ECTI Transactions on Computer and Information Technology
Journal homepage: https://ph01.tci-thaijo.org/index.php/ecticit/
Published by the ECTI Association, Thailand, ISSN: 2286-9131

Capturing Spatial Relationship Mapping Patterns between GPS Coordinates and Road Network Using Machine Learning and Partitioning Techniques

Thapana Boonchoo[1]

ABSTRACT

Map matching is a technique used to identify which path a vehicle is travelling on in a road map. Since it is a crucial fundamental step for a wide range of transportation applications, many map-matching algorithms have been devised ranging from simple geometric calculation methods to more sophisticated methods. However, the study of spatial relationship patterns between GPS coordinates and road segments mapped with map-matching has not received enough attention from researchers. This paper presents a framework, called Proxy Map Matching (PMM), to learn such patterns using machine learning techniques. However, we find that solely employing machine learning techniques on such data is not sufficient to capture the patterns. Solving this problem that way results in an inaccurate proxy model. In PMM, we construct several proxy map matchers and assign them to each group of data based on their spatial proximity, thereby achieving accuracy improvement. An experiment on real-world data shows that the framework achieves above 85% accuracy with integration of machine learning techniques, and outperforms the methods which solely employ machine learning techniques significantly. Moreover, the proposed proxy model can perform very fast matching. For 14,177 GPS coordinate pairs per second, we achieve 88.4% accuracy.

Article information:
Keywords: Map-matching, Machine Learning, Spatial Database

Article history:
Received: July 25, 2021
Revised: October 3, 2021
Accepted: March 4, 2022
Published: June 18, 2022 (Online)

DOI: 10.37936/ecti-cit.2022163.247801

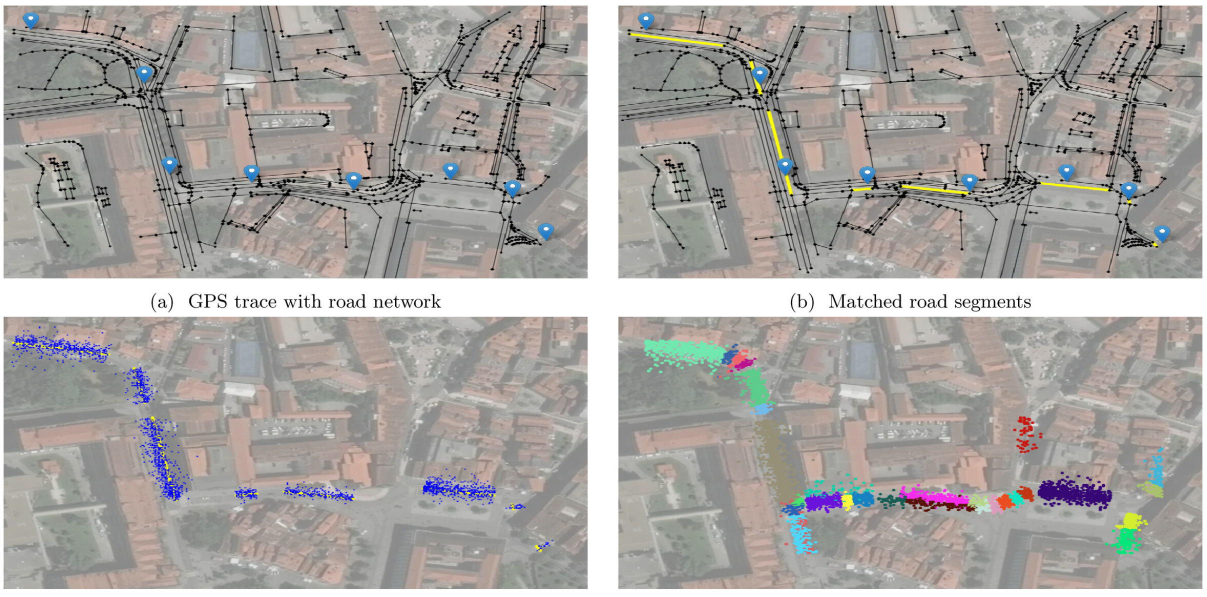
1. INTRODUCTION

Nowadays, the widespread availability of ubiquitous computing enables the collection of data about user mobility. The process of reconstructing the most likely trajectory of a device on road segments in a road network database on the basis of the observed sequential locations is known as *map-matching*. For example, Fig. 1 (a), (b) shows the observed locations (GPS trace) superimposed on a road network (shown in black lines) and the corresponding matched road trajectory (shown in yellow lines), respectively, plotted on a tile of real satellite-based map. The map-matching is an integral part for a broad range of real-world applications, such as navigation assistants [1], traffic analysis [2, 3], trajectory similarity search [4], etc. [5-7].

However, GPS data obtained from satellite-based positioning systems can be insufficiently reliable and accurate due to some conditions, such as multipath interference, sampling errors [8], urban canyons [9], and noise [8-10]. In addition, a low-sampling strategy is practical when power consumption is concerned [10-12]. As a consequence, map-matching is not always a trivial problem, and a large number of different map-matching methods have been proposed [5,13,14].

Despite advances and considerable progress in the development of map-matching methods, most of them perform the matching process by directly identifying candidate road segments that best match the observed locations. There are only a few methods considering the historical output pattern yielded by those map-matching methods. For example, the studies by Hashemi [15] have shown a strong correlation between spatial-temporal patterns and projection vectors on output road segments produced by the map-matching methods. Fig. 1 (c) shows the historical outputs of a map-matching method that maps GPS coordinates (blue points) to their matched road segments. From

[1]The author is with Department of Computer Science, Faculty of Science and Technology, Thammasat University, Pathum Thani, Thailand 12121, E-mail: thapana@cs.tu.ac.th


(a) GPS trace with road network

(b) Matched road segments

(c) Historical patterns of the matched road segment from (b)

(d) Historical patterns of all road segments in database

Fig. 1: *Mapping between GPS coordinates and road segments. The markers are GPS coordinates. The black lines are road segments in the road network. (a) GPS trace with nearby road segments (b) The GPS coordinates with corresponding matched road segments (yellow thick lines). (c) The historical mapping patterns between GPS coordinates (blue points) and road segments. (d) The historical mapping patterns between GPS coordinates and road segments. The GPS coordinates in different colors denote the mapping on different road segments.*

the figure, we can also observe the spatial relationship patterns between GPS coordinates and each matched road segment by the map-matching method, such as those GPS coordinates spread over road segments.

