Efficient Hidden Trajectory Reconstruction from Sparse Data

Ning Yang [1]
yangning@scu.edu.cn

Philip S. Yu [2,3]
psyu@uic.edu

[1]School of Computer Science, Sichuan University, Chengdu, China
[2]Department of Computer Science, University of Illinois at Chicago, USA
[3]Institute for Data Science, Tsinghua University, Beijing, China

ABSTRACT

In this paper, we investigate the problem of reconstructing hidden trajectories from a collective of separate spatial-temporal points without ID information, given the number of hidden trajectories. The challenge is three-fold: lack of meaningful features, data sparsity, and missing trajectory links. We propose a novel approach called Hidden Trajectory Reconstruction (HTR). From an information-theoretic perspective, we devise five novel temporal features and combine them into an Latent Spatial-Temporal Feature Vector (LSTFV) to characterize the dynamics of a single spatial-temporal point. The proposed features have the potential of distinguishing spatial-temporal points between trajectories. To overcome the data sparsity, we assemble the LSTFVs to a sparse Temporal Feature Tensor (TF-Tensor) and propose an algorithm called Parallel Iterative Collaborative Approximation of Sparse Tensor (PICAST). PICAST approximates the TF-Tensor by decomposing it into a tensor product of a low-rank core identity tensor and three dense factor matrices with a divide-and-conquer strategy. To achieve a dense approximate tensor with good accuracy and efficiency, PICAST minimizes a sparsity-measure and fuses an additional matrix of static geographical region features. To recover the missing trajectory links, we propose a mapping, Cross-Temporal Connectivity Preserving Transformation (CTCPT), to map the LSTFVs of the separate spatial-temporal points to an intrinsic space called Cross-Temporal Connectivity Preserving Space (CTCPS). CTCPT uses Cross-Temporal Connectivity (CTC) to evaluate whether two spatial-temporal points belong to the same trajectory and if they do, how strong the connectivity between them is. Due to the CTCPT, the hidden trajectories can be reconstructed from clusters generated in CTCPS by a clustering algorithm. At last, the extensive experiments conducted on synthetic datasets and real datasets verify the effectiveness and efficiency of our algorithms.

CIKM'16 , October 24-28, 2016, Indianapolis, IN, USA

DOI: http://dx.doi.org/10.1145/2983323.2983796

Keywords

Trajectory Reconstruction; Latent Spatial-Temporal Feature; Cross-Temporal Connectivity; Sparse Tensor Decomposition

1. INTRODUCTION

Uncertain trajectory reconstruction has attracted increasing attention of researchers recently. The existing researches often assume that most part of an uncertain trajectory is known and the trajectory ID information is available [24, 4, 23, 11]. The assumptions, however, do not hold true for some extremely uncertain situations, where only the trajectory number and separate time-stamped points are given without any available ID information. For example, in a cyber-physical system, the sensors often report the location information of moving objects without their identities [17], which results in a collection of separate time-stamped points (we refer to such points as **spatial-temporal points**) with unknown trajectory links among them (here **trajectory link** represents a directed connection from a predecessor point to its successive point in a trajectory). For another example, in radar applications, radars may just detect a series of separate spatial-temporal points of aircrafts occurrence without trajectory link information. It is thus necessary to reconstruct their respective trajectories for the tracking purpose.

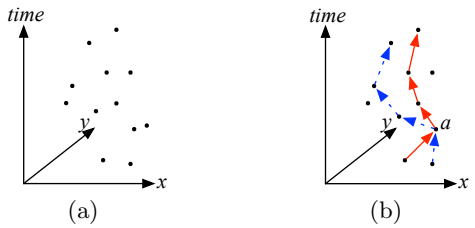

Figure 1: (a) Spatial-temporal points with unknown trajectory links. (b) Reconstructed hidden trajectories.

In this paper, we investigate the problem of reconstructing hidden trajectories from a collection of separate spatial-temporal points where only the trajectory number is known but the trajectory links are unknown. Figure 1 gives an illustration, where the space is defined by axes x and y. Figure 1(a) shows some separate spatial-temporal points among which the trajectory links are lost. If we just know there exist two hidden trajectories among these seperate points, the

task is to reconstruct the two hidden trajectories as shown in Figure 1(b), which is not easy due to the following challenges:

- **Lack of meaningful features** In the test data, we only have two features, location and time, for each point. These two features are insufficient to rebuild the trajectory link. We need to learn spatial and temporal latent features behind each location and time point to facilitate the trajectory discovery.
- **Data sparsity** The training trajectory data often span a geographical area as wide as a city, which renders the data for a specific region probably to be very sparse. Similarly, although a single trajectory often exists in only a short time period, say one hour or two hours, the whole set of training trajectories might be distributed over a long time period, say, one year, hence for a short time interval, the training trajectory points are also likely to be sparse.
- **Missing trajectory links** How to recover the trajectory links among the points that belong to the same hidden trajectory? Note that not all the given points have to be part of a trajectory, as some points are isolated and can be recognized as noises. Furthermore, the recovery of trajectory links becomes more challenging when the trajectories we consider might be entangled since moving objects could travel together for a while and then diverge. For example, in Fig. 1(b), the two hidden trajectories meet at point a, which makes it difficult to exactly determine which hidden trajectory a belongs to.

In this paper, we propose a novel approach for Hidden Trajectories Reconstruction, called HTR. Our main idea to address the above challenges stems from the intuition that *spatial-temporal points belonging to the same trajectory should be able to be grouped into the same cluster in an appropriate space*. Our major contributions can be summarized as follows:

(1) **LSTFV** To address the lack of meaningful features, we model the latent features of a point by a Latent Spatial-Temporal Feature Vector (LSTFV). An LSTFV consists of five components which can be extracted from five latent spatial-temporal feature matrixes learned from the training trajectories.

(2) **PICAST** To address the data sparsity, we propose an algorithm called Parallel Collaborative Approximation of Sparse Tensor (PICAST), where a Temporal Feature Tensor (TF-Tensor) is built based on training data. TF-Tensor is of three modes which respectively represent time slots, regions and LSTFVs, and for each input spatial-temporal point, its LSTFV can be extracted from the TF-Tensor. PICAST approximates the sparse tensor by decomposing it into a tensor product of a low-rank core identity tensor and three dense factor matrices. PICAST pursues a dense approximate tensor with good accuracy. For this purpose, PICAST minimizes a sparsity-measure and fuses an additional matrix of static geographical region features. To obtain the dense factor matrices efficiently, PICAST uses a divide-and-conquer strategy.

(3) **CTCPT** To recover missing trajectory links, we propose a mapping, Cross-Temporal Connectivity Preserving Transformation (CTCPT), which can map the LSTFVs of the input spatial-temporal points to an intrinsic space called Cross-Temporal Connectivity Preserving Space (CTCPS). CTCPT uses Cross-Temporal Connectivity (CTC) to evaluate whether two spatial-temporal points belong to the same trajectory and if they do, how strong the connectivity between them is. CTCPT can be learned from training data based on some known training trajectories. The optimization objective of the CTCPT learning is to preserve the CTC between two spatial-temporal points that belong to the same trajectory in the training data. CTCPT ensures that the transformed points in the CTCPS are close if and only if the corresponding input spatial-temporal points belong to the same hidden trajectory, so that the hidden trajectories can be reconstructed from clusters generated in the CTCPS. At last, in order to recognize the hidden trajectories that share spatial-temporal points, we employ the Fuzzy c-Means algorithm to generate overlapping clusters, where the number of clusters is exactly the number of the reconstructed hidden trajectories.

(4) We conduct extensive experiments on synthetic and real datasets to verify the performance of our approach.

The rest of this paper is organized as follows. The overview of HTR is introduced in Section 2. The definitions of TF-Tensor and LSTFV are described in Section 3. The details of PICAST are described in Section 4. The detail of CTCPT is presented in Section 5. The Hidden Trajectory Reconstruction algorithm is presented in Section 6. We analyze the experimental results in Section 7. At last, we briefly review the related work in Section 8 and conclude in Section 9.

