Deep Trajectory Recovery with Fine-Grained Calibration using Kalman Filter

Jingyuan Wang, Ning Wu, Xinxi Lu, Wayne Xin Zhao, *Member, IEEE*, and Kai Feng

Abstract—With the development of location-acquisition technologies, there are a huge number of mobile trajectories generated and accumulated in a variety of domains. However, due to the constraints of device and environment, many trajectories are recorded at low sampling rate, which increases the uncertainty between two consecutive sampled points in the trajectories. Our task is to recover a high-sampled trajectory based on the irregular low-sampled trajectory in free space, i.e., without road network information. There are two major problems with traditional solutions. First, many of these methods rely on heuristic search algorithms or simple probabilistic models. They cannot well capture complex sequential dependencies or global data correlations. Second, for reducing the predictive complexity of the unconstrained numerical coordinates, most of the previous studies have adopted a common preprocessing strategy by mapping the space into discrete units. As a side effect, using discrete units is likely to bring noise or inaccurate information. Hence, a principled post-calibration step is required to produce accurate results, which has been seldom studied by existing methods. To address the above difficulties, we propose a novel Deep Hybrid Trajectory Recovery model, named *DHTR*. Our recovery model extends the classic sequence-to-sequence generation framework by implementing a subsequence-to-sequence recovery model tailored for the current task, named subseq2seq. In order to effectively capture spatiotemporal correlations, we adopt both spatial and temporal attentions for enhancing the model performance. With the attention mechanisms, our model is able to characterize long-range correlations among trajectory points. Furthermore, we integrate the subseq2seq with a calibration component of Kalman filter (KF) for reducing the predictive uncertainty. At each timestep, the noisy predictions from the subseq2seq component will be fed into the KF component for calibration, and then the refined predictions will be forwarded to the subseq2seq component for the computation of the next timestep. Extensive results on real-world datasets have shown the superiority of the proposed model in both performance and interpretability.

Index Terms—Trajectory recovery, sequence to sequence model, spatiotemporal attention, kalman filter

1 Introduction

With the popularization of GPS-enabled mobile devices, a huge volume of trajectory data from users has become available in a variety of domains. These recorded trajectories provide an important kind of data signal to analyze, understand and predict mobile behaviors. Many studies have shown that trajectory data is useful to improve the user-centric applications, including POI recommendation [1], urban planning [2], and route optimization [3].

- *J. Wang is with the Beijing Advanced Innovation Center for Big Data and Brain Computing, School of Computer Science and Engineering, Beihang University, Beijing 100191, China. E-mail: jywang@buaa.edu.cn.*
- *N. Wu and K. Feng are with the MOE Engineering Research Center of Advanced Computer Application Technology, School of Computer Science and Engineering, Beihang University, Beijing 100191, China. E-mail: {WuNing, fengkai}@buaa.edu.cn.*
- *X. Lu is with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China. E-mail: luxinxi@buaa.edu.cn.*
- *W.X. Zhao is with the School of Information, Renmin University of China, Haidian 100872, China, and also with the Beijing Key Laboratory of Big Data Management and Analysis Methods, Beijing, China. E-mail: batmafly@gmail.com.*

Manuscript received 11 Nov. 2018; revised 7 July 2019; accepted 1 Sept. 2019. Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding author: Wayne Xin Zhao.)
Recommended for acceptance by W. Wang.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2019.2940950

However, due to the constraints of device and environment, many trajectories are recorded at a low sampling rate. As shown in previous studies [4], the low-sampled trajectories can not detail the actual route of objects, and increase the uncertainty between two consecutive sampled points in the trajectories. The high uncertainty significantly influences related research that uses trajectory data, including trajectory clustering [5], trajectory indexing [6], and trajectory classification [7]. Trajectory recovery also has an important impact on real-world applications, such as trip planning [8], movement behavior study [9], [10] and traffic condition prediction [11]. Hence, it is very important to develop effective algorithms to recover high quality trajectories based on raw low-sampled data.

Overall, the task of trajectory recovery has been studied in two different settings based on whether the map information, such as road networks, is available or not for use [9]. Under the first setting, the trajectory locations are usually mapped to road segments [4], [12], [13], [14] or Point-Of-Interests (POI) [15], [16], [17], [18], [19]. Then, the original trajectory recovery task will be simplified with such prior knowledge. While, under the second setting, the map information is not available as input, called *free space trajectory recovery* [9]. Comparing the two settings, the latter is more challenging to solve but also more common in practice. This work focuses on the second setting.

For solving the task of trajectory recovery, many efforts have been made in the literature. However, there are two potential problems with existing studies on the studied task. First, many of these methods rely on heuristic search algorithms or simple probabilistic models. They mainly model the adjacent transition patterns between locations, including depth-first search algorithm [4], [8], [20], absorbing Markov chain [21], and Gibbs sampling [12]. While, complex sequential dependencies or global data correlations can not be well captured by these methods. Second, for reducing the predictive complexity of the unconstrained numerical coordinates, most of the previous studies have adopted a common preprocessing strategy by first mapping the space into discrete units, e.g., cells [8], [20] or anchor points [21], [22], [23]. Then, their focus becomes how to develop effective recovery algorithms over the discrete units. As a side effect, using discrete units is likely to bring noise or inaccurate information. Hence, a post-calibration step is usually required to produce more accurate results. While, previous methods mainly adopt simple heuristic calibration methods, e.g., identifying frequent locations in a cell [8], [20] or simply using the centric coordinates of the cell [21], [22], [23]. They have neglected the importance of post-calibration in refining the coarse unit-level prediction results.

With the revival of neural networks, deep learning provides a promising computational framework for solving complicated tasks. Many studies try to utilize the excellent modeling capacity for better learning effective characteristics from trajectory data. Especially, sequential neural models, i.e., Recurrent Neural Networks (RNN), are widely used for modeling sequential trajectory data [24], [25], [26]. Although these studies have improved the capacity of modeling complex sequential transition patterns to some extent, they only focus on next-step or short-term location prediction in a local time window. While, our task requires that the approach should be able to effectively model and utilize the global, comprehensive information from the entire trajectory. In addition, these studies still directly produce the cell-level predictions, and do not incorporate a principled post-calibration component in their models for deriving more accurate estimations. Hence, it is difficult to directly apply existing neural network based trajectory models to the task of free space trajectory recovery.

To address the above difficulties, we propose a novel *Deep Hybrid Trajectory Recovery* model, named *DHTR*. Our recovery model substantially extends the classic sequence-to-sequence generation framework (i.e., *seq2seq*) by implementing a subsequence-to-sequence recovery model tailored for the current task, named *subseq2seq*. In order to effectively capture global spatiotemporal correlation, we adopt both spatial and temporal attentions for enhancing the model performance. With the attention mechanisms, our model is able to characterize long-range correlations among trajectory points. Furthermore, we integrate the subseq2seq component with a Kalman Filter (KF) to calibrate noisy cell prediction as accurate coordinates. KF is widely used to deal with a series of measurements observed over time, containing statistical noise and other inaccuracies. Different from conventional denoising applications that uses KF as an isolated postprocessing [27], we integrate the subseq2seq and KF in a joint deep hybrid model. Specifically, at each timestep, the noisy predictions from the subseq2seq component will be fed into the KF component for calibration, and then the refined predictions will be forwarded to the subseq2seq component for the computation of the next timestep. In this manner, our final model is endowed with the merits of both components, i.e., the capacities of modeling complex sequence data and reducing predictive noise.

Our contribution can be summarized as:

- We propose a novel deep hybrid model by integrating subseq2seq with KF for trajectory recovery. To our knowledge, it is the first time that deep learning is integrated with KF for the studied task. By using a hybrid of neural networks and KF, our model is endowed with the benefits of both components, i.e., the capacities of modeling complex sequence data and reducing predictive noise.
- As one of our major technical component, we extend the classic seq2seq framework as the subseq2seq for solving the current task. The subseq2seq approach utilizes the elaborately designed spatiotemporal attention mechanisms, which enhances the capacity of modeling complex data correlations.
- We construct the evaluation experiments using three real-world taxi trajectory datasets. Extensive results on the three datasets have shown the superiority of the proposed model in both effectiveness and interpretability.

2 Related Works

Our work is closely related to the studies on trajectory recovery and trajectory data mining. For trajectory recovery, we further divide the related works into two categories, using or not using road networks.

2.1 Trajectory Recovery with Road Networks

Given the information of road networks, previous studies usually consider trajectory reconstruction as a route inference problem of mobile objects, persons or vehicles, moving in a road network. The structure of road networks is used as the prior knowledge or constraint of the route inference algorithms. For example, Hsieh et al. propose to recommend time-sensitive trip routes, consisting of a sequence of locations associated with time stamps [16], [17]. Luo et al. study a new path finding query which finds the most frequent route during user specified time periods in large-scale historical trajectory data [14]. In [4], a history based route inference system (HRIS) has been proposed, which includes several novel algorithms to perform the inference effectively. Wu et al. propose a novel route recovery system in a fully probabilistic way which incorporates both temporal and spatial dynamics and achieve a state of art result [13]. Banerjee et al. employ Gibbs sampling by learning a Network Mobility Model (NMM) from a database of historical trajectories to infer the whole trajectories [12]. To fully utilize the road network information, some studies involve a preprocessing step called *map matching* [28], [29], [30], which aligns location coordinates onto the road segments. However, these studies highly rely on the structure of road networks, which cannot work well in free space.

2.2 Free Space Trajectory Recovery

Compared with the above works, free space trajectory recovery has no road network information as input. They usually try to identify the spatiotemporal patterns among adjacent location points, and reconstruct the trajectory using search based algorithms.

For example, Chen et al. propose a Maximum Probability Product algorithm to discover the most popular route (MPR) from a transfer network based on the popularity indicators in a breadth-first manner [21]. Wei, Liu et al. build a routable graph from uncertain trajectories, and then answers a user's online query (a sequence of point locations) by searching top-k routes on the graph [8], [20]. These algorithms mainly consider simple adjacent transitions and correlations in a small region. They cannot model long-range or global correlations among location points in a trajectory.

Especially, our work is also related to the works on trajectories similarity, since they usually involve trajectory recovery as an individual step before measuring the similarity. Su et al. propose an anchor-based calibration system that aligns trajectories to a set of anchor points [22]. Furthermore, Su et al. propose a spatial-only geometry-based calibration approach that considers the spatial relationship between anchor points and trajectories [23].

