ITAR: A Method for Indoor RFID Trajectory Automatic Recovery

Ziwen Cao[1,2], Siye Wang[1,2,3(✉)], Degang Sun[2], Yanfang Zhang[1], Yue Feng[1,2], and Shang Jiang[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{caoziwen,wangsiye,zhangyanfang,fengyue,jiangshang}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
sundegang@iie.ac.cn
[3] School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China

Abstract. With the increasing popularity of Radio Frequency Identification (RFID) technology, indoor applications based on RFID trajectory data analysis are becoming more and more extensive, such as personnel location, tracking, and heat map analysis. The effectiveness of indoor applications relies greatly on high-quality trajectory data. However, due to the constraints of the device and environment, RFID readers will miss reading data in real-world practice, which leads to a large number of indoor trajectories that are incomplete. To enhance trajectory data and support indoor applications more efficiently, many trajectory recovery methods to infer trajectories in free space have been proposed. However, existing methods cannot achieve automated inference and have low accuracy in inferring indoor trajectories. In this paper, we propose an Indoor Trajectory Automatic Recovery framework, ITAR, to recover missing points in indoor trajectories. ITAR adopts a sequence-to-sequence learning architecture to generate complete trajectories. We first construct a directed graph for each trajectory and use a graph neural network to capture complex location transition patterns. Then, we propose a multi-head attention mechanism to capture long-term correlations among trajectory points to improve performance. We conduct extensive experiments on synthetic and real datasets, and the results show that ITAR is superior in performance and robustness.

Keywords: Trajectory recovery · Sequence-to-sequence model · Radio frequency identification · Graph neural network

1 Introduction

Radio Frequency Identification (RFID) technology modernizes indoor object tracking and monitoring systems. This technology relies on two devices: tags (which can emit radio signals encoding identifying information) and readers

H. Gao et al. (Eds.): CollaborateCom 2022, LNICST 461, pp. 400–418, 2022.
https://doi.org/10.1007/978-3-031-24386-8_22

(which detect the signals emitted by tags). Today, most RFID applications use low-cost passive RFID tags, which are preferred for their small size, low price, and low energy requirements. When tags attached to people or objects appear within the reader's detection range, the deployed RFID readers detect these tags and the moving objects to which they are attached. Therefore, RFID technology realizes the identification, location, and tracking of moving objects by attaching tags on different items, such as commodities, equipment, people, etc.

Effective management of indoor RFID tracking data opens new doors for various applications that range from monitoring to analysis of indoor moving objects. This reason for monitoring is that knowing the trajectories of moving assets is helpful in several applications, such as behavior and security analyses. The reasons for such monitoring are manifold, ranging from collecting data to support the behavior analysis over the monitored entities to ensuring security for people and assets. For instance, information on the trajectory followed by monitored people can be used to prevent or look into crimes, and detect dangerous or suspicious situations. Furthermore, information on the trajectory followed by a visitor inside a museum can provide very detailed context-aware information during visiting [13].

However, the unreliability of the raw data captured by the reader is a significant factor hindering the development of such applications. Under normal circumstances, the missing rate of RFID data is between 30%–40% [10]. RFID data missed come from different sources. The read events are often missed due to reader detection capabilities, RFID tag quality, and environmental constraints [4]. Especially in indoor environments where the signal is reflected or blocked by different entities, which leads to readers missing reading tags nearby, that should be detected. Due to the limitation of the acquisition conditions of indoor trajectory data, it is often difficult to obtain a relatively complete record for indoor trajectory. Therefore, it is crucial to reconstruct RFID based indoor trajectories by estimating missing or unobserved locations.

One of the most common solutions to this problem is to impute the missing value by comparing the incomplete trajectories with possible complete trajectories to find the most similar trajectories [12]. Their performance is acceptable when only a small percentage of the locations are missing due to limited movement during a short period. However, their performance degrades significantly in highly sparse scenarios since they fail to model complex mobility regularities effectively. Another line of study is to model the regularity of user transitions among locations to impute missing locations based on the highest transition probability of observed locations [1–3]. However, this strategy is still insufficient. Because the observed RFID trajectory data are not uniform in time, so the transition regularity cannot be inferred for those persistently unobserved locations. None of the above methods can capture the complex sequential dependencies or global data correlations well in indoor environments.

With the popularity of neural networks, deep learning provides a promising computational framework for solving complex trajectory recovery tasks. Many studies have attempted to exploit excellent modeling capacity to better learn effective characteristics from trajectory data. Especially, Recurrent Neural

Networks (RNN) are widely used to model sequential trajectory data [26,33]. Although these studies have improved the ability to model complex sequential transition patterns to some extent, they only focus on the next step or short-term location prediction. However, a framework that can better simulate movement patterns is needed due to the high uncertainty between two consecutive recordings in indoor trajectories. In addition, traditional methods are often based on the hypothesis that the missing location is known [8,19], while we usually cannot directly predict the missing trajectory location, which requires that our method should be able to use the global information from the entire trajectory to detect and impute missing trajectories automatically. Therefore, it is difficult to directly apply existing neural network-based trajectory models to indoor space trajectory recovery tasks.

To address the above difficulties, we propose a new Indoor Trajectory Automatic Recovery model named ITAR. Our model achieves automatic imputation of incomplete indoor trajectories by adopting a classical sequence-to-sequence generation framework (i.e., Seq2Seq). In order to effectively capture the complex sequential dependencies between locations in the indoor environment, we adopt a gated graph neural network to encode incomplete trajectories, and adopt a gated recurrent neural network to decode the complete trajectories. Furthermore, to effectively capture the correlation between trajectory points, we employ a multi-head attention mechanism to improve the model performance. With the attention mechanism, our model is able to characterize long-range correlations among trajectory points. In this manner, our model finally enables the automatic imputing of incomplete trajectories and modeling of complex transition patterns. Overall, our main contributions can be summarized as follows:

1. We propose a novel indoor trajectory recovery model for automated RFID trajectory recovery. To the best of our knowledge, this is the first time that deep learning has been applied to the RFID trajectory data imputation research task.
2. We leverage graph neural networks for complex location transition patterns modeling and employ a multi-head attention mechanism for capturing long-range dependencies of indoor trajectory points.
3. We conduct comprehensive experimental studies using both real and synthetic data. Extensive results demonstrate the superiority of the proposed model in terms of effectiveness and robustness.

