probability distribution.

C. Estimating movement between locations

As a vehicle moves along the road network, it is observed whenever it passes by locations that at least one sensor observes. Particularly, for a vehicle v there is a time-ordered sequence of detections $\mathcal{D}_v$ that indicates where the vehicle was sensed over time. Note that, due to sensor imperfections, the vehicle may be sensed at multiple locations simultaneously. These locations, however, are discrete and scattered over the region of interest and, therefore, do not explicitly reveal any fine-grained mobility information, i.e., visits (Definition 3). Suppose that $\rho_i, \rho_{i+1} \in \mathcal{D}_v$ are two consecutive detections. A



(a) The two endpoints (b) Actual trajectory (c) Shortest path (d) Probability distribution

Fig. 2. Visit probability estimation between two endpoints. In (a) the two endpoints l_a and l_b are shown; in (b) the actual path (trajectory) is presented as a green line; in (c) the path is recovered using the fastest path method; and in (d) the probability distribution using road network-based bridgelets is plotted.

Algorithm 1 The VPE framework.

```
Output: Road Segment Visit Distribution $\Phi$
 1: procedure VPE
 2:     $\Phi \leftarrow \{\}$
 3:     $\tilde{l} \leftarrow hotEncode(l)$
 4:     for $t = 1 \ldots \tau$ do
 5:         for $l, p \in \tilde{l}$ do
 6:             if p == 0 then
 7:                 continue
 8:             end if
 9:             $\tilde{l}, \phi \leftarrow$ VPE-Step($v$, $\tilde{l}$, 1, $G$, $M$)
10:             $\Phi \leftarrow \Phi + p \times \phi$
11:         end for
12:     end for
13:     return $\Phi$
14: end procedure
15: procedure VPE-STEP($v$, $l$, $\tau$, $G$, $M$)
16:     $\bar{d} \leftarrow$ getDetectionVector($v$, $t$, $l$)
17:     $\tilde{p} \leftarrow$ calcLocationPriorProbs($l$, $G$)
18:     $\tilde{l} \leftarrow$ calcLocationPosteriorProbs($\bar{d}$, $\tilde{p}$, $M$)
19:     $\phi \leftarrow$ calcVisitProbs($l$, $\tilde{l}$, $G$)
20:     return $\tilde{l}, \phi$
21: end procedure
```

straightforward approach to transform these discrete location detections into fine-grained mobility information is assuming that the vehicle traveled on the shortest path. Intuitively, when the distance between the two locations is small, the shortest path is an obvious and, very likely, the only choice. However, as the distance and travel time ($\rho_{i+1}.\tau - \rho_i.\tau$) grows, the number of possible routes increases exponentially.

Consider the example shown in Figure 2. On the far left (Figure 2(a)), the two endpoint locations, l_a and l_b, are shown on a map. Next to it (Figure 2(b)), the actual path that the vehicle traveled on to get from l_a to l_b is shown. The shortest path between the two locations is drawn in the middle-right map (Figure 2(c)). Evidently, the shortest path method misses several segments of the path, leading to inaccurate insights. On the far right (Figure 2(d)), however, a probability distribution over the road network segments is computed, capturing the mobility uncertainty more accurately. We observe that even though the edges that fall on the shortest path still carry a lot of weight (dark red), other possible edges retain some probability depending on how likely they are to have been used (with orange and yellow indicating a higher or lower probability, respectively.)

Several recent works have delved into the problem of calculating such probability distributions, often referred to as location probability clouds, between two locations. The concept of bridglet probability clouds was introduced in [9]. The most recent work [8] proposes a time-dependent road network-based approach that discovers all possible paths between the two endpoints, weights them based on their travel time (with faster paths receiving higher weight), and assigns probabilities to each road segment by aggregating the weights of the paths that traverse it. Although this method has been shown to generate accurate results, it does not scale well to large temporal gaps due to the exponential number of distinct paths that can exist between two locations. Therefore, we propose and use a variant of this approach, named *APD+*, which builds on top of *APD** introduced in [8] and can scale well as the temporal gaps grow. For brevity, we will refer to the set of all feasible paths between two locations l_a and l_b as a bridgelet and denote it using $\beta_{a,b,\tau}$.

The first key difference in *APD+* is that, instead of computing *all* possible paths between two endpoints, it only finds the k most likely paths. Intuitively, when k is large enough, *APD** and *APD+* will produce the same set of paths. But an important observation is that the n, where $n << k$, fastest paths in this set, will receive most of the weight, leaving the rest of the paths with near-zero weights. It is important to note that weighting the paths by their travel time is natural and mostly represents human preferences in the real world [10]. Therefore, limiting the discovery to only the k most likely paths is essential to improve performance while marginally sacrificing accuracy. We empirically find that $k = 100$ is a good trade-off between the accuracy and computational performance of the algorithm.