2.3 Deep Learning for Trajectory Data Modeling

Recent years have witnessed the progress of deep learning in modeling complex data relations or characteristics. In specific, Recurrent Neural Network together with its variant Long Short-Term Memory (LSTM) have been widely used for modeling trajectory data. Zheng et al. propose a hierarchical RNN to generate Long-term trajectories [24]. Wu et al. introduce a novel RNN model constrained by the road network to model trajectory [25]. Feng et al. design a multi-modal embedding recurrent neural network with historical attention to capture the complicated sequential transitions [26]. Chang et al. employ the RNN and GRU models to capture the sequential relatedness in mobile trajectories at different levels [31]. Liu et al. extend RNN and propose a novel method called Spatial Temporal Recurrent Neural Networks to predict the next location of a trajectory [32]. Al-Molegi et al. propose a novel model called Space Time Features-based Recurrent Neural Network (STF-RNN) for predicting people next movement based on mobility patterns obtained from GPS devices logs [33]. Most of these works mainly focus on modeling the sequence of location IDs rather than the numerical coordinate information.

To our knowledge, there are very few studies that apply deep learning for trajectory recovery. Our work enhances the capacity of modeling complex trajectory sequences with neural networks, and further calibrates the predictions using the classic Kalman filter for reducing data noise. With the integration of Kalman filter, our predictive uncertainty is highly controlled.

3 Preliminaries

In this section, we first introduce the notations throughout the manuscript, and then formally define our task.

Definition 1 Location. *A location or a location point is associated with a pair of coordinate values in the given geographical space, measured by its latitude and longitude $\langle x, y\rangle$.*

Definition 2 Region cell. *We assume that the entire geographical space is divided into a set of region cells (cell for short), denoted by $\mathcal{C}$. Each cell $c \in \mathcal{C}$ is a square space with the length of l, and corresponds to a centric location with the coordinates of $\langle x_c, y_c\rangle$.*

In practice, it is a common preprocessing technique to transform the continuous measurements into discrete cells as either main input [8], [21] or auxiliary data [22]. Using cells is able to reduce the complexity of directly modeling numerical coordinate sequence to some extent, since it is easier to perform the computation over a discrete set of cell IDs. We follow the procedure proposed in [20] for dividing free geographical space into disjoint cells.

Definition 3 Trajectory point. *A trajectory point a (or b) from an moving object is a timestamped location and modeled by a quadruple $\langle x, y, s, c\rangle$, where $a.x$ is the longitude, $a.y$ is the latitude, $a.s$ is the timestamp, and $a.c$ is the cell that point a is assigned to.*

Here, longitude and latitude are real numbers, a timestamp is accurate to seconds, and a cell is denoted by an integer ID.

Definition 4 Sampling interval. *A sampling interval ε is the time difference between two consecutive sampled points for a moving object, which usually depends on the device accurracy.*

Definition 5 ε-sampling trajectory. *A ε-sampling trajectory t (trajectory for short) is a time-ordered sequence of n uniformly sampled points from the same moving object using the sampling interval ε. Formally, we have $t = a_1^{(t)} \rightarrow a_2^{(t)} \rightarrow \ldots \rightarrow a_n^{(t)}$.*

Given a ε-sampling trajectory t, we have $a_{i+1}^{(t)}.s - a_i^{(t)}.\ s = \varepsilon$ for $1 \leq i \leq n-1$ and $a_n^{(t)}.s - a_1^{(t)}.s = (n-1)\varepsilon$. For simplicity, we omit the superscript of t in the notations of $a_i^{(t)}$ and use a_i in the following content.

Definition 6 ε-sampling sub-trajectory. *Given a ε-sampling trajectory t, a corresponding sub-trajectory $\tilde{t}$ (sub-trajectory for short) is a m-length subsequence of t. We have $\tilde{t} = b_1 \rightarrow b_2 \rightarrow \ldots \rightarrow b_m$, where $b_k = a_{j_k}$, $1 \leq j_1 \leq \cdots \leq j_m \leq n$ and $m < n$.*

Note that although the trajectory is uniformly sampled, the locations in a sub-trajectory may not be uniformly distributed in timestamps. With the above definitions, $\tilde{t}$ can be equally written as $a_{j_1} \rightarrow a_{j_2} \rightarrow \ldots \rightarrow a_{j_m}$. We use the different notations (i.e., a_i and b_k) for discriminating between a trajectory and its corresponding sub-trajectories. Here, $j_{(\cdot)}$ can be considered as a mapping for transforming the current indices of a sub-trajectory into the original indices of the complete trajectory. It is easy to see that a trajectory can correspond to multiple sub-trajectories by using different mappings. Since the original trajectory is generated by uniformly sampling at each time interval ε, we will have $b_{k+1}.s - b_k.s = (j_{k+1} - j_k)\varepsilon$.

Problem Statement. Given a ε-sampling trajectory dataset and a sub-trajectory $\tilde{t}$, we would like to reconstruct or recover the corresponding trajectory t. That is to say, for each missing trajectory point a_i (i.e., $a_i \in t$ but $a_i \notin \tilde{t}$), we will infer its corresponding longitude $a_i.x$ and latitude $a_i.y$ at time $a_i.s$. The sampling interval ε and time $a_i.s$ are assumed to be given as input. Such an assumption is rational since most of the measuring instruments sample the trajectory points regularly according to some fixed sampling interval. As aforementioned, the locations in a sub-trajectory may not be uniformly distributed in timestamps. Hence, our task is very challenging when road network is not available.

4 The Proposed Model

In this section, we present the proposed *Deep Hybrid Trajectory Recovery* model, named as *DHTR*.

4.1 Overview

We first present an overview of the proposed model. Our model contains three major parts. The first is an elaborately designed *subsequence-to-sequence (subseq2seq)* neural network model. The subseq2seq component is developed on the classic seq2seq model [34]. The second part is an attention mechanism, which is used to enable the subseq2seq to capture the complex spatiotemporal correlations. Our attention mechanism considers both spatial and temporal influence among locations in an entire trajectory and therefore named as *Spatiotemporal Attention*. For reducing the complexity of directly modeling numerical coordinate sequences, the subseq2seq component captures the sequential relatedness at the cell level. In order to refine the coarse cell-level predictions, we enhance the subseq2seq model (See Section 4.4.1) with a novel post-calibration component based on Kalman filter in the third part. Instead of using a pipeline post-processing approach, we integrate the subseq2seq component and the KF component in a joint deep hybrid model, which combines the merits of both components.

We detail DHTR in an asymptotical way. In Section 4.2, we introduce the subseq2seq model for trajectory recovery. In Section 4.3, we incorporate the Spatiotemporal Attention into subseq2seq. Then, Section 4.4.1 provides the enhancement of the proposed model with the integration of Kalman filter.

4.2 A Subseq2Seq Model for Trajectory Recovery

Instead of directly predicting the numerical coordinate values, we first infer the corresponding cell of a missing trajectory point. In this manner, a sequence of trajectory points can be considered as a sequence of cell IDs. As shown in [24], it is more reliable and easy to model cell ID sequence than the original numerical sequence. Once the cell of a trajectory point can be inferred, we will use the the centric coordinate of the predicted cell to recover the missing location. In this way, our major task is how to recover the corresponding cell sequence using partial observations.

4.2.1 Motivation

In the standard seq2seq model [34] for sequence generation, it consists of two major parts, namely encoder and decoder. The encoder is to map the input sequence to a fixed-sized vector using one RNN, and then the decoder is to generate the target sequence based on the vector from the encoder with another RNN. For capturing long range temporal dependencies, improved variants such as the Long Short-Term Memory have been widely used [35]. In the decoding procedure, the decoder generates the symbol conditioned on the representation of the input sequence and the previous segment of the output sequence using the softmax function.

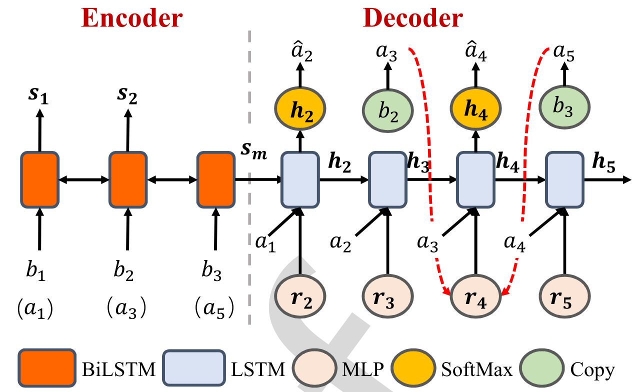


Fig. 1. The subseq2seq model. The encoder uses a BiLSTM to encode a subsequence as vector representation s_m. The decoder uses a LSTM which takes as input a region constraint vector r_i and the previous location. As an example, a missing location a_4 is bounded by the space spanned by a_3 and a_5, which is modeled as an embedding vector r_4.

Our main idea is to cast the recovery task into a sequence-to-sequence task, where the input sequence is the sub-trajectory and the output sequence is the reconstructed complete trajectory. Different from standard seq2seq tasks, our input is highly related to the corresponding output. The input sub-trajectory is a subsequence of the output trajectory. Hence, we name the approach *subsequence to sequence model*, abbreviated as *subseq2seq*.

The subseq2seq model consists of two parts: an encoder and a decoder, which is same as the standard seq2seq model. In the subseq2seq, the encoder inputs a sub-trajectory $\tilde{t}$ and the decoder predicts the corresponding complete trajectory t. The structure of the subseq2seq model is illustrated in Fig. 1. Next, we describe the two parts of the subseq2seq model in detail.

4.2.2 The Encoder for Modeling Observed Sub-Trajectories

In this part, our input (i.e., the observed cell sequence) will be encoded into a fixed-length vector. We adopt the Bidirectional Long Short Term Memory (BiLSTM) [36] as the encoder. The major benefit of BiLSTM is that it can capture both forward and backward temporal dependencies, while unidirectional RNN models can only capture the forward temporal dependencies. Such a benefit is specially important to our task, since a missing trajectory point will be closely related to both preceding and following trajectory points.