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 outlines the relevant definitions of the RFID trajectory data. Section 4 describes the trajectory recovery model in detail. Section 5 evaluates the performance of ITAR. Section 6 concludes this paper.

2 Related Work

2.1 RFID Data Imputing

There has been considerable research interest in managing incomplete RFID trajectory data [4,29]. Due to the detection ability of the reader, the quality of

the RFID tag, and the environmental limitations, the phenomenon of missing reading is unavoidable in the process of RFID data collection. Therefore, people have proposed many solutions to impute the RFID trajectory data. Gu et al. [11] first proposed to utilize the grouped trajectory information of monitored objects for RFID data imputation. The trajectory information of monitored objects is implied among massive RFID data. Hu et al. [12] proposed a motion retrospective-based filling algorithm for RFID trajectory data. The algorithm maintains a tracked trajectory event tree based on historical data. Zhao et al. [32] proposed a probabilistic model to clean RFID data for object tracking, which utilizes a Bayesian inference-based algorithm to handle RFID missed reads efficiently. Baba et al. [2,3] proposed a graph model-based RFID data cleaning method, which uses indoor deployment graphs to capture information about the indoor environment and deployed readers and proposes a probabilistic distance perception map to identify false negatives and recover missing information in indoor RFID tracking data. Fazzinga et al. [5,6] proposed a probabilistic framework and a grid-based filtering scheme to clean RFID data. Their solution uses the integrity constraints implied by maps to reduce the inherent uncertainty in trajectory data collected for RFID-monitored objects. Baba et al. [1] proposed a multivariate hidden Markov model (IR-MHMM) to capture and recover the RFID missing data in indoor environments. The method only needs to acquire a small amount of information about the RFID deployment and can learn relevant knowledge from the raw RFID data to impute the indoor trajectory.

2.2 Trajectory Recovery

The traceability of RFID data inspires us to further understand the related research on trajectory recovery [7]. Recovering missing values for trajectories has been an important problem for a long time. Traditional research mainly focuses on mining frequent human movement patterns to recover relevant data. Models such as Markov model [21] and Apriori [17] have been extensively studied. In recent years, deep learning-based models have achieved good performance. Recurrent neural networks (RNNs) and their variants, long short-term memory (LSTM), have been widely used for trajectory data modeling [30]. Existing work can be mainly divided into three categories. The first approach focuses on human mobility recovery using mobility prediction models [8,14]. However, their performance declines in highly sparse scenarios because they only exploit historical information before the missing location and fail to model the spatio-temporal dependencies between the missing location and the locations visited afterward. The second approach is to employ a model-based approach [19,27,28], which captures the multi-level and changing movement patterns of humans. However, these methods are based on the assumption that the historical trajectory of personal movement is known and is not applicable in complex transition scenarios. Recent work in this area treats trajectory recovery problems as time series data recovery problems [16,18]. However, this model fails to capture the complex transition patterns among locations.

Overall, the above methods are not suitable for RFID-based trajectory recovery in indoor scenes. Most of the existing deep learning-based models focus on recovering trajectory data in outdoor GPS-based scenes and ignore trajectory data imputing complex transition patterns in indoor scenes. In contrast, we propose an end-to-end trajectory recovery model based on graph neural networks, which is capable of capturing complex transition patterns and automatically recovering incomplete trajectories.

3 Overview

3.1 Preliminaries

Definition 1: (Trajectory Point). Trajectory points refer to track monitoring points deployed in indoor scenes. RFID reader devices are deployed at each trajectory point. When the RFID reader detects the RFID tag attached to the object, it will record the tag's ID, location, time and other information. Usually, researchers use the triple $<Tagid, Loc, Time>$ to represent the detected object information. Table 1 shows the data records read by the RFID reader device, where $TagID$ is the unique identifier of the tag, representing the detected object. Loc is represented by the number of the reader antenna. Different antennas are distributed at different locations in the room, representing where objects are detected. $Time$ is the timestamp, indicating the time the reader read the tag.

Table 1. RFID trajectory data

Tag ID	Loc	Recording time
1015000A0D3E9368E58F124E	003	2022/7/30 14:20:04
101500A99E4955022988812B	001	2022/7/30 14:23:06
1015002264B2FFBE08B0027A	002	2022/7/30 14:28:12
1015002E496B6C2C93A7794C	001	2022/7/30 14:40:32
1015002E496B6C2C93A7794C	001	2022/7/30 14:42:08
...	...	...

Definition 2: (Road Network). The indoor road network is an undirected graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_K\}$ represents the trajectory points in the indoor scene, $E = \{e_1, e_2, \ldots, e_l\}$ refers to a set of edges that indicate whether the two detection locations are directly reachable in the actual environment. Figure 1 is a monitoring area in an actual system. The topological map above is the road network map from the monitoring area.

Definition 3: (Trajectory). A trajectory T_{id} can be defined as a chronological sequence of locations of a tag ID within a given area, i.e., $T_{id} = \{l^1_{id}, l^2_{id}, \ldots, l^m_{id}\}$, where each l^i_{id} represents a location information in RFID triple $<TagID, Loc, Time>$. Note that if a trajectory point should have been passed but was not observed, the location is named a missing location.

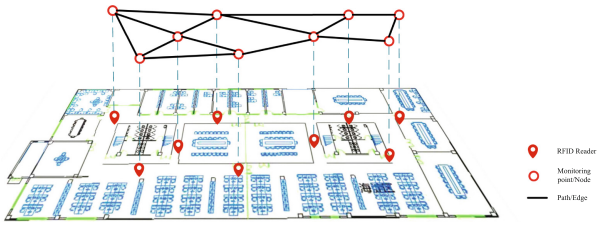

Fig. 1. Indoor road network.

3.2 Problem Definition

Given an incomplete trajectory $T_{id} = \{l_{id}^1, l_{id}^2, \ldots, l_{id}^m\}$ for an arbitrary tag ID, we aim to recover the complete trajectory $\tilde{T}_{id} = \{l_{id}^1, l_{id}^i, l_{id}^j, \ldots, l_{id}^n\}$, where $l_{id}^m = l_{id}^n$. That is to say, for each incomplete trajectory T_{id}, we attempt to infer the missing locations in it automatically with knowing the start and end trajectory points.

