RLOMM: An Efficient and Robust Online Map Matching Framework with Reinforcement Learning

MINXIAO CHEN*, Beijing University of Posts and Telecommunications, China and Beiyou Shenzhen Institute, China
HAITAO YUAN*, Nanyang Technological University, Singapore
NAN JIANG, Beijing University of Posts and Telecommunications, China
ZHIHAN ZHENG, Beijing University of Posts and Telecommunications, China
SAI WU, Zhejiang University, China
AO ZHOU, Beijing University of Posts and Telecommunications, China
SHANGGUANG WANG, Beiyou Shenzhen Institute, China and Beijing University of Posts and Telecommunications, China

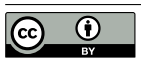
Online map matching is a fundamental problem in location-based services, aiming to incrementally match trajectory data step-by-step onto a road network. However, existing methods fail to meet the needs for efficiency, robustness, and accuracy required by large-scale online applications, making this task still challenging. This paper introduces a novel framework that achieves high accuracy and efficient matching while ensuring robustness in handling diverse scenarios. To improve efficiency, we begin by modeling the online map matching problem as an Online Markov Decision Process (OMDP) based on its inherent characteristics. This approach helps efficiently merge historical and real-time data, reducing unnecessary calculations. Next, to enhance robustness, we design a reinforcement learning method, enabling robust handling of real-time data from dynamically changing environments. In particular, we propose a novel model learning process and a comprehensive reward function, allowing the model to make reasonable current matches from a future-oriented perspective, and to continuously update and optimize during the decision-making process based on feedback. Lastly, to address the heterogeneity between trajectories and roads, we design distinct graph structures, facilitating efficient representation learning through graph and recurrent neural networks. To further align trajectory and road data, we introduce contrastive learning to decrease their distance in the latent space, thereby promoting effective integration of the two. Extensive evaluations on three real-world datasets confirm that our method significantly outperforms existing state-of-the-art solutions in terms of accuracy, efficiency and robustness.

CCS Concepts: • **Information systems** → **Spatial-temporal systems**.

Additional Key Words and Phrases: map matching; reinforcement learning; trajectory; road network

*Both authors contributed equally to this research.

Authors' Contact Information: Minxiao Chen, Beijing University of Posts and Telecommunications, China and Beiyou Shenzhen Institute, China, chenminxiao@bupt.edu.cn; Haitao Yuan, Nanyang Technological University, Singapore, haitao.yuan@ntu.edu.sg; Nan Jiang, Beijing University of Posts and Telecommunications, China, jn_bupt@bupt.edu.cn; Zhihan Zheng, Beijing University of Posts and Telecommunications, China, 26510036@bupt.edu.cn; Sai Wu, Zhejiang University, China, wusai@zju.edu.cn; Ao Zhou, Beijing University of Posts and Telecommunications, China, aozhou@bupt.edu.cn; Shangguang Wang, Beiyou Shenzhen Institute, China and Beijing University of Posts and Telecommunications, China, sgwang@bupt.edu.cn.



ACM 2836-6573/2025/6-ART209
https://doi.org/10.1145/3725346

ACM Reference Format:
Minxiao Chen, Haitao Yuan, Nan Jiang, Zhihan Zheng, Sai Wu, Ao Zhou, and Shangguang Wang. 2025. RLOMM: An Efficient and Robust Online Map Matching Framework with Reinforcement Learning. *Proc. ACM Manag. Data* 3, 3 (SIGMOD), Article 209 (June 2025), 26 pages. https://doi.org/10.1145/3725346

1 Introduction

Since the widespread availability of GPS services in the 1990s, research related to map matching has been a focal point. Map matching is the process of aligning vehicle trajectories onto the actual road network, which is a fundamental problem in location-based services [22, 48]. Online map matching is a sequential process that aligns incoming data incrementally, making it particularly well-suited for real-time applications. This process is a crucial element for various location-based online services and is vital for applications requiring real-time responses such as navigation services [39], optimal route planning [10, 31], and travel time estimation [49, 50, 54].

The map matching method can be categorized into two main types: *rule-based* [30, 45, 47] and *deep-learning-based* [7, 18, 25, 34, 52]. The former treats trajectories as observations and utilizes the Hidden Markov Model (HMM) to infer road segments as hidden states [6] or uses Markov Decision Process (MDP) with dynamic programming algorithms like value iteration to model offline matching process [45]. With advancements in deep learning, attention has shifted towards the latter approach, framing the problem as an end-to-end task. This method leverages large datasets to directly learn patterns for matching trajectories to road segments. Overall, both types of methods perform well in terms of matching accuracy in high-sampling-rate offline scenarios. However, when the problem shifts to the online scenario, they exhibit significant inefficiencies due to their inherent nature [3, 15, 32, 35].

Specifically, for online map matching scenarios, existing solutions are essentially straightforward extended forms of offline methods. For example, [30] points out that Microsoft treats the map matching problem as a batch processing task, performing matching after all data has been collected. For online scenarios, they simply use a sliding window as an adaptation of the offline method. To offer a clear perspective, we summarize the two existing extension strategies, as shown in Fig. 1. The first strategy involves repeatedly invoking the DNN or MDP-based offline methods [18, 25, 34, 45] during online matching process to match the current available trajectory at that moment. The second strategy employs HMM, incrementally matching incoming trajectories in a rule-based manner [14, 23]. However, current service providers still incur substantial costs because these two categories have the following notable deficiencies in addressing the core challenges of online map matching:

(1) **Low efficiency.** Most current methods, particularly DNN-based or MDP-based, are tailored to align complete trajectories with road networks. However, when applied incrementally, these methods necessitate the input of all preceding trajectory segments to utilize historical data for enhancing effectiveness. This process results in redundant computations and leads to significant inefficiencies. In light of this issue, it is crucial to develop an innovative solution that not only maintains the accuracy of map matching but also enhances efficiency in real-time scenarios, thereby reducing costs for service providers.

(2) **Poor robustness.** Online map matching necessitates the continuous processing of data within a streaming pipeline, emphasizing the significant influence of previous matches on subsequent ones. Earlier matches create an informational context that shapes later decisions. However, existing online methods predominantly utilize a greedy approach, neglecting the essential impact of current matches on future results, thus leading to suboptimal matches. Consequently, addressing this oversight is crucial for advancing the development of a robust online map matching framework.

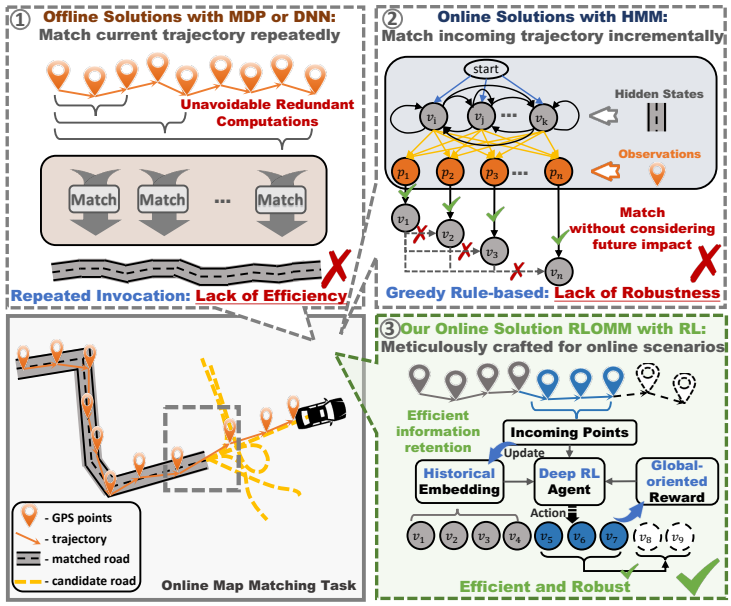


Fig. 1. Concept comparison of existing online map matching solutions and our RLOMM.

(3) **Insufficient handling of trajectory-road heterogeneity.** Intuitively, trajectory data is an unstructured sequence with significant noise, whereas road network data is a relatively fixed, structured topological graph. This means trajectory and road data are inherently heterogeneous, which demands careful address. Correlations exist both among different trajectories and between various road segments, all of which should be considered. However, existing methods lack specialized designs that effectively capture these complex correlations within and between data types, leading to suboptimal data representations and encoding techniques. We argue that an ideal solution should facilitate the integration of trajectory and road data while preserving their unique attributes and contexts.

To comprehensively address the aforementioned limitations of existing methods, as shown in Fig. 1, we introduce **RLOMM**, an efficient and robust online map matching framework with high dynamic adaptability, which is a novel paradigm to address the online map matching problem. **First, we meticulously design an Online Markov Decision Process (OMDP) specifically for online map matching scenario,** continuously extracting real-time and historical information from real-time environment to construct the information state, closely integrating with the incremental and streaming characteristics of online map matching. Compared to the MDP method previously used for offline map matching [45], this novel modeling approach effectively captures the essential data and dynamically updates the historical information throughout the matching process, which fundamentally simplifies the complexity of the problem and prevents redundant calculations, thereby enhancing efficiency. Additionally, the efficient component that captures sequence correlations effectively coordinates with this modeling approach, enhancing the integration efficiency of historical and real-time information.

Second, to endow the model with sufficient dynamic adaptability and robustness for various scenarios, we employ reinforcement learning to execute matching actions based on the states divided by the OMDP and meticulously design the reward evaluation process. Specifically, we employ the deep Q-learning method, which allows the model to consider the impact of each match on future matches. This enables the model to perform matching from a future-oriented perspective,

avoiding current optimal matches that are prone to errors, thereby discarding the simplistic greedy approach to achieve ideal overall matching accuracy. Meanwhile, a carefully designed training process enables the model to grasp the dynamic changes of the matching environment and allows for continuous updates and improvements in the decision-making process based on feedback from the rewards.

Third, considering the trajectory-road heterogeneity, we design distinct graph data structures for each type of data to capture their intrinsic connections, namely trajectory transition graph and link connection graph. By representing both types of data in effective graph structures, their heterogeneity can be mitigated, thus promoting better information integration. Additionally, we design mechanism to bridge the interaction between their representation to facilitate better fusion. Moreover, corresponding graph and recurrent neural networks are utilized to generate effective representations for both trajectories and roads. Recognizing the necessity for effective alignment and integration of trajectory data with road data in map matching task, we design a trajectory-road representation alignment module from the perspective of enhancing representational robustness. This module reduces the distance between the representations of the two data types in the latent space, thereby facilitating effective integration.

In summary, we make the following contributions:

- To the best of our knowledge, we are the first to model the online map matching problem as an Online Markov Decision Process, proposing a new solution paradigm. Additionally, we propose **RLOMM**, a novel two-stage system framework based on deep reinforcement learning, which delivers efficient map matching with high dynamic adaptability and robustness. (Sec. 2&3)
- We design graph structures meticulously tailored for both trajectory and road to capture the correlations inherent to each type of data and mitigate the heterogeneity. Our feature encoding modules utilize various representation learning and sequence correlation capture techniques to integrate trajectory and road data, facilitating the map matching process. (Sec. 4)
- We propose a novel model learning process for our **RLOMM** framework with carefully designed reward evaluation, which can perform globally optimal matching from a future-oriented perspective. We introduce a trajectory-road representation alignment module that utilizes contrastive learning to facilitate the effective integration of trajectories and roads. Combined with the temporal-difference loss of reinforcement, our framework is capable of robust learning from a variety of scenarios. (Sec. 5)
- We conduct a comprehensive evaluation on three real-world datasets. The results show that our method significantly outperforms state-of-the-art in terms of accuracy, efficiency and robustness. (Sec. 6)

2 Preliminaries

We first introduce the basic concepts of the map matching problem, namely trajectories and road networks, followed by a detailed explanation of the specifically designed data structures for this task. Finally, we formalize the online map matching problem.

2.1 Basic Concepts

Trajectory. A trajectory $\mathcal{T}$ is a sequence of GPS points, denoted as $\mathcal{T} = \langle p_1, p_2, \ldots, p_n \rangle$, where each point p_i represents a location record captured at a specific time. Each point p_i consists of three components: the geographic coordinates latitude (lat_i) and longitude (lon_i), and a timestamp (time_i).