Given an input sequence $\tilde{t} = b_1 \rightarrow b_2 \rightarrow \ldots \rightarrow b_m$, the forward LSTM reads the input embedding of cells as it is ordered and calculates a sequence of forward hidden states $(\overrightarrow{s_1}, \ldots, \overrightarrow{s_m})$, while the backward LSTM reads the input embedding in the reverse order and calculates a sequence of backward hidden states $(\overleftarrow{s_1}, \ldots, \overleftarrow{s_m})$. The output at timestep j is the composition of $\overrightarrow{s_j}$ and $\overleftarrow{s_j}$. We implement the composition as the vectorized sum, and have the final representation $s_j = \overrightarrow{s_j} + \overleftarrow{s_j}$ for the jth state of the input sequence.

4.2.3 The Decoder for Reconstructing Missing Trajectories

Compared with the standard seq2seq application, our task has two unique characteristics, and we need to make suitable adaptations for the decoder according to our task.

First, the input is a subsequence of the output sequence in our task. In our model, for a known point from the input sequence, we apply the similar idea of "copy mechanism" from the NLP field [37] to directly generate the copy at the corresponding output slot. As shown in Fig. 1, in the proposed model, the copy mechanism is formulated as follows

$$a_i = \begin{cases} \hat{a}_i, & j_k < i < j_{k+1} \text{ (an unobserved point);} \\ b_k, & i = j_k \text{ (an observed point);} \end{cases} \tag{1}$$

where $\hat{a}_i$ denotes the prediction result using the decoder. In a word, we only predict the unobserved point, while the observed point is simply copied from the sub-trajectory.

Second, as previous studies have shown [8], the trajectory recovery task can be effectively solved by searching related several local trajectory points. More specially, a single trajectory point is usually bounded in a local region. In Fig. 1, we present an illustrative example with such region constraints. An observed sub-trajectory consists of three points: a_1, a_3, and a_5. The missing point a_4 is likely to fall in the region spanned by its preceding point a_3 and successive point a_5. Hence, incorporating region constraint is important to trajectory reconstruction. Instead of using hard rules, we propose to use hidden representations for modeling such region constraints. Especially, we use an embedding vector $\boldsymbol{r}_i$ to denote the constraint information for the ith trajectory point a_i, defined as

$$\boldsymbol{r}_i = \mathrm{DNN}(a_{j_k}, a_{j_{k+1}}). \tag{2}$$

where $\mathrm{DNN}(\cdot)$ is a function consisting of an look-up layer and a Multi-Layer Perceptron (MLP), a_{j_k} and $a_{j_{k+1}}$ are the observed precursor and successor for a_i, and $j_k < i < j_{k+1}$. In this way, the prediction of a point can utilize the information from its observed precursor and successor in a trajectory.

To this end, for inferring a missing point, our decoder derives the hidden state $\boldsymbol{h_i}$ as

$$\boldsymbol{h_i} = \mathrm{LSTM}(a_{i-1}, \boldsymbol{r}_i, \boldsymbol{h}_{i-1}, \boldsymbol{s}_m). \tag{3}$$

where $\boldsymbol{r}_i$ is the region constraint vector defined in Eq. (2) and $\boldsymbol{s}_m$ is the output state derived from the encoder. Once we obtain the hidden state $\boldsymbol{h}_i$ from the decoder, we further apply the softmax function to generate the corresponding cell of the missing trajectory point conditioned on the probability of $\Pr(c|\boldsymbol{h}_i)$ as

$$\Pr(c|\boldsymbol{h}_i) = \frac{\exp(\boldsymbol{h}_i{}^{\top} \cdot \boldsymbol{w}_c)}{\sum_{c' \in \mathcal{C}} \exp(\boldsymbol{h}_i{}^{\top} \cdot \boldsymbol{w}_{c'})}, \tag{4}$$

where $\boldsymbol{w}_c$ is the cth column vector from a trainable parameter matrix $\boldsymbol{W}^C$.

4.2.4 Applying the Model for Trajectory Recovery

Given a training dataset $\mathcal{D}$ consisting of trajectory and sub-trajectory pairs, we define the following objection function

$$\mathcal{L}_1 = \sum_{\langle t, \tilde{t} \rangle \in \mathcal{D}} -\log \Pr(t|\tilde{t}), \tag{5}$$

where $\Pr(t|\tilde{t})$ is computed using the softmax following the original seq2seq model using Eq. (4).

For applying the model to our task, at each timestep i, we first infer the corresponding cell of a missing trajectory point, namely $b_i.c$. Then we use the centric coordinate of the cell $b_i.c$ as the final predictions. In practice, for accuracy, we usually set a small cell length, e.g., $100 \sim 200$ meters. The complexity for the subseq2seq model increases with the decreasing of the cell length, since there will be more cells for predictions. Hence, we need to make a trade-off between the above two aspects for setting the cell length. We will discuss the effect of the cell length on the model performance in Section 5.3.3.

4.3 Incorporating the Spatiotemporal Attention

In the aforementioned subseq2seq model, we only consider local region constraint, while long-range or global correlations can not be modeled. Such correlations mainly reflect the spatiotemporal influence among trajectory points [32], which is more important to consider in free space without using road networks. Next, we adopt the *Spatiotemporal Attention* mechanism to capture the spatiotemporal influence among trajectory points.

4.3.1 A Standard Attention Mechanism

We first present a standard attention mechanism [34] for the general seq2seq model by rewriting Eq. (3) as

$$\boldsymbol{h}_i = \mathrm{LSTM}(a_{i-1}, \boldsymbol{r}_i, \boldsymbol{h}_{i-1}, \boldsymbol{s}_m, \boldsymbol{e}_i), \tag{6}$$

where the context vector $\boldsymbol{e}_i$ is computed by a weighted sum of these hidden states $\boldsymbol{s}$ from the encoder

$$\boldsymbol{e}_i = \sum_{k=1}^{m} \alpha_{i,k} \boldsymbol{s}_k. \tag{7}$$

An important part is how to compute the attention coefficients $\{\alpha_{i,j}\}$. Following [38], we can apply the softmax function to derive $\{\alpha_{i,j}\}$ as

$$\alpha_{i,k} = \frac{\exp(u_{i,k})}{\sum_{k'=1}^{m} \exp(u_{i,k'})}, \tag{8}$$

$$u_{i,k} = \boldsymbol{v}^{\top} \cdot \tanh(\boldsymbol{W}^H \boldsymbol{h}_i + \boldsymbol{W}^S \boldsymbol{s}_k), \tag{9}$$

where v, $\boldsymbol{W}^H$ and $\boldsymbol{W}^S$ are the parameter vector or matrices to learn. This attention mechanism mainly captures the general correlations among hidden states in the sequence. While, in our task, we need to explicitly model spatiotemporal influence with the attention mechanism.

4.3.2 Modeling the Spatiotemporal Influence

For modeling the temporal influence, given a target point, we assume that temporally adjacent points have a larger

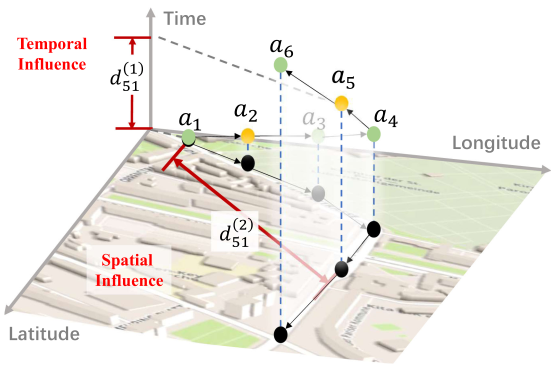


Fig. 2. The proposed spatiotemporal attention mechanism. Here, we are given a sequence of seven trajectory points from a_1 to a_6, where a_1, a_3, a_4 and a_6 (b_2 and b_5) are observed and the rest are for prediction. For region constraints, a_5 is only bounded by the pair of a_4 and a_6. We also plot the computation for both the temporal and spatial influence of a_1 to a_5. Utilizing the attention mechanism makes it feasible to explore long-range correlations or dependencies.

influence than temporally distant points in a trajectory. In other words, the influence decreases with the increasing of the temporal distance. Inspired by the method for modeling positional information from the NLP field [39], [40], [41], the temporal distance between the ith predicted point and kth observed point is defined as

$$d_{i,k}^{(1)} = |j_k - i|. \tag{10}$$

We have presented an illustrative example in Fig. 2. Here $d_{1,5}^{(1)} = 4$ since there are four sample intervals between the two trajectory points.

Similarly, the spatial influence can be computed with the euclidean distance of locations as

$$d_{i,k}^{(2)} = \sqrt{(a_{j_k}.x - b_i.x)^2 + (a_{j_k}.y - b_i.y)^2}. \tag{11}$$

However, during the inferring process, we do not have the coordinate information of $b_i.x$ and $b_i.y$, which are our goal to learn. Hence, we use the (inferred) coordinate information of the previous point to approximate the computation of $d_{i,k}^{(2)}$. In the Eq. (11), $d_{i,k}^{(2)}$ is in a real number. We further discretize the distance into integers via dividing $d_{i,k}^{(2)}$ by the cell length.

For a given trajectory, the scales of both $d_{i,k}^{(1)}$ and $d_{i,k}^{(2)}$ are usually bounded in a limited small range. So we propose to associate each discretized value for each kind of distance with an embedding vector. Then, the spatial-temporal influence is finally modeled as

$$u_{i,k} = \boldsymbol{v}^\top \tanh\left(\boldsymbol{W}^H \boldsymbol{h}_i + \boldsymbol{W}^S \boldsymbol{s}_k + \boldsymbol{W}^P \boldsymbol{p}_{d_{i,k}^{(1)}} + \boldsymbol{W}^Q \boldsymbol{q}_{d_{i,k}^{(2)}}\right), \tag{12}$$

where $\{\boldsymbol{p}\}$ and $\{\boldsymbol{q}\}$ are the embedding parameters corresponding to the temporal and spatial influences respectively, which are indexed by the discretized distance values of $d_{i,k}^{(1)}$ and $d_{i,k}^{(2)}$. The matrices $\boldsymbol{W}^P$ and $\boldsymbol{W}^Q$ are the parameters to learn.