4 Methodology

To solve the indoor trajectory recovery problem, we are inspired by the classic Seq2Seq model. The trajectory recovery problem is similar to the machine translation problem, where the incomplete trajectory T_{id} can be treated as the original sentence, and the imputed trajectory $\tilde{T}_{id}$ can be regarded as the translated sentence. Therefore, the Seq2Seq structure can potentially be used to solve the trajectory recovery problem. Our model consists of three main parts. The first part is a sequence-to-sequence (Seq2Seq) neural network model [20], which generates trajectory points incrementally. The second part is a multi-head attention mechanism to capture the correlation between trajectory points. Our attention mechanism considers the dependencies between locations in a global perspective. We use the cross-entropy loss function for model training in the third part.

In this section, we specify ITAR in an asymptotic manner. In Sect. 4.1, we firstly introduce the basic structure of a Seq2Seq-based model for solving the trajectory automatic recovery problem. In Sect. 4.2, we incorporate multi-head attention into Seq2Seq. Then, Sect. 4.3 trains the model by using the cross-entropy loss function. In the following, we elaborate them in details.

4.1 Seq2seq Model

As illustrated in Fig. 2, ITAR is composed of an encoder and a decoder. The encoder learns the spatial dependencies among the trajectory points, while the decoder iteratively predicts the trajectory points using the previous output as the input vector.

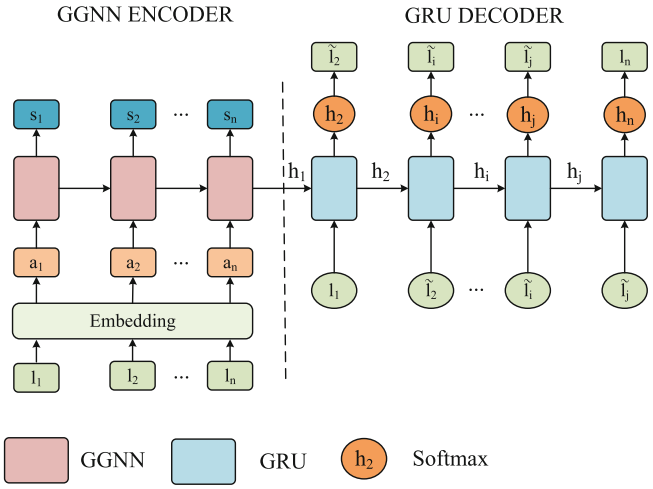

Fig. 2. Structure of ITAR.

Encoder. Previous studies show that gated GNN (ggnn) is able to capture complex transition patterns among nodes, which makes gated GNN suitable for our problem [19]. We process each trajectory separately to capture the complex transition patterns hidden in each trajectory. Specifically, we first construct a directed graph for each trajectory, and then apply gated GNN on each directed graph, which incorporates transition patterns into the encoder.

First, the graph neural network encoder constructs a graph for each incomplete trajectory. Given a trajectory $T_{id} : \{l_1, l_2, \ldots, l_n\}$, we treat each location l_i as a graph node v_i, (l_{i-1}, l_i) is considered as an edge between nodes, where the direction of the edge is from l_{i-1} to l_i. Specifically, let $M_I, M_O \in \mathbb{R}^{d \times d}$ denote the weighted connections of incoming and outgoing edges in the trajectory graph. For example, considering a trajectory $T_{id} : \{l_1, l_2, l_4, l_3, l_2\}$, the corresponding graph and matrix $(i.e, M_I, M_O)$ are shown in Fig. 3. We assign a normalized weight to each edge, which is the number of occurrences of the edge divided by the outdegree of that edge's start node. ITAR updates the graph structure by tracking the movement of tags between different locations.

Next, we describe how to embed the trajectory graph into the location vector. We first embed each location $l \in \mathbb{L}$ into an unified low-dimensional latent space s_l, and the location vector $s_l \in \mathbb{R}^d$ denotes a d-dimensional latent space location vector of l. For each location l of the trajectory, a_v extracts contextual information about the neighborhood of location l in the trajectory graph, which can be formalized as:

$$\begin{aligned} \mathbf{a}_v = \text{Concat} \left(\mathbf{M}_I^v \left([\mathbf{s}_1, \ldots, \mathbf{s}_N] \, \mathbf{W}_I^a + \mathbf{b}_I \right) \right. \\ \left. \mathbf{M}_O^v \left([\mathbf{s}_1, \ldots, \mathbf{s}_N] \, \mathbf{W}_O^a + \mathbf{b}_O \right) \right) \end{aligned} \tag{1}$$

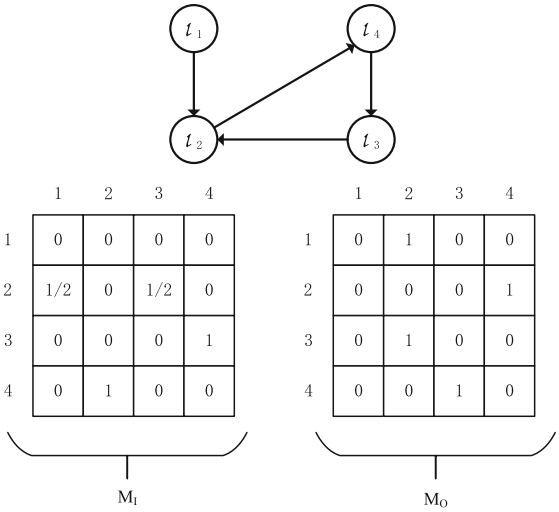

Fig. 3. An example of a trajectory graph structure.

where $\mathbf{W}_I^a, \mathbf{W}_O^a \in \mathbb{R}^{d\times d}$ are variable parameters. $\mathbf{b}_I, \mathbf{b}_O \in \mathbb{R}^d$ are the bias vectors. N is the number of unique locations in the trajectory. $\mathbf{M}_I^v, \mathbf{M}_O^v$ are the incoming and outgoing matrices of the node v in the graph corresponding to the location l.