Considering how the observed locations are processed, the map-matching methods can be classified into online and offline. Online map-matching [16-18] requires low computational effort to produce outputs in an incremental manner to respond users in real time. On the other hand, offline methods [19-22] process a whole database after it has been collected to ensure adequately high accuracy. Here, we focus on the offline map-matching methods. Specifically, we exploit the reusability of historical outputs provided by an offline map-matching method to model the spatial relationship patterns between GPS coordinates and road segments. The model can be considered as a proxy for the map-matching method. That is, given GPS coordinates of a location, the proxy model can immediately report the most likely matched road segment directly to users. Interestingly, one could regard the model as a bridge to emulate the online manner using offline map-matching methods.

In particular, the model employs machine learning techniques, such as classification algorithms, to learn spatial relationship patterns between GPS coordinates of observed locations and output road segments produced by an offline map-matching method. In other words, this can be viewed as a multiclass classification (supervised learning) where the GPS coordinates of observed locations as input variables are mapped to road segments as target variables. Fig. 1 (d) illustrates an example. Namely, different colors denote different classes. In practice, the number of different road segments in a road network can be very large. For example, there are almost $3 \cdot 10^5$ road segments in the city of Porto. This high number of road segments might degrade the efficiency and performance of some machine learning classification algorithms due to noise in the training [23].

In this paper, we propose a proxy map matching framework (PMM for short) to model the above-mentioned spatial relationship patterns between GPS coordinates and output road segment produced by a map-matching method. The proposed framework has four processes: a map-matching process, a process to extract matched pairs, a partitioning process, and a training process. In the map-matching process, given a historical trajectory database, it runs an offline map-matching method on the database to obtain the collection of GPS coordinate and road-segment pairs. Second, the matched pair extraction process constructs a matched pair database to facilitate the model learning with a machine learning technique. Third, the partitioning process divides the samples in the matched pair database into different partitions based on their spatial proximity. Lastly, the training process employs a machine learning technique to learn the pattern relationship between GPS coordinate pairs and road segments to build a proxy map matcher. We also conduct extensive experiments to show the performance of PMM in terms of both accuracy and running time. The experimental results show that PMM can achieve an improvement of about 76% in accuracy and runs 30 times faster compared to solely applying machine learning techniques.

The contributions of this paper can be summarized as follows:

- We study the spatial relationship mapping patterns between GPS coordinates and road networks and propose a framework to capture such mapping patterns using machine learning classifiers.
- We find that solely employing existing machine learning classifiers, for example k-Nearest Neighbor or Logistic Regression, cannot capture such mapping patterns well. As a result, the pattern mapping accuracy performance is degraded. We then propose a partitioning process in the framework to enhance the accuracy. The experiments show that the partitioning process can improve the accuracy of the existing machine learning classifiers by up to 76%.
- We conduct extensive experiments to show the performance of PMM in terms of both accuracy and running time. The experimental results demonstrate that the proposed framework can capture such mapping patterns between GPS coordinates and road networks effectively and efficiently as it achieves significant improvements on both accuracy and running time in most cases.

The rest of the paper is organized as follows. We survey the related work in Section 2. We present the preliminaries and the problem statement in Section 3. In Section 4, we detail each process of our proposed PMM framework. Section 5 provides the experimental results and discussion. Finally, we conclude the paper in Section 6.

2. RELATED WORK

Different map-matching methods are designed to be used in specific applications and which to select depends strongly on the expected accuracy and assumptions about the data. For example, some methods expect a low-sampling rate and noisy GPS [8]. Map-matching problems have been intensively explored and various methods have been proposed in the literature ranging from simple geometry to more advanced techniques [5, 13, 14].

Early map-matching methods considered the geometric shape of road segment [24-26]. White et al. [25] provided a discussion about the advantages and disadvantages of four fundamental methods for the map-matching problem, which rely on simple geometry and road network topology. Greenfeld [26] and Brakatsoulas et al. [27] performed greedy matching on the basis of curves and orientation of consecutive GPS observations and road segments. Brakatsoulas et al. [27] and Alt et al. [28] further proposed using Frechet-based distances to perform a global match mapping for the whole trajectory to a candidate road segment curve. Wenk and Salas [29] improved the speed of Frechet-based global methods [27, 28] by exploiting tracking error estimates to prune the road network graph.

Moreover, probabilistic methods have also received considerable attention and have been widely exploited in developing map-matching methods. This includes techniques employing Kalman filters, fuzzy logic, belief theory, and Hidden Markov Models.

The Hidden Markov Model (HMM) is a powerful technique and has been successfully employed to deal with sequential data. In HMM, road segments are treated as the states and GPS as the noisy external observations. Once the transition probabilities, the relation between states and other states, and emission probabilities are known, the relationship between states and observations is constructed. Given a sequence of observed locations, the optimal path of the road segments can be discovered efficiently by running the Viterbi algorithm. Hummel [30] proposed a map-matching method based on HMM, where the emission probabilities were computed on the basis of a Bayesian classifier, and the transition probabilities were assumed to be uniformly distributed over all possible outgoing road segments. Similar to [30], Krumm et al. [31] additionally incorporated a probabilistic travel time constraint in the model such that temporally unlikely paths are discarded. Newson and Krumm [21] modeled the transition probabilities on the intuition that the route distance between projected points on road segments and the great circle distance between consecutive observations should be close for correct matching. Recently, Yang and Gidofalvi [20] mitigated the bottleneck of repeated routing queries by preprocessing the shortest paths within a certain length in the road network, thereby gaining a performance improvement. Online HMM-based map-matching has also been proposed using a *sliding window technique* [16] with an online Viterbi algorithm. Recently, an RNN-based map-matching algorithm [32] was developed to identify the most-likely trajectory of roads given a sequence of cell towers. In their work, they focus on data from cell towers instead of GPS coordinate pairs. In our work, we propose a framework that incorporates machine learning techniques to learn the relationship between GPS coordinates and road segments matched by a HMM-based method [21] with partitioning strategies to improve the performance of the learning process in terms of both accuracy and efficiency.

3. PRELIMINARIES

In this section, we introduce important definitions required for our framework and give the problem statement.

Definition 1: **Road Network**: A road network is defined as a directed graph $R = \{V, E\}$, where V and

E are the sets of nodes and edges, respectively. In the graph G, an edge $e \in E$ represents a road segment from a node $v \in V$ to another node $w \in V$, where the start and end of edge e are represented by $e.s = v$ and $e.e = w$, where $v \neq w$.

Definition 2: **Trajectory**: A trajectory is a geographical coordinate history of a moving object recorded during a travel in a chronological order, denoted by $T = \langle p_1, p_2, \cdots, p_k \rangle$, where p_i=(x_i, y_i, t_i), $x_i, y_i \in \mathbb{R}$, $\forall_{0 \leq i \leq k}\ t_i < t_{i+1}$, and k is the trajectory length.