Trajectory Transition Graph. As shown in the left part of Fig. 2, we utilize the widely adopted grid representation method [25, 34, 44] to split the city into $N_g = H \times W$ grids with side length l_g. This process results in each trajectory point p falling into a specific grid denoted $g_{ij} = Grid(p)$,

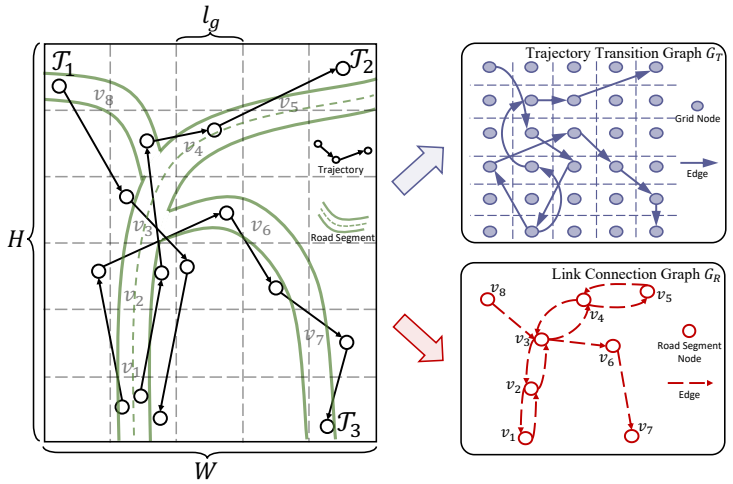


Fig. 2. An illustration of Grids Partition, Trajectory Transition Graph, and Link Connection Graph.

where g_{ij} is the grid indexed by (i, j) and $Grid(\cdot)$ can be considered as the function that maps trajectory points to grids. Therefore, a trajectory $\mathcal{T}$ can be represented by a sequence of grids with timestamps $\mathcal{T} = \langle g_1, \ldots, g_n \rangle$. By doing so, the relationships between different trajectories can be effectively captured from a global perspective. Specifically, we construct a directed trajectory transition graph $G_T = (V_T, E_T)$, where V_T is the node set of grids and E_T is the edge set representing trajectory transition. Specifically, as shown in the upper right part of Fig. 2, for any trajectory $\mathcal{T}$ in trajectory set, if there is a trajectory point transition from $p_i \in \mathcal{T}$ to $p_{i+1} \in \mathcal{T}$ (p_i, p_{i+1} are consecutive and $Grid(p_i)$, $Grid(p_{i+1})$ are not identical), then there is an edge form $Grid(p_i)$ to $Grid(p_{i+1})$. Moreover, the weight assigned to each edge reflects the count of such trajectory transitions observed across the trajectory set.

Road Network. A road network is a graph where the nodes represent locations, such as intersections, and the edges represent road segments connecting these locations.

Link Connection Graph. As shown in the lower right part of Fig. 2, a directed link connection graph $G_R = (V_R, E_R)$ is used to model the road network, where each $u_i \in V_R$ is a road segment, and $e_{ij} = (u_i, u_j)$ represents road segment u_i and u_j are connected. To maintain consistency with the grid representation of trajectories, we choose the grid coordinates of the start and end points of road segments, along with their latitude and longitude as node features.

2.2 Problem Formulation

In the online scenario, we need to match newly arriving trajectory points at regular intervals. For each matching, we have the link connection graph G_R which represents road network, the k new incoming trajectory points $\mathcal{T}_{i:i+k} = \langle p_i, \ldots, p_{i+k-1} \rangle$ to match, the historical prefix trajectory $\mathcal{T}_{\triangleleft i} = \langle p_1, \ldots, p_{i-1} \rangle$ that is already available for previous matching, and the matched road $U_{\triangleleft i} = \langle u_1, \ldots, u_{i-1} \rangle$ of the historical prefix trajectory. Formally, we define the Online Map Matching problem and Online Markov Decision Process as follows.

Definition 1 (Online Map Matching). Given G_R, $\mathcal{T}_{i:i+k}$, $\mathcal{T}_{\triangleleft i}$ and $U_{\triangleleft i}$, the goal is to match k road segments $U_{i:i+k} = \langle u_i, \ldots, u_{i+k-1} \rangle$ from their candidate sets $C_{i:i+k}$. The candidate sets are precomputed based on spatial distance, which is a common practice in related works [14, 30, 34]. The matching process is incremental and requires multiple steps to complete the matching of the entire trajectory $\mathcal{T}$.

Definition 2 (Online Markov Decision Process). The Markov Decision Process (MDP) provides a powerful framework for representing problems as a sequence of states and transitions, offering both effectiveness and efficiency. Inspired by this, we innovatively model the online map matching problem as an online MDP, meticulously tailored to capture the core aspects of the problem. Specifically, it is characterized by *State*, *Action*, *Transition*, and *Reward*.

- *State*: The state s_i of time step i represents the real-time information of online map matching scenario (i.e., *previously matched road segments* U_i, *current trajectory points* $\mathcal{T}_i$, and *candidate road segments* C_i), and the historical information of previous matches (i.e., *historical information* of road H_i^r and trajectory H_i^t).
- *Action*: Action a_i represents the selection of a road segment from the candidate road segments C_i.
- *Transition*: In our modeling, the selected candidate road segments influence the update of historical information, which in turn affects the next state. Meanwhile, the previously matched road segments within the state are determined by the selected candidate road segments. This can be considered as state transitions in the context of map matching.
- *Reward*: Reward $r = R(s_i, a_i)$ denotes the reward received after taking action a_i under state s_i with reward function R, which is associated with the accuracy of the action.

By defining these components, the online map matching problem can be effectively modeled as an online MDP. In particular, previously defined $\mathcal{T}_{\triangleleft i}$, $U_{\triangleleft i}$, $\mathcal{T}_{i:i+k}$ and $C_{i:i+k}$ are effectively incorporate into *State*, the *Action* determines how to select $U_{i:i+k}$, the *Transition* indicates the evolution of information for $\mathcal{T}_{\triangleleft i}$ and $U_{\triangleleft i}$ within *State*, and the *Reward* is the assessment of the matched results $U_{i:i+k}$. This approach enables the use of reinforcement learning algorithms to optimize decision-making for efficient and robust map matching in dynamic environments.

3 Framework

In this section, we outline the framework of our proposed **RLOMM**. As shown in Fig. 3, **RLOMM** contains two stages: the online inference and the offline training.

Online Inference. At this stage, we first extract real-time information and historical information to form *State*. Next, we design feature encoding modules to generate effective representations for trajectories and roads, and generate score for each candidate road segment to guide the matching decisions.

1. Extracting State: For a given time step i, we extract real-time information from the online map matching scenario, and preserve historical information from previous matches to construct state s_i. Specifically, the *previously matched road segments* U_i, *current trajectory points* $\mathcal{T}_i$, and *candidate road segments* C_i are incorporate as real-time information since they are basic elements of map matching. Moreover, to maximize the efficiency of information transfer during the online matching process and to prevent redundant computations, we retain historical information by preserving the hidden states from the encoding module of the preceding step, which contains *historical information* of road H_i^r and trajectory H_i^t.

2. Encoding Feature: After the construction of the state s_i, we design the trajectory and road encoding module to obtain effective encodings of the corresponding information within the state. In the trajectory encoding module, we first construct the trajectory transition graph G_T based on the set of trajectories, aiming to capture the inter-trajectory correlations to obtain informative trajectory representations. Subsequently, to bridge the interaction between representations of road segments and trajectory, we incorporate the representations of the corresponding road segment for the *current trajectory point* $\mathcal{T}_i$ and apply a graph neural network to derive the trajectory representation. This is then combined with the historical information of trajectory H_i^t to generate the trajectory

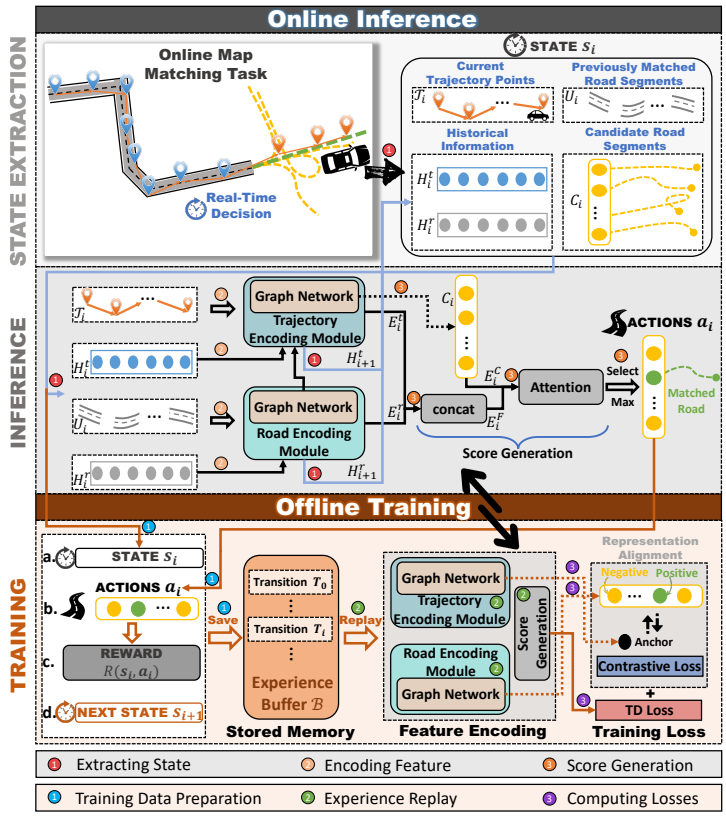


Fig. 3. The architecture of RLOMM, which consists of two parts: Online Inference and Offline Training.

encoding E_i^t for time step i. In the design of the road encoding module, we initially employ the road network to construct the link connection graph G_R. Subsequently, analogous to the trajectory encoding module, we deploy a graph neural network to generate representations for road segments, and in conjunction with the historical information of road H_i^r, we obtain the road encoding E_i^r. Moreover, as mentioned before, the corresponding road segment representation for the *current trajectory point* $\mathcal{T}_i$ is transmitted to the trajectory encoding module. Finally, the hidden states H_{i+1}^r and H_{i+1}^t produced by the encoding module are retained as the new historical information for the next time step $i+1$.

3. Score Generation: In this step, we first employ the graph neural network within the road encoding module to encode the candidate road segments C_i, generating candidates encoding E_i^C. Then, we concatenate the trajectory encoding E_i^t and the road encoding E_i^r to obtain fusion encoding E_i^F, and together with the candidate encoding E_i^C, we generate attention scores and select the maximum value as the matching decision action a_i.

Offline Training. At this stage, we first prepare the training data through multiple inferences and reward evaluations, saving it to the experience buffer. Next, we replay the experience through the feature encoding module to prepare for loss calculation. Finally, we compute different losses from the perspectives of temporal difference and contrastive learning.

1. Training Data Preparation: Given the action a_i from the online inference matching decision and state s_i, we devise a meticulously reward evaluation function to generate reward $R(a_i, s_i)$. Notably, the reward evaluation function thoroughly considers the complexity of online matching

scenarios, performing a comprehensive assessment from a future-oriented perspective, which helps to minimize incorrect matches right from the start. After taking action a_i, we can obtain the next state s_{i+1}, which along with the state s_i, action a_i, and reward $R(a_i, s_i)$, are stored as a transition T_i in the experience buffer. We repeat the aforementioned steps multiple times to prepare a collection of training data. This process will be conducted repeatedly to facilitate the model's continuous enhancement and refinement across various scenarios.

2. Experience Replay: After a certain number of online inferences and transition storages, we randomly select a batch of transitions from the experience buffer and replay them for training purposes. Specifically, these transitions are processed through our feature encoding module and score generation component, supplemented by double DQN techniques to prepare for loss calculation while reducing score overestimation.