2. OVERVIEW

Definition 1. Spatial-Temporal Point: A spatial-temporal point q is defined as a 3-tuple (σ, τ, id), where $q.\sigma$ is the region where q occurs, $q.\tau$ is the time slot number of the time-stamp of q, $q.id$ is the ID of the trajectory that q belongs to.

Note that (1) in our setting, $q.id$ is assumed to be unknown unless q is one of the points in the training data we use to learn the temporal features and CTCPT; (2) we do not need the accurate coordinates of a spatial-temporal point, but the region where a point appeared, e.g., a square. In this paper, we just consider road networks as the geographic space where a region represents a road segment. Besides, we consider the time of day as the important factor due to the periodicity of human movements [22], and divide one day into slots with width of 30 minutes and use the slot number in conjunction with the date to represent a time-stamp.

Definition 2. Trajectory: A trajectory s is a time ordered sequence of spatial-temporal points $\langle q_1, q_2, \cdots, q_{n_s} \rangle$, where n_s is the length of trajectory s, and all $q_i.id$ are from the same trajectory ID which may be unknown, and $q_i.\tau < q_{i+1}.\tau$.

If a moving object stays at the same region for multiple consecutive time points, we will merge those points into one spatial-temporal point, which ensure that the region of the ith spatial-temporal point is always different from that of the no. $(i+1)$ spatial-temporal point.

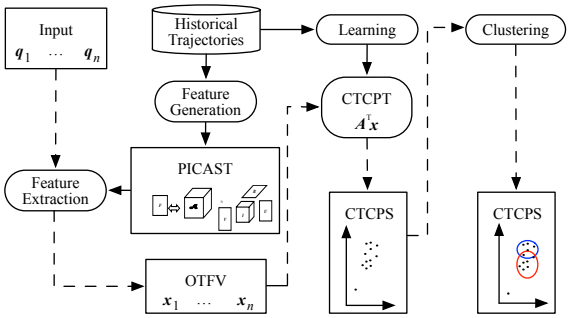

Figure 2: Work Flow of HTR

Figure 2 shows the work flow of our proposed approach, which contains a learning part and a reconstructing part. The learning part includes two subparts, Feature Generation and CTCPT learning. During the Feature Generation, we generate 5 temporal features for each region in each time slot from trajectories in the training data and build the TF-Tensor. To fill in the sparse cells of the TF-Tensor, we use PICAST to produce an approximate dense tensor in a parallel way. CTCPT Learning produces a transformation matrix from the trajectories in the training data. The path along the dashed arrows is the reconstructing part, which fulfils the hidden trajectory reconstruction from the input of separate spatial-temporal points q_i whose trajectory IDs are unknown. It first extracts the LSTFV $\boldsymbol{x}_i$ of each input point from the approximation tensor produced by CAST, then maps the LSTFVs to the CTCPS by applying the CTCPT. Finally, hidden trajectories are reconstructed from clusters generated in the CTCPS by applying the Fuzzy c-Means algorithm.

3. LATENT SPATIAL-TEMPORAL FEATURE VECTOR

In this section, we first define Latent Spatial-Temporal Feature Vector (LSTFV), and then justify it.

3.1 LSTFV

For a spatial-temporal point q_i, its LSTFV is defined as a 5-dimensional vector $\boldsymbol{x}_i = (e, g, d, h, u)^T$, where $\boldsymbol{x}_i.e = e(q_i.\sigma, q_i.\tau)$, $\boldsymbol{x}_i.g = g(q_i.\sigma, q_i.\tau)$, $\boldsymbol{x}_i.d = d(q_i.\sigma, q_i.\tau)$, $\boldsymbol{x}_i.h = h(q_i.\sigma, q_i.\tau)$, and $\boldsymbol{x}_i.u = u(q_i.\sigma, q_i.\tau)$ represent respectively the 5 temporal features, *transition randomness*, *visiting normality*, *stability*, *horizon*, and *significance*.

3.1.1 Transition Randomness

We define *transition randomness* to measure the uncertainty of the regions that a moving object will visit after visiting region r. At first we introduce the concept of *transition link* from a given region to another region. If there exists a trajectory visiting region r at time t, then there exists a transition link from r to another region visited by that trajectory later than t.

Definition 3. Transition Link: For a given region r and a time t, there exists a transition link, $r \xrightarrow{t} r'$, from r to another region r' at t if $\exists q_j, q_k$ such that $q_j.id = q_k.id, q_j.\sigma = r, q_k.\sigma = r', q_j.\tau = t$, and $q_j.\tau < q_k.\tau$.

Definition 4. Transition Randomness: Let R' be a set of regions to which there exist transition links from a region r at t, i.e., $R' = \{r' | r \xrightarrow{t} r'\}$, then the transition randomness $e(r,t)$ of region r at time t is defined as $e(r,t) = -\sum_{r' \in R'} P(r \xrightarrow{t} r') log_2 P(r \xrightarrow{t} r')$ where $P(r \xrightarrow{t} r')$ is the probability of a transition link $r \xrightarrow{t} r'$.

The probability $P(r \xrightarrow{t} r')$ can be estimated from the training trajectory set S. Let S_{rt} be the subset of S consisting of trajectories that visit r at time t, i.e., $S_{rt} = \{s | s \in S \ \& \ \exists q, q.id = s, q.\sigma = r, q.\tau = t\}$, and let $E_{rt}^{r'}$ be the subset of S_{rt} consisting of trajectories that visit r' at time $t' > t$, i.e., $E_{rt}^{r'} = \{s | s \in S_{rt} \ \& \ \exists q, q.id = s, q.\sigma = r', q.\tau > t\}$, then $P(r \xrightarrow{t} r') = \frac{|E_{rt}^{r'}|}{|S_{rt}|}$.

3.1.2 Visiting Normality

We define *visiting normality* to indicate how normal it is that region r is visited at t by a moving object. Intuitively, if the visiting time to r is rather random (which means the visits to r are almost uniformly distributed over time slots), any visit to r at any time is normal. Otherwise, only the visits at the time of a high probability are considered normal. Based on this intuition, we can define the visiting normality of a region r at t as follow:

Definition 5. Visiting Normality: The visiting normality $g(r,t)$ of region r at time t is defined as $g(r,t) = \frac{P(t|r)}{H_r}$, where $H_r = -\sum_{t'} P(t'|r) log_2 P(t'|r)$, $P(t|r)$ is the probability of a visit to r at time t, which can be estimated as $P(t|r) = \frac{|S_{rt}|}{|S_r|}$, where $S_r = \{s | \exists q, q.id = s, q.\sigma = r\}$

3.1.3 Stability

We define *stability* to measure the uncertainty of the time length for which a moving object will stay at a region. At first we define the stay time of a trajectory at a region r.

Definition 6. Stay Time: For a trajectory s visiting r at time t, the stay time $t_d(s,r,t)$ of s at r is defined as $t_d(s,r,t) = q_{i+1}.\tau - q_i.\tau$, where $q_i.id = q_{i+1}.id = s$, $q_i.\tau = t$ and $q_i.\sigma = r$.

Definition 7. Stability: The stability of region r at time t is defined as $d(r,t) = -\sum_{t' \in T_d(r,t)} P(t') log_2 P(t')$, where $T_d(r,t)$ is the set of different stay time lengths at region r at time t, and $P(t')$ is the probability of stay time length t', which can be estimated as $P(t') = \frac{|S_{rt}^{t'}|}{|S_{rt}|}$, where $S_{rt}^{t'} = \{s | s \in S_{rt} \ \& \ t_d(s,r,t) = t'\}$.

3.1.4 Horizon

We define the *horizon* of a region r at time t as the expected length of trajectories that visit r at t, which measures how large the moving range of a object who passes through region r usually is.

Definition 8. Horizon: Let $L(r,t)$ be the set of different lengths of the trajectories that visit region r at time t, then the horizon of region r at t is defined as $h(r,t) = \sum_{l \in L(r,t)} l \times P(l)$, where $P(l)$ can be estimated as $P(l) = \frac{|Z_{rt}^l|}{|S_{rt}|}$, where $Z_{rt}^l = \{s | s \in S_{rt} \ \& \ |s| = l\}$.

3.1.5 Significance

We define the *significance* to measure how indispensable the region r is for a trajectory which visits it at t.