The second key difference is that, unlike *APD**, *APD+* pre-processes the road network to precompute the bridgelets between certain locations. Particularly, APD+ employs the well-known ALT [11] algorithm to select several landmarks in the region of interest strategically. Then, the $1.5 * k$ fastest paths between each pair of landmarks are computed using the speed limit of each road segment. This set is referred to as the time-agnostic landmark bridgelet. The rationale behind using speed limit and computing more than n paths is simple. First, computing the set of paths for every time step becomes prohibitively expensive as the time step size decreases and/or the number of landmarks increases. Second, as the traffic dynamics change throughout the day, a path that was in the n fastest during freeflow (speed limit) may not be during rush hours and vice-versa. Precomputing more paths allows the algorithm to adaptively re-rank them and assign weights to each path by recalculating the travel time based on the time of day.

Subsequently, when *APD+* is employed to compute the bridgelet between two endpoints, the ALT algorithm is used along with the landmarks to recover the relevant precomputed bridgelets. Obviously, only the bridgelets between landmarks are precomputed and can be directly retrieved, whereas the bridgelets between a given nonlandmark endpoint and a landmark need to be computed on the fly. This means that, instead of having to compute a potentially large and expensive bridgelet between two endpoints, we need to compute several much smaller bridgelets; from the departing location to one or more nearby landmarks, and from one or more landmarks to the destination location. Of course, when the distance between the two endpoints is small, i.e., when the endpoints are closer to each other than to any landmark, the bridgelet is directly computed without using landmarks.

Ultimately, we are not interested in the bridgelet itself but in the probability that a road segment is visited. As in [8], we weight the discovered paths using an inverse travel time weighting (ITTW) function and calculate the probability of each path $\rho \in \beta_{a,b,\tau}$ as shown in Equation 1.

$$P(\pi = \rho) = \frac{w_\rho}{\sum_{\rho' \in \beta_{a,b,\tau}} w_{p'}} \tag{1}$$

$$w_\rho = \frac{1}{traveltime(\rho)^\lambda} \tag{2}$$

where, $traveltime(\cdot)$ is a function that returns the travel time of a path, and $\lambda \in \mathbb{R}^+$ is the power parameter. Then, the probability that an edge e is visited is calculated as the sum of the probabilities of each path that contains that edge. Equation 3 summarizes the calculation.

$$P(e|\beta_{a,b,\tau}) = \sum_{\rho_i \in \beta_{a,b,\tau}} \mathbb{I}_{e\in\rho_i} * P(\pi = \rho_i) \tag{3}$$

Here $\mathbf{I}_{e\in\rho_i}$ is equal to 1 iff the edge e is visited by path ρ_i, or 0 otherwise.

D. Estimating movement between location distributions

In the previous section, we make one strong assumption: that sensors are (near) perfect and that the vehicle is precisely sensed whenever it passes by a location that a sensor monitors. However, in reality, sensors are far from perfect, especially the ones that do not uniquely identify vehicles, and often lead to false positives (and false negatives). This introduces uncertainty about the true location of the vehicle, which accumulates over time.

To account for this uncertainty, we generalized the idea of bridgelets into *uncertain bridgelets* where the two endpoints are now location distributions rather than specific locations. We denote uncertain bridgelets using $\beta_{\overrightarrow{a},\overrightarrow{b},\tau}$, where $\overrightarrow{a}, \overrightarrow{b} \in [0,1]^{|\mathcal{L}|}$ are vectors encoding location distributions. When the location is certain, the vector one-hot encoded, i.e., the probability is 1.0 at the index of the location, and everything else is 0.0. On the other extreme, where the location uncertainty is maximized, the probability distribution will be uniform, i.e., every location will have a probability of $\frac{1}{|\mathcal{L}|}$. In either case, we treat them as probability distributions and propose a mathematical extension to *APD+* to deal with the uncertainty.

Suppose that the location distribution at the current time step is $\overrightarrow{a}$. For every location l with non-zero probability, i.e., $\overrightarrow{a}[l] > 0$, we employ a mathematical model, *VPE* (Section II), to estimate the probability that the vehicle travels to any particular location at the next step. The output of *VPE* is a location distribution $\overrightarrow{b}$ that considers the sensor detections and reliability. In a later section, we describe the details of *VPE*. Then, for every location l' with non-zero probability, i.e., $\overrightarrow{b}[l'] > 0$, we compute the bridgelet $\beta_{l,l'}$. A total of $\mathbb{I}_{\overrightarrow{a}[l]>0} \times \mathbb{I}_{\overrightarrow{b}[l']>0}$ bridgelets are computed, where $\mathbb{I}_x$ is a function that counts the number of times the condition x holds.

The road segment visit probability from an uncertain bridgelet is calculated as the sum of the probabilities of the individual bridgelets we compute in the previous step multiplied by the probability that the origin and destination locations are visited. Equation 4 summarizes this aggregation.

$$P(e|\beta_{\overrightarrow{a},\overrightarrow{b}}) = \sum_{l\in\mathcal{L}} \sum_{l'\in\mathcal{L}} \overrightarrow{a}[l] \times \overrightarrow{b}[l'] \times P(e|\beta_{l,l'}) \tag{4}$$

This is essentially an expected bridgelet, in the probabilistic sense, where the expectation is a probability-weighted sum of the constituent bridgelets.