Finally, due to the spatial dependencies among trajectory points, we employ a gated graph neural network as an encoder to obtain context vectors for incomplete trajectories. The gated graph neural network sequentially updates the hidden state by introducing an update gate $\mathbf{z}_v$ and a reset gate $\mathbf{r}_v$, $\mathbf{z}_v$ and $\mathbf{r}_v$ decide which information to keep and discard, respectively. After that, we construct the candidate state $\tilde{\mathbf{s}}^v$ from the previous state $\mathbf{s}_{v-1}$, the current state $\mathbf{a}_v$ and the reset gate $\mathbf{r}_v$, as described in the Eq. 2. Under the control of the update gate $\mathbf{z}_v$, we combine the previously hidden state $\mathbf{s}_{v-1}$ with the candidate state $\tilde{\mathbf{s}}_v$ to get the final state $\mathbf{s}_v$.

$$\begin{aligned}
\mathbf{z}_v &= \sigma\left(\mathbf{W}_z\mathbf{a}_v + \mathbf{U}_z\mathbf{s}_{v-1}\right) \\
\mathbf{r}_v &= \sigma\left(\mathbf{W}_r\mathbf{a}_v + \mathbf{U}_r\mathbf{s}_{v-1}\right) \\
\tilde{\mathbf{s}}_v &= \tanh\left(\mathbf{W}_h\mathbf{a}_v + \mathbf{U}_h\left(\mathbf{r}_v \odot \mathbf{s}_{v-1}\right)\right) \\
\mathbf{s}_v &= \left(1 - z_v\right) \odot \mathbf{s}_{v-1} + z_v \odot \tilde{\mathbf{s}}_v
\end{aligned} \tag{2}$$

where $\mathbf{W}_z, \mathbf{W}_r, \mathbf{W}_h \in \mathbb{R}^{d\times 2d}$, $\mathbf{U}_z, \mathbf{U}_r, \mathbf{U}_h \in \mathbb{R}^{d\times d}$ are learnable parameters. σ denotes the sigmoid function and $\odot$ represents element-wise multiplication. $\mathbf{s}_v$ is the final latent location vector with transition-aware after gated graph neural network. To simplify, for getting the context vector of incomplete trajectory, the encoder derives the hidden state $\mathbf{s}_v$ as:

$$\mathbf{s}_v = GGNN(\mathbf{a}_v, \mathbf{s}_{v-1}) \tag{3}$$

where the last state $\mathbf{s}_n$ will be considered as the context vector as well as the initial hidden state for the decoder.

Decoder. The decoder is used to recover incomplete trajectories to complete trajectories. In ITAR, we utilize gated recurrent unit (GRU) for trajectory decoding because the network architecture of GRU is close to the gated graph neural network. GRU also sequentially updates the hidden state by introducing an update gate and a reset gate, as shown in Eq. 2. In contrast, GRU does not need to compute trajectory graphs to obtain embeddings of spatial transition information. At each time step, the GRU decoder decodes the input (the location generated at the previous time step) and the encoded intermediate state into the output location at the current time step. The decoder expresses the hidden state obtained by each round of decoding as $\mathbf{h}_j$ and derives it as:

$$\mathbf{h}_j = GRU(\mathbf{l}_{j-1}, \mathbf{h}_{j-1}) \tag{4}$$

where $\mathbf{l}_{j-1}$ represents the location generated at the previous time step and $\mathbf{h}_{j-1}$ represents the intermediate hidden state. Once we get the hidden state $\mathbf{h}_j$ from the decoder, we use the $softmax$ function to predict the location l_j, where w_l is the $l-th$ column vector from a trainable parameter matrix W_l.

$$\Pr\left(l \mid \boldsymbol{h}_i\right) = \frac{\exp\left(\boldsymbol{h}_i^{\top} \cdot \boldsymbol{w}_l\right)}{\sum_{l' \in \mathcal{L}} \exp\left(\boldsymbol{h}_i^{\top} \cdot \boldsymbol{w}_{l'}\right)} \tag{5}$$

Furthermore, compared to other seq2seq-based applications applied to trajectory recovery, our task is unique in that the length of the recovered trajectory is unclear. RFID-based indoor trajectories are usually not sampled uniformly in time, making the number of padding points problematic. Fortunately, we usually know the start and end locations of the target trajectory. Therefore, in the training process, we determine whether the location has reached the end location of the target trajectory whenever the current location is predicted. Once the conditions are met, the task of trajectory recovery ends.

4.2 Multi-head Attention

The traditional SeqSeq structure is not ideal for long trajectory imputing. As the trajectory sequence grows, the location information in front of the trajectory sequence will be seriously lost. Even though many papers propose some tricks, such as inputting sentences in reverse order (such as bidirectional LSTM model [31]). However, the improvement in model performance is not apparent. Inspired by the widely used attention mechanism in natural language translation [22], we introduce an attention mechanism into the decoder to model the global correlation of incomplete trajectories. On this basis, we further extend the attention mechanism into a multi-head attention mechanism.

The goal of the attention mechanism is to automatically extract those parts of the incomplete trajectory relevant to the target recovery trajectory and to represent the implicit relationship between the output and the input by generating a context vector $\mathbf{c}$. Multi-head attention utilizes multiple query vectors to compute and select multiple dimensions from the input information in parallel. Each independent attention head focuses on different parts of the input information, and then concentrates to get the final context $\mathbf{c}_j$, which is expressed as:

$$\mathbf{c}_j = \mathbf{c}_j^{(1)} \| \mathbf{c}_j^{(2)} \| \ldots \| \mathbf{c}_j^{(H)} \tag{6}$$

Each context vector $\mathbf{c}_j^{(h)}$ is calculated from the weighted sum of all output vectors s from the encoder:

$$\mathbf{c}_j^{(h)} = \sum_{i=1}^{n} \alpha_{j,v}^{(h)} \mathbf{s}_v \tag{7}$$

where $\alpha_{j,v}^{(h)}$ represents the similarity between the query vector (i.e. current hidden state in the decoder) and the key vector (i.e. output from the encoder), which is formulated as:

$$\begin{aligned} \alpha_{j,v}^{(h)} &= \frac{\exp\left(u_{j,v}^{(h)}\right)}{\sum_{v'=1}^{N} \exp\left(u_{j,v'}^{(h)}\right)} \\ u_{j,v}^{(h)} &= \mathbf{v}^{(h)\top} \cdot \tanh\left(\mathbf{W}_{\mathbf{h}}^{(h)} \mathbf{h}_j + \mathbf{W}_{\mathbf{s}}^{(h)} \mathbf{s}_v\right) \end{aligned} \tag{8}$$

where $\mathbf{W}_{\mathbf{h}}^{(h)}, \mathbf{W}_{\mathbf{s}}^{(h)}, v^{(h)}$ are learnable parameters, $\mathbf{h}_j$ denotes the hidden location status from the decoder and $\mathbf{s}_v$ is the output from the encoder.