4.4 Incorporating Kalman Filter

Above, we first apply the subseq2seq model to characterize the cell sequence for a trajectory, and then the centric coordinate of the predicted cell is treated as the final prediction. The approach has two potential shortcomings. First, the prediction model is likely to be affected by noise, e.g., the instrumental errors. Second, the final estimations are coarse since we use the corresponding cell coordinate as a surrogate.

To address these issues, we propose to integrate the above neural network model with Kalman filter. Kalman Filter [42] is particularly useful in dealing with varying temporal information. Especially, several studies have applied the KF model to calibrate noisy estimates in object tracking [43]. Compared with sequence neural networks, the standard KF is a linear system model, which can not capture long-range temporal dependencies. To develop our approach, our idea is to combine the benefits of sequence neural networks and KF with a hybrid model.

4.4.1 The General Description of Kalman Filter

Generally speaking, Kalman Filters (KFs) are optimal state estimators under the assumptions of linearity and Gaussian noise. In the KF model, we use a state vector $\boldsymbol{g}_i$, which could consists of the location and/or speed, to denote the state of a mobile object at the time i. The object linearly updates the state $\boldsymbol{g}_i$ with a Gaussian noise $\boldsymbol{e}_g$ as

$$\boldsymbol{g}_i = \boldsymbol{\Phi} \boldsymbol{g}_{i-1} + \boldsymbol{e}_g, \qquad \boldsymbol{e}_g \sim \mathcal{N}(0, \boldsymbol{M}), \tag{13}$$

where $\boldsymbol{M}$ is the covariance of $\boldsymbol{e}_g$, and $\boldsymbol{\Phi}$ is a state update matrix. In the KF model, the real value $\boldsymbol{g}_i$ can be measured by a measurement vector $\boldsymbol{z}_i$ as

$$\boldsymbol{z}_i = \boldsymbol{\Psi} \boldsymbol{g}_i + \boldsymbol{e}_z, \qquad \boldsymbol{e}_z \sim \mathcal{N}(0, \boldsymbol{N}), \tag{14}$$

where $\boldsymbol{\Psi}$ is measurement matrix, $\boldsymbol{e}_z$ is a Gaussian measurement noise and $\boldsymbol{N}$ is the covariance of $\boldsymbol{e}_z$. In the KF model, the measurement vector $\boldsymbol{z}_i$ is observable, the real state $\boldsymbol{g}_i$ is the unknown variable to be estimated. The matrices $\boldsymbol{\Phi}$, $\boldsymbol{\Psi}$, $\boldsymbol{M}$, and $\boldsymbol{N}$ in Eqs. (13) and (14) are known as a priori.

The KF model uses two procedures, *Prediction* and *Update*, to iteratively estimate the true value of $\boldsymbol{g}$ and calculate a covariance matrix, denoted as $\boldsymbol{H}$, to express the uncertainty of $\boldsymbol{g}$.

Prediction. In the prediction procedure, the KF model uses following equations to predict the state $\boldsymbol{g}$ and the covariance matrix $\boldsymbol{H}$ at the timestep i

$$\boldsymbol{g}_{i|i-1} = \boldsymbol{\Phi} \boldsymbol{g}_{i-1|i-1}, \tag{15}$$

$$\boldsymbol{H}_{i|i-1} = \boldsymbol{\Phi} \boldsymbol{H}_{i-1|i-1} \boldsymbol{\Phi}^\top + \boldsymbol{M}, \tag{16}$$

where $\boldsymbol{g}_{i-1|i-1}$ and $\boldsymbol{H}_{i-1|i-1}$ with the subscript "$i-1|i-1$" denotes the variables generated by the update procedure at the timestep $i-1$, while $\boldsymbol{g}_{i|i-1}$ and $\boldsymbol{H}_{i|i-1}$ with the subscript "$i|i-1$" denotes the predicted state and covariance.

Update. In the update procedure, the KF model use the observable measurement vector $\boldsymbol{z}_i$ to update/collate the predicted $\boldsymbol{g}_{i|i-1}$ and $\boldsymbol{H}_{i|i-1}$ as

$$\boldsymbol{g}_{i|i} = \boldsymbol{g}_{i|i-1} + \boldsymbol{K}_i(\boldsymbol{z}_i - \boldsymbol{\Psi} \boldsymbol{g}_{i|i-1}), \tag{17}$$

$$H_{i|i} = H_{i|i-1} - K_i \Psi H_{i|i-1}, \tag{18}$$

where K_i is named as the optimal Kalman gain, which combines the predicted state and measured state as the updated state. The matrix K_i is calculated as

$$K_i = H_{i|i-1}\Psi^\top (\Psi H_{i|i-1}\Psi^\top + N_i)^{-1}. \tag{19}$$

We can see K_i is a tradeoff coefficient matrix between the covariance matrices H_i of the estimation error and N_i of the measurement error. Note that in the standard KF, the covariance matrix N_i is a preset constant. Here, we incorporate the subscript of i for ease of our subsequent extension.

At every timestep i, the KF takes the noisy measurements of z_i and its corresponding covariance matrix N_i as input and produces the "filtered" measurements $\hat{z}_i$ as

$$\hat{z}_i = \Psi g_{i|i}. \tag{20}$$

4.4.2 Utilizing Kalman Filter to Calibrate Trajectory Estimations

The main motivation in integrating subseq2seq with KF is to effectively combine the merits of both models. On one hand, we apply the subseq2seq to capture the long-range dependencies or correlations over the cell sequence; on the other hand, we feed the coarse, noisy predictions from the subseq2seq into the KF for detailed calibration.

Next, we study how to integrate subseq2seq with KF as a hybrid model. The key point is to bind the output of subseq2seq with the input of KF. In our solution, the original predictions from subseq2seq are considered as the noisy observations. While, KF takes them as input and calibrates them for output. As we mentioned in the Section 4.4.1, KF has two inputs, the noisy measurements z_i and its corresponding covariance matrix N_i. We present how to set the two kinds of input in our model as below.

Setting z_i. The subseq2seq model is able to produce coarse estimates using the centric coordinates of the predicted cell, which is used to set z_i of our model. At timestamp i, let c_i denote the cell predicted by the subseq2seq model. Then we set z_i as the centric coordinates of c_i, i.e. $z_i = \langle x_{c_i}, y_{c_i}\rangle$.

Setting N_i. The covariance matrix N_i represent the uncertainty of predicted z_i. In the standard KF model, N_i is set as a fixed priori parameter. However, intuitively, the predictive uncertainties of the estimations for different timesteps should be highly varying. Therefore, in our model, we propose to adopt a dynamic covariance matrix. Given the cell set $\mathcal{C}$, each cell $c \in \mathcal{C}$ is associated with the centric coordinate of $\langle x_c, y_c\rangle$. We aggregate the coordinates for all the cells into a matrix L of size $2 \times |\mathcal{C}|$, where each column corresponds to the longitude and latitude of a unique cell. For each timestamp i, the expected coordinate vector for the current estimate is calculated as

$$\bar{l}_i = \sum_{c' \in \mathcal{C}} \Pr(c'|h_i) \cdot l_{c'}, \tag{21}$$

where $\Pr(c'|h_i)$ is the prediction probability for cell c' using the softmax function in Eq. (4), and $l_{c'}$ is the c'th column

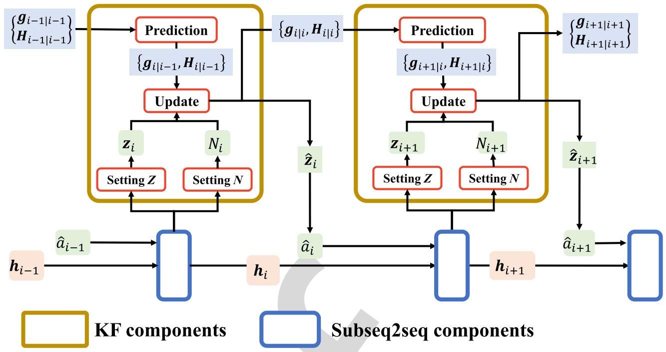


Fig. 3. The illustration for the integration of subseq2seq with Kalman Filter. We can see there are three data flows in the model: i) The flow of the hidden state h_i in the subseq2seq component (highlighted by pink), which captures the long-range dependencies or correlations over the cell sequence. ii) The flow of g, H in the KF component (highlighted by blue), which calibrate coarse cell coordinates as fine-grained trajectory point coordinates. iii) The flow of the predicted cell coordinate z_i and its calibrated coordinate $\hat{z}_i$ (including the middle variables N_i and $\hat{a}_i$, highlighted by green), which are passed between the KF component and the subseq2seq component to integrate the two component as a hybrid model.

vector of L. As such, we compute the mean coordinate vector as an expectation of the coordinates over all the cells. Then, the covariance matrix N for the timestamp i is calculated as

$$N_i = \sum_{c' \in \mathcal{C}} \Pr(c'|h_i) \cdot (l_{c'} - \bar{l}_i) \cdot (l_{c'} - \bar{l}_i)^\top. \tag{22}$$

Here, we use the probability distribution $\{\Pr(c'|h_i)\}_{c' \in \mathcal{C}}$ to combine the covariance matrix of each cell as an expected covariance of the measured coordinates z_i.

It is not easy to directly estimate the concrete latitude and longitude values in free space. So we adopt a two-stage approach for prediction. We first utilize the RNN component to locate more predictable cells, and then calibrate the centric location of the predicted cell. Our model jointly integrates the two components in a principled way. We present the illustration for the hybrid model in Fig. 3. As we can see, it contains two components, namely the subseq2seq component and the KF component. The two components are integrated by binding their input and output correspondingly. At each timestep, the subseq2seq component first generates the predicted cell of a trajectory point, and then the corresponding coordinates z_i of the predicted cell will be calibrated as $\hat{z}_i$ by the KF component using the prediction and update procedures. Moreover, the KF calibrated z_i are further discretized as a cell id $\hat{a}_i$, which is used as the input of the decoder of the subseq2seq in Eq. (1). In this manner, the subseq2seq component and the KF components are intergraded as a whole, and the original noisy, coarse cell coordinates are refined to a more reliable, accurate prediction by explicitly reducing the predictive noise. To our knowledge, it is the first time that a joint RNN-KF hybrid model has been proposed for trajectory related tasks. The model elegantly combines the merits of the RNN and KF components. As will be shown later, the two components can be optimized in a joint way, which produces a better estimation than using a loose combination.