One challenge is that the more time steps we aggregate across, the higher the likelihood of the probabilities becoming more uniformly distributed over a large number of locations, especially if there are many unreliable sensors. In other words, as the uncertainty accumulates over time and the set of locations the vehicle could have visited increases in size, the expected probability of a location goes closer and closer to zero. To address this, we clip every location distribution so that locations with very low probability are forced to zero. This allows us to discard locations that are very unlikely to be visited and prevent the probabilities from imploding. As a byproduct, clipping the location distribution also improves the computational performance of the algorithm since a smaller number of bridgelets need to be computed. We empirically find that a probability threshold $\theta = 10^{-4}$ strikes a good trade-off between precision, recall, and computation time.

III. Location Estimation Model

In this section, we introduce *LEM*, a novel mathematical model that estimates the location probability of a vehicle while taking into account the reliability and uncertainty of sensor data.

A. From uncertain data to detections

As previously discussed, a variety of sensors deployed along the road network can detect or even identify vehicles. For example, some sensors may be very specific, such as license plate readers, while others may be very generic, such as a detector that merely detects the presence of any vehicle. In between are detectors tuned to see vehicles of a certain color or with a certain number of axles (e.g., freight vehicle detectors). In addition, each detector has a given level of reliability or trustworthiness. Lastly, some sensors may be well-focused on a given road segment, while others may detect vehicles on a set of nearby segments. This diverse variety of sensors makes the task of estimating the location of a vehicle a challenge.

To address this challenge, we propose a mathematical model that accounts for the sensors' specificity, reliability, and spatial coverage. Affecting specificity is the mix of vehicles on the road. At one extreme, if there is only a single vehicle on the entire road network, then a simple vehicle detector is adequate to pinpoint that single vehicle. At the other extreme, if the vehicle is visually indistinguishable from multiple other vehicles currently on the road, then the target vehicle could only be reliably recognized by a very specific detector, such as a license plate reader. Our model accounts for the mix of vehicles on the road along with the detection features of the sensors. Affecting reliability is the potential of hardware (or software) failures and the ability of a sensor to successfully detect a vehicle without generating false positives or negatives. For example, a computer vision model that processes the stream of a CCTV camera may fail to detect a vehicle under certain conditions, e.g., low visibility at night or occlusions during rush hour. Lastly, spatial coverage directly depends on the type of sensor and its ability to sense vehicles in an entire range or simply at one distinct location. For example, double inductive loop detectors are installed in the road itself and detect vehicles that drive directly on top of them, whereas CCTV cameras have a field of view that can span both sides of a motorway.

The following model applies to a single, specific vehicle. We discretize 2D space on the map into the set of possible locations $\mathcal{L}$. The vehicle can only be at one of the locations $l \in \mathcal{L}$. In reality, the detectors are versatile, e.g., a CCTV camera can detect vehicles of any color; however, for this model for a single vehicle, we will assume that detectors are tuned to detect a specific vehicle of interest v. For instance,

TABLE I
Various example behaviors of detectors with $p_e(d_j|l_i)$

	l_1	l_2	l_3	l_4	l_5	$\dots$	$l_{\lvert\mathcal{L}\rvert}$
s_1	1	0	0	0	0	0	0
s_2	0	0.8	0	0	0	0	0
s_3	0	0.7	0.9	0.5	0	0	0
s_4	1	1	1	1	1	1	1
s_5	0	0	0	0	0	0	0
$\dots$							
$s_{\lvert\mathcal{S}\rvert}$							

if the detector recognizes license plates, we expect that it will trigger only if it sees the license plate of v. Likewise, for detectors that detect a vehicle's size, color, weight, etc., we expect that the detectors will trigger only if they see a vehicle that matches these features of v. For a different vehicle, we would use a different instance of the model, where that model's sensors are tuned to detect that vehicle. At any given time, a detector $s_j \in \mathcal{S}$ may be either triggering if it detects v, or else not triggering. If the vehicle is at location l_i, the probability of triggering detector s_j is $p_e(s_j|l_i)$. The subscript e indicates that this probability applies when the roads are otherwise empty of other vehicles that could confuse the detector. With this probability, we model the uncertainty of the detector for this vehicle. When the vehicle is at l_i, the detector's true positive rate is $p_e(s_j|l_i)$, and its false negative rate is $1 - p_e(s_j|l_i)$.

Modeling the detectors with $p_e(s_j|l_i)$ gives a rich representation of different detector behaviors. We can represent $p_e(s_j|l_i)$ as a table, with an example in Table I. In this example, detector s_1 (first row) is perfectly tuned to detect the vehicle if the vehicle is at location l_1 and, as such, $p_e(s_1|l_1) = 1$ and the detection probability is zero at all locations other than l_1. Detector s_2 works almost as well for location l_2, although it does not always detect the vehicle if it is there, with $p_e(s_2|l_2) = 0.8$. Detector s_3 detects the vehicle somewhat reliably if it is at locations l_2, l_3, or l_4, with $p_e(s_3|l_2) = 0.7$, $p_e(s_3|l_3) = 0.9$, $p_e(s_3|l_4) = 0.5$, and a detection probability of zero at other locations. The next detector, s_4, has a low spatial resolution because it reports detecting the vehicle no matter where the vehicle is located, with $p_e(s_4|l_i) = 1$ for $\forall l_i$. Conversely, detector s_5 does not detect the vehicle no matter where it is.