Therefore, the hidden state $\mathbf{h}_j$ in the decoder is updated to:

$$\mathbf{h}_j = GRU(\mathbf{h}_{j-1}, \mathbf{l}_{j-1}, \mathbf{c}_j) \tag{9}$$

4.3 Model Training

Finally, we elaborate the training process of the end-to-end trained model. We adopt cross-entropy as the loss function:

$$\mathcal{L}(\theta) = -\sum_{(T,\tilde{T}) \in \mathcal{D}} \sum_{j=1}^{|\tilde{T}|} \sum_{i=1}^{\mathbb{L}} l_i^j \log\left(\tilde{l}_i^j\right) \tag{10}$$

where T is the incomplete trajectory, $\tilde{T}$ is the complete target trajectory, $|\hat{T}$— is the length of the $\tilde{T}$ trajectory. L is the set of indoor trajectory points, which represents the category of the output location. l is the ground truth of the target trajectory, and $\tilde{l}_j$ is the predicted location. $\mathcal{D}$ denotes a dataset consisting of incomplete trajectories T and complete trajectories $\tilde{T}$.

During the training process, we apply AdamW stochastic gradient descent to update the parameters θ [15]. First, we construct a suitable training set including

incomplete trajectories T and complete trajectories $\tilde{T}$. Then, we initialize the training model parameters θ and randomly shuffle the training set. Finally, we update the ITAR parameter θ using the Eq. 10, where η is the training step size. The Algorithm 1 illustrates the training process of the ITAR model. In addition, our model is implemented in Python and Pytorch. All models are done on Apple computers with M1 chips.

Algorithm 1. ITAR Training.

Input: Trajectories T, $\tilde{T}$, max iteration *epochs*, batch size *batch*
Output: Trained Model θ
Construct training instances T and $\tilde{T}$.
initialize the model parameters θ.
for $i \in \{1, 2, 3, \ldots, epochs\}$ **do**
  Shuffle the training instances D into mini-batches
  for $j \in \{1, 2, 3, \ldots, batch\}$ **do**
    Calculate gradient $\nabla\mathcal{L}(\theta)$ using Eq. 10.
    Update $\theta \leftarrow \theta - \eta\nabla\mathcal{L}(\theta)$.
  end for
end for

5 Experiment

In this section, we first introduce the relevant settings of the experiments and then perform a performance comparison and analysis of the experimental results.

5.1 Experimental Setup

Construction of the Evaluation Set. In this experiment, we use two datasets to demonstrate the effectiveness of our proposed algorithm, including a synthetic dataset and an RFID tracking dataset. To facilitate trajectory recovery, we preprocess the raw RFID data. First, we use the redundant filtering algorithm to remove duplicate points and outliers [24], and then use environmental constraints to impute some missing data to form the ground truth trajectory. Finally, we filter out trajectories with lengths less than 7. Table 2 summarizes the final detailed static data for the two RFID trajectory datasets.

– **Synthetic:** We simulate the trajectory of objects in indoor space, record the sampling data of objects at different trajectory points, and synthesize RFID trajectory data accordingly. We use the original trajectory data to represent the ground truth trajectory of the object and form incomplete trajectories by masking different proportions of the data. Since the synthetic data is a trajectory formed by simulation, the model will perform better.

– **RFID Tracking:** We use data from an RFID-based person tracking system as the dataset for this experiment. This dataset was collected at a product expo held in 2018 [9]. The dataset has about 400,000 pieces of data generated by about 10,000 tags over three days.

Table 2. Basic statistics of datasets.

Dataset	Duration	Average Traj length	Loctions	Traj pairs
Synthetic	7 days	11	36	12208
RFID tracking	3 days	14	18	4262

Evaluation Metrics. Our aim is to recover incomplete indoor trajectories. Following previous work [18,25], we mainly adopt three metrics *Accuracy*, *Recall*, and *Precision* to demonstrate the performance of our model and baseline methods.

Accuracy. *Accuracy* is the primary evaluation metric to judge whether the predicted location is accurate. We use TP to indicate that the predicted location matches the ground truth and FN to indicate that the predicted location does not match. *Accuracy* is formulated as:

$$accuracy = \frac{TP}{TP + FN} \tag{11}$$

Recall. *Recall* is defined as the proportion of correctly classified locations to the length of the target trajectory, which can reflect the recovery trajectory's integrity. The *recall* calculation is shown in Eq. 12, where $\cap$ represents the longest common subsequence of the recovered trajectory T_R and the ground truth T_G, and $||$ represents the length of the trajectory.

$$recall = \frac{\mid T_R \cap T_G \mid}{\mid T_G \mid} \tag{12}$$

Precision. *Precision* refers to the ratio of the number of correctly classified locations to the length of the recovered trajectory generated by the model, which can reflect the quality of the recovered trajectory. We use *Precision* to evaluate the performance by comparing the recovered trajectory T_R with the ground truth T_G. The formula of *precision* is:

$$precision = \frac{\mid T_R \cap T_G \mid}{\mid T_R \mid} \tag{13}$$

Task Setting. For each of the two datasets, we split the dataset into training, validation, and test sets with a split ratio of 7:2:1. Since the preprocessed trajectory data is complete, we generate incomplete trajectories by randomly masking the complete trajectories. The inherent data missing rate of data collected by RFID equipment is 30% [10], coupled with other factors such as environmental interference, so the missing rate in actual scenarios will be higher. We evaluate the robustness of our ITAR by masking different rates of trajectory points. The masking location indicates that the trajectory point is missing. We choose the missing rate mr as 30%, 50%, 70% respectively. A higher mr indicates that the number of missing points in the incomplete trajectory is larger, and the difficulty of trajectory recovery is greater. We generate incomplete trajectories for prediction and use the complete trajectories as the ground truth for evaluation. We repeat the above procedure three times for reliable evaluation and report the average results for both datasets.