For optimizing the KF component, we adopt the widely used mean squared error by minimizing the following loss

$$\mathcal{L}_2 = \frac{1}{2} \sum_{(\tilde{t},t)\in\mathcal{D}} \sum_{a_i \in t \text{ and } a_i \notin \tilde{t}} \left(\begin{bmatrix} a_i.x \\ a_i.y \end{bmatrix} - \hat{z}_i \right)^2, \tag{23}$$

where $\hat{z}_i$ is the output of KF component at timestep i and $\begin{bmatrix} a_i.x \\ a_i.y \end{bmatrix}$ is the actual coordinate vector for the ith point.

4.5 Model Learning

The final model loss consists of two parts, and we use a linear combination way to integrate both loss functions

$$\mathcal{L}_{total} = \mathcal{L}_1 + \lambda \cdot \mathcal{L}_2, \tag{24}$$

where $\mathcal{L}_1$ and $\mathcal{L}_2$ are the loss functions defined in Eqs. (5) and (23) respectively, and λ is a tuning parameter to balance the gradients of two loss functions in our work. Since our method is a hybrid model, it will be difficult to directly optimize the whole model. We adopt a separated approach to train the model parameters. Specifically, at each iteration, we first optimize the subseq2seq component and then update the KF component using the learned RNN parameters.

For learning the subseq2seq component, it is relatively straightforward to optimize the $\mathcal{L}_1$ in Eq. (5). We apply the cross entropy as the loss function to train our subseq2seq model. Once we obtain the parameters for the subseq2seq component, we next optimize the KF component.

While, the optimization of KF is more difficult. In the KF component, we have the following parameter matrices to learn, including $\boldsymbol{M}$, $\boldsymbol{N}$, $\boldsymbol{\Phi}$, $\boldsymbol{\Psi}$. Note that with our model, $\boldsymbol{N}$ can be directly computed using the parameters from the subseq2seq component. The transformation matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ are set as a priori based on general knowledge. Then, our focus is how to learn the state covariance matrix $\boldsymbol{M}$.

Since the KF component is coupled with the subseq2seq, we can not directly use the traditional optimization algorithm, such as the discriminative training approach in [43], to infer the KF parameter of our model. Inspired by the learning methods of backprop Kalman filter proposed in [44], we propose an *error Back Propagation Through Time for Kalman Filter* algorithm, abbreviated to *BPTT-KF*, to optimize the parameter matrix M of the KF component in our model.

The gradients from timestep $i+l$ to the procedure covariance at timestep i can be derived as follows

$$\begin{aligned} &\frac{\partial \widetilde{\mathcal{L}}_{i+l}}{\partial \boldsymbol{M}_i} \\ &= \frac{\partial \widetilde{\mathcal{L}}_{i+l}}{\partial \hat{z}_{i+l}} \frac{\partial \hat{z}_{i+l}}{\partial \boldsymbol{g}_{i+l|i+l}} \prod_{k'=i+1}^{i+l} \frac{\partial \boldsymbol{g}_{k'|k'}}{\partial \boldsymbol{g}_{k'-1|k'-1}} \frac{\partial \boldsymbol{g}_{i|i}}{\partial \boldsymbol{M}_i} \\ &= \left(\begin{bmatrix} a_i.x \\ a_i.y \end{bmatrix} - \hat{z}_{i+l} \right) \boldsymbol{\Psi}_{i+l} \prod_{k'=i+1}^{i+l} (\boldsymbol{I} - \boldsymbol{K}_{k'} \boldsymbol{\Psi}_{k'}) \boldsymbol{\Phi}_{k'} \frac{\partial \boldsymbol{g}_{i|i}}{\partial \boldsymbol{M}_i}, \end{aligned}$$

where $\widetilde{\mathcal{L}}_i$ is the squared error between the real coordinates and the predicted values, which is defined as

$$\widetilde{\mathcal{L}}_i = \frac{1}{2} \left(\begin{bmatrix} a_i.x \\ a_i.y \end{bmatrix} - \hat{z}_i \right)^2. \tag{25}$$

TABLE 1
Statistics of Our Datasets After Preprocessing

Datasets	Beijing	Shenzhen	Porto
Duration	1 month	2 weeks	1 year
#taxis	18,298	17,998	442
#trajectories	313,560	149,230	284,100
#records	31,356,000	14,923,000	8,523,000
#cells	15,870	19,306	6,351
Period	1 minute	5 minutes	1 minute

For a trajectory with a length n, we can accumulate the gradients of $\boldsymbol{M}_i$ from the current position to the end of the trajectory as

$$\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{M}_i} = \sum_{i'=i}^{n} \frac{\partial \widetilde{\mathcal{L}}_{i'}}{\partial \boldsymbol{M}_i}. \tag{26}$$

Then the parameter $\boldsymbol{M}_i$ is optimized using the gradient descent approach.

5 Experiments

In this section, we first set up the experiments, and then present the performance comparison and result analysis.

5.1 Experimental Setup

5.1.1 Construction of the Evaluation Set

To measure the performance of our proposed model, we use three real-world taxi trajectory datasets collected from Beijing, Shenzhen and Porto respectively. The taxi trajectory data from Beijing is sampled every minute, while the dataset from Shenzhen is sampled every five minutes. The dataset from Porto is a public trajectory dataset, and originally released for a taxi trajectory prediction competition on Kaggle.[1] The original sampling period of Porto dataset is 15 seconds. Here, we convert it into one minute in the preprocessing procedure. In the three datasets, we do not have the road network data. For our work, we need to partition the entire space into disjoint cells. For different datasets, we set different cell lengths. The cell length of Beijing and Porto datasets is set to 100 meters, and the cell length of Shenzhen dataset is set to 200 meters. We summarize the detailed statistics of the datasets in Table 1.

5.1.2 Evaluation Metrics

To evaluate our approach, we adopt a variety of evaluation metrics widely used in previous works [8], [20], [21], [22].

- *RMSE* is the *Root Mean Squared Error* between the real values and predicted values for the coordinates of the missing trajectory points.
- *NDTW* is the *Normalized Dynamic Time Warping* distance, that is used for evaluating the task of trajectory recovery [8]. NDTW is an enhanced version of Dynamic Time Warping distance (DTW) [45], which divides DTW by the number of reconstructed trajectory points.

1. https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i

TABLE 2
Performance Comparison of Four Metrics on Three Data Sets

Metric (km)	Datasets Sampling Rate	Beijing 30%	Beijing 50%	Beijing 70%	Shenzhen 30%	Shenzhen 50%	Shenzhen 70%	Porto 30%	Porto 50%	Porto 70%
RMSE	DHTR	**0.324**	**0.164**	**0.069**	**1.041**	**0.632**	**0.369**	**0.301**	**0.215**	**0.126**
	DeepMove	0.959	0.479	0.295	4.101	1.953	1.115	0.518	0.313	0.195
	STRNN	1.071	0.514	0.314	4.346	2.182	1.283	0.659	0.351	0.224
	MPR	0.759	0.636	0.548	2.973	2.364	2.013	0.857	0.786	0.731
	RICK	0.574	0.378	0.287	2.092	1.485	1.013	0.763	0.497	0.387
NDTW	DHTR	**0.139**	**0.059**	**0.027**	**0.522**	**0.338**	**0.125**	**0.167**	**0.099**	**0.045**
	DeepMove	0.433	0.166	0.069	1.076	0.546	0.219	0.219	0.109	0.055
	STRNN	0.486	0.185	0.082	1.126	0.587	0.246	0.237	0.128	0.060
	MPR	0.402	0.368	0.318	1.432	1.127	0.983	0.615	0.579	0.553
	RICK	0.263	0.174	0.092	1.038	0.699	0.408	0.599	0.358	0.308
LCSS	DHTR	**0.208**	**0.116**	**0.060**	**0.230**	**0.102**	**0.029**	**0.188**	**0.130**	**0.066**
	DeepMove	0.457	0.245	0.132	0.364	0.221	0.103	0.254	0.152	0.085
	STRNN	0.472	0.293	0.158	0.393	0.263	0.115	0.271	0.164	0.091
	MPR	0.451	0.409	0.394	0.396	0.353	0.317	0.591	0.559	0.528
	RICK	0.443	0.392	0.368	0.372	0.259	0.153	0.538	0.392	0.277
EDR	DHTR	**0.169**	**0.080**	**0.041**	**0.318**	**0.127**	**0.053**	**0.232**	**0.146**	**0.069**
	DeepMove	0.387	0.204	0.143	0.434	0.254	0.96	0.254	0.163	0.086
	STRNN	0.406	0.222	0.164	0.455	0.274	0.102	0.273	0.185	0.091
	MPR	0.701	0.685	0.679	0.506	0.453	0.342	0.623	0.586	0.540
	RICK	0.524	0.465	0.296	0.706	0.561	0.448	0.517	0.331	0.221

All the performance scores are better with smaller values for the four metrics.

- *LCSS* [46] is the measurement for the length of the *Longest Common Sub-Sequence* between two target sequences. It allows the skip trajectory points when necessary, which is helpful to reduce the influence of noises.
- *EDR* [47] is the *Edit Distance on Real sequence*, which is also robust to noise and addresses some deficiencies in LCSS.

Note that the original LCSS and EDR are intended to deal with sequences of discrete symbols. Here, we assume two continuous points are the same if their distance is smaller than a predefined threshold of 0.2 kilometers. For ease of analysis, we subtracting the value of LCSS and EDR from one, so all the four metrics have the same tendency: smaller is better.

5.1.3 Task Setting

For each of the three datasets, we divide it into three parts with the splitting ratio of $7:1:2$, namely training set, validation set and test set. In our datasets, all the trajectories are completely sampled. Hence, we randomly generate the sub-trajectories using a sampling rate of $r\%$. In other words, for each complete trajectory, we only keep $r\%$ of sampled trajectory points from it randomly. For the training set, we generate random sub-trajectories at each iteration. While, for the test set, we generate and fix the sub-trajectories for prediction, and the complete trajectory are held out as ground-truth for evaluation. We further vary the sampling rate of $r\%$ in the set $\{30\%, 50\%, 70\%\}$ For reliable evaluation, we repeat the above process five times, and report the average results on the five evaluation sets.