The empty road detection probability $p_e(s_j|l_i)$ models the case where the vehicle of interest is the only one on the roads, where the detectors would not be confused by other vehicles. Thus $p_e(s_j|l_i)$ is the basis for the true positive rate and the false negative rate, both of which cover the case where the vehicle is actually present. We must also model how the detectors behave when the vehicle in question is not present, but other vehicles may be in the detectors' fields of view. This will be the basis for the true negative rate and the false positive rate, both of which cover the case where the vehicle is not actually present. The detection probability when the vehicle of interest is not present is $p_b(s_j)$, where the "b" subscript stands for "background". This is the probability that the detector will trigger in the absence of the vehicle of interest but in the presence of any other vehicles. This can model the case where a detector may confuse another vehicle for the vehicle of interest, and it should ideally be zero. For instance, a weigh-in-motion sensor will trigger on any vehicle whose weight is near that of the vehicle of interest. Conversely, a license plate sensor should trigger for only the vehicle of interest, so $p_b(s_j)$ should be very small in this case. The background detection probability $p_b(s_j)$ depends on the number of vehicles that the detector may confuse with the vehicle of interest. We can estimate $p_b(s_j)$ from the population of vehicle types we expect to be on the road. The false positive rate for detector j is $p_b(s_j)$, and the true negative rate is $1 - p_b(s_j)$.

We combine $p_e(s_j|l_i)$ and $p_b(s_j)$ to get the overall probability $p(s_j|l_i)$ of triggering detector s_j when the vehicle of interest v is at l_i. This is the probability of detection of the vehicle of interest *or* a mistake triggering on a background vehicle, given that v is at l_i. Note that the probability of *not* triggering the detector is the product of the false negative rate $(1 - p_e(s_j|l_i))$ and the true negative rate $(1 - p_b(s_j))$.

$$p(s_j|l_i) = 1 - \Big(1 - p_e(s_j|l_i)\Big)\Big(1 - p_b(s_j)\Big) \tag{5}$$

B. From Detection to Location

From the detector results, our goal is to compute a distribution of the vehicle's location over the locations in the set $\mathcal{L}$. Each detector gives a binary result (vehicle either detected or not detected), and the set of results is gathered into a binary detection vector $\overline{d} \in \{0,1\}^{|\mathcal{S}|}$. From Bayes theorem, we have

$$p(l_i|\overline{d}) = \frac{p(\overline{d}|l_i)p(l_i)}{\sum_{j=1}^{|L} p(\overline{d}|l_j)p(l_j)} \tag{6}$$

Here $p(l_i|\overline{d})$ is the *a posteriori* probability over locations that we seek. The quantity $p(\overline{d}|l_i)$ is the likelihood of the detection vector $\overline{d}$. We assume the individual detector likelihoods, from Equation 5, are independent, giving a likelihood term of

$$p(\overline{d}|l_i) = \prod_{j=1}^{|D|} p(d_j|l_i) \tag{7}$$

The independence assumption sidesteps the problem of explicitly assessing the correlations between the detectors. Vehicle detectors normally operate independently of each other, although regional phenomena like fog or an electric power failure could affect the detectors as a group.

In the Bayes equation (Equation 6), $p(l_i)$ is the *a priori* location distribution of the vehicle of interest. A straightforward approach is to represent the *a priori* probabilities using a uniform distribution. However, this does not account for the temporal dependencies between locations. A more data-driven method is to weight locations using an inverse distance or inverse travel time function. Even though this approach captures the spatio-temporal dependencies more realistically, it still fails to account for the feasibility of transitioning from one particular location to another. As an example, suppose that the last known location of a vehicle is l and that five minutes later it is detected at locations l_1 and l_2. All previously discussed methods will assign some non-zero probability to both locations. Now suppose that the fastest path to l_2 is 10 minutes. This means that the vehicle could not have been detected at that location. Therefore, l_2 should be assigned a zero probability. For our formulation, we compute the prior starting with a binary reachability variable r_i for each location l_i, where $r_i = 1$ means that the vehicle could have reached l_i from its previous location, and $r_i = 0$ otherwise. Then $p(l_i) = b_i / \sum_{k=1}^{|L|} b_i$. The prior could also model the popularity of different locations and the probability of speeds necessary to reach l_i.

IV. Experiments

This section describes the results of experiments to demonstrate how our technique works under a variety of realistic conditions.

A. Experimental Setup

All experiments were performed on an Ubuntu server equipped with an Intel(R) Core(TM) i9-9980XE CPU at 3.00GHz and 128GB of RAM. The processor has 18 cores (36 threads) with private L1 (32KB for data and 32KB for instructions) and L2 (1MB) caches and a shared L3 (24.75MB) cache.