Baselines. We compare the proposed ITAR with several representative baselines. Among them, the first is the latest RFID-based indoor trajectory data imputation algorithm. The last three are state-of-the-art deep learning-based trajectory imputation models which can extract more complex features.

- IR-MHMM [1]: This is the latest research in the field of RFID trajectory recovery, which uses a Multi-variate Hidden Markov Model (IR-MHMM) to capture and recover the missing data of RFID-based trajectory in indoor environments.
- PeriodicMove [19]: This is a latest model-based trajectory recovery method. It exploits gated graph neural networks to mine user movement preferences and utilizes various attention mechanisms to model regularity and periodic patterns of user movements. We adapt to the current task by reserving the fill location.
- DHTR [23]: The method designs a subseq2seq model with a Kalman filter to recover trajectories in free space. We mainly refer to the bidirectional LSTM in the main part of the method for experimental comparison.
- MTrajRec [18]: This is the state-of-the-art method in the field of trajectory recovery, which models the forward sequential mobility transition through a gated neural network and will enhance the performance of trajectory recovery with an attention mechanism. In order to adapt to our task, we only refer to the sequence numbers of the recovered trajectories without considering the specific coordinates of the recovered trajectories.

5.2 Result and Analysis

Overall Performance. We compare ITAR with other baseline models in terms of *Accuracy*, *Precision*, and *Precision*. Table 3 gives the trajectory recovery performance of different methods with different missing rates. We can have the following observations.

Table 3. Overall performance comparison in terms of *Accuracy*, *Recall* and *Precision*. The best result for each evaluation metric is in bold. A larger missing rate indicates a larger number of missing points in incomplete trajectories.

Dataset	Methods	30%			50%			70%		
		Accuracy	Recall	Precision	Accuracy	Recall	Precision	Accuracy	Recall	Precision
Synthetic	IR-MHMM	0.5213	0.6212	0.6614	0.4253	0.5023	0.5528	0.3438	0.4249	0.4423
	PeriodicMove	0.8648	0.8839	0.8839	0.7413	0.8014	0.8014	0.6013	0.7333	0.7333
	DHTR	0.8270	0.8687	0.8747	0.7205	0.7930	0.8036	0.5912	0.7058	0.7299
	MTrajRec	0.8811	0.9108	0.9175	0.7338	0.8008	0.8112	0.6035	0.7193	0.7416
	ITAR	**0.9252**	**0.9467**	**0.9498**	**0.8067**	**0.8681**	**0.8807**	**0.6895**	**0.7936**	**0.8168**
Tracking	IR-MHMM	0.3827	0.4623	0.4766	0.3024	0.3635	0.3865	0.2283	0.3398	0.3245
	PeriodicMove	0.6413	0.7812	0.7812	0.5097	0.6690	0.6690	0.4142	0.5849	0.5849
	DHTR	0.6542	0.7270	0.7968	0.5147	0.6229	0.6683	0.3647	0.5272	0.6361
	MTrajRec	0.6608	0.7523	0.8143	0.5278	0.6308	0.6712	0.3809	0.5533	0.6364
	ITAR	**0.7311**	**0.8243**	**0.8874**	**0.5995**	**0.6904**	**0.7303**	**0.4402**	**0.6043**	**0.6874**

1. Traditional Hidden Markov-based approaches perform the worst among all evaluation metrics on both datasets. Although the method attempts to find the transition patterns of tags, its performance is still unacceptable, especially with a high missing rate, as it fails to capture complex transition patterns between locations.
2. In recent years, RNN-based deep learning methods have outperformed traditional methods, and state-of-the-art deep learning methods, including DHTR and MTrajRec, have achieved satisfactory performance because they can model simple translation patterns between locations. Although DHTR uses bidirectional LSTM for encoding, in our application, its actual effect is no better than that of MTrajRec with unidirectional GRU. One possible reason is that the missing trajectory points in our scene are irregular. As a result, the before and after dependencies between locations are not obvious. Furthermore, our ITAR achieves further performance gains over state-of-the-art deep learning methods, as ITAR can capture complex location transition patterns.
3. ITAR outperforms all baselines on all evaluation metrics on both datasets. These vast improvements show that our proposed ITAR can well simulate position transition patterns to recover trajectories. When the missing rate is 50%, *Accuracy*, *Recall*, and *Precision* of ITAR outperform the best baseline MTrajRec by 9.9%, 8.4%, and 8.4% on the Synthetic dataset, respectively. *Accuracy*, *Recall*, and *Precision* outperform the best baseline MTrajRec by 7.1%, 9.4%, and 8.8% on the RFID tracking dataset, which proves the effectiveness of our ITAR in indoor trajectory recovery. The fundamental reason for this progress is that, on the one hand, we utilize GGNN to model complex transition patterns, and on the other hand, we employ a multi-head attention mechanism to capture correlations between locations.

Robustness Analysis. We evaluate the robustness of ITAR by varying the proportion of missing trajectory points. An increase in the missing rate means an increase in the number of missing points, leading to increased uncertainty

between any two consecutive points. We increased the percentage of missing trajectory points in trajectories from 30% to 70%, and the results are shown in Table 3. We can find that as the missing rate increases, the performance of ITAR and other baselines on all evaluation metrics decreases. However, the higher the missing rate, the more pronounced the performance improvement of ITAR. Specifically, when the missing rate is 70%, ITAR still maintains a good recovery effect and is better than the best baselines on average of 14.91%, 9.78%, and 9.08% in terms of *Accuracy*, *Recall*, and *Precision*. This validates the robustness of our model in capturing complex transition patterns between locations and demonstrates its robust ability to reconstruct individual indoor trajectories.