Definition 3: **Matched Pair**: A matched pair is a pair between GPS coordinates of an observed location and a road segment e in a road network, denoted by γ=$[(x, y) \rightarrow e]$ or γ=$[p \rightarrow e]$.

Definition 4: **Map Matcher**: A map matcher is a function $\mathcal{M} : \mathcal{T} \rightarrow \mathcal{S}$ that matches a trajectory with a sequence of road segments.

Next we present the problem statement of this paper. Given a trajectory database $\mathcal{T} = \{T_1, T_2, \cdots, T_n\}$, for each trajectory $T_i = \langle p^i_1, p^i_2, \cdots, p^i_{|T_i|} \rangle$, we obtain a corresponding sequence of road segments $S_i = \langle e^i_1, e^i_2, \cdots, e^i_{|T_i|} \rangle$ through a map matcher $\mathcal{M}$ (Definition 4), where $1 \leq i \leq n$. For each matched pair p^i_j and e^i_j, we also consider to view it as a matched pair $\gamma = [p^i_j \rightarrow e^i_j]$ (Definition 3), where $1 \leq j \leq |T_i|$. Once all the trajectories in the dataset $\mathcal{T}$ have been matched with a sequence of road networks by $\mathcal{M}$, we can construct a dataset of every matched pair, denoted by $\Gamma = \{\gamma_1, \gamma_2, \cdots, \gamma_m\}$. Our goal is to study the pattern exhibited in Γ, and to use machine learning techniques to learn such a pattern to build a proxy model M, called the *proxy map matcher*, to match a given coordinate pair with a road segment, which is defined next.

Definition 5: **Proxy Map Matcher**: A proxy map matcher is a function $M : \mathbf{R}^2 \rightarrow E$ that matches a coordinate pair with a road segment.

The problem statement of this paper is to produce a framework that captures the characteristics of matching patterns of the traditional map matcher $\mathcal{M}$ to match the trajectory observations in the database $\mathcal{T}$ on the road network R. More specifically, a proxy map matcher (Definition 5) in the framework can be considered as a model for the matching patterns and can be used as a surrogate for a map matcher for any given unseen GPS coordinates. We provide a detailed explanation of each process in the framework in Section 4.

4. METHODOLOGY

In this section, we detail the four processes in the framework: map-matching process, extraction of matched pairs process, classifier partitioning process, and training proxy matcher process. The overall framework of this paper is shown in Fig. 2.

4.1 Map-matching Process

Our framework starts by collecting a trajectory database. Once the trajectory database has been constructed, the map-matching process will employ a map matcher (Definition 4) to convert each trajectory into a corresponding sequence of road segments on road network R. Specifically, we use the Hidden Markov Model map matching algorithm, which provides accurate matching of the sequence of coordinate pairs with the sequence of actual road segments, as our map matcher (HMM matcher for short). The HMM matcher is implemented based on the well-known Hidden Markov model which considers all possible sequences of road segments as the hidden states while considering the sequence of coordinate pairs as observations or input of the model. The key idea is that the model finds the sequence of road segments that maximize the probability of the given sequence of coordinate pairs. This can be seen as the trade-off between the road segments suggested by observed coordinate pairs and the probabilities that a vehicle can move from one road segment to the others.

To construct the model, the HMM matcher mainly requires computing the emission and transition probabilities. The *emission probability* is the conditional probability that a coordinate pair p is observed given that the vehicle is moving on a road segment e, i.e. $P(p|e)$. We can also model $P(p|e)$ by zero-mean Gaussian, i.e. $P(p|e) = \frac{1}{\sqrt{2\pi}\sigma_p} exp(-\frac{1}{2}(\frac{||p-p_e||_{gc}}{\sigma_p})^2)$, where p_e is the projected location of p on the road segment e, and σ_p is a standard error. According to the formula of $P(p|e)$, the value of $P(p|e)$ will become higher when the coordinate pair gets closer to the road segment. Moreover, the model also computes the probability that a given moving object travels or makes a transition from the current road segment to the other possible roads which is known as *transition probability*. The formula of transition probability is $P(d) = \frac{1}{\beta} e^{\frac{-d}{\beta}}$, where d is the difference of distances between the current road segment and the other. This distance can be derived from the road network graph and the great circle distance between them (i.e. the absolute distance). This means that if these two distances are closer to each other, the probability that one is able to travel from the current road segment to the other will be higher, which means it is a more reasonable path to travel. Note that β is an estimated parameter.

Given the HMM matcher $\mathcal{M}$ and a sequence of coordinate pairs, we can build a matched pair database Γ (see Definition 4). The process of creating the matched pair database Γ is explained next.

4.2 Process of Extracting Database of Matched Pairs

In this section, we present the process of creating the matched pair database from a given trajec-

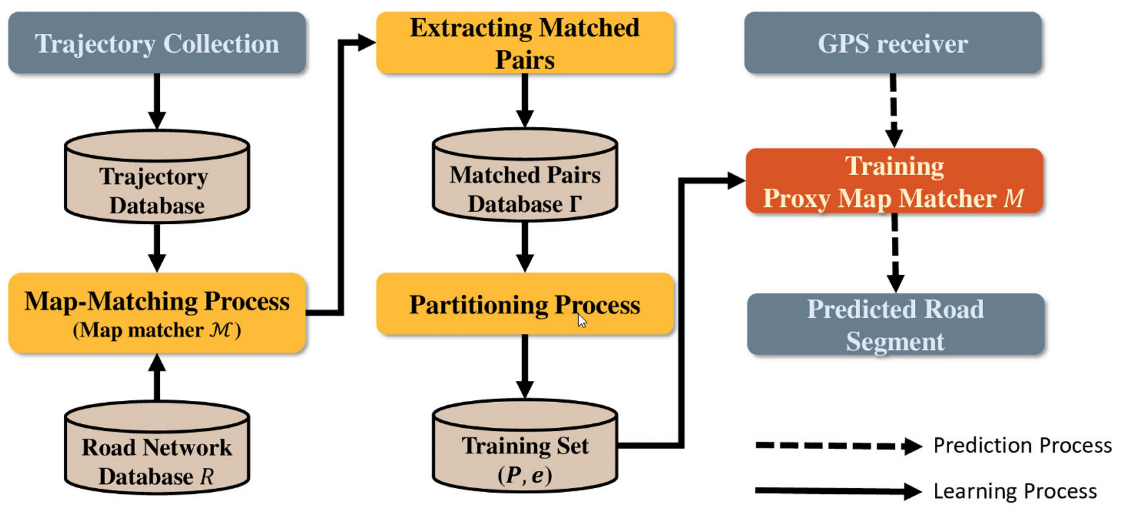


Fig.2: *This flow diagram shows the overall framework. The framework starts by collecting trajectories to build a trajectory database. Then, the map matching process will be performed by integrating a road network database (R) that corresponds to the trajectory database to extract the matched pairs in order to construct a database of matched pairs (Γ). After Γ has been constructed, the partitioning process will be employed to extract the training set to be used to build the proxy map matchers in the training process. Once the proxy map matchers have been constructed, we can use the proxy map matchers to predict the road segments corresponding to any unseen GPS coordinates from a GPS receiver.*