Datasets:

- **Road Network:** We obtain the road network for the metropolitan region of Los Angeles from OpenStreetMap. The road network graph contains $3,239,158$ nodes and $4,190,761$ edges.
- **Traffic Data:** We use the ADMS [12] system as the traffic data source and use the methods described in [13] to estimate the traffic at every road segment.
- **Trajectories:** We use a synthetic trajectory generator [14] to simulate 1000 realistic trajectories in a controlled environment where the traffic data is known.
- **Sensors:** We randomly distribute 300 sensors on the road network and replay the synthetic trajectories to generate sensor observations. We simulate various settings, from very reliable to very unreliable sensors.

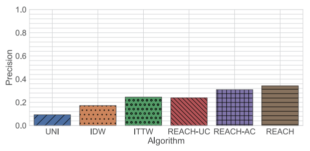
B. Quantitative Evaluation

We evaluate the performance of LEM using the *Precision* (Equation 8), *Recall* (Equation 9), and *F1* scores to measure how often it generates a non-zero probability for the locations the vehicle visited. Specifically, we use the sets $\tilde{\mathcal{V}}$ and $\mathcal{V}$ to denote the inferred and true location visits, respectively.

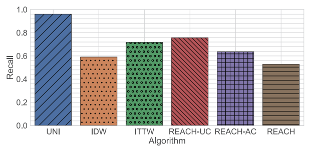
$$Precision = \frac{|\mathcal{V} \cap \tilde{\mathcal{V}}|}{|\tilde{\mathcal{V}}|} \tag{8}$$

$$Recall = \frac{|\mathcal{V} \cap \tilde{\mathcal{V}}|}{|\mathcal{V}|} \tag{9}$$

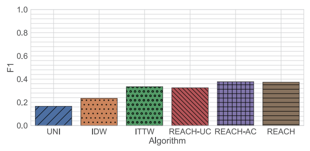
Additionally, we use Jensen–Shannon divergence as defined in Equation 10. This metric measures how close the estimated



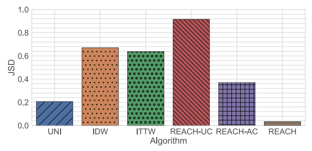
(a) Precision



(b) Recall

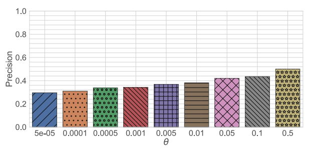


(c) F1 score

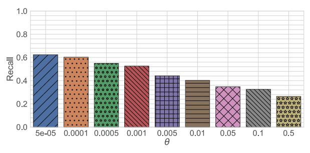


(d) JS-divergence

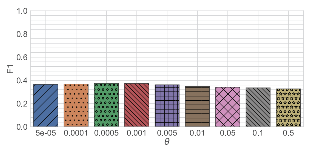
Fig. 3. Comparison of LEM against baselines using precision, recall, F1 score, and KL-divergence.



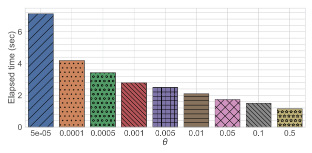
(a) Precision



(b) Recall



(c) F1 score



(d) Elapsed time

Fig. 4. Varying the probability clipping parameter θ.

location probability distribution is to the true locations. Here, the sample space is the set $\mathcal{L}$ of all locations, and the probability distributions are formed by mapping locations to their visit probability as generated by each algorithm. M is a mixture of the distributions P (estimated location probability distribution) and Q (true location, i.e., pseudo-probability distribution equivalent to one-hot encoded vector) and is defined as $\frac{1}{2}(P+Q)$.

$$JSD(P||Q) = \frac{1}{2}KL(P||M) + \frac{1}{2}KL(Q||M) \tag{10}$$

$$KL(P||Q) = \sum_{l \in \mathcal{L}} P(l) \log \frac{P(l)}{Q(l)} \tag{11}$$

Example: Suppose that there exist three possible locations l_1, l_2, and l_3 and the vehicle in question visits l_2 at the given time step t. We represent this distribution with the one-hot encoded vector $Q = \{0, 1, 0\}$. Now, suppose that the estimated distribution is $P = \{0.2, 0.8, 0.0\}$. Then, the divergence is equal to 0.936. Contrarily, if the estimated distribution is $P = \{0.8, 0.2, 0.0\}$, then the divergence is 0.641. This indicates that the former is closer to the true distribution.

1) Effect of prior probabilities: We vary the method used to generate the location prior probabilities and evaluate how the accuracy of LEM is affected. Specifically, we compare the following methods:

- **UNI**: Assigns the same probability to all locations in the detection vector (uniform).
- **IDW**: Uses inverse distance weighting to assign probabilities so that further away locations receive lower probability.
- **ITTW**: Similar to IDW but uses travel time instead of distance.
- **REACH-UC**: A variant of our approach that does not clip the probabilities.
- **REACH-AC**: A variant of our approach that only clips the final aggregated location probabilities using a threshold $\theta = 10^{-4}$.
- **REACH**: Our proposed reachability-based method using a clipping threshold $\theta = 10^{-4}$. The difference between this method and REACH-AC is that probabilities at intermediate time steps are also clipped.