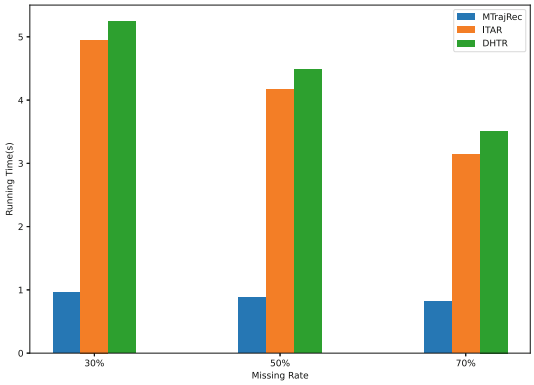

Fig. 4. Running efficiency.

Running Efficiency. To evaluate the efficiency of ITAR, we compared the algorithm's running time with DHTR and MTrajRec models, which also employ the seq2seq architecture for easy comparison. As shown in Fig. 4, we find that the running time of ITAR and other baseline models decreases as the missing rate increases because ITAR utilizes an automatic inference model. As the missing rate increases, the accuracy of the model decreases, and thus the system's run time decreases slightly. In addition, due to the multi-head attention mechanism adopted by ITAR, its running speed is slightly slower than that of the MTrajRec algorithm, but the accuracy rate is much higher. Therefore, this difference can be ignored.

Importance of the Attention Mechanism. In this section, we remove the attention mechanism from ITAR to test its contribution, as shown in Table 4. Results for ITAR-noAttn were significantly lower compared to ITAR. We enumerate the average performance change in ITAR on the two datasets after removing the attention mechanism with a missing rate of 50%. Among them, *Accuracy* decreased by 32.83%, *Recall* decreased by 25.79%, and *Precision* decreased by

24.55%, which shows that the attention mechanism occupies a critical position in our model. One possible reason is that the attention mechanism can effectively enforce the spatial constraints on missing locations and establish dependencies between locations.

Table 4. Importance of the attention mechanism.

Method	Accuracy	Recall	Precision
ITAR	0.7031	0.7793	0.8055
ITAR-noatten	0.4723	0.5783	0.6077
Reduced performance	**32.83%**	**25.79%**	**24.55%**

Parameter Tuning. Apart from evaluating the components of our proposed model MTrajRec, there are two important parameters to tune in our model.

Hidden size d. We refer to previous studies [19] to observe performance changes by adjusting the hidden size d in the range of {16, 32, 64, 128, 256, 512}. From the results presented in Fig. 5, it can be seen that as the hidden size increases, the performance gradually improves, and when it is larger than a certain value, the performance begins to decrease slightly. On the one hand, it shows that a moderate d can better represent the hidden information between locations, which is enough to capture the transition patterns. On the other hand, using redundant dimensions increases model complexity and forces the model to overfit to training, which may reduce the generalization ability of our model on the test set. We find that when the hidden layer size is 128, ITAR can achieve a trade-off between model efficiency and accuracy. Therefore, the default hidden layer size of ITAR is 128.

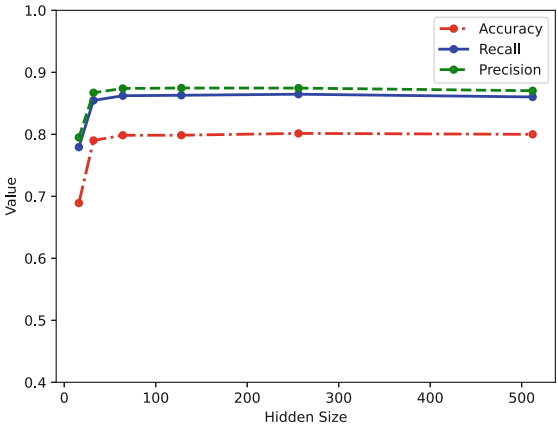

Fig. 5. Impart of hidden size d.

Head number H. We adjust the head number H in the range of {1, 2, 3, 4, 5, 8}, and Fig. 6 shows the performance change. We can see that in most cases, the number of attention heads does not show a clear trend in the performance change, which means that the effect of attention head number is not significant. Considering that more heads leads to stronger model expressiveness and more computational cost, we finally fixed the number of heads to 4 in order to make a compromise between performance and efficiency.

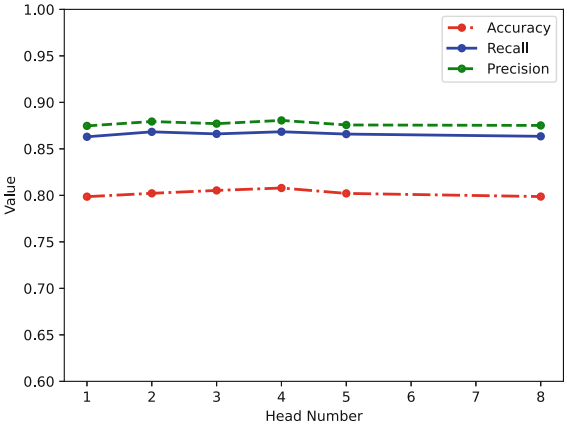

Fig. 6. Impart of head number H.

6 Conclusion

In this paper, we propose a novel end-to-end deep learning model ITAR for automatic trajectory recovery in indoor spaces. We introduce a graph neural network in the Seq2Seq model to capture complex transition patterns and employ an attention mechanism to improve performance. We test the proposed model with synthetic and real RFID tracking data. The experimental results show that when the missing rate is 50%, ITAR outperforms the best baseline by 8.5%, 8.9% and 8.6% in terms of *Accuracy*, *Recall* and *Precision* respectively. In future work, we plan to further enhance the proposed model by incorporating more environmental constraints.

Acknowledgments. This work was supported by the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No. XDC02040300.