Algorithm 1: Process of Extracting Matched Pair Database

```
Input: $\mathcal{T}$: trajectory database, $R$ road network
Output: $\Gamma$: matched pair database
1 $\Gamma \leftarrow \emptyset$
2 foreach Trajectory $T \in \mathcal{T}$ do
3     $\tau \leftarrow \mathcal{M}(\mathcal{T}, \mathcal{R})$
      // $\tau$ is the sequence of road segments
         corresponding to trajectory $T$ obtained
         from the HMM map matcher $\mathcal{M}$
4     foreach point $p \in T$ do
5         $\gamma \leftarrow [(p.x, p.y) \rightarrow \tau[p]]$
          // $\gamma$ denotes the matched pair
          // $\tau[p]$ denotes the road segment that is
             matched with the point $p$
6         $\Gamma.insert(\gamma)$
7 return $\Gamma$
```

tory database. The whole process of constructing the matched pair database is shown in Algorithm 1.

The process starts by looping through each trajectory T in database $\mathcal{T}$ and the HMM map matcher in Section 4 is applied to match the trajectory to a sequence of road segments, denoted by τ in the algorithm (Lines 2-3). Next, the inner loop (Lines 4-6) constructs the matched pair database Γ by collecting the matched pairs γ (see Definition 3) which are formed by Line 5. After Algorithm 1 finishes, the matched pair database Γ is constructed to be used for the next step of the framework.

4.3 Partitioning Process

After the matched pair database Γ is established from the previous step, we construct the proxy matcher M (see Definition 5). Our proposed proxy matcher is composed of a number of independent matchers. Each matcher corresponds to a classifier trained to learn the spatial pattern between GPS coordinates and road segments. To construct such independent matchers, we partition the matched pair database Γ according to the spatial proximity of each sample $\gamma \in \Gamma$. Here, the spatial proximity is coordinate pairs (x, y) in γ, denoted by $\gamma.(x, y)$ or $\gamma.p$. So, given a matched pair database Γ, we partition it into k partitions $\Gamma_1, \Gamma_2, \cdots, \Gamma_k$, where $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \cdots \cup \Gamma_k$. To perform partitioning, we divide the matched pair database Γ by placing the coordinate pairs into equal-sized cells.

Fig. 3 shows the partitioning process in our framework. The left figure shows the data layout of coordinate pairs, while the right shows the data layout after the partitioning process is applied. In the figure, we have 14 samples in the matched pair database, $\Gamma = \{\gamma_1, \gamma_2, \cdots, \gamma_{14}\}$, where $\gamma_i = [p_i \rightarrow e_i]$ (Definition 3), $1 \leq i \leq 14$, and coordinate pairs p_i correspond to the points depicted in the figure. We will partition such samples in γ into cells according to $\gamma.p$, so each cell can be seen as a partition of our database. Fig. 3 (right) shows 5 non-empty cells: $c_3, c_7, c_9, c_{10}, c_{12}$. So in this example, we have 5 partitions. Formally, we can write $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4 \cup \Gamma_5$, where $\Gamma_1 = c_3$, $\Gamma_2 = c_7$, $\Gamma_3 = c_9$, $\Gamma_4 = c_{10}$, and

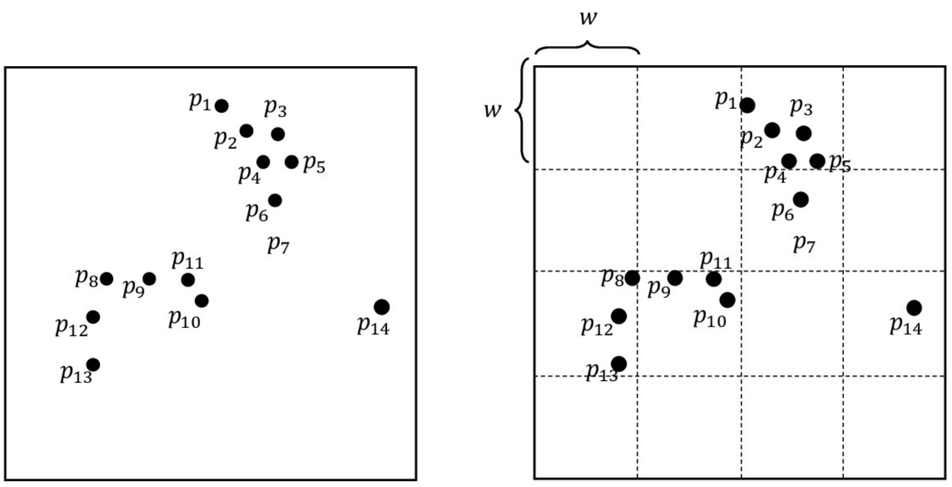


Fig.3: *Partitioning Process. The left shows the data layout without partitioning. The right shows the data layout that were partitioned into equal-sized (w) cells*

$\Gamma_5 = c_{12}$.

4.4 Training Proxy Matcher

After the partitioning process, the matched pair database Γ is divided into k partitions according to the spatial proximity of each sample, i.e. $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \cdots \cup \Gamma_k$. In this subsection, we construct a proxy matcher M for the map matcher $\mathcal{M}$ (see mmprocess). Here, the proxy matcher M is composed of k independent matchers, denoted by $M_1, M_2, \cdots, M_k$. Each independent matcher M_j corresponds to a machine learning technique which is used to learn the pattern of data or more specifically the relationship between a coordinate pair $\gamma_j.(x,y)$ and a matched road segment $\gamma_j.e$ in Γ_j. In this paper, we investigate four well-known classifiers: k-Nearest Neighbor (kNN), Multi-layer Perceptron (MLP), Logistic Regression (LS), and Gaussian Naive Bayes model (GNB). Besides the four classifiers, our framework also makes room for other classifiers in this process for future research. The task for a matcher M_j is to train a classifier by considering $\gamma_j.(x,y)$ (2D feature vector) as the input and $\gamma_j.e$ as the target variable. Readers can find the fundamental concepts for such techniques in [33].