5.1.4 Methods to Compare

We consider using the following successful methods for comparison:

- *RICK* [8]: It builds a routable graph from uncertain trajectories, and then answers a users online query (a sequence of point locations) by searching top-k routes on the graph.
- *MPR* [21]: It can discover the most popular route from a transfer network based on the popularity indicators in a breadth-first manner.
- *DeepMove* [26]: It is a multi-modal embedding recurrent neural network that can capture the complicated sequential transitions by jointly embedding the multiple factors that govern the human mobility.
- *STRNN* [32]: It models local temporal and spatial contexts in each layer with transition matrices for different time intervals and geographical distances.

Among the four baselines, RICK and MPR are classic search algorithms, while DeepMove and STRNN are newly-proposed deep learning methods for next-step trajectory prediction. To our knowledge, no deep learning methods are directly applicable to trajectory recovery, and here we adapt DeepMove and STRNN to this task by consecutively predicting each missing point. In the trajectory recovery task, we follow their original way to make the next-point prediction. To recovery a missed point, we repeat the next-point prediction several times according to the time interval until the cell of the missed point is predicted. When the cell is predicted, we use the centric location as the final prediction.

All the models have some parameters to tune. We either follow the reported optimal parameter settings or optimize each model separately using the validation set. For our model, we adopt a two-layer LSTM network, the embedding size of locations is set to 512. More detailed parameter configuration can be found in Table 3. We will give the detailed analysis on the parameter sensitivity of our model in Section 5.3.3.

TABLE 3
Parameter Configuration for Our Model

Component	Notation	Configuration
RNN	$\boldsymbol{s}_j$ (Eq. (3))	$\mathbb{R}^{512\times1}$
	$\boldsymbol{r}_j$ (Eq. (2))	$\mathbb{R}^{512\times1}$
	$\boldsymbol{h}_j$ (Eq. (3))	$\mathbb{R}^{512\times1}$
	$\boldsymbol{v}$ (Eq. (12))	$\mathbb{R}^{256\times1}$
	$\boldsymbol{W}^H$ (Eq. (12))	$\mathbb{R}^{256\times512}$
	$\boldsymbol{W}^S$ (Eq. (12))	$\mathbb{R}^{256\times512}$
	$\boldsymbol{W}^P$ (Eq. (12))	$\mathbb{R}^{256\times512}$
	$\boldsymbol{W}^Q$ (Eq. (12))	$\mathbb{R}^{256\times512}$
KF	$\boldsymbol{\Phi}$ (Eq. (13))	$\mathbb{R}^{4\times4}$
	$\boldsymbol{\Psi}$ (Eq. (14))	$\mathbb{R}^{2\times4}$
	$\boldsymbol{g}_i$ (Eq. (13))	$\mathbb{R}^{4\times1}$
	z_i (Eq. (14))	$\mathbb{R}^{2\times1}$
	$\boldsymbol{H}_{i\|i-1}$ (Eq. (16))	$\mathbb{R}^{4\times4}$
	$\boldsymbol{M}$ (Eq. (16))	$\mathbb{R}^{4\times4}$
	$\boldsymbol{N}_i$ (Eq. (19))	$\mathbb{R}^{2\times2}$

TABLE 4
The Effect of Different Attention Mechanisms on Beijing Dataset

Sampling	30%	40%	50%	60%	70%
NA	0.598	0.436	0.331	0.256	0.221
BA	0.551	0.383	0.292	0.239	0.194
SA	0.490	0.353	0.252	0.214	0.177
TA	0.475	0.338	0.259	0.217	0.163
STA	**0.344**	**0.257**	**0.176**	**0.115**	**0.075**

TABLE 5
The Effect of Different Attention Mechanisms on Porto Dataset

Sampling	30%	40%	50%	60%	70%
NA	0.416	0.335	0.301	0.264	0.225
BA	0.352	0.298	0.245	0.229	0.179
SA	0.346	0.285	0.239	0.224	0.168
TA	0.348	0.284	0.234	0.221	0.162
STA	**0.316**	**0.274**	**0.229**	**0.194**	**0.138**

5.2 Result and Analysis

The performance of all methods has been presented in Table 2. It can be observed that:

(1) Comparing the two traditional algorithms, we can see RICK is much better than MPR. RICK is based on the classic A* search algorithm, and MPR is based on the graph search algorithm. MPR involves much computation over the construction of the graph, which leads it is not suitable for large-scale data.

(2) For the two neural network models, DeepMove is better than STRNN, since it incorporates more kinds of context information such as history information. Overall, DeepMove is better than the two traditional methods RICK and MPR. RICK is a competitive baseline, since it uses a series of heuristic refinement techniques for enhancing the prediction performance. As a comparison, DeepMove and STRNN mainly rely on the automatic learning of useful patterns or characteristics from original data.

(3) Finally, our proposed model DHTR consistently outperform all the baselines on three datasets with four metrics. Especially, the improvement ratios with a larger sampling rate is more significant. The underlying reasons for improvement lie in two aspects. First, we specially design a subseq2seq model equipped with the spatiotemporal attention for the task of trajectory recovery, which is able to fully utilize the spatiotemporal information in observed sub-trajectory. Second, we use the KF component to calibrate the coarse estimate by reducing prediction noise.

5.3 Detailed Analysis of Our Model

As shown in previous experiments, our model has achieved a significant improvement over all the baselines. In this part, we construct more detailed analysis of the proposed model for better understanding why it works well. Our model has two major technical contributions. First, we adopt the subseq2seq model for trajectory recovery, and incorporate the spatiotemporal attention mechanism to enhance the capacity of modeling complex dependencies or correlations for trajectory data. Second, we integrate the subseq2seq model with the KF component, which further uses a dynamic covariance for prediction. Next, we analyze the effect of these two contributions, and then report the results of parameter tuning.

For ease of analysis, we only report the average RMSE performance on the Beijing dataset and Porto dataset, while the results on the Shenzhen dataset using other metrics are similar and omitted here.

5.3.1 The Effect of the Spatiotemporal Attention

For the task of trajectory recovery, spatiotemporal influence is very important to consider. In our model (Section 4.3), we proposed an elaborately designed spatiotemporal attention mechanism for effectively utilizing such context. In this part, we analyze the effectiveness of the proposed attention mechanism. Based on the subseq2seq model, we prepare 4 variants for comparison:

- *NA:* It is the subseq2seq model without the attention mechanism.
- *BA:* It is subseq2seq model with the original Bahdanau attention mechanism proposed in [38].
- *SA:* It is the subseq2seq model which only utilizes the spatial attention in Eq. (12).
- *TA:* It is the subseq2seq model which only utilizes the temporal attention in Eq. (12).
- *STA:* It is the subseq2seq model which utilizes both the spatial and temporal attention in Eq. (12).

As showed in Tables 4 and 5, the variants with the attention mechanism are better than the one without the attention mechanisms. It indicates that it is important to utilize the attention mechanism to better capture the dependencies or correlations in trajectory data. Among all the attention mechanisms, the general attention mechanism works worst, since it essentially does not utilize the spatiotemporal information. In our model, both the spatial and temporal attentions are helpful to improve the performance. The combination between both spatial and temporal attention yields the best performance among these variants.

TABLE 6
The Effect of the KF Component with Different Covariance Matrices on Beijing Dataset

Sampling Rate	30%	50%	70%	90%
Dynamic Covariance	**5.90%**	**6.98%**	**8.09%**	**9.23%**
Static Covariance	1.18%	2.34%	4.18%	6.52%
Post-processing	0.83%	1.94%	3.65%	5.27%

TABLE 7
The Effect of the KF Component with Different Covariance Matrices on Porto Dataset

Sampling Rate	30%	50%	70%	90%
Dynamic Covariance	**4.96%**	**6.37%**	**8.58%**	**10.18%**
Static Covariance	2.41%	3.71%	4.85%	6.12%
Post-processing	1.78%	2.48%	3.27%	4.57%

5.3.2 The Effect of the Kalman Filter Component

A key contribution of our model is the integration of the subseq2seq model with the KF component. In this section, we examine the effect of KF component on the model performance. Especially, in the standard KF, the covariance matrix N_i for the observation is static and all the timesteps share the same covariance matrix. In our model, we use the dynamic covariance matrices for modeling varying predictive uncertainty with time.

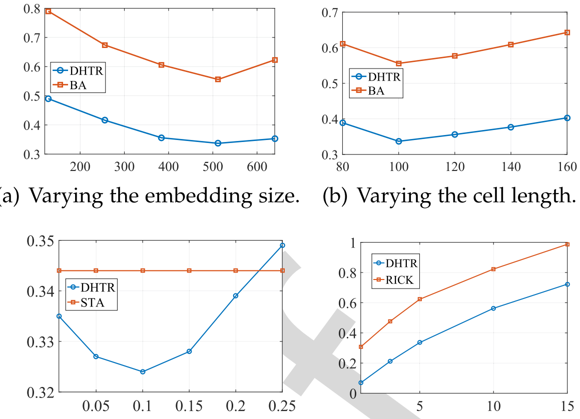
Here, we prepare four variants of our model for comparison. The first variant is the subseq2seq model (including the spatiotemporal attention) without the KF component. The second and third variants are the subseq2seq model integrated with the KF component using static and dynamic covariances respectively. While, the fourth variant is using the standard KF as a post-processing to filter the trajectory recovered by the subseq2seq model.

For ease of comparison, we take the variant without KF as the reference, and compute the improvement ratios of the other three variants over it. We also consider the comparisons with different sampling rates. We present the results in Tables 6 and 7. As we can see, the two variants integrating the KF component and the subseq2seq component as a whole are better than the one using KF as a post-processing. The dynamic covariance have better performance than the static covariance. Interestingly, with the increase of the sampling rate, the improvements become larger. It indicates that when we have more observed data, the estimation of KF can become more accurate and the calibration of KF component are more efficient.