References

1. Baba, A.I., Jaeger, M., Lu, H., Pedersen, T.B., Ku, W.S., Xie, X.: Learning-based cleansing for indoor RFID data. In: Proceedings of the 2016 International Conference on Management of Data, pp. 925–936 (2016)

2. Baba, A.I., Lu, H., Pedersen, T.B., Xie, X.: Handling false negatives in indoor RFID data. In: 2014 IEEE 15th International Conference on Mobile Data Management, vol. 1, pp. 117–126. IEEE (2014)
3. Baba, A.I., Lu, H., Xie, X., Pedersen, T.B.: Spatiotemporal data cleansing for indoor RFID tracking data. In: 2013 IEEE 14th International Conference on Mobile Data Management, vol. 1, pp. 187–196. IEEE (2013)
4. Derakhshan, R., Orlowska, M.E., Li, X.: RFID data management: challenges and opportunities. In: 2007 IEEE International Conference on RFID, pp. 175–182. IEEE (2007)
5. Fazzinga, B., Flesca, S., Furfaro, F., Parisi, F.: Cleaning trajectory data of RFID-monitored objects through conditioning under integrity constraints. In: EDBT, pp. 379–390 (2014)
6. Fazzinga, B., Flesca, S., Furfaro, F., Parisi, F.: Offline cleaning of RFID trajectory data. In: Proceedings of the 26th International Conference on Scientific and Statistical Database Management, pp. 1–12 (2014)
7. Fazzinga, B., Flesca, S., Furfaro, F., Parisi, F.: Interpreting RFID tracking data for simultaneously moving objects: an offline sampling-based approach. Expert Syst. Appl. **152**, 113368 (2020)
8. Feng, J., et al.: DeepMove: predicting human mobility with attentional recurrent networks. In: Proceedings of the 2018 World Wide Web Conference, pp. 1459–1468 (2018)
9. Feng, Y., Huang, W., Wang, S., Zhang, Y., Jiang, S.: Detection of RFID cloning attacks: a spatiotemporal trajectory data stream-based practical approach. Comput. Netw. **189**, 107922 (2021)
10. Floerkemeier, C., Lampe, M.: Issues with RFID usage in ubiquitous computing applications. In: Ferscha, A., Mattern, F. (eds.) Pervasive 2004. LNCS, vol. 3001, pp. 188–193. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24646-6_13
11. Gu, Yu., Yu, G., Chen, Y., Ooi, B.C.: Efficient RFID data imputation by analyzing the correlations of monitored objects. In: Zhou, X., Yokota, H., Deng, K., Liu, Q. (eds.) DASFAA 2009. LNCS, vol. 5463, pp. 186–200. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00887-0_15
12. Hu, K.F., Li, L., Lu, Z.P.: AgCleaning: a track data filling algorithm based on movement recency for RFID track data. In: Applied Mechanics and Materials, vol. 490, pp. 1330–1337. Trans Tech Publ (2014)
13. Huang, W., Zhang, Y., Feng, Y.: ACD: an adaptable approach for RFID cloning attack detection. Sensors **20**(8), 2378 (2020)
14. Liu, Q., Wu, S., Wang, L., Tan, T.: Predicting the next location: a recurrent model with spatial and temporal contexts. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
15. Loshchilov, I., Hutter, F.: Fixing weight decay regularization in Adam (2017)
16. Luo, Y., Cai, X., Zhang, Y., Xu, J., et al.: Multivariate time series imputation with generative adversarial networks. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
17. Morzy, M.: Prediction of moving object location based on frequent trajectories. In: Levi, A., Savaş, E., Yenigün, H., Balcısoy, S., Saygın, Y. (eds.) ISCIS 2006. LNCS, vol. 4263, pp. 583–592. Springer, Heidelberg (2006). https://doi.org/10.1007/11902140_62
18. Ren, H., et al.: Mtrajrec: map-constrained trajectory recovery via seq2seq multi-task learning. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 1410–1419 (2021)

19. Sun, H., Yang, C., Deng, L., Zhou, F., Huang, F., Zheng, K.: Periodicmove: shift-aware human mobility recovery with graph neural network. In: Proceedings of the 30th ACM International Conference on Information and Knowledge Management, pp. 1734–1743 (2021)
20. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, vol. 27 (2014)
21. Tong, C., Chen, H., Xuan, Q., Yang, X.: A framework for bus trajectory extraction and missing data recovery for data sampled from the internet. Sensors **17**(2), 342 (2017)
22. Vaswani, A., et al.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS 2017, pp. 6000–6010. Curran Associates Inc., Red Hook (2017)
23. Wang, J., Wu, N., Lu, X., Zhao, W.X., Feng, K.: Deep trajectory recovery with fine-grained calibration using Kalman filter. IEEE Trans. Knowl. Data Eng. **33**(3), 921–934 (2019)
24. Wang, S., Cao, Z., Zhang, Y., Huang, W., Jiang, J.: A temporal and spatial data redundancy processing algorithm for RFID surveillance data. Wirel. Commun. Mob. Comput. **2020** (2020)
25. Wheeb, A.H.: Performance analysis of VOIP in wireless networks. Int. J. Comput. Netw. Wirel. Commun. (IJCNWC) **7**(4), 1–5 (2017)
26. Wu, H., Chen, Z., Sun, W., Zheng, B., Wang, W.: Modeling trajectories with recurrent neural networks. In: IJCAI (2017)
27. Xi, D., Zhuang, F., Liu, Y., Gu, J., Xiong, H., He, Q.: Modelling of bi-directional spatio-temporal dependence and users' dynamic preferences for missing poi check-in identification. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 5458–5465 (2019)
28. Xia, T., et al.: AttnMove: history enhanced trajectory recovery via attentional network. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 4494–4502 (2021)
29. Xie, L., Yin, Y., Vasilakos, A.V., Lu, S.: Managing RFID data: challenges, opportunities and solutions. IEEE Commun. Surv. Tutor. **16**(3), 1294–1311 (2014)
30. Yang, C., Sun, M., Zhao, W.X., Liu, Z., Chang, E.Y.: A neural network approach to jointly modeling social networks and mobile trajectories. ACM Trans. Inf. Syst. (TOIS) **35**(4), 1–28 (2017)
31. Zhao, J., Xu, J., Zhou, R., Zhao, P., Zhu, F.: On prediction of user destination by sub-trajectory understanding: a deep learning based approach. In: the 27th ACM International Conference (2018)
32. Zhao, Z., Ng, W.: A model-based approach for RFID data stream cleansing. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, pp. 862–871 (2012)
33. Zheng, S., Yue, Y., Hobbs, J.: Generating long-term trajectories using deep hierarchical networks. In: Advances in Neural Information Processing Systems, vol. 29 (2016)