Definition 9. Significance: Let $N(r)$ be the set of r's neighbor regions. The significance of region r at time t is defined as the probability that r is the intermediary of any pair of its neighbor regions at t, i.e., $u(r,t) = \frac{|M^r_{rt}|}{|M_{rt}|}$, where $M_{rt} = \{s | \exists q_i, q_k \in s, k > i, q_i.\sigma \in N(r), q_k.\sigma \in N(r), q_i.\tau < t < q_k.\tau\}$, and $M^r_{rt} = \{s | s \in M_{rt}, \exists q_j \in s, q_j.\sigma = r, q_j.\tau = t\}$.

Definition 9 shows that the significance of region r at t is the proportion of the training trajectories that visit r at t over all the training trajectories that visit the neighbors of r before and after t. The higher significance of region r at t, the more likely region r is the intermediary point when an object moves from its one neighbor to another neighbor at time t, or in other words, for objects that pass from r's one neighbor to another, r is indispensable as a bridge.

3.2 Justification

We argue why the five proposed temporal features make sense. The reason starts from our attempts to answer the question that what makes trajectories look different. Our answer is grounded on an information-theoretic idea that each trajectory has a unique randomness which makes it distinguishable from others. We use LSTFV to evaluate the randomness of a single point from 5 aspects corresponding to the five features, all of which are defined by using an information entropy or a probability. Note that two single points belonging to different trajectories might expose similar randomness (that is why we can not reconstruct the hidden trajectories by clustering the points directly in the LSTFV space). However, the randomness of a trajectory is still unique because it depends not only on the randomness of a single point, but also on a combination of the randomness of the whole set of the points belonging to that trajectory. We argue that it is such uniqueness that makes the points belonging to the same trajectory be in the same cluster in CTCPS, as we will see in Section 5.

4. PARALLEL COLLABORATIVE APPROXIMATION OF SPARSE TENSOR

Once we generate the LSTFVs for training points, we can assemble them to a Temporal Feature Tensor (TF-Tensor). As shown in Figure 3, the TF-Tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{M \times N \times 5}$ consists of three modes which respectively represent M regions $(r_1, \cdots, r_M)$, N time slots $(t_1, \cdots, t_N)$, and the 5 temporal features. An entry $\boldsymbol{\mathcal{T}}(i,j,q)$ stores the value of the q-th feature of region r_i in time slot t_j, where $i = 1, \cdots, M, j = 1, \cdots, N, q = 1, \cdots, 5$. The 5 temporal features, $a_1, \cdots, a_5$, respectively correspond to the temporal features described in the last section, *transition randomness*, *visiting normality*, *stability*, *horizon*, and *significance*. For a spatial-temporal point q_i, its LSTFV $\boldsymbol{x}_i$ can be retrieved as a tensor fiber, i.e., $\boldsymbol{x}_i = \boldsymbol{\mathcal{T}}(q_i.\tau, q_i.\sigma, :)$.

As we have mentioned, TF-Tensor might be very sparse, since the training trajectories span a city-wide area and are distributed over a long time period. To address this problem, we propose a Parallel Iterative Collaborative Approximation of Sparse Tensor (PICAST). In the following subsections, we first define the model of PICAST, then present its implementation.

4.1 Model of PICAST

As illustrated in Figure 3, we approximate the original tensor $\boldsymbol{\mathcal{T}}$ by a CP decomposition [9],

$$\boldsymbol{\mathcal{T}} \approx \widetilde{\boldsymbol{\mathcal{T}}} = \boldsymbol{\mathcal{I}} \times_1 \boldsymbol{B} \times_2 \boldsymbol{U} \times_3 \boldsymbol{V}, \tag{1}$$

i.e., a core identity tensor $\boldsymbol{\mathcal{I}} \in \mathbb{R}^{L \times L \times L}$ multiplied by three latent factor matrices, $\boldsymbol{B} \in \mathbb{R}^{M \times L}$, $\boldsymbol{U} \in \mathbb{R}^{N \times L}$, $\boldsymbol{V} \in \mathbb{R}^{5 \times L}$, along its three modes respectively, where L is the target rank, and the symbol $\times_i$ $(1 \le i \le 3)$ stands for the tensor multiplication along the i-th mode.

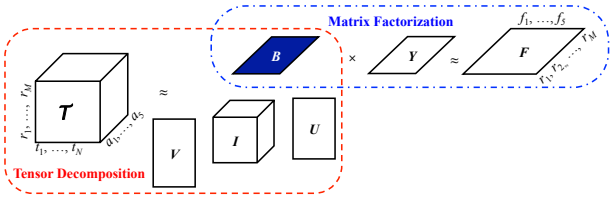

Figure 3: Model of PICAST

We argue that the decomposition shown in Equation (1) is meaningful. Actually, $\boldsymbol{V}$ can be regarded as the information about LSTFV, $\boldsymbol{U}$ as the information about time, and $\boldsymbol{B}$ as the information about region.

To make the approximate tensor $\widetilde{\boldsymbol{\mathcal{T}}}$ as dense as possible, we choose the classical sparsity-measure, the n-rank of tensor (in our case, $n = 3$) [9], as part of the objective function that PICAST is going to minimize. The n-rank of $\widetilde{\boldsymbol{\mathcal{T}}}$ is an n-dimension vector consisting of the ranks of the unfolding matrices of $\widetilde{\boldsymbol{\mathcal{T}}}$, which is defined as: $\text{rank}_n(\widetilde{\boldsymbol{\mathcal{T}}}) = \langle \text{rank}(\widetilde{T}_{(1)}), \cdots, \text{rank}(\widetilde{T}_{(n)}) \rangle$, where $\widetilde{T}_{(i)}, 1 \le i \le n$, is the unfolding matrix along the mode-i of $\widetilde{\boldsymbol{\mathcal{T}}}$. However, minimizing the sum $\sum_{i=1}^{n} \text{rank}(\widetilde{T}_{(i)})$ is difficult since it is non-convex [6]. So we relax it to become the convex function: $\|\widetilde{\boldsymbol{\mathcal{T}}}\|_{\text{tr}} = \sum_{i=1}^{n} \|\widetilde{T}_{(i)}\|_{\text{tr}}$, where $\|X\|_{tr}$ is the trace norm of matrix X. We argue that $\|\widetilde{\boldsymbol{\mathcal{T}}}\|_{\text{tr}}$ is a reasonable convex approximation of $\sum_{i=1}^{n} \text{rank}(\widetilde{T}_{(i)})$, because for each $\widetilde{T}_{(i)}$, $\|\widetilde{T}_{(i)}\|_{\text{tr}} \le \text{rank}(\widetilde{T}_{(i)})$, which leads to $\|\widetilde{\boldsymbol{\mathcal{T}}}\|_{\text{tr}} \le \sum_{i}^{n} \text{rank}(\widetilde{T}_{(i)})$.

In order to achieve a higher accuracy of the approximation, we collaboratively decompose $\boldsymbol{\mathcal{T}}$ with a static geographical feature matrix $\boldsymbol{F} \in \mathbb{R}^{M \times 5}$ we build as an extra information source, as shown in Figure 3. The 5 static geographical features $f_1, \cdots, f_5$ of a road segment (i.e. a region) r are respectively r's length, direction (e.g. one-way or bi-directional), the number of r's lanes, the number of r's neighbors, and the number of Point of Interests (POIs) around r. An entry $\boldsymbol{F}(i,j)$ stores the value of the j-th static geographical feature of region r_i, where $i = 1, \cdots, M, j = 1, \cdots, 5$. $\boldsymbol{F}$ can be factorized as $\boldsymbol{F} = \boldsymbol{B} \times \boldsymbol{Y}$, where $\boldsymbol{Y} \in \mathbb{R}^{L \times 5}$ is another latent factor matrix. Note that $\boldsymbol{F}$ shares $\boldsymbol{B}$ with $\boldsymbol{\mathcal{T}}$. The idea here is that the dynamical temporal knowledge of the spatial-temporal points, which is represented by the tensor $\boldsymbol{\mathcal{T}}$, can fuse, through $\boldsymbol{B}$, with the static geographical features of the regions, which is represented by $\boldsymbol{F}$.