5.3.3 Parameter Tuning

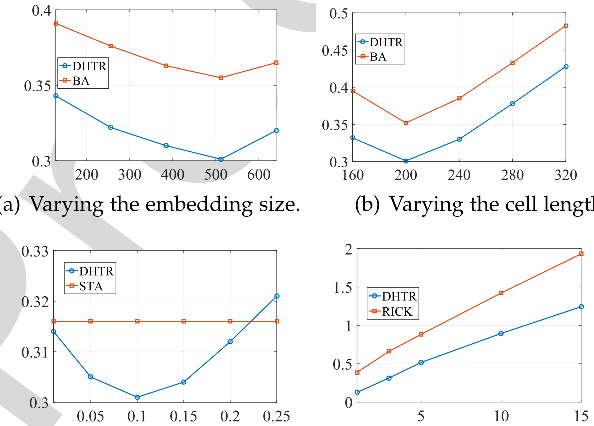
In addition to the model components, there are several parameters to tune in our model. Here we incorporate the best baseline for comparison.

Since the subseq2seq model is developed based on the discrete symbol set (i.e., the cell set), each cell ID is associated with an embedding. An important parameter to consider is the embedding size for cell IDs. We vary the embedding size from 128 to 640 with a gap of 128. As shown in Figs. 4a and 5a, the optimal embedding



(a) Varying the embedding size. (b) Varying the cell length.
(c) Varying λ. (d) Varying the sampling period.

Fig. 4. Parameter tuning for our model on Beijing dataset using the RMSE measure.



(a) Varying the embedding size. (b) Varying the cell length.
(c) Varying λ. (d) Varying the sampling period.

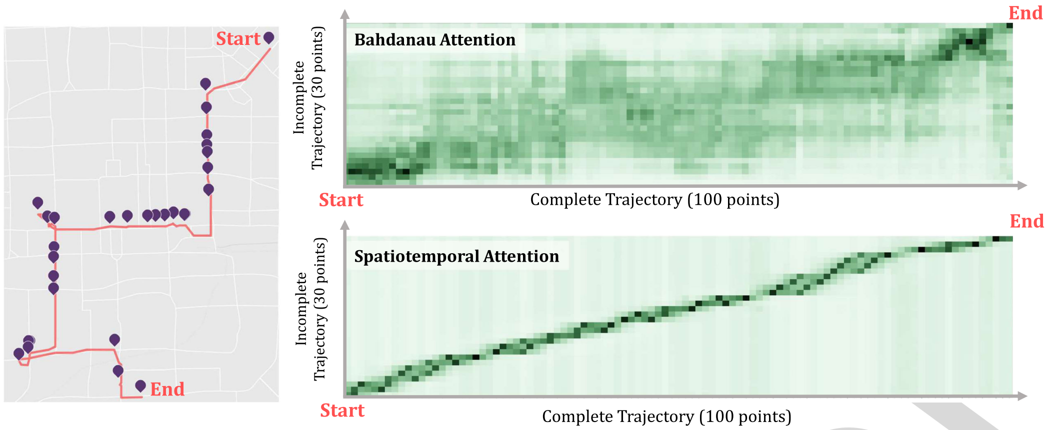
Fig. 5. Parameter tuning for our model on Porto dataset using the RMSE measure.

size is around 500. Overall, the range from 400 to 600 gives good performance.

Another parameter to tune is the cell length. When a larger cell length is used, we will have fewer cells to model. In this case, we have a small complexity for the cell sequence, but the numerical predictions for the coordinates become coarse and vice versa. Hence, the setting of the cell length should make a trade-off on the two above aspects. We vary the cell length from 80 meters to 160 meters with a gap of 20 meters for the Beijing dataset and vary the cell length from 160 meters to 320 meters with a gap of 40 meters for the Porto dataset. Figs. 4d and 5d present the varying results for different cell lengths. It can be observed that the length gives the best performance is 100 meters for the Beijing dataset and is 200 meters for the Porto dataset.

In our loss function, we incorporate a tuning parameter λ for balancing the RNN and KF components. We tune λ from 0.01 to 0.25 with a gap of 0.05 for the Beijing and Porto datasets. From Figs. 4c and 5c, it can be observed that the performance remains relatively stable when $\lambda \in [0.05, 0.15]$. And a value of 0.1 leads to the optimal performance for the both datasets.

In previous experiments, we use the sampling interval (or period) as known input (Table 3). Indeed, our model itself does not rely on a specific sampling interval and can



(a) Real Trajectory (b) Attention Weights

Fig. 6. The visualization of the attention weights for the recovery of a sample trajectory. This trajectory originally consists of 100 trajectory points. In this example, 30 points are kept as input, and the rest are hidden for prediction. For each known point, we calculate its attention weights over the complete 100 points. Hence, the attention weights are plotted as a matrix of the size of 30×100. Here, we compare two kinds of attention mechanisms for calculating the weights. The left figure gives the overview of the trajectory sequence, where the observed points are marked as purple. The right figure presents the comparison between the Bahdanau attention and the proposed spatiotemporal attention.

work with different sampling periods. To see this, we vary the sampling period in the set $\{1, 3, 5, 10, 15\}$. Figs. 4d and 5d present the results with different sampling periods for the Beijing and Porto datasets. Overall, a larger sampling period will yield a worse performance, since the input has contained fewer known points.

5.4 Qualitative Analysis on the Model Interpretability

Another major benefit of our model is that the intermediate results for predictions are highly interpretable. In this section, we present some qualitative analysis by visualizing the attention weights and the covariance matrices.

5.4.1 Visualizing Attention Weights of the Subseq2seq Model

We first present a qualitative example for understanding the attention weights in our model. We take a trajectory sequence from the Beijing dataset. The complete sequence consists of 100 trajectory points sampled by minute. After removing the points for predictions, we keep 30 random points as the observed sub-trajectory. The task is to reconstruct the missing 70 points in the original trajectory. Fig. 6a gives the overview of the trajectory sequence, where the observed points are marked as purple.

Fig. 6b presents the comparison between our proposed spatiotemporal attention mechanism and the Bahdanau attention mechanism in [38]. In the experiment, we respectively use the Bahdanau attention and the spatiotemporal attention in our model to recovery the missing trajectory points. The attention weights of the two type of attentions are plotted in Fig. 6b. As shown in the figure, there are two matrices with the size of 30×100. The upper is for the Bahdanau attention, and the lower is for our spatiotemporal attention. The horizontal axes of the matrices express the 100 trajectory points of the complete sequence, and the vertical axes express the 30 random points. The cells of the Bahdanau attention matrix denote $u_{i,j}$ that are calculated by Eq. (9), and of the spatiotemporal attention denote $u_{i,j}$ that are calculated for Eq. (12). The values of the matrix cells are the darker the higher.

As shown in the figure, the Bahdanau attention does not consider the spatiotemporal information for modeling trajectory data, and it produces dispersive attention weights over the observed points. As a comparison, our proposed method generates a skew, focused distribution of attention weights. With our attention mechanism, a trajectory point to be predicted is mainly influenced by the nearby sampled points. Hence, our attention mechanism is more capable of modeling the spatiotemporal characteristics of trajectory data.

5.4.2 Visualizing the Dynamic Covariance of Kalman Filter

In our model, we model the dynamic covariance for the predictions with time, and trace the varying of the predictive uncertainty for the estimations. Therefore, besides the estimated results, both the subseq2seq component and the KF component are able to give confidence distribution over the predictions.

Fig. 7 illustrates an example of how prediction uncertainty in our model changes with time. In this example, we present three trajectory points from a trajectory sequence, namely a_{i-2}, a_{i-1} and a_i. The prediction uncertainty of the these points are plotted on the figure as heat map. A darker color indicates a more confident prediction. At timestamp i, we give a detail of the prediction uncertainty for different components of our model. Here, the subseq2seq model outputs a noisy prediction with a confidence distribution labeled as "*Predicted by subseq2seq at i*". The noisy prediction is subsequently fed into the KF component. Recall that the KF component involves two steps. In the prediction step, KF makes a state prediction with a confidence distribution labeled as "*Predicted by KF at i*", which pulls the prediction towards the target point. Then, after the update step, our model given a final prediction with a confidence distribution labeled as "*Calibrated by KF at i*", which is more close to the target point.

Different from previous deep learning models for trajectory data, our model is able to explicitly characterize the

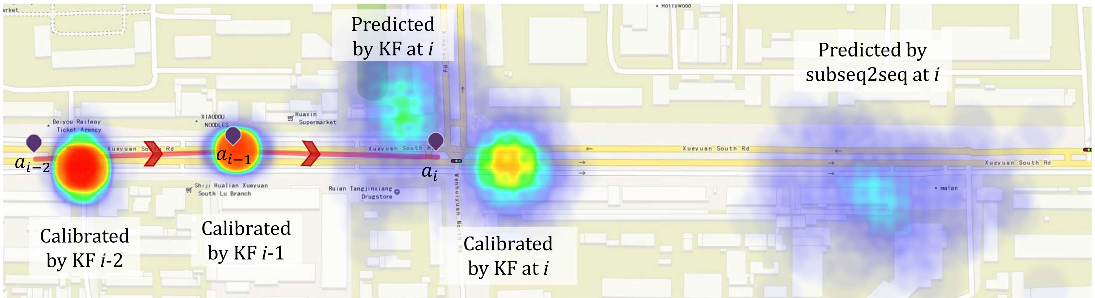


Fig. 7. The visualization of the internal procedures for the KF calibrations. Here, we have three sample points, namely a_{i-2}, a_{i-1} and a_i, whose ground-truth location indicated by the purple points. We use the heat maps to show the confidence distribution of sample locations that are predicted by KF, predicted by subseq2seq, and calibrated by KF. A darker color indicates a more confident prediction.

predictive uncertainty (e.g., the heat map in Fig. 7). It is useful to understand the internal working mechanism of the prediction model.

6 Conclusions

In this paper, we proposed a novel deep hybrid model for trajectory recovery in free space. Our model integrated the subseq2seq component with the KF component. It was endowed with the merits of both components, i.e., the capacities of modeling complex sequence data and reducing predictive noise. We constructed three large trajectory datasets for evaluation. The experimental results have shown that our model is superior to previous methods in the task of trajectory recovery.

Currently, we test the proposed model with three taxi trajectory datasets. We will consider applying our model to trajectory data in more domains, e.g., animal trace. In our work, we mainly focus the spatiotemporal correlations among trajectory points. As future work, we will extend the proposed model by incorporate more kinds of context information, e.g., POI labels and user profiles.