3. Computing Losses: After the experience replay, we employ reinforcement learning strategies grounded in Temporal Difference (TD) learning principles. By leveraging the Temporal Difference (TD) Loss, the model can enhance the capability to estimate the q-value associated with action a_i given the state s_i. Moreover, considering that the map matching problem necessitates effective encoding of both trajectories and road segments to integrate their information for accurate matching, we design a trajectory-road representation alignment module to align the representations in latent space. In particular, we select the representations of trajectory points as anchors, and take the representations of candidate road segments as positive and negative samples. Within the candidates, the ground truth and the remaining parts are considered as positive and negative, respectively. Finally, the contrastive loss is used together with the TD loss to train our model.

4 Feature Encoding

As shown in Fig. 4, we design road encoding module $\mathcal{M}_r$, trajectory encoding module $\mathcal{M}_t$ and score generation module $\mathcal{M}_s$. In particular, we first elaborate on the details of deriving informative representations for trajectories and roads in Sec. 4.1, which is based on graph structures tailored to trajectories and roads. Then, we focus on efficiently capturing the sequence correlation in online map matching scenarios in Sec. 4.2. At last, we explain how to generate the score of each candidate in Sec. 4.3.

4.1 Trajectory & Road Representation

Link Connection Graph Convolution. The representation learning of road segments is crucial in the map matching problem. Considering that it can be naturally presented as a graph structure (i.e., the link connection graph G_R), we obtain its representation through a graph neural network. Specifically, considering the complex topological structure of road networks, motivated by [25], we adopt the Graph Isomorphism Network (GIN) [46]. GIN has the optimal ability to distinguish various graph structures, thus accurately representing the nuances of road networks. Its aggregation mechanism, which merges a node's features with those of its neighbors, is enhanced by a learnable parameter that precisely adjusts the balance between a node's own characteristics and its neighbors' influences. In our implementation, we adopt a lightweight version with a limited number of layers and a simple MLP, ensuring that this component is efficient for practical use. The update rule for feature $z_{u_j}^{(n)} \in \mathbb{R}^{d_r}$ of node $u_j \in V_R$ at the n-th layer can be formulated as follows:

$$z_{u_j}^{(n)} = \mathrm{MLP}^{(n)}((1 + \epsilon^{(n)}) \cdot z_{u_j}^{(n-1)} + \sum_{v \in \mathcal{N}(u_j)} z_v^{(n-1)}) \tag{1}$$

where $\mathrm{MLP}^{(n)}$ denotes the multi-layer perceptron at the n-th layer, $\epsilon^{(n)}$ is a learnable parameter that adjusts the weight of a node's own features relative to its neighbors' features, $\mathcal{N}(u_j)$ represents the

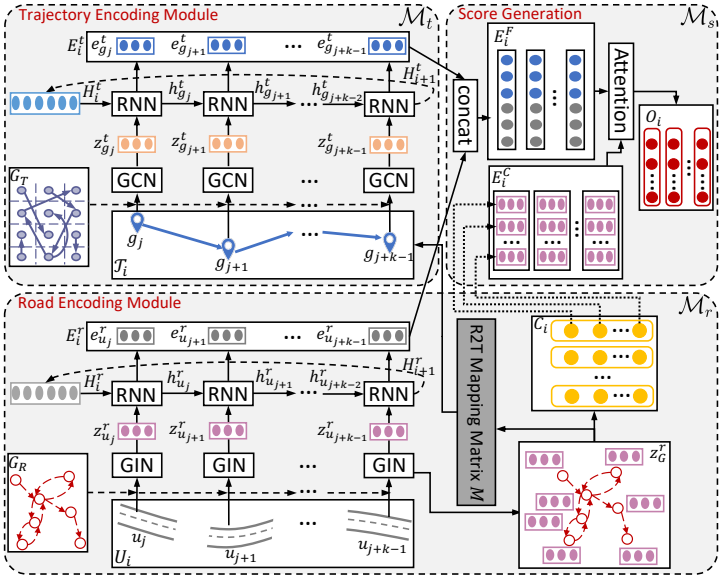


Fig. 4. The framework of feature encoding.

set of neighbor nodes of u_j, and $z_{u_j}^{(n-1)}$ and $z_v^{(n-1)}$ are the feature vectors of node u_j and its neighbors, respectively. For previously matched road segments $U_i = U_{j:j+k} = \langle u_j, u_{j+1}, \ldots, u_{j+k-1} \rangle$, we use the feature $\langle z_{u_j}^{(n)}, z_{u_{j+1}}^{(n)}, \ldots, z_{u_{j+k-1}}^{(n)} \rangle$ after n-th GIN layer as the representations $\langle z_{u_j}^r, z_{u_{j+1}}^r, \ldots, z_{u_{j+k-1}}^r \rangle$ of road segments.

Trajectory Transition Graph Construction. The map matching problem is inherently graph-natured because the road network is represented as a graph composed of nodes (links) and edges (connections). Therefore, representing trajectories in a graphical form can also unify the representation of the two types of data, thereby alleviating the impact of their heterogeneity. Motivated by this, we further design trajectory transition graph $G_T = (V_T, E_T)$ to model the trajectories as a graph to address the heterogeneity between roads and trajectories. Specifically, for trajectory set $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n\}$ and trajectory point g mapped to the grid, we construct the node set V_T and E_T, which is defined as follows:

$$V_T = \cup_{i=1}^{n} \{g \mid g \in \mathcal{T}_i\} \tag{2}$$

$$E_T = \cup_{i=1}^{n} \{(g_j, g_{j+1}, w_{(g_j, g_{j+1})}) \mid g_j, g_{j+1} \in \mathcal{T}_i, g_j \neq g_{j+1}\} \tag{3}$$

$$w_{(g_j, g_{j+1})} = \sum_{i=1}^{n} \sum_{g_j, g_{j+1} \in \mathcal{T}_i} \mathbb{I}(g_j \neq g_{j+1}) \tag{4}$$

where w is the weight of each edge, $\mathbb{I}(\cdot)$ is the indicator function, which takes a value of 1 if g_j is different from g_{j+1}, and 0 otherwise. By doing so, the trajectory transition graph G_T is able to model the relationships between trajectories from a global perspective of the trajectory set, where the edges and weights in the graph explicitly indicate the degree of correlation between the grids.

Remark. In terms of complexity, the trajectory transition graph can be constructed with time and space complexity linear to the number of points in the trajectory set, both of which can be represented as $O(\sum_{i=1}^{n} |\mathcal{T}_i|)$, where $|\mathcal{T}_i|$ denotes the number of points in $\mathcal{T}_i$. This endows the trajectory transition graph with good scalability, enabling our model to be rapidly deployed in practical application scenarios with large amounts of trajectory data, and to incrementally update the graph structure during service provision.

Trajectory Transition Graph Convolution. To enhance the consistency between trajectory and road representations and bridge their interaction, benefiting from our adoption of the same grid representation method for both trajectories and roads, we select the representations of road segments that intersect with the corresponding grid of the trajectory point as its initial features. Specifically, we construct a road-to-trajectory mapping matrix M, which distributes the road representations z_r to various nodes of the trajectory transition graph G_T based on whether the roads intersect with the nodes. It can be formulated as follows:

$$M_{ij} = \begin{cases} 1 & \text{if } g_i \cap u_j \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

where $\cap$ denotes the intersection of grid node and road segment. Then, for gird $g_i \in V_T$, its initial representation $z_{g_i}^{(0)} \in \mathbb{R}^{d_t}$ is:

$$z_{g_i}^{(0)} = \frac{\sum_{j=1}^{N_g} M_{ij} \cdot z_{u_j}^r}{\sum_{j=1}^{N_g} M_{ij}} \tag{6}$$

where N_g is the number of nodes in G_T, $z_{u_j}^r$ denotes the representation of road segments u_j of the global graph representation z_G^r. By doing this, the initial representations of the nodes in the trajectory transition graph G_T sufficiently incorporate the associated road information. This strategy for linking road and trajectory representations lays a robust groundwork for the subsequent derivation of effective trajectory representations.

By definition, all transitions from grid g_i to grid g_j are aggregated into a single edge e_{ij}, resulting in a relatively sparse Trajectory Transition Graph. Given this sparsity, Graph Convolutional Networks (GCNs) are a suitable choice for encoding the graph, as they are computationally efficient on sparse graphs [21]. Using GCNs in this context allows for efficient encoding of the trajectory while aligning well with the graph's structural characteristics. With the initial representations, we apply n layers of Graph Convolutional Networks (GCN) to get the final representations. The update rule for feature $z_{g_i}^{(n)} \in \mathbb{R}^{d_t}$ of node $g_i \in V_T$ at the n-th layer can be formulated as follows:

$$z_{g_i}^{(n)} = \sigma(\mathrm{BN}^{(n)}(z_{g_i}^{(n-1)} \times W_a^{(n)} + \sum_{v \in \mathcal{N}(g_i)} \frac{1}{\sqrt{\hat{d}_{g_i}\hat{d}_v}} z_v^{(n-1)} \times W_b^{(n)})) \tag{7}$$

where σ denotes the activation function ReLU, $\mathrm{BN}^{(n)}$ denotes the batch normalization at the n-th layer, $W_a^{(n)} \in \mathbb{R}^{d_t \times d_t}$ and $W_b^{(n)} \in \mathbb{R}^{d_t \times d_t}$ are learnable weights, $\mathcal{N}(g_i)$ denotes the set of neighboring nodes of g_i, $\hat{d}_{g_i}$ and $\hat{d}_v$ are normalized degree terms for g_i and v. For current trajectory points $\mathcal{T}_i = \mathcal{T}_{j:j+k} = \langle g_j, g_{j+1}, \ldots, g_{j+k-1} \rangle$, we use the feature after n-th GCN layer as the representations $\langle z_{g_j}^t, z_{g_{j+1}}^t, \ldots, z_{g_{j+k-1}}^t \rangle$ of trajectory.

4.2 Sequence Correlation Capture

In online map matching scenarios, we need to match newly generated trajectory points within a certain time interval. It's critical to take into account not only the historical information but also the intrinsic correlations present in the newly emerging sequence of trajectory points. In light of integrating them, a RNN model [12] is employed. Specifically, we leverage the sequential dependencies inherent in trajectory data by initializing the RNN with the hidden state derived from the previous trajectory points. This methodology facilitates a dynamic learning process, where the network progressively refines its understanding of the trajectory patterns, thereby improving the precision and efficiency of map matching process.

For road representations $\langle z^r_{u_j}, z^r_{u_{j+1}}, \ldots, z^r_{u_{j+k-1}} \rangle$ and trajectory representations $\langle z^t_{g_j}, z^t_{g_{j+1}}, \ldots, z^t_{g_{j+k-1}} \rangle$, the update of hidden states and generation of output is denoted as follows:

$$h_{x_j} = \tanh(h_{x_{j-1}} \times W_h + b_h + z_{x_j} \times W_z + b_z) \tag{8}$$

$$e_{x_j} = h_{x_j} \times W_e + b_e \tag{9}$$

where x_j denotes road segment u_j or current trajectory point g_j, h_{x_j} and e_{x_j} are hidden states and outputs, respectively. W_h, W_z, W_e are the learnable weights, b_h, b_z, b_e are the learnable bias. In particular, within our RLOMM framework's operation, $h^r_{u_{j-1}}$ and $h^t_{g_{j-1}}$ are included in the state s_i of the online MDP we model, represented as historical information H^r_i and H^t_i, respectively. Moreover, $h^r_{u_{j+k-1}}$ and $h^t_{g_{j+k-1}}$ are treated as the historical information H^r_{i+1} and H^t_{i+1} and saved for the next state s_{i+1}. After the sequence correlation capture of both trajectory and road segments, we get the trajectory embeddings $E^t_i = \langle e^t_{g_j}, e^t_{g_{j+1}}, \ldots, e^t_{g_{j+k-1}} \rangle$ and the road embeddings $E^r_i = \langle e^r_{u_j}, e^r_{u_{j+1}}, \ldots, e^r_{u_{j+k-1}} \rangle$ of state i.

Remark. In online map matching, where immediate processing is essential, RNN [12] demonstrates advantages over GRU [5] and LSTM [11] due to its simpler structure and faster computational capabilities. Additionally, compared to Transformer [43], which lacks sequential hidden state transfer, RNN maintains a continuous context flow that aligns well with MDP modeling, where the current state encapsulates all necessary past information, reflecting the Markov property essential for sequential data processing. In this scenario, the relevance of long-term historical data to current matching is minimal, as distant trajectory information has little impact on immediate decisions. This context is congruent with RNN's operational strengths and mitigates their limitation of handling long-term dependencies [24], making their efficient processing of short-term, relevant sequences particularly well-suited for the quick-paced, real-time requirements of online map matching.