Figure 3 shows the performance of all the algorithms. We observe that UNI exhibits the lowest precision and highest recall as expected because it gives all detections equal probability, even if it is not feasible for the vehicle to travel from its last known location to that of the detection. Similar observations can be made for IDW and ITTW though these methods are more accurate than UNI because they implicitly assume that the farther away or the longer the vehicle travels to reach the detection, the less likely it is to be a true positive. They both fail to take into account the temporal dependency between the current and detection time steps, i.e., even if a detection is far, it may still be feasible to reach there in time. On the other hand, the three reachability-based methods exhibit better performance, with REACH performing the best in terms of F1 score. Intuitively, this is attributed to the fact that detections that are not reachable are ignored. Finally, the distributions calculated by REACH exhibit the lowest JSD score. This means that the estimated distribution is very similar to the real distribution.

2) Effect of clipping threshold θ: We study the effect of the clipping parameter θ on LEM's accuracy and computational performance. Figure 4 summarizes the results. Overall, increasing θ does not seem to affect the accuracy significantly in terms of the F1 score. Precision increases as unlikely locations are ignored, but at the same time, recall decreases because there are cases where true locations may be assigned a low probability. Overall, this shows that, in most cases, LEM accurately assigns high probabilities to the true locations the vehicle visited. On the other hand, execution time decreases as

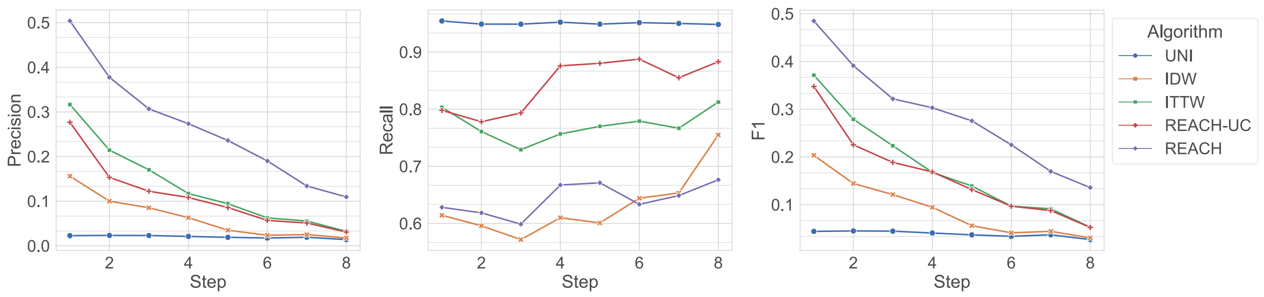


Fig. 5. The effect of uncertainty propagation on the accuracy of VPE at a depth of eight time steps. In this example, each time step is equivalent to three minutes.

θ increases. This is attributed to the fact that fewer possibilities need to be explored.

3) Effect of propagating uncertainty: Finally, we examine how the accuracy is affected as the uncertainty propagates from one time step to the next. Figure 5 plots precision, recall, and F1 scores up to eight time steps. Each time step is equivalent to three minutes. We observe that for all methods, precision drops as the uncertainty propagates through time steps. At the same time, recall goes up. The root cause of this is the explosion of possibilities. Suppose that at the first time step, we only have two probable locations. Then, at the next step, we have another two locations for each of those two locations, and so on and so forth. At any given time step, we keep track of a set of possibilities, whereas in reality, very few of them are valid. Mathematically, this translates to low precision and high recall. Evidently, this has a smaller impact on REACH, which outperforms all other methods.

C. *Visual Evaluation*

In Figure 6, we provide a visual example of a trajectory and the vehicle's estimated locations at four different time steps. In the example, we plot the sensors using grey dots and the real trajectory of the vehicle as a red line for reference. Note that *VPE* is unaware of the trajectory but only the sensors' observations. The only information provided to the algorithm for this example is the vehicle's initial location marked with a pin. We observe that at the first time step, *VPE* assigns a non-zero probability (color-coded circles) to a location that the vehicle did not visit. This is attributed to the fact that a sensor that observes that location is triggered at a time that is feasible to reach from the vehicle's current location. However, at subsequent time steps, the algorithm makes more accurate estimations. The reason behind this is that only a few detections are reachable by the vehicle.

V. Related Work

A. *Trajectory recovery*

A variety of methods have been proposed to recognize vehicle mobility from low-sampled trajectories (GPS, RFID [15], etc) and narrow down the true paths of the vehicles generating them. These methods include conventional approaches such as the shortest path algorithm and HMM that use heuristics to reduce the uncertainty between consecutive points. The authors in [16] propose a history-based route-inference system named HRIS which extracts mobility patterns from historical data and uses them to inform the recovery process. In [17] HMM is extended to leverage temporal information derived from historical traffic data and speed limits.