ALGORITHM 1: *PICAST*$(\boldsymbol{\mathcal{T}}, \boldsymbol{F}, \epsilon)$

Input: Tensor $\boldsymbol{\mathcal{T}}$, static geographical feature matrix $\boldsymbol{F}$, and error threshold ϵ.
Output: Factor matrices $\boldsymbol{V}, \boldsymbol{U}, \boldsymbol{B}$.
Partition $\boldsymbol{\mathcal{T}}$ into a grid of sub-tensors, $\{\boldsymbol{\mathcal{T}}^{(\vec{k})}\}$;
Set η as step size;
for *each sub-tensor* $\boldsymbol{\mathcal{T}}^{(\vec{k})}$ **do**
  Initialize $\boldsymbol{V}^{(\vec{k})}, \boldsymbol{U}^{(\vec{k})}, \boldsymbol{B}^{(\vec{k})}, \boldsymbol{Y}^{(\vec{k})}$ with small random values;
  while $\Gamma_t - \Gamma_{t+1} > \epsilon$ **do**
    for *each* $\boldsymbol{\mathcal{T}}^{(\vec{k})}_{ijl} \neq 0$ **do**
      $\boldsymbol{V}^{(\vec{k})}_{i*} = \boldsymbol{V}^{(\vec{k})}_{i*} - \eta \partial_{\boldsymbol{V}^{(\vec{k})}_{i*}} \Gamma$;
      $\boldsymbol{U}^{(\vec{k})}_{j*} = \boldsymbol{U}^{(\vec{k})}_{j*} - \eta \partial_{\boldsymbol{U}^{(\vec{k})}_{j*}} \Gamma$;
      $\boldsymbol{B}^{(\vec{k})}_{l*} = \boldsymbol{B}^{(\vec{k})}_{l*} - \eta \partial_{\boldsymbol{B}^{(\vec{k})}_{l*}} \Gamma$;
      $\boldsymbol{Y}^{(\vec{k})} = \boldsymbol{Y}^{(\vec{k})} - \eta \partial_{\boldsymbol{Y}^{(\vec{k})}} \Gamma$;
    end
  end
end
Concurrently build $\boldsymbol{V}, \boldsymbol{U}, \boldsymbol{B}$ by iteratively concatenating the factors of the sub-tensors, $\boldsymbol{V}^{(\vec{k})}, \boldsymbol{U}^{(\vec{k})}, \boldsymbol{B}^{(\vec{k})}$, in terms of Lemma 1;

Now we can define the optimization objective of PICAST as to minimize the following function:

$$\Gamma(\boldsymbol{V}, \boldsymbol{U}, \boldsymbol{B}, \boldsymbol{Y}) = \frac{1}{2}\|\boldsymbol{\mathcal{T}} - \widetilde{\boldsymbol{\mathcal{T}}}\|_2^2 + \frac{1}{2}\|\widetilde{\boldsymbol{\mathcal{T}}}\|_{\text{tr}}^2 + \frac{1}{2}\|\boldsymbol{F} - \boldsymbol{B}\boldsymbol{Y}\|_2^2 + \frac{1}{2}(\|\boldsymbol{V}\|_2^2 + \|\boldsymbol{B}\|_2^2 + \|\boldsymbol{U}\|_2^2 + \|\boldsymbol{Y}\|_2^2) \tag{2}$$

where $\|\cdot\|_2$ represents the 2-norm of matrix. The first term of the right side of Equation (2) controls the decomposition error, the second term controls the sparsity, the third term controls the error of factorization of $\boldsymbol{F}$, and the last term is the regularizing term used to avoid overfitting. Note that the collaborative decomposition part is inspired by [20], but the sparsity control part is a novel contribution of this paper, by which PICAST is able to make an optimal tradeoff between the density and the accuracy of the approximate tensor.

4.2 Divide-and-Conquer Strategy

The sparse tensor $\boldsymbol{\mathcal{T}}$ will incur a very high computational cost of Equation (1) due to its huge volume. To overcome this issue, we use a divide-and-conquer strategy. Algorithm 1 gives the procedures of PICAST. PICAST first partitions the tensor $\boldsymbol{\mathcal{T}}$ into a grid of sub-tensors (), $\mathfrak{T} = \{\boldsymbol{\mathcal{T}}^{(\vec{k})} | \vec{k} \in \mathcal{K}\}$, where $\mathcal{K}$ is a collection of sub-tensor indexes, $\mathcal{K} = \{[k_1, k_2, k_3] | 1 \leq k_1 \leq K_1, 1 \leq k_2 \leq K_2, 1 \leq k_3 \leq K_3\}$, and $K_i (1 \leq i \leq 3)$ is the number of sub-tensors along ith mode. Then PICAST concurrently factorizes the sub-tensors with respect to the objective Equation (2) by using a stochastic gradient descent decomposition algorithm [8], and finally integrates the partial results to produce the final factor matrices for the whole tensor. The following lemma ensures that a large-scale tensor decomposition can be obtained by integrating the factors of its sub-tensors [14].

LEMMA 1. *If a tensor $\boldsymbol{\mathcal{T}}$ can be partitioned into two sub-tensors $\boldsymbol{\mathcal{T}}^{(1)}$ and $\boldsymbol{\mathcal{T}}^{(2)}$, and they are factorized as $\boldsymbol{\mathcal{T}}^{(1)} = \boldsymbol{\mathcal{I}}^{(1)} \times_1 \boldsymbol{V}^{(1)} \times_2 \boldsymbol{U}^{(1)} \times_3 \boldsymbol{B}^{(1)}$ and $\boldsymbol{\mathcal{T}}^{(2)} = \boldsymbol{\mathcal{I}}^{(2)} \times_1 \boldsymbol{V}^{(2)} \times_2 \boldsymbol{U}^{(2)} \times_3 \boldsymbol{B}^{(2)}$ respectively, then the tensor $\boldsymbol{\mathcal{T}}$ can be factorized as $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{I}} \times_1 \boldsymbol{V} \times_2 \boldsymbol{U} \times_3 \boldsymbol{B}$, where $\boldsymbol{\mathcal{I}} = \begin{bmatrix} \boldsymbol{\mathcal{I}}^{(1)} & \\ & \boldsymbol{\mathcal{I}}^{(2)} \end{bmatrix}$, $\boldsymbol{V} = \begin{bmatrix} \boldsymbol{V}^{(1)} & \boldsymbol{V}^{(2)} \end{bmatrix}$, $\boldsymbol{U} = \begin{bmatrix} \boldsymbol{U}^{(1)} & \boldsymbol{U}^{(2)} \end{bmatrix}$, and $\boldsymbol{\mathcal{B}} = \begin{bmatrix} \boldsymbol{\mathcal{B}}^{(1)} & \\ & \boldsymbol{\mathcal{B}}^{(2)} \end{bmatrix}$.*

For the approximation of a sub-tensor, $\boldsymbol{\mathcal{T}}^{(\vec{k})} \approx \widetilde{\boldsymbol{\mathcal{T}}}^{(\vec{k})} = \boldsymbol{\mathcal{I}}^{(\vec{k})} \times_1 \boldsymbol{V}^{(\vec{k})} \times_2 \boldsymbol{U}^{(\vec{k})} \times_3 \boldsymbol{B}^{(\vec{k})}$, the gradients of the objective function $\Gamma(\boldsymbol{V}^{(\vec{k})}, \boldsymbol{U}^{(\vec{k})}, \boldsymbol{B}^{(\vec{k})}, \boldsymbol{Y}^{(\vec{k})})$ are given by the following equations:

$$\begin{aligned}
\partial_{\boldsymbol{V}^{(\vec{k})}_{i*}} \Gamma &= (\widetilde{\boldsymbol{\mathcal{T}}}^{(\vec{k})}_{ijl} - \boldsymbol{\mathcal{T}}^{(\vec{k})}_{ijl}) \times \boldsymbol{\mathcal{I}}^{(\vec{k})} \times_2 \boldsymbol{U}^{(\vec{k})}_{j*} \times_3 \boldsymbol{B}^{(\vec{k})}_{l*} \\
&\quad + \partial_{\boldsymbol{V}^{(\vec{k})}_{i*}} \|\widetilde{\boldsymbol{\mathcal{T}}}^{(\vec{k})}\|_{\text{tr}} + \boldsymbol{V}^{(\vec{k})}_{i*}, \\
\partial_{\boldsymbol{U}^{(\vec{k})}_{j*}} \Gamma &= (\widetilde{\boldsymbol{\mathcal{T}}}^{(\vec{k})}_{ijl} - \boldsymbol{\mathcal{T}}^{(\vec{k})}_{ijl}) \times \boldsymbol{\mathcal{I}}^{(\vec{k})} \times_1 \boldsymbol{V}^{(\vec{k})}_{i*} \times_3 \boldsymbol{B}^{(\vec{k})}_{l*} \\
&\quad + \partial_{\boldsymbol{U}^{(\vec{k})}_{j*}} \|\widetilde{\boldsymbol{\mathcal{T}}}^{(\vec{k})}\|_{\text{tr}} + \boldsymbol{U}^{(\vec{k})}_{j*}, \\
\partial_{\boldsymbol{B}^{(\vec{k})}_{l*}} \Gamma &= (\widetilde{\boldsymbol{\mathcal{T}}}^{(\vec{k})}_{ijl} - \boldsymbol{\mathcal{T}}^{(\vec{k})}_{ijl}) \times \boldsymbol{\mathcal{I}}^{(\vec{k})} \times_1 \boldsymbol{V}^{(\vec{k})}_{i*} \times_2 \boldsymbol{U}^{(\vec{k})}_{j*} \\
&\quad + (\boldsymbol{B}^{(\vec{k})}_{l*} \times \boldsymbol{Y}^{(\vec{k})} - \boldsymbol{F}^{(\vec{k})}_{l*}) \times \boldsymbol{Y}^{(\vec{k})} + \partial_{\boldsymbol{B}^{(\vec{k})}_{l*}} \|\widetilde{\boldsymbol{\mathcal{T}}}^{(\vec{k})}\|_{\text{tr}} \\
&\quad + \boldsymbol{B}^{(\vec{k})}_{l*}, \\
\partial_{\boldsymbol{Y}^{(\vec{k})}} \Gamma &= (\boldsymbol{B}^{(\vec{k})}_{l*} \times \boldsymbol{Y}^{(\vec{k})} - \boldsymbol{F}^{(\vec{k})}_{l*}) \times \boldsymbol{B}^{(\vec{k})}_{l*} + \boldsymbol{Y}^{(\vec{k})},
\end{aligned}$$

where $\widetilde{\boldsymbol{\mathcal{T}}}^{(\vec{k})}_{ijl} = \boldsymbol{\mathcal{I}}^{(\vec{k})} \times_1 \boldsymbol{V}^{(\vec{k})}_{i*} \times_2 \boldsymbol{U}^{(\vec{k})}_{j*} \times_3 \boldsymbol{B}^{(\vec{k})}_{l*}$. Note that it is not difficult to derive $\partial_{\boldsymbol{V}^{(\vec{k})}_{i*}} \|\widetilde{\boldsymbol{\mathcal{T}}}^{(\vec{k})}\|_{\text{tr}} = \boldsymbol{V}^{(\vec{k})}_{i*} \times (ZZ^T + Z^TZ)$, where $Z = \boldsymbol{\mathcal{I}}^{(\vec{k})} \times_2 \boldsymbol{U}^{(\vec{k})}_{j*} \times_3 \boldsymbol{B}^{(\vec{k})}_{l*}$, and the remaining derivatives of $\|\widetilde{\boldsymbol{\mathcal{T}}}^{(\vec{k})}\|_{\text{tr}}$ can be calculated similarly.

5. CROSS-TEMPORAL CONNECTIVITY PRESERVING TRANSFORMATION (CTCPT)

5.1 Definition of Cross-Temporal Connectivity (CTC)

Cross-Temporal Connectivity (CTC) evaluates whether two spatial-temporal points belong to the same trajectory and if they do, how strong the connectivity between them is.

Definition 10. Cross-Temporal Connectivity: The cross-temporal connectivity between any two spatial-temporal points q_i and q_j is defined as:

$$c_{ij} = \begin{cases} e^{-(|q_i.\tau - q_j.\tau|)(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)} & \text{if } \exists s,\ q_i \in s \text{ and } q_j \in s \\ 0 & \text{otherwise} \end{cases},$$

where s is a trajectory, and $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are the LSTFVs of q_i and q_j, respectively.

Definition 10 is reasonable due to its two properties. First, CTC is nonzero only between two spatial-temporal points belonging to the same trajectory. Second, the exponential

function is used to assign a greater value to the CTC between q_i and q_j if they are closer in time, which is consistent with the intuition that the neighboring points in a trajectory should have a stronger connectivity than others. Similarly, the CTC between the points with smaller $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$ is also assigned a greater value.

5.2 Definition of CTCPT

CTCPT maps the LSTFV of a spatial-temporal point onto the CTCPS by a transformation matrix $\boldsymbol{A}$ of $5 \times p$, where p is the dimensionality of the CTCPS. The objective is to make the transformed points of any two spatial-temporal points belonging to the same trajectory closer in CTCPS.

Definition 11. CTCPT: For any spatial-temporal point q_i, CTCPT is defined as the mapping, $\boldsymbol{x}_i \mapsto \boldsymbol{y}_i = \boldsymbol{A}^T \boldsymbol{x}_i$, such that $\boldsymbol{y}_i$ is a p-dimensional vector, and for any two spatial-temporal points q_i and q_j belonging to the same trajectory, i.e. $q_i.id = q_j.id$, their transformed points, i.e. $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$, are closer to each other than to any other transformed point of $q_k, q_k.id \neq q_i.id$.

5.3 Learning CTCPT

Now the problem turns out to be how to learn from training data a transformation matrix $\boldsymbol{A}$ satisfying Definition 11. In this subsection, we first define the data structure we use to learn the CTCPT, then define the objective function we want to optimize during learning, at last describe our Laplacian graph based learning algorithm.

5.3.1 Data Structure

The key data structure we used in the learning of CTCPT is the weighted adjacent matrix of the training spatial-temporal points, which is defined as follow.

Definition 12. Weighted Adjacent Matrix: Given a training trajectory dataset S, its weighted adjacent matrix $\boldsymbol{W}$ is an $m \times m$ matrix, where m is the total number of spatial-temporal points in S. Each element w_{ij} is the weight between the training spatial-temporal points q_i and q_j, and $w_{ij} = c_{ij}$.

5.3.2 Objective Function

Inspired by the idea proposed by M. Belkin *et al.* [3], we choose the following function, $\gamma(\boldsymbol{A}) = \frac{1}{2}\sum_{i,j}(\boldsymbol{y}_i - \boldsymbol{y}_j)^2 w_{ij}$, as the optimization criterion of the learning of the transformation matrix $\boldsymbol{A}$ from the training data, where $\boldsymbol{y}_i = \boldsymbol{A}^T \boldsymbol{x}_i$ and $\boldsymbol{y}_j = \boldsymbol{A}^T \boldsymbol{x}_j$ are the transformed points, and $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are the LSTFVs of q_i and q_j respectively. It is easy to check that a small value of $\gamma(\boldsymbol{A})$ ensures the points belonging to the same trajectory to be closer in the CTCPS than the points belonging to different trajectories. The following theorem gives a more tractable form of function $\gamma(\boldsymbol{A})$.