4.4.1 k-Nearest Neighbor (kNN)

The kNN model is simple and very well-known for classification tasks. It is trained by building indices of feature vectors of data in the training set in a multi-dimensional space. The model takes a constant k as a parameter and determines a class for a query input feature vector based on the majority classes among k training feature vectors nearest to the query vector. For the matcher M_j, we will use the corresponding Γ_j as the training set for the model. We employ Euclidean distance as the measure of nearness as

$$Dist_{Euclid}(v_1, v_2) = \sqrt{(v_1.x - v_2.x)^2 + (v_1.y - v_2.y)^2}, \tag{1}$$

where v_1 and v_2 are 2D-dimensional vectors. Consider the cell c_3 in Fig. 3 (right). If we assume $k = 3$, p_4 is the query, and the other points are the training set. The 3-nearest neighbors for feature vector p_4 will be p_2, p_3 and P_5. Note that even though p_6 is closer to p_4 than p_2, p_3, and P_5, it will not be included in the nearest neighbor set because it is not in the same cell. If p_2, p_3, and P_5 belong to classes e_1, e_2, and e_1, respectively, then this kNN will classify the query p_4 as being in the same class as e_1 since e_1 has two votes which are greater than the vote of class of e_2.

4.4.2 Logistic Regression (LR)

Logistic Regression (LR) is a classifier extended from Linear Regression which is a linear model. The Logistic Regression can non-linearly learn the mapping from input to output for the classification task. LR is trained by fitting the model to estimate the probabilities of binary classes by maximizing

$$P(C = c|x;\theta) = h(x)^c(1 - h(x))^{1-c}, \tag{2}$$

where $c \in \{0, 1\}$, $h(x)$ is a logistic function defined as

$$h(x) = \frac{1}{1 + e^{-\theta^T x}}, \tag{3}$$

and θ is a model parameter to be estimated using the Gradient ascent technique. LR has a number of variants that extend it to support multi-class classification. These include multinomial logistic regression, one-vs-rest, etc.

With LR as the matcher M_j, we will estimate the parameter θ using Γ_j as the training set by considering $\gamma.(x,y)$ and $\gamma.e$ as the input x and the target variable c in Equation 2. In practice, the number of possible classes c is normally much greater than two classes. We will employ one-vs-rest logistic regression in this paper.

4.4.3 Multi-layer Perceptron (MLP)

A Multi-layer Perceptron (MLP) is a more complex non-linear model which can be considered as a type of Artificial Neural Network (ANN) that consists of 3 layers: input layer, hidden layer, and output layer. Each layer contains at least one node, and all nodes in adjacent layers will be fully connected. The number of nodes in the input layer corresponds to the number of input features. The number of nodes in the output layer corresponds to the number of classes in the problem. Unlike input and output layers, the hidden layer can have more than one layer. The links between nodes of adjacent layers are the weights or parameters of MLP which can be learned by using Stochastic Gradient Descent (SGD).

With MLP as the matcher M_j, the input layer has two nodes which are $\gamma.x$ and $\gamma.y$. The number of nodes in the output layer will be determined by the number of different classes in the training set Γ_j.

4.4.4 Gaussian Naive Bayes (GNB)

The Naive Bayes model is based on the well-known Bayes Theorem in the form

$$P(C|x) = \frac{P(x|C)P(C)}{P(x)} \tag{4}$$

on the assumption that each feature in x is conditionally independent of each other given the class C. Therefore, we assume the probability of $P(x|C) = \prod_{l=1}^{m} P(x_l|C)$, where m is the number of features of x. The GNB classifier will determine the class c for a given input feature vector x that has the highest probability as follows:

$$\operatorname*{argmax}_c P(C) \prod_{l=1}^{m} P(x_l|C). \tag{5}$$

Note that for the classification task we can ignore the computation of $P(x)$ in Equation 4 since it does not affect the correctness of the result.

With GNB as the matcher M_j, we will use *Gaussian Naive Bayes* which further assumes the Gaussian distribution on each feature in the input x. Here, the input feature vector is $\gamma.(x, y)$, so we can compute the probability $P(x_l|C = c)$ based on the Gaussian assumption as follows:

$$P(x_l|C = c) = \frac{1}{\sqrt{2\pi\sigma_{lc}^2}} e^{-\frac{1}{2}(\frac{x_l - \mu_{lc}}{\sigma_{lc}})^2}, \tag{6}$$

where σ_{lc} and μ_{lc} are the model parameters of a specific class c and feature l to be estimated. We can use the Maximum Likelihood Estimation (MLE) method to estimate such parameters.

4.5 Prediction Process

Given a new GPS point $p_{new} = (x_{new}, y_{new})$ and the constructed proxy map matcher M, the prediction process (the mapping procedure) has two steps. First, we need to identify the proxy map matcher that corresponds to (x_{new}, y_{new}). Assume that the correponding proxy matcher for p_{new} is M_j. In the second step, the framework will make the prediction of road segment that p_{new} should be matched with according to the machine learning technique adopted to implement the proxy matcher M_j.

5. EXPERIMENTS

We now present the experimental results of our proposed framework. We first evaluate the effectiveness of proxy map matchers in terms of accuracy and runtime performance of the proxy map matcher framework, and then discuss further the effects of different cell sizes on the performance of the framework.

5.1 Data

We used a real trajectory database of taxis in the city of Porto, Portugal [34], and obtained the digital road network database of the city of Porto from OpenStreetMap [35]. The Porto database contains approximately 1.7 million trajectories collected from 442 taxicabs in the city of Porto, Portugal. Each trajectory has even sampling intervals between sampled points where a sampling interval is 15 seconds long. The road network of Porto contains about 0.26 million nodes with about 0.29 million edges. In the experiments, we randomly sample $50,000$ trajectories from the Porto database as the dataset for our experiments. We split those into 85% of trajectories for the training set, and the remaining 15% were used as the test set. We use the training and test samples which are located in the tile as shown in Fig. 4. Totally, there were 155333(83%) training matched pair samples and 26441(17%) test samples.

5.2 Performance Evaluation

We examine the performance of the proposed framework through the following four compared proxy matchers:

kNN as Proxy Map Matcher (kNN-PMM): We employed kNN as the Proxy Map Matcher in our proposed framework with a partitioning process applied. We set $k = min(1, 0.05n)$ as the parameter to train the classifier, where n is the number of training samples.

Logistic Regression as Proxy Map Matcher (LR-PMM): We employed LR as the Proxy Map Matcher in our proposed framework with a partitioning process applied. We use one-over-rest logistic regression where the $l1$ penalty is employed in the model, and the sigmoid function is used as the activation function in the model.