4.3 Score Generation

Based on considerations of consistency, for the candidate road segments $C_i \in \mathbb{R}^{k \times n_c}$ within state i, we use the GIN in the road encoding module to obtain their embeddings, where k is the matching interval and n_c is the number of candidates. Specifically, we select the representations of the corresponding road segments from the global graph representation z^r_G as the candidates embedding $E^C_i \in \mathbb{R}^{k \times n_c \times d_r}$. Subsequently, we concatenate the trajectory embedding $E^t_i \in \mathbb{R}^{k \times d_t}$ and road embedding $E^r_i \in \mathbb{R}^{k \times d_r}$ of state i to obtain a fused embedding $E^F_i \in \mathbb{R}^{k \times (d_r + d_t)}$. Next, we apply an attention mechanism to calculate attention scores $O_i \in \mathbb{R}^{k \times n_c}$ based on the fused embedding E^F_i and candidate road segments embedding E^C_i, serving as the scores for each candidate road segment. In particular, we transform E^F_i and E^C_i respectively and use them as the query $Q = \tanh(E^F_i \times W_F + b_F)$ and key $K = \tanh(E^C_i \times W_C + b_C)$ in the attention mechanism to generate the score $O_i = Q \cdot K^\top$, where $W_F \in \mathbb{R}^{(d_r + d_t) \times d_a}$ and $W_C \in \mathbb{R}^{d_r \times d_a}$ are the learnable weights, $b_F \in \mathbb{R}^{d_a}$ and $b_C \in \mathbb{R}^{d_a}$ are the learnable bias.

5 Model Learning

In this section, we first focus on the trajectory-road representation alignment module in Sec 5.1. Next, we describe the reward evaluation in Sec. 5.2. Finally, in Sec. 5.3, we provide the details of the entire training process of the model and complexity analysis.

5.1 Trajectory-Road Representation Alignment

The core challenge of map matching lies in accurately aligning trajectories to the corresponding road network. To address this, we design a trajectory-road representation alignment module that

fundamentally enhances the ability of our model to discern and align trajectory data with road segments.

Based on the effective trajectory and road representations detailed in Sec. 4.1, we first obtain representations for trajectory requiring matching and its corresponding candidate road segments. Specifically, we use the trajectory representation $z^t_{\mathcal{T}_i}$ for trajectory $\mathcal{T}_i$ as anchor $\mathcal{A}_i \in \mathbb{R}^{k\times d_t}$, and the embeddings for candidates $E^C_i \in \mathbb{R}^{k\times n_c\times d_r}$ as sample set. Among the sample set, we select the representation for ground truth as positive samples $\mathcal{P}_i \in \mathbb{R}^{k\times d_r}$ and the representation for remaining parts as negative samples $\mathcal{N}_i \in \mathbb{R}^{k\times(n_c-1)\times d_r}$. The process can be formulated as follows:

$$\{\mathcal{A}_i, \mathcal{P}_i, \mathcal{N}_i\} = \begin{cases} \mathcal{A}_i = z^t_{\mathcal{T}_i} \\ \mathcal{P}_i = \{z^r_{u_j} \in E^C_i \mid u_j \in y_i\} \\ \mathcal{N}_i = \{z^r_{u_j} \in E^C_i \mid u_j \notin y_i\} \end{cases} \tag{10}$$

where $z^t_{\mathcal{T}_i} \in \mathbb{R}^{k\times d_t}$ is the representation for trajectory $\mathcal{T}_i$ generated by the trajectory encoding module, $z^r_{u_j} \in \mathbb{R}^{d_r}$ denotes the representation of candidate road segment u_j generated by the road encoding module, and y_i denotes the ground truth road segments for trajectory $\mathcal{T}_i$.

Then, we introduce the InfoNCE loss [41] as an unsupervised alignment loss $\mathcal{L}_a$ to bring the anchor and positive samples closer together while distancing them from negative samples.

$$\mathcal{L}_a = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp\left(\frac{\mathcal{A}_i^\top \mathcal{P}_i}{\tau}\right)}{\exp\left(\frac{\mathcal{A}_i^\top \mathcal{P}_i}{\tau}\right) + \sum_{z^r_{u_j}\in\mathcal{N}_i} \exp\left(\frac{\mathcal{A}_i^\top z^r_{u_j}}{\tau}\right)} \tag{11}$$

where N is the number of anchor-sample pairs, and τ scales the softmax function. By doing so, the robustness of the representations for trajectories and roads is enhanced, enabling effective integration to achieve better matching performance.

5.2 Reward Evaluation

After the score generation for each candidate road segment in Sec. 4.3, we get the scores $O_i \in \mathbb{R}^{k\times n_c}$ for k time step of state s_i. Then, we select the maximum value in O_i as the matching result for the current state s_i, which also serves as the action a_i:

$$a^{(n)}_i = u_{k^*}, \text{ where } k^* = \arg\max_{j\in\{1,2,\ldots,n_c\}} O^{(n)}_j \tag{12}$$

where $a^{(n)}_i$ denotes the time step n of action for state i, arg max denotes the argument of the maximum, which is used to find the index j that maximizes the value of $O^{(n)}_j$.

Upon obtaining the action a_i corresponding to the current state s_i, in alignment with the OMDP modeling, we devise a reward evaluation mechanism to ascertain the reward. Specifically, we consider the reward from four aspects: *Accuracy*, *Consecutive Success*, *Detour Penalty*, and *Road Connectivity*.

Accuracy. The primary goal of map matching is to align the observed trajectory accurately with the corresponding road segments. The matching accuracy reward is designed to directly incentivize the model to select road segments that best fit the GPS points, which is formulated as follows:

$$r_{ac} = \begin{cases} 1 & \text{if } a^{(n)}_i = y^{(n)}_{s_i} \\ -1 & \text{otherwise} \end{cases} \tag{13}$$

where $a^{(n)}_i$ denotes the action of time step n, and $y^{(n)}_{s_i}$ is the ground truth road segments for state s_i at time step n.

Consecutive Success. In real-world scenarios, trajectory points are collected sequentially as a vehicle moves. Consecutive success reward is given for consecutive successful matches along a trajectory, which motivates the model to not only focus on individual, isolated matchings but also to ensure that these matchings are logically consistent over time. This reward helps model to maintain continuity in the sequence of matched road segments, thereby improving overall accuracy from a future-oriented perspective.

$$r_{cs} = \alpha \cdot \mathbb{I}(a_i^{(n)} = y_{s_i}^{(n)} \text{ and } m \geq \theta) \tag{14}$$

where $\mathbb{I}(\cdot)$ is the indicator function, which takes a value of 1 if the condition is satisfied and 0 otherwise. m denotes the current number of consecutive success, θ is the predefined threshold, and α is the reward value.

Detour Penalty. Unnecessary detours are a significant issue in map matching, often caused by GPS errors where consecutive trajectory points erroneously match upstream of the previous point [3, 14, 36, 38]. This issue severely affects the accuracy of matches, leading to abnormal and circuitous results in the matched routes. Motivated by this, the detour penalty is implemented to discourage the model from choosing circuitous or uncharacteristically long paths that deviate significantly from the most direct route between consecutive GPS points.

$$r_{dp} = -\beta \cdot \mathbb{I}(a_i^{(n)} \neq y_{s_i}^{(n)} \text{ and } a_i^{(n)} \in H) \tag{15}$$

where H is a historical matching queue that stores recent matching results, and β is the penalty value.

Road Connectivity. Ensuring that the selected road segments are interconnected is crucial for generating viable routes in map matching. A road connectivity reward is designed to promote the selection of navigable paths and help the model understand that accurate matching is constrained by the connectivity between roads. We specifically consider the shortest path distance between the two road segments within the link connection graph G_R as the degree of connectivity. The rationale behind this is that a shorter shortest path distance signifies a stronger connection between the segments.

$$r_{rc} = \gamma \cdot \frac{1}{\delta(a_i^{(n)}, a_i^{(n-1)})} \tag{16}$$

where $\delta(\cdot)$ represents the degree of connectivity of current road and previous road, and γ is the reward value. Note that in this paper, we use a fixed road network, meaning we do not consider dynamic changes such as temporary road closures, which is a common practice in existing works [14, 18, 25, 34]. However, we can address this issue to some extent by introducing a dynamic connectivity discrimination function $\delta^d(\cdot)$. This is difficult for other existing methods, as they do not explicitly account for the impact of road connectivity on matching results, making it hard for them to extend this capability.

Finally, we can evaluate the overall reward as follows:

$$r = r_{ac} + r_{cs} + r_{dp} + r_{rc} \tag{17}$$

5.3 Model Training

To leverage the effective sequential decision-making capabilities of reinforcement learning, as shown in Algorithm 1, we develop a novel model learning algorithm that enhances the model's adaptability and robustness across diverse scenarios. For each epoch, we first fetch a batch of trajectories, candidates and ground truths $\{\mathcal{T}, C, y\}$ from whole set of trajectories, candidates and ground truths $\{\mathbf{T}, \mathbf{C}, \mathbf{y}\}$ (line 5 of Algorithm 1). Then, the systematic accumulation of transitions through *Experience Inference* is conducted, which are then stored in an experience buffer (lines

Algorithm 1: Model Learning for **RLOMM**

Input : training inputs $\mathbf{T}, \mathbf{C}, \mathbf{y}$, network module $\mathcal{M}_t$, $\mathcal{M}_r$, $\mathcal{M}_s$, trajectory transition graph G_T, link connection graph G_R, experience buffer $\mathcal{B}$, matching steps k, learning rate lr, training epochs ep, batch size bs, loss weight λ, update interval t, discount factor γ.

Output : parameters $\theta_t, \theta_r, \theta_s$, for the three parts $\mathcal{M}_t$, $\mathcal{M}_r$, $\mathcal{M}_s$.

initialize $\theta_t, \theta_r, \theta_s$ with normal distribution;
for $m \leftarrow 1 \dots ep$ **do**
  iterations $I = \lfloor \frac{|\mathbf{T}|}{bs} \rfloor$;
  for $n \leftarrow 1 \dots I$ **do**
    fetch the batch $\{\mathcal{T}, C, y\}$ from $\{\mathbf{T}, \mathbf{C}, \mathbf{y}\}$;
    inference times $IT = \lfloor \frac{|\mathcal{T}|}{k} \rfloor$;
    for $i \leftarrow 1 \dots IT$ **do**
      $s_i \leftarrow \{\mathcal{T}_i, U_i, H_i^t, H_i^r, C_i\}$;
      $a_i, r_i \leftarrow ExperienceInference(s_i, G_T, G_R)$;
      $s_{i+1} \leftarrow \{\mathcal{T}_{i+1}, U_{i+1}, H_{i+1}^t, H_{i+1}^r, C_{i+1}\}$;
      construct transition $T_i = \{s_i, a_i, r_i, s_{i+1}\}$;
      store T_i into $\mathcal{B}$ based on First In First Out;
    $\theta_t, \theta_r, \theta_s \leftarrow ModelTrain(\mathcal{B}, G_T, G_R, lr, \lambda, \gamma)$;
    update the main network with $\theta_t, \theta_r, \theta_s$;
    if $n \mod t = 0$ **then**
      using parameters of the main network to update the target network;

Function: ExperienceInference

Input : current state s_i, trajectory transition graph G_T, link connection graph G_R.