More recently, learning-based methods have been proposed. DHTR [18], MTrajRec [5], JCRNT [19], and RNTrajRec [4] define an ϵ-sampling strategy to generate a uniformly sampled time-ordered sequence of points. Then, sequence-to-sequence methods such as RNN, Transformers [20], and BERT [21] can be employed to reconstruct a high-resolution trajectory.

However, all these methods assume that a time-ordered sequence of certain locations exists beforehand and their task is to increase the resolution whereas, in our task, locations are not only uncertain but also not always attributed to the vehicle that generates them.

B. *Next location prediction*

The problem of location prediction has received significant attention in the last decade. Predicting the next location of a user necessitates modeling human mobility for the sequential prediction task. Markov Chain (MC) based methods, Matrix Factorization (MF) techniques, and Neural Network models (NN) are the schemes of choice for this objective. MC-based methods utilize a per-user transition matrix comprised of location-location transition probabilities computed from the historical record of check-ins [22]. The mth-order Markov chains emit the probability of the user visiting the next location based on the latest m visited locations. Recurrent Neural Networks (RNN) can model sequential data effectively, especially language sentences [23]. Recurrent nets have also been adapted for location sequences [24], [25]. However, RNNs assume that temporal dependency changes monotonically with the position in a sequence. This is often a poor assumption in sparse location data. As a result, the state-of-art employs the skip-gram model to learn the distributional context of users' behavior. Extensions incorporate temporal [26], textual, and other contextual features.

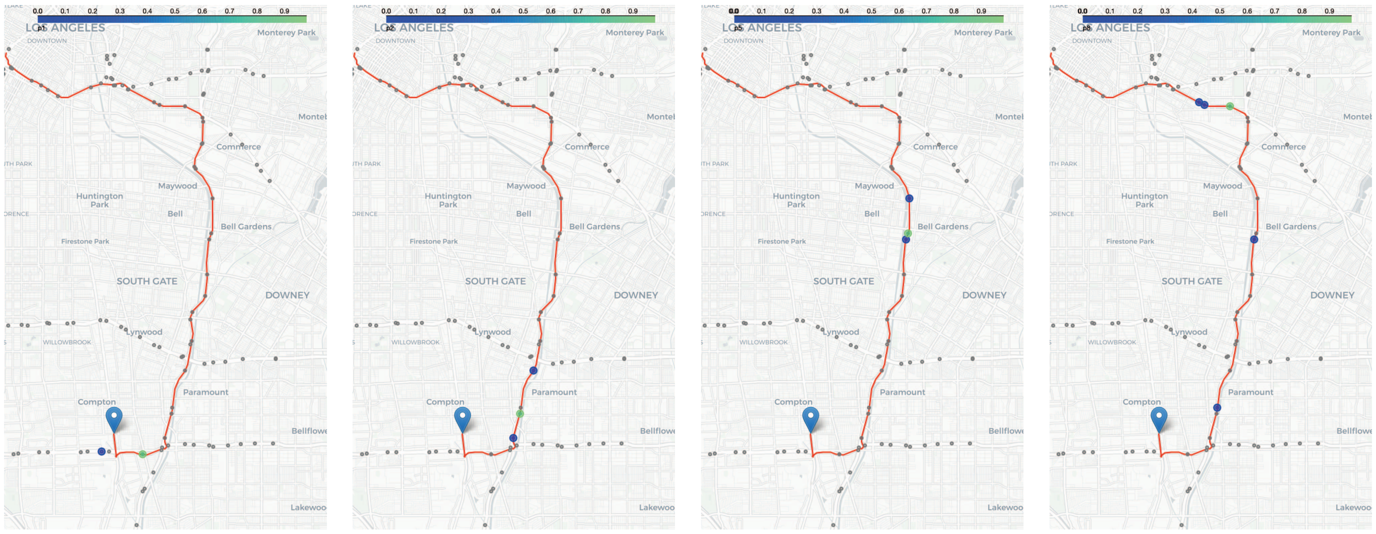


Fig. 6. Visual example of a trajectory (red line), the set of sensors (grey dots), and the estimated location distributions (color-coded circles) at four different time steps.

These methods attempt to predict the next location given the history and profile of a user. However, in our task, vehicles are not necessarily identified and therefore we cannot build a profile to leverage prior information. Additionally, we do not have a certain location history for each vehicle. Instead, we only have location distributions.

VI. Conclusion

In this paper, we introduced a novel framework named *VPE* that processes uncertain roadside sensor data to estimate mobility at the road network level probabilistically. *VPE* employs a mathematical model, *LEM*, to estimate the probability distribution of the next location while taking into account not only the reliability of sensors but also the feasibility of traveling from one particular location to another. These distributions are subsequently translated to road segment visits using APD+, which computes the most likely paths that could have been used to travel between locations. Our experiments show that *VPE* can accurately estimate the visit probability distribution of each road segment.

As future work, we are planning to develop methods and algorithms for approximate but fast bridgelet computation to further speed up the performance on large and complex road networks. Additionally, we will study how data sources beyond roadside sensors, e.g., moving sensors such as camera-equipped drones, can be integrated into the framework to improve the accuracy of estimations.