THEOREM 1. *Let $\boldsymbol{L}$ be the Laplacian matrix of the weighted adjacent matrix $\boldsymbol{W}$, i.e., $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$, where $\boldsymbol{D}$ is a diagonal matrix of which the element $d_{ii} = \sum_j w_{ij}$, and Let $\boldsymbol{X}$ be a matrix where the ith column vector is $\boldsymbol{x}_i$, then $\gamma(\boldsymbol{A}) = \boldsymbol{A}^T \boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^T \boldsymbol{A}$.*

PROOF.

$$\begin{aligned}
\gamma(\boldsymbol{A}) &= \frac{1}{2}\sum_{i,j}(\boldsymbol{y}_i^2 + \boldsymbol{y}_j^2 - 2\boldsymbol{y}_i\boldsymbol{y}_j)w_{ij} \\
&= \frac{1}{2}(\sum_i(\boldsymbol{y}_i^2\sum_j w_{ij}) + \sum_j(\boldsymbol{y}_j^2\sum_i w_{ij}) - 2\sum_{i,j}\boldsymbol{y}_i\boldsymbol{y}_j w_{ij}) \\
&= \frac{1}{2}(2\sum_i \boldsymbol{y}_i^2 d_{ii} - 2\sum_{i,j}\boldsymbol{y}_i\boldsymbol{y}_j w_{ij}) \\
&= \sum_i \boldsymbol{y}_i^2 d_{ii} - \sum_{i,j}\boldsymbol{y}_i\boldsymbol{y}_j w_{ij}.
\end{aligned}$$

Since $\boldsymbol{y}_i = \boldsymbol{A}^T\boldsymbol{x}_i$ (Definition 11), then $\gamma(\boldsymbol{A}) = \boldsymbol{A}^T\boldsymbol{X}(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{X}^T\boldsymbol{A} = \boldsymbol{A}^T\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^T\boldsymbol{A}$. □

In terms of Theorem 1, the solution of the following optimization problem can be chosen as $\boldsymbol{A}$:

$$\underset{\boldsymbol{A}}{\operatorname{argmin}}\ \boldsymbol{A}^T\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^T\boldsymbol{A}, \text{s.t. } \boldsymbol{X}^T\boldsymbol{A}\boldsymbol{D}\boldsymbol{A}^T\boldsymbol{X} = \boldsymbol{I}_k. \quad (3)$$

where $\boldsymbol{I}_l$ is an identity matrix of $p \times p$. According to the Spectral Graph Theory [5], the solution of Equation (3) is $\boldsymbol{A} = (\boldsymbol{a}_1, \boldsymbol{a}_1, \cdots, \boldsymbol{a}_p)$, where the ith column $\boldsymbol{a}_i (1 \leq i \leq p)$ is the solution of the following generalized eigenvalue problem corresponding to the ith smallest eigenvalue λ_i:

$$\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^T\boldsymbol{a} = \lambda\boldsymbol{X}\boldsymbol{D}\boldsymbol{X}^T\boldsymbol{a}. \quad (4)$$

5.3.3 Learning Algorithm

The outline of the CTCPT learning algorithm is presented in Algorithm 2. The learning algorithm takes two inputs, the training trajectory dataset S and the dimensionality l of the CTCPS we want to generate. Note that since an LSTFV is 5-dimensional, p can be any nonzero value less than 5. The later experiment shows that $p = 2$ is the best choice in practice.

ALGORITHM 2: *LearningCTCPT*(S, p)

Input: Historical trajectory dataset S, dimensionality of CTCPS p.
Output: Transformation matrix $\boldsymbol{A}$.
Build weight matrix $\boldsymbol{W}$ of S according to Definition 12;
Compute the solution $(\boldsymbol{a}_1, \boldsymbol{a}_1, \cdots, \boldsymbol{a}_p)$ of Equation (4), where $\boldsymbol{A} = \boldsymbol{a}_i (1 \leq i \leq p)$ is in ascending order by its corresponding eigenvalue;

6. HIDDEN TRAJECTORY RECONSTRUCTION

The algorithm for Hidden Trajectory Reconstruction (HTR) is presented in Algorithm 3. HTR first extracts the LSTFVs of the input spatial-temporal points from the TF-Tensor and applies the CTCPT to them, then groups the transformed points by Fuzzy c-Means algorithm, and finally reconstructs the hidden trajectories from the clusters.

7. EXPERIMENTAL EVALUATION

7.1 Settings

Datasets The first real dataset is Geolife which contains over 17K GPS trajectories collected in Beijing (BJ), China,

ALGORITHM 3: $HTR(\mathcal{T}, \boldsymbol{A}, Q)$

Input: TF-Tensor $\mathcal{T}$, CTCPT matrix $\boldsymbol{A}$, collection of separate spatial-temporal points Q, hidden trajectory number K.
Output: Set of hidden trajectories S_h.

```
S_h = {};
for each q_i in Q do
    x_i = T(q_i.τ, q_i.σ, :);
    y_i = A^T x_i;
end
Build Y, where ith column is y_i;
Apply Fuzzy c-Means to Y;
for each cluster C_j, j ∈ {1, ··· , K} do
    Sort the points y_i ∈ C_j in ascending order of their
    timestamps q_i.τ;
    Build trajectory s = ⟨q_1, ··· , q_|C_j|⟩ according to the
    order of y_i, where q_i is the spatial-temporal point
    whose LSTFV x_i corresponds to y_i, 1 ≤ i ≤ |C_j|;
    S_h = S_h ∪ {s};
end
```

during a period of four years [25]. The second real dataset contains over 10K GPS trajectories of approximately 500 taxis collected over 30 days in San Francisco (SF) [15]. Each dataset is split into two parts in order of time. We use the earlier 80% of each dataset as the training data for PICAST and CTCPT learning, and the later 20% as the test data in which the trajectory ID information is removed.

Baselines To evaluate the precision of PICAST, we compare it with four methods for inferring missing values: (1) AVR infers a missing value by averaging the non-zero values of all the entries within the same region; (2) AVT infers a missing value by averaging the non-zero values of all the entries in the same time slot; (3) Linear Regression (LR) infers missing values by the interpolated values that are modelled by a linear model; (4) The Context-Aware Tensor Decomposition (CATD) proposed by Y. Wang *et al.* [20]. To verify the efficiency of PICAST, we compare it with its Serial variant (SCAST) without any optimization and CATD. To evaluate the performance of HTR, we compare it with LiSM (LS) [17], the Kalman Filtering (KF), and TruAlarm (TA) [18].

Metrics for PICAST We measure the precision of PICAST by Mean Square Error (MSE), defined as $MSE = \frac{\sum_i^n (y_i - \hat{y}_i)^2}{n}$, where $\hat{y}_i$ is the estimate of the ith instance $\hat{y}_i$, and n is the number of instances. We evaluate the performance of HTR, in terms of precision and recall. There are two precisions here. One is the **trajectory precision**, which is the proportion of reconstructed true hidden trajectories over all reconstructed trajectories. The other precision is the **point precision**, which is the proportion of the points that are correctly assigned to the trajectory those points belong to over all the points that are assigned to that trajectory. Similarly, there are two recalls. One is the **trajectory recall**, which is the proportion of reconstructed true hidden trajectories over the ground truth. The other is the **point recall**, which is the proportion of the points that are correctly assigned to a trajectory over the ground truth of all the points belonging to that trajectory.

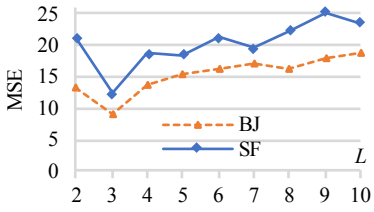

Figure 4: PICAST MSE over Different Target Ranks.

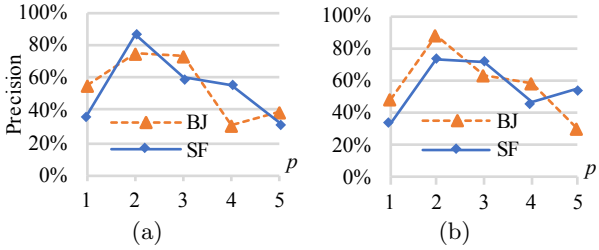

Figure 5: Effect of p on Precision (a) Trajectory Precision (b) Point Precision.

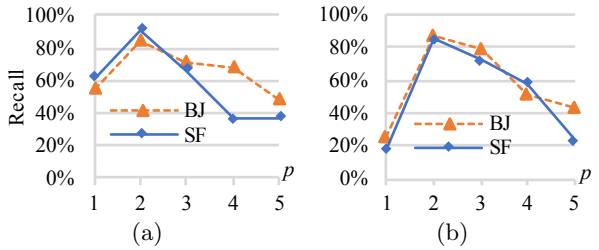

Figure 6: Effect of p on Recall (a) Trajectory Recall (b) Point Recall.