Fig.4: *The map tile in the Porto city, Portugal, used in the experiments.*

Multi-Layer Perceptron as Proxy Map Matcher (MLP-PMM): We employed MLP as the Proxy Map Matcher in our proposed framework with a partitioning process applied. We use 1-layer for the hidden layer with 20 neural nodes in the hidden layer, and we used the activation function *relu* in the hidden layer.

Gaussian Naive Bayes as Proxy Map Matcher (GNB-PMM): We employed GNB as the Proxy Map Matcher in our proposed framework with a partitioning process applied. We employed the continuous version of Naive Bayes Classifier which assumes the Gaussian distribution over each feature.

Other than the four classifiers, we additionally added one simple classifier which always predicts the road segment which has the highest likelihood in the cell. This model is denoted as **Maximum Likelihood (ML-PMM)**.

5.2.1 Accuracy

Table 1 shows the accuracy of the compared classifiers with different cell size settings and the column **No partition** indicates the experimental results by PMM that do not employ the partitioning process. Note that we use the matched pair resulting from the HMM map matcher as the ground truth of this experiment. We show the highest accuracy value of each column in bold face. To examine the effectiveness of the proposed framework, we added the last column which depicts the accuracy of solely employing just machine learning technique *without* any partitioning process. The accuracy measure is computed as

$$Accuracy = \frac{\#correct\ prediction}{\#testset}. \qquad (7)$$

The correct prediction denotes the number of correct predictions that a compared classifier predicted exactly the same as the HMM map matcher, given the same GPS coordinates.

LR-PMM achieves about 88% of accuracy ($w = 0.0001$), which is the best accuracy, outperforming other compared methods significantly. The value w is the cell size. The other three compared methods, kNN-PMM, MLP-PMM, GNB-PMM, have comparable accuracy scores. GNB-PMM is in second place, providing also about 88% of accuracy ($w = 0.001$), while MLP-PMM and kNN-PMM have comparable scores of accuracy which are approximately almost 88% ($w = 0.0001$) and 87% ($w = 0.0001$), respectively. This finding demonstrates that LR-PMM and GNB-PMM are capable of capturing the spacial relationship patterns between GPS coordinates of observed locations and road segments, while the other three methods provide comparable performance for the same task.

Next, we compared the accuracy of all methods with the cases when no partitioning process was performed. Our proposed framework with the partitioning process significantly outperforms the one without the partitioning process. For example, the accuracy of LR-PMM increases by about 75% when the partitioning process is employed. Similar observations can be seen for other methods as well. This demonstrates the effectiveness of the partitioning process in improving the accuracy performance of all compared machine learning techniques. The reason for this is that the performance of some classifiers can be degraded due to a large number of classes during the training process [23]. However, the partitioning process in PMM, which divides the data into cells before the training process with each cell containing a smaller number of classes, can mitigate the problem, as demonstrated in the accuracy performance improvement of all classifiers shown in Table 1.

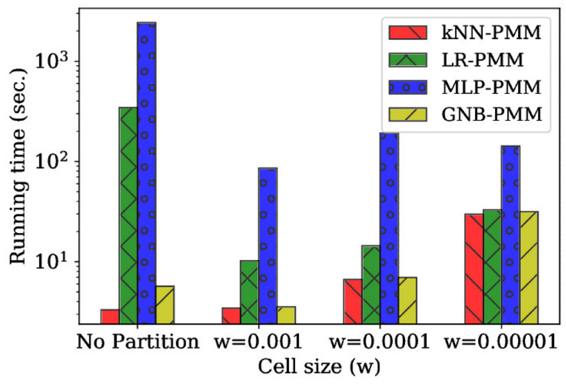
Moreover, we observe that ML-PMM, which naively predicts the road segments based on the statistical likelihood of training data, achieves the worst accuracy scores compared to the other four machine learning techniques. For example, ML-PMM achieves about 42% less accuracy than GNB-PMM ($w = 0.001$), and about 6% less than LR-PMM ($w = 0.0001$). This suggests that machine learning techniques perform significantly better than the basic statistical method (ML-PMM).

5.2.2 Running Time

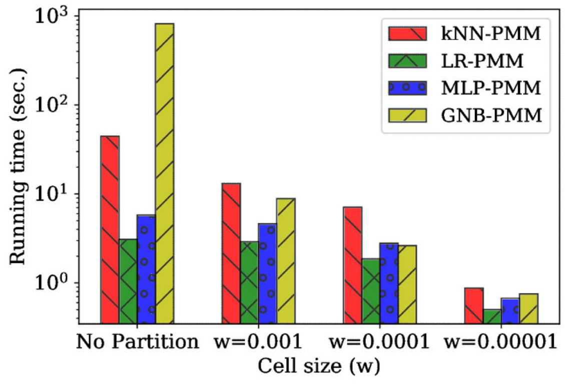
We also evaluate the running time of the four machine learning techniques with different cell sizes. The running time of each method on the training and prediction steps is presented in Fig. 5 (a) and Fig. 5 (b), respectively, in log scale. First, we examine the efficiency of the training step. The running time of the training step for kNN increases when the cell size becomes larger. The reason for that is that when the cell size is larger, the method requires some overhead time taken to build the indexing structures for the

Table 1: *Accuracy of Compared Classifiers. Note that the column* ***No partition*** *indicates the experimental results by PMM that do not employ the partitioning process.*

Classifier/Cell size (w)	0.001	0.0001	0.00001	0.000001	No partition
ML-PMM	45.645	82.527	37.612	31.753	2.046
kNN-PMM	86.479	87.080	37.600	31.662	19.102
LR-PMM	77.307	**88.468**	**37.702**	**31.765**	12.631
MLP-PMM	85.159	87.958	37.687	31.712	73.809
GNB-PMM	**88.063**	87.776	37.509	31.670	**84.527**



(a) Training



(b) Prediction

Fig.5: *Running times for the training and prediction steps of the four machine learning classifiers: kNN-PMM, LR-PMM, MLP-PMM and GNB-PMM. The cell sizes in both training and prediction are varied, i.e.* $w = 0.001, 0.0001, 0.00001$*, while No partition means the framework did not employ the partitioning process.*