Acknowledgments

This work was supported by Caltrans Award, 2660-65A0674TO057. The opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the sponsors' views. The authors would like to thank Caltrans District 7 for providing CCTV data and the California Department of Transportation Traffic Operations Weigh-in-Motion for providing WIM data.

References

[1] M. Yue, Y. Li, H. Yang, R. Ahuja, Y. Chiang, and C. Shahabi, "DETECT: Deep trajectory clustering for mobility-behavior analysis," in *2019 IEEE International Conference on Big Data (Big Data)*. Los Alamitos, CA, USA: IEEE Computer Society, dec 2019, pp. 988–997. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/BigData47090.2019.9006561

[2] E. Chambers, B. T. Fasy, Y. Wang, and C. Wenk, "Map-matching using shortest paths," *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, vol. 6, no. 1, pp. 1–17, 2020.

[3] M. M. Elshrif, K. Isufaj, and M. F. Mokbel, "Network-less trajectory imputation," in *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, 2022, pp. 1–10.

[4] Y. Chen, H. Zhang, W. Sun, and B. Zheng, "Rntrajrec: Road network enhanced trajectory recovery with spatial-temporal transformer," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 829–842.

[5] H. Ren, S. Ruan, Y. Li, J. Bao, C. Meng, R. Li, and Y. Zheng, "Mtrajrec: Map-constrained trajectory recovery via seq2seq multi-task learning," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 1410–1419.

[6] T. Xia, Y. Qi, J. Feng, F. Xu, F. Sun, D. Guo, and Y. Li, "Attnmove: History enhanced trajectory recovery via attentional network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4494–4502.

[7] A. Tok, K. Hyun, S. Hernandez, K. Jeong, Y. Sun, C. Rindt, and S. G. Ritchie, "Truck activity monitoring system for freight transportation analysis," *Transportation Research Record*, vol. 2610, no. 1, pp. 97–107, 2017.

[8] C. Anastasiou, J. Krumm, and C. Shahabi, "Time-variant road network-based bridgelets," in *2023 24th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 2023, pp. 265–273.

[9] J. Krumm, "Maximum entropy bridgelets for trajectory completion," in *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, ser. SIGSPATIAL '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3557915.3561015

[10] T. Brinkhoff, "A framework for generating network-based moving objects," *GeoInformatica*, vol. 6, no. 2, 2002.

[11] A. V. Goldberg and C. Harrelson, "Computing the shortest path: A* search meets graph theory," in *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '05. USA: Society for Industrial and Applied Mathematics, 2005, p. 156–165.
[12] C. Anastasiou, J. Lin, C. He, Y.-Y. Chiang, and C. Shahabi, "ADMSv2: A modern architecture for transportation data management and analysis," in *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Advances on Resilient and Intelligent Cities*, ser. ARIC'19. New York, NY, USA: Association for Computing Machinery, 2019, p. 25–28. [Online]. Available: https://doi.org/10.1145/3356395.3365544
[13] C. Anastasiou, J. Zhao, S. H. Kim, and C. Shahabi, "Data-driven traffic index from sparse and incomplete data," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2022, pp. 2593–2598.
[14] C. Anastasiou, S. H. Kim, and C. Shahabi, "Generation of synthetic urban vehicle trajectories," in *2022 IEEE International Conference on Big Data (Big Data)*, 2022, pp. 359–366.
[15] W. Cheng, S. Wang, and X. Cheng, "Virtual track: Applications and challenges of the rfid system on roads," *IEEE Network*, vol. 28, no. 1, pp. 42–47, 2014.
[16] K. Zheng, Y. Zheng, X. Xie, and X. Zhou, "Reducing uncertainty of low-sampling-rate trajectories," in *2012 IEEE 28th international conference on data engineering*. IEEE, 2012, pp. 1144–1155.
[17] E. Ozdemir, A. E. Topcu, and M. K. Ozdemir, "A hybrid hmm model for travel path inference with sparse gps samples," *Transportation*, vol. 45, pp. 233–246, 2018.
[18] J. Wang, N. Wu, X. Lu, W. X. Zhao, and K. Feng, "Deep trajectory recovery with fine-grained calibration using kalman filter," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 3, pp. 921–934, 2019.
[19] Z. Mao, Z. Li, D. Li, L. Bai, and R. Zhao, "Jointly contrastive representation learning on road network and trajectory," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 1501–1510.
[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
[22] C. Zhang, K. Zhang, Q. Yuan, L. Zhang, T. Hanratty, and J. Han, "Gmove: Group-level mobility modeling using geo-tagged social media," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1305–1314.
[23] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *Interspeech*, vol. 2, no. 3. Makuhari, 2010, pp. 1045–1048.
[24] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.
[25] S. Zhao, T. Zhao, I. King, and M. R. Lyu, "Geo-teaser: Geo-temporal sequential embedding rank for point-of-interest recommendation," in *Proceedings of the 26th international conference on world wide web companion*, 2017, pp. 153–162.
[26] S. Feng, G. Cong, B. An, and Y. M. Chee, "Poi2vec: Geographical latent representation for predicting future visitors," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.