Environments We conduct the experiments on a Hadoop cluster consisting of 3 PCs, each of which is with an Intel Core i7 CPU (2.3 GHz) and 16 GB RAM. The operating system is Ubuntu with Linux kernel 3.18. All the programs are implemented in Python and Matlab 2011b.

7.2 Sensitiveness of Parameters

Figure 4 shows the MSEs of PICAST over $L = 2, 3, 4, 5, 6, 7, 8, 9, 10$. One can see that the MSE of PICAST over the both two real datasets achieves the minimum when $L = 3$, so we choose value 3 as the target rank in the following experiments.

Figure 5 and Figure 6 show the effects of different values of p. It is easy to see that no matter for precision or recall, $p = 2$ is the best choice of CTCPS dimensionality on the datasets we use in the experiments. When $p = 1$, underfitting exists since the spatial-temporal points are actually projected onto a line. On the other hand, when $p > 2$, overfitting happens.

7.3 Performance of PICAST

7.3.1 *MSE of PICAST*

We evaluate the precision of PICAST using the approach proposed in [20]. We build a sparse tensor on each dataset, and randomly remove 30% non-zero entries from the ten-

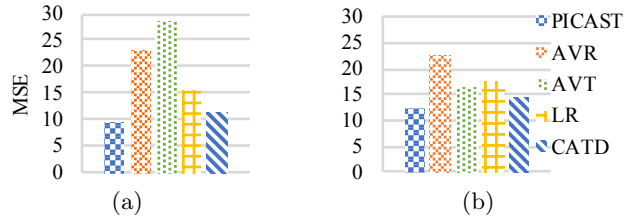

Figure 7: MSE Comparison on (a) BJ and (b) SF.

sors, then infer these entries using PICAST. At last, we compute the MSE by comparing the inferred values with the corresponding original values. As shown in Figure 7, the MSE of PICAST on both datasets is significantly lower than the MSE of the baseline methods. At the same time, we note that PICAST performs better on BJ than on SF, due to the sparser tensor on SF, while PICAST outperforms CATD due to the sparsity control captured in the objective function (Equation (2)).

7.3.2 *Efficiency of PICAST*

We compare PICAST with SCAST and CATD on running time first over synthetic datasets, then over the real datasets, and the result is shown in Figure 8.

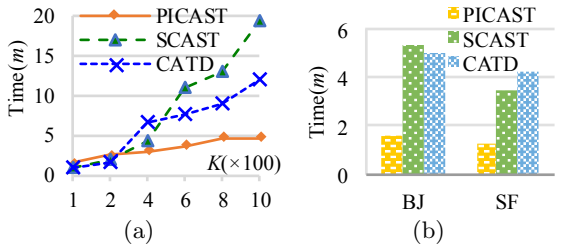

Figure 8: Running Time Comparison

We generate a series of synthetic tensors of $\mathbb{R}^{K\times K\times K}$, where $K = 1, 2, \cdots, 10$. As we can see from Figure 8(a), the PICAST consumes no more 3 minutes of time, while the running time of SCAST and CATD increases sharply as the increase of the synthetic tensor size. Figure 8(b) shows that on the dataset BJ, PICAST, SCAST and CATD take 1.58, 5.31 and 5.01 minutes respectively, and on the dataset SF, 1.33, 3.44 and 4.21 minutes respectively. Clearly, PICAST makes a significant improvement in running time on all datasets.

7.4 Performance of HTR

7.4.1 *Trajectory Precision and Recall*

The trajectory precisions of HTR, LiSM, KF and TA on different datasets are shown in Figure 9. As we can see from Figure 9, the trajectory precision of HTR is significantly higher than those of all the competitors on both datasets, due to HTR's stronger ability to discriminate between true positive cases and false positive cases. Figure 9 shows that on both datasets, HTR achieves a precision of near 98%, which indicates that almost all the trajectories reconstructed by HTR are true hidden trajectories occurring in reality.

The trajectory recalls of HTR, LiSM, KF and TA on different datasets are shown in Figure 10. One can see that

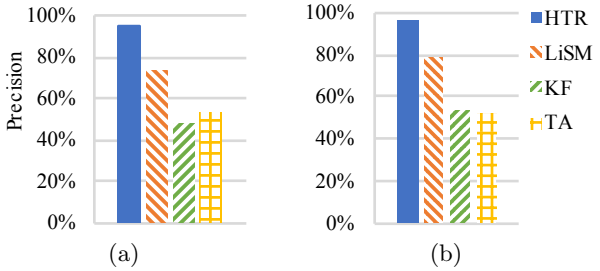

Figure 9: Trajectory Precision on Different Datasets (a) BJ (b) SF.

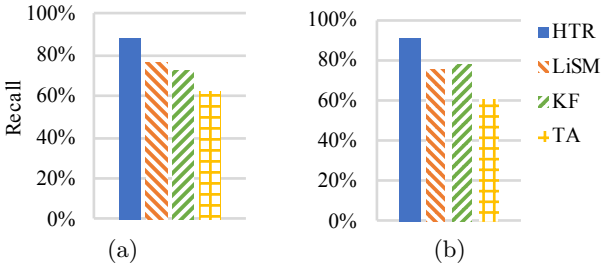

Figure 10: Trajectory Recall on Different Datasets (a) BJ (b) SF.

HTR has an average trajectory recall of about 90% on both datasets, which is higher than those of the alternative approaches. The root cause of such superiority is that HTR takes a completely different strategy to discover candidate trajectories. HTR discovers the candidate trajectories all at once by clustering the transformed points in a CTCPS. In contrast, all the alternative approaches take some sorts of tracking strategies to determine a trajectory. Tracking strategies require accurately determining the start point and the end point of a trajectory, which is hard to achieve in an extremely uncertain situation without any trajectory link information.

7.4.2 *Point Precision and Point Recall*

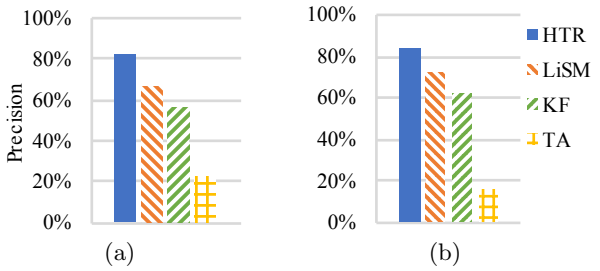

Figure 11: Point Precision on Different Datasets (a) BJ (b) SF.

The point precisions and the point recalls of HTR, LiSM, KF and TA on different datasets are respectively shown in Figure 11 and Figure 12. One can see that HTR outperforms the alternative approaches since the point precision and the point recall of HTR are both over 80%. These results indicate that in sharp contrast to HTR, the tracking methods

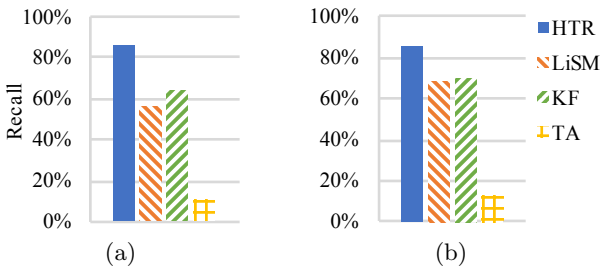

Figure 12: Point Recall on Different Datasets (a) BJ (b) SF.

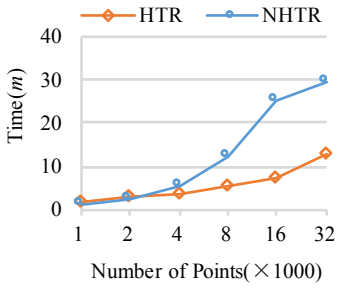

Figure 13: Overall Running Time Comparison.

like LiSM, KF and TA are not suitable for the trajectories each of which spans across a geographical area as wide as a city. Particularly, TA is much worse than the other alternatives as it likely confuses multiple intersecting trajectories due to its tracking strategy of choosing the nearest spatial-temporal point as the next point of a candidate trajectory.