$\mathcal{T}_i, U_i, H_i^t, H_i^r, C_i \leftarrow s_i$;
$H_{i+1}^t, E_i^t \leftarrow \mathcal{M}_t(\mathcal{T}_i, H_i^t, G_T)$;
$H_{i+1}^r, E_i^r \leftarrow \mathcal{M}_r(U_i, H_i^r, G_R)$;
$E_i^C \leftarrow \mathcal{M}_r(C_i, G_R)$;
$O_i \leftarrow \mathcal{M}_s(E_i^t, E_i^r, E_i^C)$;
$a_i \leftarrow$ using Equation 12 with O_i;
$r_i \leftarrow$ using Equations 13-17 with a_i and s_i;
return a_i, r_i;

Function: ModelTrain

Input : experience buffer $\mathcal{B}$, trajectory transition graph G_T, link connection graph G_R, learning rate lr, loss weight λ, discount factor γ.

random sample transitions T from $\mathcal{B}$;
training iterations $TI = |T|$;
for $i \leftarrow 1 \dots TI$ **do**
  $s_i, a_i, r_i, s_{i+1} \leftarrow T_i$;
  $q \leftarrow Q(s_i, a_i; \theta_{main})$;
  $a_{i+1}^* \leftarrow \arg\max_{a_{i+1}} Q(s_{i+1}, a_{i+1}; \theta_{main})$;
  $q_{target} \leftarrow r_i + \gamma \cdot Q(s_{i+1}, a_{i+1}^*; \theta_{target})$;
  $\mathcal{A}_i, \mathcal{P}_i, \mathcal{N}_i \leftarrow$ using Equation 10;
  $\mathcal{L} \leftarrow$ using Equations 11, 18-19 and λ;
return $\theta_t, \theta_r, \theta_s$;

Table 1. Complexity Analysis of the Main Methods

	#Params	Complexity
MDP	$O(n_R \times c)$	$O(n_T \times \frac{l_T}{a} \times K \times n_R \times c)$
DNN	$O(n_R \times d)$	$O(n_T \times \frac{l_T^2}{2a} \times n_R \times d)$
HMM	$O(n_R \times c)$	$O(n_T \times l_T \times c \times M_{\max})$
RLOMM	$O(d^2)$	$O(n_T \times \frac{l_T}{a} \times d^2)$

7-12 of Algorithm 1). Next, the model continuously samples from these stored transitions, enabling robust *Model Train* that incorporates a wide range of experiences (lines 13-16 of Algorithm 1). Through this alternating process, this learning approach enhances the model's generalization ability while reducing the risk of overfitting, achieving high adaptability and robustness, making it suitable for real-world applications.

Experience Inference. The process begins by using the model to encode features and generate the scores (lines 1-5 of Function: ExperienceInference), which are detailed in Sec 4. Then, we select the action a_i and evaluate the reward for action a_i under state s_i to obtain the reward r_i that encompasses multiple aspects (lines 6-7 of Function: ExperienceInference). Together with the next state s_{i+1}, we construct a transition $T_i = \{s_i, a_i, r_i, s_{i+1}\}$ as a piece of experience and store it in the experience buffer $\mathcal{B}$. We refer to this process as the *experience inference* process, and execute it multiple times to accumulate a sufficient amount of transitions.

Model Train. After a certain number of experience inferences, the model randomly samples a batch of transitions from the experience buffer $\mathcal{B}$ for experience replay. Specifically, we employ a deep Q-learning method, which learns the value of an action in a particular state, intending to maximize the total reward received [28]. Moreover, to reduce the overestimation of action values that often occurs in standard Q-learning, we apply Double DQN [42] to increase the stability of our training. In particular, we use two networks with identical structure that are composed of $\mathcal{M}_t$, $\mathcal{M}_r$, and $\mathcal{M}_s$, namely the main network and the target network. Note that we only train the main network, and use its parameters to update the target network periodically (lines 15-16 of Algorithm 1). We first use the main network to estimate the q-value q (line 5 of Function: ModelTrain). Then, we use the main network to select the action a^*_{i+1} with the max q-value (line 6 of Function: ModelTrain). Next, we use the target network to estimate the Q-value of taking action a^*_{i+1} under next state s_{i+1}, multiply it by the discount factor γ, and add it with reward r_i to compute the target q-value q_{target} (line 7 of Function: ModelTrain). Finally, we apply the Huber Loss [16] as the Temporal Difference (TD) loss for Q-learning, which is formulated as follows:

$$\mathcal{L}_{td} = \begin{cases} 0.5 \times (q - q_{\text{target}})^2, & \text{if } |q - q_{\text{target}}| < 1 \\ |q - q_{\text{target}}| - 0.5, & \text{otherwise} \end{cases} \tag{18}$$

where q is the estimated q-value and q_{target} is the target q-value. Combined with the alignment loss $\mathcal{L}_a$ and corresponding weights λ, we can compute the overall learning objective as follows:

$$\mathcal{L} = \mathcal{L}_{td} + \lambda \cdot \mathcal{L}_a \tag{19}$$

Complexity Analysis. We compare **RLOMM** with **MDP-based** [45], **DNN-based** [18, 25, 34] offline methods, and **HMM-based** [14, 30, 47] online method in terms of complexity. We analyze the complexity of parameters and computation for the four types of methods, with the results shown in Table 1. Specifically, we model the problem input as follows. The road network contains n_R road segments. The number of trajectories is n_T, the longest trajectory length is l_T, and each time interval has a new trajectory points. For each trajectory point, the maximum candidates road segments is c. The dimension of all hidden representations is d.

Table 2. Statistics of Datasets

	Beijing	Porto	Chengdu
Time Span	03/01-06/23, 2022	01/07-06/30, 2014	11/01-11/30, 2016
Cover Area	8.69km × 7.67km	5.86km × 3.34km	8.32km × 8.36km
Road Segments	8.5K	4.2K	6.5K
Trajectories	10K	129.5K	898.1k
Sampling frequency	15s	15s	15s
Average Error	23.9m	7.8m	12.7m

For **MDP** methods, the parameters mainly consist of transition probabilities for each state-action pair, which has the complexity of $O(n_R \times c)$. The computational complexity primarily arises from calculating the value function for each state-action pair during the iterative process (e.g., value iteration). Assuming the number of iterations required for convergence is K, the time complexity for a single computation is $O(K \times n_R \times c)$. In the online scenario, the method needs to recompute the MDP multiple times, with the trajectory length increasing each time. The cumulative computational complexity becomes $O(n_T \times \frac{l_T}{a} \times K \times n_R \times c)$, where $\frac{l_T}{a}$ represents the number of calculations for each trajectory. For **DNN** methods, the FC layer and the road segment embedding layer have $O(n_R \times d)$ parameters, where $n_R >> d$, making these parameters significantly greater than the $O(d^2)$ parameters of other layers. This results in an overall parameter complexity of $O(n_R \times d)$. In the online scenario, the model needs to recompute over increasingly longer trajectories, leading to an accumulated computational complexity of $O(n_T \times \frac{l_T^2}{2a} \times n_R \times d)$, where the term $\frac{l_T^2}{2a}$ comes from summing the lengths of all steps for a trajectory, which forms an arithmetic series. For **HMM** methods, the parameter complexity is $O(n_R \times c)$, which comes from emission probabilities and transition probabilities between states. Assuming the maximum number of transitions per candidate is $M_{\max}$, at each time step, HMM methods can perform incremental online matching, with a total time complexity of $O(n_T \times l_T \times c \times M_{\max})$. For our **RLOMM** method, it does not require embedding encoding of n_R road segments, therefore the parameter complexity is $O(d^2)$. Since our method efficiently retains historical information and can match a trajectory points at a time, the time complexity is reduced to $O(n_T \times \frac{l_T}{a} \times d^2)$.

Remark: The complexity of **MDP** and **DNN** method has the large term n_R, K and the quadratic term $\frac{l_T^2}{2a}$, making them inefficient because they redundantly process the entire trajectory. In contrast, in our method, d is much smaller than n_R, and $\frac{l_T}{a}$ achieves optimal complexity with respect to the trajectory length term. Additionally, since $M_{\max}$ is relatively small compared to n_R, the complexity of **HMM** methods is relatively acceptable. However, it's term l_T still exceeds our method's $\frac{l_T}{a}$. Therefore, our **RLOMM** achieves superior efficiency by eliminating redundant computations, making it well-suited for real-world applications.

6 Experiments

6.1 Experimental Setup

Datasets. As shown in Table 2, we conduct a series of experiments based on three real-world datasets.

(1) **Beijing** [25]: This dataset is collected from the real-world application Tencent Maps[1] and is a large-scale dataset in terms of the number of road segments.

(2) **Porto** [1]: This dataset of trajectories is open-sourced on Kaggle. The road network of this dataset is extracted from OpenStreetMap[2].

[1]https://map.qq.com

[2]https://www.openstreetmap.org

(3) **Chengdu** [2]: This dataset is open-sourced by DiDi to support related research. The road network of this dataset is extracted from OpenStreetMap[2]. This dataset contains a large number of trajectories, making it suitable for evaluating the scalability of various methods in large-scale data scenarios.
(4) **Data Labels**: In the map matching problem, data labels typically come from two sources: real-world collection or HMM annotation under high sampling rates. For Beijing, thanks to work by Tencent Maps, the labels are entirely real, which is rare in the realm of trajectory data. For Porto and Chengdu, we use HMM to annotate the data under high sampling rates (15s). This is a common practice in the related works [18, 27, 38], as the effectiveness of HMM for high-sampling-rate data is already a consensus. In practical scenarios, high-sampling-rate data can be used to generate high-quality labels, serving as reliable training data to support the training of deep learning methods. Once the model is trained, deploying it in low-sampling-rate scenarios is more cost-effective, as it avoids the communication, storage, and computational overhead associated with high-sampling-rate data. Therefore, different sampling rates can be utilized at different stages, ensuring the quality of training data while balancing cost and performance in practical deployment scenarios.
(5) **Training, Validation and Test**: We divide a dataset into training, validation and test data with the splitting rate of 7:2:1. We set the sampling rate of the dataset as 50%, 25%, and 12.5%, corresponding to the time intervals of 30s, 60s, and 120s. In particular, it makes the trajectory points sparse and therefore challenging for map matching algorithms. We set the matching interval to 120 seconds, which is a common processing interval in practical applications [25], thus the number of matching steps k for sampling rates of 50%, 25%, and 12.5% is 4, 2 and 1. The reward value α, β, γ is set to 0.01, 0.05 and 0.02, respectively. Moreover, we set 10 as the number of candidates n_c, thereby selecting 10 candidate road segments closest to each trajectory point. For Beijing, Porto and Chengdu, this allows our model to consider nearly all possible road segments within distances of 160 meters, 140 meters and 175 meters from the trajectory points, respectively. This is already significantly larger than the Average Error of 23.9 meters for Beijing, 7.8 meters for Porto and 12.7 meters for Chengdu.
Baseline methods. We compare the following baselines:

- **MDP**: We employ the value iteration algorithm based on Markov Decision Process (MDP) to perform map matching.
- **HMM** [30]: Hidden Markov Model (HMM) is a classical probabilistic framework that models potential road segments as states and GPS data as observations.
- **FMM** [47]: This is a HMM-based method store shortest paths in a precomputed table with quick hash table searches, which achieves high processing speed.
- **AMM** [14]: This is the current state-of-the-art method for online map matching. It designs a collaborative evaluation model with a retrospective correction mechanism.
- **MTrajRec** [34]: This is a method that innovatively integrates trajectory recovery and map matching for urban applications, using a sequence-to-sequence learning model to enhance low-sampling-rate GPS data.
- **L2MM** [18]: This method generates representations of low-quality trajectories through high-frequency trajectory enhancement and data distribution augmentation, and incorporates mobility patterns into the map matching task.
- **GraphMM** [25]: This is the current state-of-the-art method for offline map matching, which proposes a graph-based solution to capture the correlations of trajectories and road segments.

For the offline map matching methods, we adjust them to online form to suit online scenarios. Specifically, because offline methods lack the transmission of historical information of the current matching trajectory, we perform incremental multiple matches to meet the requirements of online

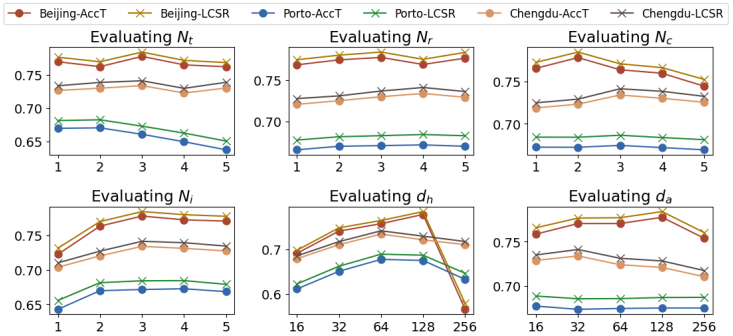


Fig. 5. AccT & LCSR vs. Hyper-parameters

scenarios. Note that it's fair for them, as these offline methods obtain all the currently available trajectories each time a match is made, which is consistent with the setting of offline matching problem they aim to solve.