Acknowledgments

The work was supported by the National Natural Science Foundation of China (Grant No. 61572059, 61872369, 71531001), and the Fundamental Research Funds for the Central Universities. Dr. Wang's work was partially supported by the National Key Research and Development Program of China (Grant No. 2017YFC0820405) and the Science and Technology Project of Beijing (Grant No. Z.181100003518001). Dr. Zhao's work was partially supported by the Research Funds of Renmin University of China (Grant No. 18XNLG22). Dr. Feng's work was partially supported by the Open Research Program of Shenzhen Key Laboratory of Spatial Smart Sensing and Services (Shenzhen University).

References

[1] W. X. Zhao, N. Zhou, A. Sun, J. R. Wen, J. Han, and E. Y. Chang, "A time-aware trajectory embedding model for next-location recommendation," *Knowl. Inf. Syst.*, vol. 56, no. 3, pp. 559–579, Sep. 2018.

[2] J. Wang, J. Wu, Z. Wang, F. Gao, and Z. Xiong, "Understanding urban dynamics via context-aware tensor factorization with neighboring regularization," *IEEE Trans. Knowl. Data Eng.*, early access, May 7, 2019, doi: 10.1109/TKDE.2019.2915231.

[3] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "T-drive: Enhancing driving directions with taxi drivers' intelligence," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 220–232, Jan. 2013.

[4] K. Zheng, Y. Zheng, X. Xie, and X. Zhou, "Reducing uncertainty of low-sampling-rate trajectories," in *Proc. IEEE 28th Int. Conf. Data Eng.*, 2012, pp. 1144–1155.

[5] M. Ulm and N. Brandie, "Robust online trajectory clustering without computing trajectory distances," in *Proc. IEEE 21st Int. Conf. Pattern Recognit.*, 2012, pp. 2270–2273.

[6] D. Pfoser, C. S. Jensen, and Y. Theodoridis, "Novel approaches in query processing for moving object trajectories," in *Proc. ACM 26th Int. Conf. Very Large Data Bases*, 2000, pp. 395–406.

[7] Y. Zheng, L. Liu, L. Wang, and X. Xie, "Learning transportation mode from raw GPS data for geographic applications on the web," in *Proc. ACM 16th Conf. World Wide Web*, 2008, pp. 247–256.

[8] L. Y. Wei, Y. Zheng, and W. C. Peng, "Constructing popular routes from uncertain trajectories," in *Proc. ACM 18th SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 195–203.

[9] Y. Zheng, "Trajectory data mining: An overview," *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 3, 2015, Art. no. 29.

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

[11] B. Liao, J. Zhang, C. Wu, D. McIlwraith, T. Chen, S. Yang, Y. Guo, and F. Wu, "Deep sequence learning with auxiliary information for traffic prediction," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 537–546.

[12] P. Banerjee, S. Ranu, and S. Raghavan, "Inferring uncertain trajectories from partial observations," in *Proc. IEEE 14th Int. Conf. Data Mining*, 2014, pp. 30–39.

[13] H. Wu, J. Mao, W. Sun, B. Zheng, H. Zhang, Z. Chen, and W. Wang, "Probabilistic robust route recovery with spatio-temporal dynamics," in *Proc. ACM 22nd SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1915–1924.

[14] W. Luo, H. Tan, L. Chen, and L. M. Ni, "Finding time period-based most frequent path in big trajectory data," in *Proc. ACM 32th SIGMOD Int. Conf. Manage. Data*, 2013, pp. 713–724.

[15] H. Liang and K. Wang, "Constructing top-k routes with personalized submodular maximization of POI features," *CoRR*, vol. abs/1710.03852, 2017. [Online]. Available: http://arxiv.org/abs/1710.03852

[16] H. P. Hsieh, C. T. Li, and S. D. Lin, "Exploiting large-scale check-in data to recommend time-sensitive routes," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 55–62.

[17] H. Hsieh, C. Li, and S. Lin, "Measuring and recommending time-sensitive routes from location-based data," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2015, pp. 4193–4196.

[18] F. Xu, Z. Tu, Y. Li, P. Zhang, X. Fu, and D. Jin, "Trajectory recovery from ash: User privacy is not preserved in aggregated mobility data," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 1241–1250. [Online]. Available: https://doi.org/10.1145/3038912.3052620

[19] K. Ouyang, R. Shokri, D. S. Rosenblum, and W. Yang, "A non-parametric generative model for human trajectories," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 3812–3817. [Online]. Available: https://doi.org/10.24963/ijcai.2018/530

[20] H. Liu, L.-Y. Wei, Y. Zheng, M. Schneider, and W.-C. Peng, "Route discovery from mining uncertain trajectories," in *Proc. IEEE 11th Int. Conf. Data Mining Workshops*, 2011, pp. 1239–1242.

[21] Z. Chen, H. T. Shen, and X. Zhou, "Discovering popular routes from trajectories," in *Proc. IEEE 27th Int. Conf. Data Eng.*, 2011, pp. 900–911.

[22] H. Su, K. Zheng, H. Wang, J. Huang, and X. Zhou, "Calibrating trajectory data for similarity-based analysis," in *Proc. ACM 32th SIGMOD Int. Conf. Manage. Data*, 2013, pp. 833–844.

[23] H. Su, K. Zheng, J. Huang, H. Wang, and X. Zhou, "Calibrating trajectory data for spatio-temporal similarity analysis," *Int. J. Very Large Data Bases*, vol. 24, no. 1, pp. 93–116, 2015.

[24] S. Zheng, Y. Yue, and P. Lucey, "Generating long-term trajectories using deep hierarchical networks," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1543–1551.

[25] H. Wu, Z. Chen, W. Sun, B. Zheng, and W. Wang, "Modeling trajectories with recurrent neural networks," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 3083–3090.

[26] J. Feng, Y. Li, C. Zhang, F. Sun, F. Meng, A. Guo, and D. Jin, "DeepMove: Predicting human mobility with attentional recurrent networks," in *Proc. ACM 26th Conf. World Wide Web*, 2018, pp. 1459–1468.

[27] R. G. Brown, P. Y. Hwang, et al., *Introduction to Random Signals and Applied Kalman Filtering*, vol. 3. New York, NY, USA: Wiley, 1992.

[28] O. Pink and B. Hummel, "A statistical approach to map matching using road network geometry, topology and vehicular motion constraints," in *Proc. IEEE 11th Int. Conf. Intell. Transp. Syst.*, 2008, pp. 862–867.

[29] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Map-matching for low-sampling-rate GPS trajectories," in *Proc. ACM 17th SIGSPATIAL Int. Conf. Advances Geographic Inf. Syst.*, 2009, pp. 352–361.

[30] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun, "An interactive-voting based map matching algorithm," in *Proc. IEEE 11th Int. Conf. Mobile Data Manage.*, 2010, pp. 43–52.

[31] C. Yang, M. Sun, W. X. Zhao, Z. Liu, and E. Y. Chang, "A neural network approach to jointly modeling social networks and mobile trajectories," *ACM Trans. Inf. Syst.*, vol. 35, no. 4, pp. 36:1–36:28, 2017.

[32] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 194–200.

[33] A. Al-Molegi, M. Jabreel, and B. Ghaleb, "STF-RNN: Space time features-based recurrent neural network for predicting people next location," in *Proc. IEEE Symp. Series Comput. Intell.*, 2016, pp. 1–7.

[34] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

[35] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[36] M. Schuster and K. K. Paliwal, *Bidirectional Recurrent Neural Networks*. Piscataway, NJ, USA: IEEE Press, 1997.

[37] J. Gu, Z. Lu, H. Li, and V. O. K. Li, "Incorporating copying mechanism in sequence-to-sequence learning," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics ACL'16*, 2016, pp. 1631–1640.

[38] Y. Bengio and Y. LeCun, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015, pp. 1–15.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. U. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.

[40] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proc. 35th Int. Conf. Mach. Learn.*, 2017, pp. 1243–1252.

[41] Q. Chen, Q. Hu, J. X. Huang, L. He, and W. An, "Enhancing recurrent neural networks with positional attention for question answering," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 993–996.

[42] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Eng. Trans.*, vol. 82, pp. 35–45, 1960.

[43] P. Abbeel, A. Coates, M. Montemerlo, A. Y. Ng, and S. Thrun, "Discriminative training of kalman filters," in *Proc. Robot.: Sci. Syst. I*, 2005, pp. 289–296.

[44] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, "Backprop KF: Learning discriminative deterministic state estimators," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 4376–4384.

[45] J. B. Kruskal, "An overview of sequence comparison: Time warps, string edits, and macromolecules," *Int. J. Mach. Tools Manufacture*, vol. 25, no. 2, pp. 201–237, 1983.

[46] J. K. Kearney and S. Hansen, "Stream editing for animation," The University of Iowa, Iowa City, IA, Rep. no. ADA231316, 1990.

[47] L. Chen and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proc. ACM 24th SIGMOD Int. Conf. Manage. Data*, 2005, pp. 491–502.

Jingyuan Wang received the PhD degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He is currently an associate professor of the School of Computer Science and Engineering, Beihang University, Beijing, China. He is also the head of the Beihang Interest Group on SmartCity (BIGSCity), and vice director of the Beijing City Lab (BCL). He published more than 30 papers on top journals and conferences, as well as named inventor on several granted US patents. His general area of research is data mining and machine learning, with special interests in smart cities.

Ning Wu received the BEng degree in computer science from Beihang University, China, in 2018. He is currently working toward the master's degree in the School of Computer Science and Engineering, Beihang University. His research mainly focuses on urban data analysis.

Xinxi Lu received the PhD degree from the School of Electronic and Information Engineering, Beihang University, Beijing, China. He is currently an associate professor of the School of Software, Beihang University, Beijing, China. He has engaged in information technology for more than 10 years, mainly focusing on cloud computing, exploration and research in big data. He developed dozens of software applications, published more than 10 books and papers, as well as named inventor on several granted patents.

Wayne Xin Zhao received the PhD degree from Peking University, in 2014. He is currently an associate professor with the School of Information, Renmin University of China. His research interests are social data mining and recommender systems. He has published more than 60 referred papers in international conferences and journals. He is a member of the IEEE and ACM.

Kai Feng received the BEng degree in computer science from Beihang University, China, in 2018. He is currently working toward the master's degree in the School of Computer Science and Engineering, Beihang University. His research mainly focuses on time series analysis.