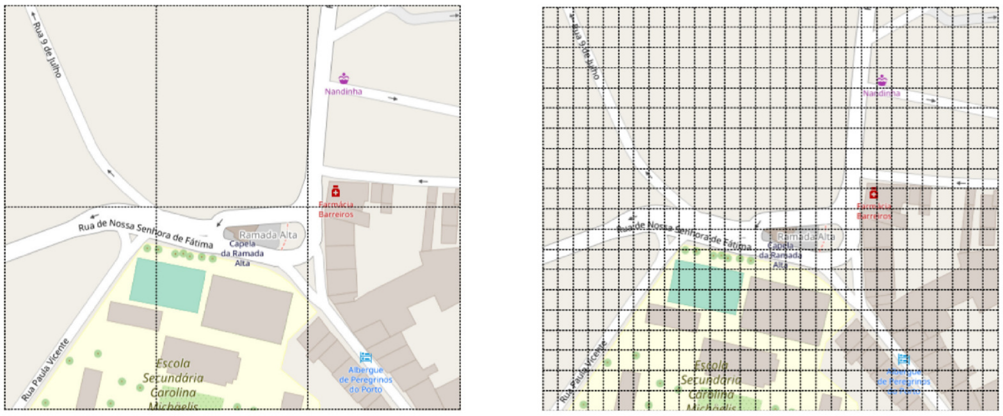
kNN classifier in each cell. For LR-PMM, the training time slightly increases. MLP-PMM is more complex and unsurprisingly runs slower than LR in all cases because the number of parameters for MLP to be estimated is much greater than that of LR. However, MLP can sometimes be struck when it performs optimizing, and can require a very long time to converge, like in the case when cell size equals to 0.0001 of MLP, which takes about 192 seconds to finish. The running time of training step of GNB is relatively good in almost cases compared to the others.

When comparing the running time in the case of no partitioning process, we can see that the runtime performance of model-based methods, LR-PMM, MLP-PMM and GNB-PMM, are dramatically improved. For example, the training time for LR-PMM with partitioning process is 10 to 30 times faster than when no partitioning process employed.

Next, we examine the efficiency of the prediction step. We observe that the running times of the prediction step of kNN-PMM, LR-PMM, MLP-PMM and GNB-PMM decrease as the cell size becomes smaller. The reason for that might be that the number of parameters for each classifier is smaller according to the cell size, so the running time of the prediction step will be flattened and distributed across all cells. For example, when the cell size is decreased from 0.001 to 0.0001, the prediction time decreases by about 4 times. While comparing the prediction time between the cases of no partitioning process and methods that do employ partitioning, we find that they are comparable for LR-PMM and MLP-PMM, while the case employing the partitioning process and kNN-PMM is slightly better. We surprisingly observe that the case of employing the partitioning process and GNB-PMM significantly outperforms the case of no partitioning process in all cases. For example, the GNB-PMM with a partitioning process runs 100 times faster than GNB-PMM without a partitioning process.

5.2.3 Effects of Cell Size

As discussed in Section 5.2.1 and Section 5.2.2, the cell size (w) in the partitioning process has a great impact on both accuracy (Table 1) and running time (Fig. 5). This section provides a further discussion on the effects of cell size. Fig. 6 shows the map tile partitioned into different cell sizes. It is interesting to notice that some machine learning techniques achieve relatively good accuracy performance when the cell size is smaller. For example, LR-PMM with a cell size of 0.0001 achieves 88.4% accuracy which is about 11.1% better than LR-PMM with a cell size of 0.001. Also, MLP-PMM with a cell size of 0.0001 achieves 87.958% accuracy which is about 2.8% better than MLP-PMM with a cell size of 0.001. This demonstrates that the performance of some machine learn-


Rua 9 de Julho
Rua de Nossa Senhora de Fátima
Ramada Alta
Capela da Ramada Alta
Nandinha
Farmácia Barreiros
Escola Secundária Carolina Michaelis
Albergue de Peregrinos do Porto
Rua Paula Vicente

(a) $w = 0.001$ (b) $w = 0.0001$

Fig.6: *A map tile partitioned into cells with two different sizes: (a) setting* $w = 0.001$*, (b) setting* $w = 0.0001$

ing techniques might be degraded for insufficiently small cell sizes. For kNN-PMM and GNB-PMM, the accuracy is stable when the cell size varies, with only about 0.5% and 0.3% accuracy differences for kNN-PMM and GNB-PMM, respectively.

Next, setting cell sizes too small can cause the model performance to degrade. Table 1 shows that all machine learning methods in the framework achieve similar poor accuracy scores when the cell size is less than 0.00001. Let us consider Fig. 6 (b) where we set the cell size to 0.001. If we further reduce the cell size into 0.000001 or even smaller to 0.000001, this may cause each cell to contain only a small number of data values to train a proxy matcher. Consider the extreme case where each cell contains only one training data sample. In such cases, the machine learning methods cannot learn any patterns, so they will always predict the same class as the training one. In addition, it is possible that when we are mapping a new GPS point, it will be assigned an empty proxy matcher. In this case, the framework will fail to make the prediction. Therefore, the above reasons explain why setting cell size too small might cause the framework to perform the mapping task poorly. In general, this might suggest setting the cell size as 0.0001 for general machine learning techniques when we have no prior knowledge about the machine learning techniques that will be adopted in PMM.

6. CONCLUSION

In this paper, we propose a proxy map matching framework (PMM for short) which uses mainstream machine learning techniques to capture the relationship pattern between GPS coordinate pairs which are mapped with road segments on road network with a map-matching algorithm. We employed the HMM map matcher in this work. We clearly provide the whole process to build the proxy map matcher framework from map-matching process, extraction of matched pairs process, partitioning process, training proxy matcher process, to the prediction process. The empirical evidence shows that only applying machine learning techniques is not sufficient to achieve good performance. By combining it with the proposed partitioning process in the framework, we can significantly improve the mapping accuracy performance so that it can achieve up to 76% improvement for a logistic regression classifier with a partitioning process over non-partitioning one. Apart from improving the accuracy, the partitioning process is still capable of improving the runtime performance significantly by working to 30 times faster. In conclusion, the experiments show that the proposed framework as a proxy model for map-matching can achieve good performance in in terms of both accuracy and running time.

For future work, it would be interesting to also investigate the other features, rather than just the spatial feature, to include them in the framework. This would include things such as temporal features and density of GPS coordinate pairs. We could consider using more sophisticated strategies to perform the partitioning process, rather than using naive static cell partitioning. Lastly, we might study more complex machine learning techniques, for example, deep learning, to use as classifiers in each cell.

7. DATA AVAILABILITY STATEMENT

The data that supports the findings of this study are available in Taxi Service Trajectory (TST) Prediction Challenge 2015 [36]. This data was derived from the following resource available in the public domain [34].

References

[1] D. Bernstein and A. Kornhauser, "An introduction to map matching for personal navigation assistants," *New Jersey TIDE Center Technical Report*, 1996.