Evaluation Metrics. We evaluate our proposed methods and baseline methods based on two metrics: AccT (Trajectory-level Accuracy) and LCSR (Longest Common Subsequence Ratio). Compared to traditional metrics such as precision and recall, these two metrics emphasize the sequential nature of matching, which is crucial for online scenarios and adopted by real-world applications [25]. Specifically, for each trajectory $\mathcal{T}_i$ in trajectory set T, suppose the ground truth and matched result of model of road segments is represented as $\langle u_1, \cdots, u_{l_i} \rangle$ and $\langle \hat{u}_1, \cdots, \hat{u}_{l_i} \rangle$, the accuracy of trajectory set T is computed as follows: $AccT(T) = \frac{1}{n} \sum_{i=1}^{n} \frac{\sum_{j=1}^{l_i} \mathbb{I}(u_j = \hat{u}_j)}{l_i}$, where n is number of trajectories in trajectory set T, l_i is the length of trajectory $\mathcal{T}_i$, and $\mathbb{I}(\cdot)$ is the indicator function, which takes a value of 1 if the matched road segment is identical to the ground truth, and 0 otherwise. The LCSR metric characterizes the similarity between two sequences and is well-suited for evaluating tasks like online map matching that emphasize the order of sequences. In particular, for trajectory set T, it is compute as follows: $LCSR(T) = \frac{1}{n} \sum_{i=1}^{n} \frac{LCS(\mathcal{T}_i, \hat{\mathcal{T}}_i)}{l_i}$, where $LCS(\cdot)$ is longest common subsequence function.

Experimental Settings. All deep learning methods were implemented with PyTorch 2.0.1 and Python 3.11.5, and trained with a Tesla V100 GPU. All rule-based methods were implemented with python 3.11.5, and tested with a Intel Xeon Gold 6148 CPU @ 2.40GHz. The platform ran on Ubuntu 18.04 OS. In addition, we used Adam [20] as the optimization method with the mini-batch size of 512. The learning rate is set as 0.001, and the training epoch is set as 200, and an early stopping mechanism is adopted.

6.2 Setting of Model's Hyper-parameters

We consider the following hyper-parameters: (1) the number of recurrent neural network (RNN) layers (N_t and N_r) in trajectory encoding module and road encoding module; (2) the number of graph convolution network (GCN) layers (N_c) and graph isomorphism network (GIN) layers (N_i); (3) the settable-dimension size of RNN hidden representations for trajectory and road (d_h); (4) the settable-dimension size of attention mechanism (d_a). In particular, given a hyper-parameter, we first select its value range according to the experience under some constraints (e.g., the limitation of GPU memory). Then, we conduct experiments on the validation set of Beijing, Porto and Chengdu to determine its optimal value. By default, we conduct experiments at a sampling rate of 50%. As shown in Fig. 5, we plot the AccT and LCSR for different hyper-parameters. In summary, we set each hyper-parameter with the value corresponding to the optimal performance as follows: (1) For Beijing, we have $N_t = 3$, $N_r = 3$, $N_c = 2$, $N_i = 3$, $d_h = 128$, $d_a = 128$. (2) For Porto, we have $N_t = 2$,

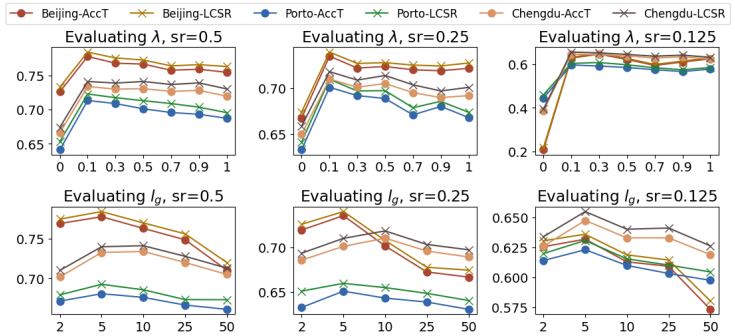


Fig. 6. AccT & LCSR vs. the loss weight λ and the grids side length l_g

$N_r = 4$, $N_c = 3$, $N_i = 4$, $d_h = 64$, $d_a = 16$. (3) For Chengdu, we have $N_t = 3$, $N_r = 4$, $N_c = 3$, $N_i = 3$, $d_h = 64$, $d_a = 32$.

6.3 Effectiveness of Loss Weight and Partition

To fine-tune the loss weight λ, we vary it from 0 to 1 when training. We compute the two metrics for the validation data. The result is shown in Fig. 6, from which we can observe a noticeable improvement in model performance as λ varies from 0 to 0.1. The improvement in performance proofs the effectiveness of our alignment module, particularly beneficial for scenarios with extremely sparse trajectory points. Based on the majority voting rule, the best values of λ for Beijing, Porto and Chengdu are both 0.1, which is set as the default values in subsequent experiments.

Additionally, we further optimize the grid partition by adjusting the side length l_g of grids, the outcomes are illustrated in Fig. 6. In essence, the size of grids represents the granularity of regions, which determines the precision and generalization of the trajectories. Specifically, finer granularity can provide more precise GPS coordinates, while coarser granularity can enhance the generalizability of the trajectory. By default, the grids side length of Beijing and Porto are optimally set to 5m, and for Chengdu it is set to 10m.

Table 3. Effectiveness Results on Test Data

	Beijing						Porto						Chengdu					
	50% (30s)		25% (60s)		12.5% (120s)		50% (30s)		25% (60s)		12.5% (120s)		50% (30s)		25% (60s)		12.5% (120s)	
	AccT	LCSR	AccT	LCSR	AccT	LCSR	AccT	LCSR	AccT	LCSR	AccT	LCSR	AccT	LCSR	AccT	LCSR	AccT	LCSR
MDP	50.18%	0.5172	49.19%	0.5081	46.93%	0.4730	52.07%	0.5281	48.69%	0.5003	47.35%	0.4831	51.05%	0.5174	46.80%	0.4788	45.21%	0.4611
HMM	53.55%	0.5462	51.43%	0.5290	51.19%	0.5226	52.58%	0.5359	50.51%	0.5209	50.25%	0.5077	52.93%	0.5388	50.10%	0.5125	46.61%	0.4780
FMM	54.38%	0.5471	53.29%	0.5431	51.37%	0.5281	53.59%	0.5391	52.13%	0.5290	48.91%	0.4964	53.12%	0.5390	52.40%	0.5319	48.95%	0.5003
AMM	58.31%	0.5917	57.21%	0.5819	51.85%	0.5289	56.22%	0.5693	55.83%	0.5649	51.03%	0.5227	57.72%	0.5843	56.13%	0.5700	50.41%	0.5122
MTrajRec	61.54%	0.6352	55.92%	0.5797	47.06%	0.4971	57.53%	0.5912	55.77%	0.5641	52.27%	0.5399	60.57%	0.6172	57.48%	0.5881	50.89%	0.5207
L2MM	62.39%	0.6355	57.98%	0.5951	52.40%	0.5367	56.41%	0.5759	55.51%	0.5649	53.76%	0.5488	59.19%	0.6021	56.90%	0.5801	50.10%	0.5149
GraphMM	66.01%	0.6768	61.96%	0.6302	55.06%	0.5616	59.68%	0.6172	57.07%	0.5860	52.33%	0.5406	62.14%	0.6328	60.31%	0.6142	53.44%	0.5455
NC	51.37%	0.5247	48.29%	0.4955	41.76%	0.4291	44.67%	0.4576	41.50%	0.4266	38.71%	0.3993	53.29%	0.5413	50.61%	0.5174	42.53%	0.4341
NI	44.31%	0.4521	43.27%	0.4429	38.82%	0.3956	33.21%	0.3489	32.39%	0.3345	30.89%	0.3191	45.56%	0.4667	43.91%	0.4503	41.54%	0.4237
NM	69.66%	0.7036	68.72%	0.6956	60.48%	0.6135	59.24%	0.6007	57.47%	0.5841	54.93%	0.5603	63.28%	0.6409	60.13%	0.6129	52.98%	0.5388
RLOMM	**77.78%**	**0.7843**	**73.53%**	**0.7398**	**63.19%**	**0.6360**	**68.05%**	**0.6924**	**65.12%**	**0.6599**	**62.31%**	**0.6308**	**73.39%**	**0.7412**	**71.02%**	**0.7183**	**64.75%**	**0.6569**

6.4 Effectiveness Comparison

Apart from comparing **RLOMM** with baseline methods, we replace our **RLOMM** by three variations to conduct ablation study, namely **NC**, **NI**, and **NM**, to evaluate the effectiveness of different parts of encoding in **RLOMM**. In **NC**, we use fully connected (FC) layers to replace graph convolutional network in trajectory encoding module. In **NI**, we use FC layers to replace graph isomorphism network in road encoding module. In **NM**, we replace the road-to-trajectory mapping mechanism with raw graph node coordinates. Note that the effectiveness of our proposed trajectory-road representation alignment module is evaluated in Sec. 6.3.

Table 4. Efficiency of Different Methods

	memory size (MB)			training time (s)			matching time (s)		
	Beijing	**Porto**	**Chengdu**	**Beijing**	**Porto**	**Chengdu**	**Beijing**	**Porto**	**Chengdu**
MDP	1819MB	2039MB	2122MB	-	-	-	389.14s	361.15s	599.51s
HMM	1209MB	1388MB	1361MB	-	-	-	427.97s	380.05s	589.08s
FMM	897MB	931MB	981MB	-	-	-	1.13s	1.02s	1.87s
AMM	957MB	1013MB	1124MB	-	-	-	3.42s	3.05s	5.16s
MTrajRec	9045MB	12428MB	11265MB	182.4s	2200.2s	25672.4s	51.22s	42.27s	73.68s
L2MM	9087MB	11875MB	10898MB	189.1s	2314.2s	27032.2s	6.71s	5.26s	9.10s
GraphMM	8537MB	11752MB	10378MB	48.4s	645.2s	7311.4s	8.06s	6.96s	11.18s
RLOMM	2530MB	2299MB	2357MB	11.9s	126.4s	1507.8s	1.09s	0.95s	1.65s

Table 3 reports the results of all methods with different sampling rate, and we have the following observations:
(1) Traditional rule-based offline methods (i.e., **MDP**, **HMM**, **FMM**) perform the worst because they are designed to model offline complete trajectories and are unable to learn general patterns from sparse and large-scale online scenarios, lacking robustness.
(2) When examining the results of **NC**, **NI**, **NM**, and **RLOMM**, we observe that the graph neural networks in the encoding part are crucial, which validates the effectiveness of our graph structure in addressing the trajectory-road heterogeneity. Furthermore, using road representations as the initial representations for nodes in the trajectory transition graph to bridge the representation interaction also further enhances performance, as **NM** performs worse.
(3) **RLOMM** exhibits the best performance across all metrics. For instance, **RLOMM** outperforms the best existing methods (e.g., **GraphMM**) by 17% on AccT for the test data of Beijing at a sampling rate of 50%. In Porto, the best baseline method, GraphMM, achieves lower accuracy at the highest sampling rate than our method does at the lowest sampling rate. This demonstrates that, by using our method, service providers can reduce the sampling rate of trajectories to save costs while still achieving better performance than existing methods, as our method maintains strong performance even at lower sampling rates.
(4) When comparing metrics across different datasets, the methods perform similarly on Beijing and Chengdu, with slightly better results than that on Porto. We attribute this primarily to the Porto dataset's higher road density, which results in a greater number of road segments close to the correct one, making the matching task more challenging.
(5) The LCSR is consistently better than the AccT across all methods. By definition of LCSR and AccT, AccT penalizes every incorrect match equally while LCSR emphasizes the accuracy of the entire sequence match, so when there are matching errors, the decline in AccT will be more pronounced than in LCSR. Moreover, this indicates that most map matching methods are not prone to frequent alternations between correct and incorrect matches, which possess good global matching capabilities and fault tolerance.
(6) As the sampling rate decreases, the performance of all methods declines, with the impact being more significant for learning-based methods due to the inevitable influence of data sparsity on the quality of the models' learning.