7.4.3 *Overall Efficiency of HTR*

To verify the overall efficiency of HTR, over six synthetic datasets of different sizes we compare its running time with the running time of the Naive HTR (NHTR) which directly factorizes a TF-Tensor without any optimization, and the results are shown in Figure 13. We can draw three conclusions from the results. First, due to PICAST, the overall running time of HTR almost linearly increases with the growth of dataset size, while the running time of NHTR sharply increases when dataset contains more than 8,000 points. Second, when dataset size is less than 2,000, HTR performs slightly poorer than NHTR. This is because on small datasets, the contribution of PICAST to time saving is disproportionately less than the extra overhead incurred by its divide-and-conquer strategy. Third, the running time of no matter HTR or NHTR is increasing with the number of points, because more points are more likely distributed over more regions which results in a larger size of the TF-Tensor.

8. RELATED WORK

Moving Object Detecting and Tacking Recently, moving object detecting and tracking have been attracting much attention of the researchers of wireless networks [1, 12, 16, 18, 17] and trajectory mining [24, 4, 23, 11, 2]. Arora *et al.* [1] propose a detection model based on acoustic and magnetic sensors. Ozdemir *et al.* [12] propose a particle filtering-based algorithm to detect intruders in cyber-physical space. Sheng *et al.* [16] propose an algorithm using maximum likelihood estimation. Tang *et al.* [18] propose the Tru-Alarm filtering algorithm, which uses a nearest-neighboring strategy to distinguish different moving objects. To overcome the defect of Tru-Alarm, Tang *et al.* [17] further propose cone-model based algorithm LiSM, which builds a watching network from untrustworthy sensor data and estimates the appearances of moving objects by using the link information of the watching network. Zheng *et al.* [24] propose an algorithm to rebuild uncertain piece of a low sampled trajectory by using road network information. Cheng *et al.* [4] propose a probabilistic approach to track uncertain moving objects. Zheng *et al.* [23] investigate the problem of probabilistic queries on uncertain trajectories on road networks. Larusso *et al.* [11] propose a Kalman Filter-based model to update the current location of uncertain mobile objects with a probability bounds of errors. Banerjee *et al.* [2] propose a technique called InferTra to infer an uncertain trajectory by generating a summary route from partial observations. However, these methods cannot serve our goal since they cannot tackle the extremely uncertain situations where no trajectory link and ID information are available.

Uncertain Trajectory Modeling Hornsby *et al.* [7] propose a cone model based on the fact that the movement range of an object is constrained by maximal speed between two observations [21]. Tang *et al.* [17] propose a cone-model based algorithm LiSM, which builds a watching network from untrustworthy sensor data and estimates the appearances of moving objects by using the link information of the watching network. Trajcevski *et al.* [19] propose a cylinder model in which an uncertain trajectory is represented as a cylinder made up of horizontal circles at different time. Pelekis *et al.* [13] present a grid model of uncertain trajectories, which partitions a given space into a set of disjoint cells. The models of uncertain trajectories on road networks have different concerns, where the context of uncertainty metrics turns to a graph. Kuijpers *et al.* [10] develop the cone model under the constraint of road networks, where only a subset of the edges of the road network are taken into consideration for computing an uncertain area. However, these traditional models are often built for small and dense data, not for huge sparse data like the datasets we deal with in this paper.

9. CONCLUSION

In this paper, we propose a novel approach for the Hidden Trajectory Reconstruction, called HTR. At first, we model a spatial-temporal point by an Latent Spatial-Temporal Feature Vector (LSTFV) based on which a Temporal Feature Tensor (TF-Tensor) is built. To overcome the sparsity of the TF-Tensor, we propose an algorithm called Parallel Iterative Collaborative Approximation of Sparse Tensor (PICAST) to approximate the TF-Tensor by decomposing it into a tensor product of a low-rank core identity tensor and three dense factor matrices. We also propose a Cross-Temporal Connectivity Preserving Transformation (CTCPT), to map the LSTFVs of the input spatial-temporal points to an intrinsic space, called Cross-Temporal Connectivity Preserving Space (CTCPS), in which the hidden trajectories can be reconstructed from fuzzy clusters generated. The results of the extensive experiments conducted on real datasets verify the effectiveness and efficiency of our proposed HTR.

Acknowledgment

This work is supported by National Science Foundation of China through grant 61173099, and in part by NSF through grants III-1526499 and CNS-1115234.

10. REFERENCES

[1] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, et al. A line in the sand: a wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5):605–634, 2004.

[2] P. Banerjee, S. Ranu, and S. Raghavan. Inferring uncertain trajectories from partial observations. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 30–39. IEEE, 2014.

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, pages 585–591, 2001.

[4] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Querying imprecise data in moving object environments. *Knowledge and Data Engineering, IEEE Transactions on*, 16(9):1112–1127, 2004.

[5] F. R. Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

[6] S. Gandy, B. Recht, and I. Yamada. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27(2):025010, 2011.

[7] K. Hornsby and M. J. Egenhofer. Modeling moving objects over multiple granularities. *Annals of Mathematics and Artificial Intelligence*, 36(1-2):177–194, 2002.

[8] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010.

[9] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[10] B. Kuijpers, H. J. Miller, T. Neutens, and W. Othman. Anchor uncertainty and space-time prisms on road networks. *International Journal of Geographical Information Science*, 24(8):1223–1248, 2010.

[11] N. D. Larusso and A. Singh. Efficient tracking and querying for coordinated uncertain mobile objects. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 182–193. IEEE, 2013.

[12] O. Ozdemir, R. Niu, and P. K. Varshney. Tracking in wireless sensor networks using particle filtering: Physical layer considerations. *Signal Processing, IEEE Transactions on*, 57(5):1987–1999, 2009.

[13] N. Pelekis, I. Kopanakis, E. E. Kotsifakos, E. Frentzos, and Y. Theodoridis. Clustering trajectories of moving objects in an uncertain world. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 417–427. IEEE, 2009.

[14] A. H. Phan and A. Cichocki. Parafac algorithms for large-scale problems. *Neurocomputing*, 74(11):1970–1984, 2011.

[15] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser. A parsimonious model of mobile partitioned networks with clustering. In *Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International*, pages 1–10. IEEE, 2009.

[16] X. Sheng and Y.-H. Hu. Maximum likelihood multiple-source localization using acoustic energy measurements with wireless sensor networks. *Signal Processing, IEEE Transactions on*, 53(1):44–53, 2005.

[17] L.-A. Tang, X. Yu, Q. Gu, J. Han, A. Leung, and T. La Porta. Mining lines in the sand: On trajectory discovery from untrustworthy data in cyber-physical system. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 410–418. ACM, 2013.

[18] L.-A. Tang, X. Yu, S. Kim, J. Han, C.-C. Hung, and W.-C. Peng. Tru-alarm: Trustworthiness analysis of sensor networks in cyber-physical systems. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 1079–1084. IEEE, 2010.

[19] G. Trajcevski, O. Wolfson, F. Zhang, and S. Chamberlain. The geometry of uncertainty in moving objects databases. In *Advances in Database Technology—EDBT 2002*, pages 233–250. Springer, 2002.

[20] Y. Wang, Y. Zheng, and Y. Xue. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 25–34. ACM, 2014.

[21] S. Winter and Z.-C. Yin. The elements of probabilistic time geography. *GeoInformatica*, 15(3):417–434, 2011.

[22] N. Yang, X. Kong, F. Wang, and P. S. Yu. When and where: Predicting human movements based on social spatial-temporal events. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 515–523. SIAM, 2014.

[23] K. Zheng, G. Trajcevski, X. Zhou, and P. Scheuermann. Probabilistic range queries for uncertain trajectories on road networks. In *Proceedings of the 14th International Conference on Extending Database Technology*, pages 283–294. ACM, 2011.

[24] K. Zheng, Y. Zheng, X. Xie, and X. Zhou. Reducing uncertainty of low-sampling-rate trajectories. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 1144–1155. IEEE, 2012.

[25] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining interesting locations and travel sequences from gps trajectories. In *Proceedings of the 18th international conference on World wide web*, pages 791–800. ACM, 2009.