6.5 Efficiency Comparison

For efficiency evaluation, we record the memory usage, training time and inference time at a sampling rate of 50%. The memory usage represents the required memory size for applying corresponding methods, which is used to evaluate the memory efficiency. The training time is used to evaluate the offline learning efficiency for neural network based methods. In particular, we compute the average time of an epoch for each method. The matching time (i.e., inference time) can evaluate the online matching efficiency. Specifically, we record the time for each method to match 10K trajectories at a sampling rate of 0.5. Note that for rule-based methods running on the CPU,

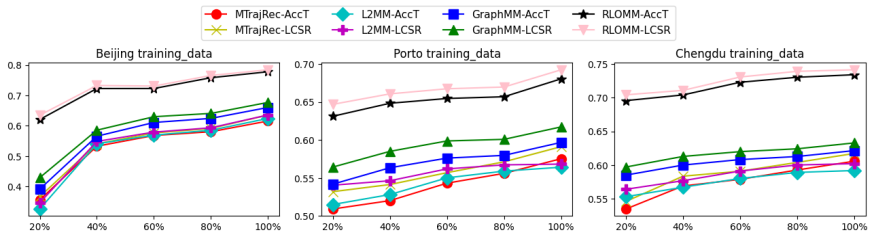


Fig. 7. AccT & LCSR vs. the Scalability.

we apply parallel mechanisms to ensure fairness compared to learning-based methods running on the GPU with batch. The outcomes of these evaluations are presented in Table 4. From our observations, we deduce the following:
(1) Rule-based methods (i.e., **MDP**, **HMM**, **FMM**, and **AMM**) require less memory than learning-based methods (i.e., **MTrajRec**, **L2MM**, **GraphMM** and **RLOMM**) because they do not need to keep network structures in memory. Among learning-based methods, **RLOMM** is the most memory efficient.
(2) Offline map matching methods, such as **MDP**, **HMM**, **MTrajRec**, **L2MM** and **GraphMM**, which have not been specifically optimized for efficiency, perform poorly in terms of online matching efficiency. This demonstrates that the naive strategies for adapting offline methods to online scenarios are unsuitable. Additionally, among all HMM-based models, the efficiency of **HMM** itself is significantly lower than other models due to its lack of improvements to enhance efficiency. Furthermore, the lack of support for online scenarios in naive **MDP** and **HMM** models leads to repeated invocations, making their matching time in online scenarios almost unacceptable for practical applications.
(3) Among all learning-based methods, our model exhibits the best offline learning efficiency. Furthermore, among all methods, our model demonstrates the best online matching efficiency. Compared to the method that specifically designed for online scenarios (i.e., **AMM**), our model achieves more than 3× improvement in online matching efficiency, making it highly practical and feasible for real-world applications. This is attributed to the seamless integration of our designed online MDP and reinforcement learning methods, which effectively extract key information while dynamically updating and utilizing historical data. Our approach thus avoids redundant computations to achieve high efficiency.
(4) When comparing results across different datasets, training one epoch takes significantly longer on the Chengdu due to the higher number of trajectories. For online matching time, Chengdu requires more time as well, primarily because its trajectories are generally longer than those in Porto and Beijing.

6.6 Scalability Comparison

To compare the scalability of the learning based methods (i.e., **MTrajRec**, **L2MM**, **GraphMM**, and **RLOMM**), we train different models by varying the training data size. In particular, we sample 20%, 40%, 60%, 80% and 100% from the training data, and collect the associated AccT and LCSR of the online prediction over the test data at a sampling rate of 50%. From Fig. 7, we have the following observations:
(1) All methods perform better with an increased amount of training data, since a larger dataset encompasses a wider range of situations, allowing the model to learn more effectively.
(2) Our **RLOMM** consistently achieves the best performance. Notably, in many instances (e.g., the AccT and LCSR metrics for Beijing), as the sampling rate decreases, the performance gap between our **RLOMM** and other methods widens.

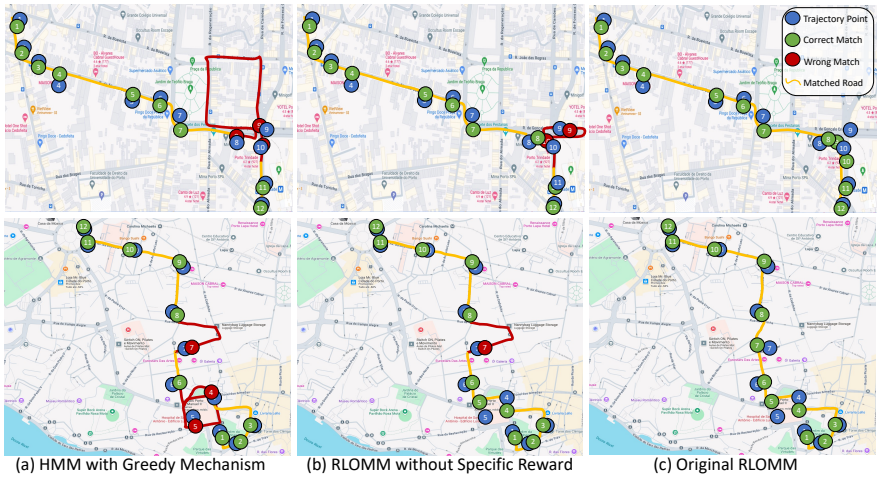


Fig. 8. An illustration of matching result for two trajectories of Porto. The blue points are the trajectory points need to be matched, the green points are the correct matches and the red points are the wrong matches.

(3) The performance degradation varies across datasets due to differences in the amount of trajectory data. Beijing, with its relatively smaller number of trajectories, experiences the most significant decline, whereas Chengdu shows notably less degradation.

6.7 Case Study

To visually illustrate the matching result and further demonstrate the superiority of our method over the greedy approach, we conduct a case study. Specifically, we select two trajectories from Porto to compare the matching performance of three methods: the **HMM** with greedy mechanism, **RLOMM** without detour penalty and road connectivity reward designed in Sec. 5.2, and the original **RLOMM**. As shown in Fig. 8, we have the following observations:
(1) For HMM with a greedy mechanism, its matching results often exhibit incorrect detours because of selecting closer road segments to the trajectory point (e.g., the 9th point of the first trajectory, and the 4th and 5th point of the second trajectory). In particular, such greedy-induced matching errors often impact subsequent matches. For example, to maintain road continuity, the incorrect match of the 4th point in the second trajectory directly leads to the incorrect match of the 5th point. This inherent flaw in greedy methods results the unsuitability of the HMM approach for online scenarios.
(2) Comparing RLOMM with and without the detour penalty and road connectivity reward reveals that, thanks to the carefully designed rewards, our reinforcement learning-based framework can consider each match from a global perspective. For instance, in the case of the first trajectory, since the erroneous matching of the 9th trajectory point is not connected to the preceding 8th matched road, the road connectivity reward guides the model to avoid this incorrect matching. Furthermore, when matching the 10th trajectory point, the detour penalty encourages the model to consider previously unmatched road segments, leading to a correct match. The various rewards we design work in conjunction to guide the model in providing accurate matching results when faced with complex and error-prone scenarios.

7 Related Work

Map matching problem can be classified to two broad categories [3]: *Offline Map Matching* and *Online Map Matching*. For *Offline Map Matching*, researchers focus on matching complete trajectories

in offline scenarios, which aims for optimal matching route with less constraint on processing time [22]. For *Online Map Matching*, trajectory points are continuously sampled and processed in a streaming manner, which imposes high demands on real-time performance.

7.1 Offline Map Matching

Early research on the offline map matching problem was primarily based on the concept of similarity comparison, that is, by defining similarity to identify the road segments most similar to the target trajectory. The authors in [33, 53, 55] design algorithms with different measure of similarity (e.g., spatial distance, longest common subsequence) to find the most similar road segment as the matching result.

When maps and trajectories are complex, similarity-based methods often perform poorly. To enhance performance, Hidden Markov Models (HMM) and HMM-enhanced models are used. HMMs assume that the current state depends on the previous one and consider the correct road segments as hidden states of the trajectory points, which are the observations. Newson et al. [30] are the first to use HMM to address the offline map matching problem. Yang et al. [47] integrate HMM with precomputation to achieve high processing speed. The authors in [45] use MDP to address the limitations of HMM, designing dynamic outlier resolution and improved preprocessing functions for map matching. Moreover, considering both the spatial structures and the temporal constraints of the trajectories, [13, 26, 51] design methods that characteristic relationships between GPS points. However, although HMM is widely used in commercial software, its performance in low-sampling-rate scenarios is poor, limiting its potential for cost reduction and efficiency improvement [35].

Recently, deep-learning-based map matching methods have attracted the attention of researchers. In order to learn general patterns from trajectory data, [7, 52] build deep learning based models to utilize all the trajectory data for joint training and knowledge sharing. The authors in [34] propose MTrajRec to recover the fine-grained points in trajectories and map match them on the road network in an end-to-end manner. Jiang et al. [18] propose L2MM to generate high-quality representations of low-quality trajectories through high-frequency trajectory enhancement. Liu et al. [25] present a graph-based approach that incorporates conditional models to leverage various road and trajectory correlations. As for cellular trajectory map matching, [27, 37, 38] design different learning methods (e.g., multi-relational graph learning, federated learning, heuristic reinforcement learning optimizer) to learn the mapping function from cellular trajectory points to road segments. As remote sensing imagery is innovatively applied to various spatiotemporal data tasks [4, 19], it is conceivable that the map matching task could similarly leverage diverse types of novel data in the future.

7.2 Online Map Matching

For online map matching, research primarily focuses on the Hidden Markov Model (HMM) or enhanced versions of HMM, focusing on improving efficiency.

Considering noise and sparseness of GPS data, Goh et al. [9] propose an enhanced HMM with a Variable Sliding Window method to integrate spatial, temporal, and topological information to map the GPS trajectories to the road network in real time. To further address complex and changeable city traffic conditions, Liang et al. [23] design OLMM, which uses online learning to improve accuracy without requiring any prior human labeling. [8] proposes an online map matching algorithm based on a second-order HMM with an extended Viterbi algorithm and a self-adaptive sliding window mechanism to enhance trajectory data processing accuracy in complex urban road networks. To cope with high noise, Jagadeesh et al. [17] design a HMM to generate partial map-matched paths in an online manner, and use a route choice model to reassess each HMM-generated partial path along with a set of feasible alternative paths. Furthermore, the authors in [40] propose an algorithm that

replaces future GPS points with a probabilistic route prediction model. Recently, Hu et al. [14] design an adaptive online map matching algorithm called AMM, which calibrate GPS observation data for various measurement errors and under complex urban conditions. Additionally, in addressing the issue of cellular trajectory map matching, which involves handling trajectories based on cellular positioning data, [29] design an incremental HMM algorithm that combines digital map hints and a number of heuristics to reduce the noise and provide real-time estimations.

However, aforementioned online methods are limited by their rule-based nature, which prevents them from fully capturing the spatio-temporal correlations [32]. Additionally, these methods are predominantly greedy in their matching process, ignoring the impact of current matches on future outcomes, leading to poor robustness and accuracy [3, 15].

8 Conclusion

This paper introduces **RLOMM**, a reinforcement learning framework designed for the online map matching problem, offering high efficiency and robustness tailored for real-world applications. We have addressed three key limitations present in existing methods: low efficiency, poor robustness, and insufficient handling of trajectory-road heterogeneity. Initially, we modeled the problem as an Online Markov Decision Process with efficient integration of real-time and historical information. Next, we adopted reinforcement learning and meticulously designed rewards to enhance dynamic adaptability and robustness. Additionally, we incorporated an effective graph structure for trajectories and roads to address the heterogeneity, facilitating better fusion between them. Moreover, we proposed a trajectory-road representation alignment module to enhance the robustness of representations and reduce their distance in latent space. Extensive evaluations on three real-world datasets have confirmed the efficacy of **RLOMM**.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No.62425203, No.62032003, No.U21B2016). Haitao Yuan is the corresponding author of the work.

References

[1] 2015. kaggle. https://www.kaggle.com/c/pkdd-15-taxi-trip-time-prediction-ii
[2] 2021. GAIA. https://outreach.didichuxing.com/research/opendata/
[3] Pingfu Chao, Yehong Xu, Wen Hua, and Xiaofang Zhou. 2020. A Survey on Map-Matching Algorithms. In *ADC*, Renata Borovica-Gajic, Jianzhong Qi, and Weiqing Wang (Eds.), Vol. 12008. Springer, 121–133.
[4] Minxiao Chen, Haitao Yuan, Nan Jiang, Zhifeng Bao, and Shangguang Wang. 2024. Urban Traffic Accident Risk Prediction Revisited: Regionality, Proximity, Similarity and Sparsity. In *CIKM*. ACM, 281–290.
[5] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*, Alessandro Moschitti, Bo Pang, and Walter Daelemans (Eds.). ACL, 1724–1734.
[6] Sean R Eddy. 1996. Hidden markov models. In *Current opinion in structural biology*, Vol. 6. Elsevier, 361–365.
[7] Jie Feng, Yong Li, Kai Zhao, Zhao Xu, Tong Xia, Jinglin Zhang, and Depeng Jin. 2022. DeepMM: Deep Learning Based Map Matching With Data Augmentation. In *TMC*, Vol. 21. 2372–2384.
[8] Xiao Fu, Jiaxu Zhang, and Yue Zhang. 2021. An Online Map Matching Algorithm Based on Second-Order Hidden Markov Model. In *JAT*, Vol. 2021. 9993860.
[9] Chong Yang Goh, Justin Dauwels, Nikola Mitrovic, Muhammad Tayyab Asif, Ali Oran, and Patrick Jaillet. 2012. Online map-matching based on Hidden Markov model for real-time traffic sensing applications. In *ITSC*. IEEE, 776–781.
[10] Chenjuan Guo, Bin Yang, Jilin Hu, and Christian S. Jensen. 2018. Learning to Route with Sparse Trajectory Sets. In *ICDE*. IEEE Computer Society, 1073–1084.
[11] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. In *Neural Comput.*, Vol. 9. 1735–1780.
[12] John J Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities.. In *PNAS*, Vol. 79. National Acad Sciences, 2554–2558.

[13] Gang Hu, Jie Shao, Fenglin Liu, Yuan Wang, and Heng Tao Shen. 2017. IF-Matching: Towards Accurate Map-Matching with Information Fusion. In *TKDE*, Vol. 29. 114–127.
[14] Hanwen Hu, Shiyou Qian, Jingchao Ouyang, Jian Cao, Han Han, Jie Wang, and Yirong Chen. 2023. AMM: An Adaptive Online Map Matching Algorithm. In *TITS*, Vol. 24. 5039–5051.
[15] Zhenfeng Huang, Shaojie Qiao, Nan Han, Chang-an Yuan, Xuejiang Song, and Yueqiang Xiao. 2021. Survey on vehicle map matching techniques. In *CAAI Trans. Intell. Technol.*, Vol. 6. 55–71.
[16] Peter J. Huber. 1964. Robust Estimation of a Location Parameter. In *The Annals of Mathematical Statistics*, Vol. 35. 73–101.
[17] George R Jagadeesh and Thambipillai Srikanthan. 2017. Online map-matching of noisy and sparse location data with hidden Markov and route choice models. In *TITS*, Vol. 18. IEEE, 2423–2434.
[18] Linli Jiang, Chao-Xiong Chen, and Chao Chen. 2023. L2mm: learning to map matching with deep models for low-quality gps trajectory data. In *TKDD*, Vol. 17. ACM, 1–25.
[19] Nan Jiang, Haitao Yuan, Jianing Si, Minxiao Chen, and Shangguang Wang. 2024. Towards Effective Next POI Prediction: Spatial and Semantic Augmentation with Remote Sensing Data. In *ICDE*. IEEE, 5061–5074.
[20] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*, Yoshua Bengio and Yann LeCun (Eds.).
[21] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[22] Matej Kubicka, Arben Çela, Hugues Mounier, and Silviu-Iulian Niculescu. 2018. Comparative Study and Application-Oriented Classification of Vehicular Map-Matching Methods. In *ITSM*, Vol. 10. 150–166.
[23] Biwei Liang, Tengjiao Wang, Shun Li, Wei Chen, Hongyan Li, and Kai Lei. 2016. Online Learning for Accurate Real-Time Map Matching. In *PAKDD*, James Bailey, Latifur Khan, Takashi Washio, Gillian Dobbie, Joshua Zhexue Huang, and Ruili Wang (Eds.), Vol. 9652. Springer, 67–78.
[24] Zachary Chase Lipton. 2015. A Critical Review of Recurrent Neural Networks for Sequence Learning. In *CoRR*, Vol. abs/1506.00019.
[25] Yu Liu, Qian Ge, Wei Luo, Qiang Huang, Lei Zou, Haixu Wang, Xin Li, and Chang Liu. 2024. GraphMM: Graph-Based Vehicular Map Matching by Leveraging Trajectory and Road Correlations. In *TKDE*, Vol. 36. 184–198.
[26] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. 2009. Map-matching for low-sampling-rate GPS trajectories. In *SIGSPATIAL*, Divyakant Agrawal, Walid G. Aref, Chang-Tien Lu, Mohamed F. Mokbel, Peter Scheuermann, Cyrus Shahabi, and Ouri Wolfson (Eds.). ACM, 352–361.
[27] Huali Lu, Feng Lyu, Huaqing Wu, Jie Zhang, Ju Ren, Yaoxue Zhang, and Xuemin Shen. 2023. FL-AMM: Federated Learning Augmented Map Matching With Heterogeneous Cellular Moving Trajectories. In *IJSAC*, Vol. 41. 3878–3892.
[28] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. In *CoRR*, Vol. abs/1312.5602.
[29] Reham Mohamed, Heba Aly, and Moustafa Youssef. 2017. Accurate Real-time Map Matching for Challenging Environments. In *TITS*, Vol. 18. 847–857.
[30] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *SIGSPATIAL*, Divyakant Agrawal, Walid G. Aref, Chang-Tien Lu, Mohamed F. Mokbel, Peter Scheuermann, Cyrus Shahabi, and Ouri Wolfson (Eds.). ACM, 336–343.
[31] Simon Aagaard Pedersen, Bin Yang, and Christian S. Jensen. 2020. Anytime Stochastic Routing with Hybrid Learning. In *VLDB*, Vol. 13. 1555–1567.
[32] Lin Qu, Yue Zhou, Jiangxin Li, Qiong Yu, and Xinguo Jiang. 2023. HMM-Based Map Matching and Spatiotemporal Analysis for Matching Errors with Taxi Trajectories. In *IJGI*, Vol. 12. 330.
[33] Mohammed A Quddus, Washington Y Ochieng, and Robert B Noland. 2007. Current map-matching algorithms for transport applications: State-of-the art and future research directions. In *Transportation research part c: Emerging technologies*, Vol. 15. Elsevier, 312–328.
[34] Huimin Ren, Sijie Ruan, Yanhua Li, Jie Bao, Chuishi Meng, Ruiyuan Li, and Yu Zheng. 2021. MTrajRec: Map-Constrained Trajectory Recovery via Seq2Seq Multi-task Learning. In *SIGKDD*, Feida Zhu, Beng Chin Ooi, and Chunyan Miao (Eds.). ACM, 1410–1419.
[35] Siavash Saki and Tobias Hagen. 2022. A Practical Guide to an Open-Source Map-Matching Approach for Big GPS Data. In *SN Comput. Sci.*, Vol. 3. 415.
[36] Zhihao Shen, Wan Du, Xi Zhao, and Jianhua Zou. 2020. DMM: fast map matching for cellular data. In *MobiCom*. ACM, 60:1–60:14.
[37] Zhihao Shen, Kang Yang, Xi Zhao, Jianhua Zou, Wan Du, and Junjie Wu. 2024. DMM: A Deep Reinforcement Learning based Map Matching Framework for Cellular Data. In *TKDE*. IEEE.
[38] Weijie Shi, Jiajie Xu, Junhua Fang, Pingfu Chao, An Liu, and Xiaofang Zhou. 2023. LHMM: A Learning Enhanced HMM Model for Cellular Trajectory Map Matching. In *ICDE*. IEEE, 2429–2442.

[39] Avneesh Sud, Russell Gayle, Erik Andersen, Stephen J. Guy, Ming C. Lin, and Dinesh Manocha. 2008. Real-time navigation of independent agents using adaptive roadmaps. In *SIGGRAPH*. ACM, 56:1–56:10.
[40] Shun Taguchi, Satoshi Koide, and Takayoshi Yoshimura. 2019. Online Map Matching With Route Prediction. In *TITS*, Vol. 20. 338–347.
[41] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. In *CoRR*, Vol. abs/1807.03748.
[42] Hado van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-Learning. In *AAAI*, Dale Schuurmans and Michael P. Wellman (Eds.). AAAI Press, 2094–2100.
[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. 5998–6008.
[44] Jingyuan Wang, Ning Wu, Xinxi Lu, Wayne Xin Zhao, and Kai Feng. 2021. Deep Trajectory Recovery with Fine-Grained Calibration using Kalman Filter. In *TKDE*, Vol. 33. 921–934.
[45] Adrian Wöltche. 2023. Open source map matching with Markov decision processes: A new method and a detailed benchmark with existing approaches. In *Trans. GIS*, Vol. 27. 1959–1991.
[46] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*.
[47] Can Yang and Gyözö Gidófalvi. 2018. Fast map matching, an algorithm integrating hidden Markov model with precomputation. In *IJGIS*, Vol. 32. 547–570.
[48] Haitao Yuan and Guoliang Li. 2021. A Survey of Traffic Prediction: from Spatio-Temporal Data to Intelligent Transportation. In *Data Sci. Eng.*, Vol. 6. 63–85.
[49] Haitao Yuan, Guoliang Li, and Zhifeng Bao. 2022. Route Travel Time Estimation on A Road Network Revisited: Heterogeneity, Proximity, Periodicity and Dynamicity. In *VLDB*, Vol. 16. 393–405.
[50] Haitao Yuan, Guoliang Li, Zhifeng Bao, and Ling Feng. 2020. Effective Travel Time Estimation: When Historical Trajectories over Road Networks Matter. In *SIGMOD*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 2135–2149.
[51] Jing Yuan, Yu Zheng, Chengyang Zhang, Xing Xie, and Guangzhong Sun. 2010. An Interactive-Voting Based Map Matching Algorithm. In *MDM*, Takahiro Hara, Christian S. Jensen, Vijay Kumar, Sanjay Madria, and Demetrios Zeinalipour-Yazti (Eds.). IEEE Computer Society, 43–52.
[52] Kai Zhao, Jie Feng, Zhao Xu, Tong Xia, Lin Chen, Funing Sun, Diansheng Guo, Depeng Jin, and Yong Li. 2019. DeepMM: Deep Learning Based Map Matching with Data Augmentation. In *SIGSPATIAL*, Farnoush Banaei Kashani, Goce Trajcevski, Ralf Hartmut Güting, Lars Kulik, and Shawn D. Newsam (Eds.). ACM, 452–455.
[53] Kai Zheng, Yu Zheng, Xing Xie, and Xiaofang Zhou. 2012. Reducing Uncertainty of Low-Sampling-Rate Trajectories. In *ICDE*, Anastasios Kementsietsidis and Marcos Antonio Vaz Salles (Eds.). 1144–1155.
[54] Zhihan Zheng, Haitao Yuan, Minxiao Chen, and Shangguang Wang. 2025. RLER-TTE: An Efficient and Effective Framework for En Route Travel Time Estimation with Reinforcement Learning. In *SIGMOD*, Vol. 3. 71:1–71:26.
[55] Lei Zhu, Jacob R Holden, and Jeffrey D Gonder. 2017. Trajectory segmentation map-matching approach for large-scale, high-resolution GPS data. In *Transportation Research Record*, Vol. 2645. 67–75.

Received October 2024; revised January 2025; accepted February 2025