[2] Q. Huang, Y. Yang, Y. Xu, F. Yang, Z. Yuan and Y. Sun, "Citywide road-network traffic monitoring using large-scale mobile signaling data," *Neurocomputing*, 2021.

[3] Z. Wang, M. Lu, X. Yuan, J. Zhang and H. Van De Wetering, "Visual traffic jam analysis based on trajectory data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2159–2168, 2013.

[4] T. Boonchoo, X. Ao, and Q. He, "Multi-aspect embedding for attribute-aware trajectories," *Symmetry*, vol. 11, p. 1149, 09 2019.

[5] M. Kubicka, A. Cela, H. Mounier and S. Niculescu, "Comparative study and application-oriented classification of vehicular map-matching methods," *IEEE Intelligent Transportation Systems Magazine*, vol. 10, pp. 150–166, Summer 2018.

[6] N. Podevijn, J. Trogh, M. Aernouts, R. Berkvens, L. Martens, M. Weyn, W. Joseph and D. Plets, "Lorawan geo-tracking using map matching and compass sensor fusion," *Sensors*, vol. 20, no. 20, 2020.

[7] L. d. C. Gomes and L. H. M. K. Costa, "Vehicular dead reckoning based on machine learning and map matching," in *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, pp. 1–5, 2020.

[8] D. Pfoser and C. S. Jensen, "Capturing the uncertainty of moving-object representations," *Advances in Spatial Databases Lecture Notes in Computer Science*, vol. 1651, pp. 111–131, 1999.

[9] Youjing Cui and Shuzhi Sam Ge, "Autonomous vehicle positioning with gps in urban canyon environments," *IEEE Transactions on Robotics and Automation*, vol. 19, pp. 15–25, Feb 2003.

[10] Y. Wang, Y. Zhu, Z. He, Y. Yue, and Q. Li, "Challenges and opportunities in exploiting large-scale gps probe data," in *HP Laboratories, Technical Report HPL-2011-109*, 07 2011.

[11] Y.-L. Hsueh and H.-C. Chen, "Map matching for low-sampling-rate gps trajectories by exploring real-time moving directions," *Information Sciences*, vol. 433-434, pp. 55–69, 2018.

[12] K. Zheng, Y. Zheng, X. Xie, and X. Zhou, "Reducing uncertainty of low-sampling-rate trajectories," in *2012 IEEE 28th International Conference on Data Engineering*, pp. 1144–1155, 2012.

[13] M. A. Quddus, W. Y. Ochieng, and R. B. Noland, "Current map-matching algorithms for transport applications: State-of-the art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 15,no. 5, pp. 312 – 328, 2007.

[14] M. Hashemi and H. A. Karimi, "A critical review of real-time map-matching algorithms: Current issues and future directions," *Computers, Environment and Urban Systems*, vol. 48, pp. 153–165, 2014.

[15] M. Hashemi, "Reusability of the output of map-matching algorithms across space and time through machine learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, pp. 3017–3026, Nov 2017.

[16] C. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, and P. Jaillet, "Online map-matching based on hidden markov model for real-time traffic sensing applications," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pp. 776–781, 2012.

[17] S. Taguchi, S. Koide, and T. Yoshimura, "Online map matching with route prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 338–347, 2019

[18] L. Luo, X. Hou, W. Cai, and B. Guo, "Incremental route inference from low-sampling gps data: An opportunistic approach to online map matching," *Information Sciences*, vol. 512, pp. 1407–1423, 2020.

[19] P. Chao, Y. Xu, W. Hua, and X. Zhou, "A survey on map-matching algorithms," in *Databases Theory and Applications* (R. Borovica-Gajic, J. Qi, and W. Wang, eds.), (Cham), pp. 121–133, Springer International Publishing, 2020.

[20] C. Yang and G. Gidófalvi, "Fast map matching, an algorithm integrating hidden markov model with precomputation," *International Journal of Geographical Information Science*, vol. 32, no. 3, pp. 547–570, 2018.

[21] P. Newson and J. Krumm, "Hidden markov map matching through noise and sparseness," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS'09*, (New York, NY, USA), pp. 336–343, ACM, 2009.

[22] D. Zhang, Y. Dong, and Z. Guo, "A turning point-based offline map matching algorithm for urban road networks," *Information Sciences*, vol. 565, pp. 32–45, 2021.

[23] M. R. Gupta, S. Bengio, and J. Weston, "Training highly multiclass classifiers," *J. Mach. Learn. Res.*, vol. 15, p. 1461–1492, Jan. 2014

[24] J. Kim, "Node based map matching algorithm for car navigation system," in *International Symposium on Automotive Technology Automation Global Deployment of Advanced Transportation Telematics/its*, 1996.

[25] C. E. White, D. Bernstein, and A. L. Kornhauser, "Some map matching algorithms for personal navigation assistants," *Transportation Research Part C: Emerging Technologies*, vol. 8, no. 1, pp. 91–108, 2000.

[26] J. S. Greenfeld, "Matching gps observations to locations on a digital map," in *Transportation Research Board. Meeting, National Research*

Council, (Washington, DC, USA), 2002

[27] S. Brakatsoulas, D. Pfoser, R. Salas and C. Wenk, "On map-matching vehicle tracking data," in *Proceedings of the 31st International Conference on Very Large Data Bases, VLDB'05*, pp. 853–864, VLDB Endowment, 2005.

[28] H. Alt, A. Efrat, G. Rote, and C. Wenk, "Matching planar maps," *Journal of Algorithms*, vol. 49, no. 2, pp. 262–283, 2003.

[29] C. Wenk, R. Salas, and D. Pfoser, "Addressing the need for map-matching speed: Localizing global curve-matching algorithms," in *18th International Conference on Scientific and Statistical Database Management (SSDBM'06)*, pp. 379–388, July 2006.

[30] B. Hummel, "Map matching for vehicle guidance," *Dynamic and Mobile GIS: Investigating Space and Time*, 2006

[31] J. Krumm, E. Horvitz, and J. Letchner, "Map matching with travel time constraints," in *SAE Technical Paper*, SAE International, 2007.

[32] Z. Shen, W. Du, X. Zhao, and J. Zou, "Dmm:Fast map matching for cellular data," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking, MobiCom '20*, (New York, NY, USA), Association for Computing Machinery, 2020.

[33] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[34] "Porto taxi database." `https://www.kaggle.com/c/pkdd-15-predict-taxi\\-service-trajectory-i/data`, 2015. Accessed: 2022-05-31.

[35] OpenStreetMap, "Open street map." `www.openstreetmap.org`. Accessed: 2022-05-31.

[36] "Taxi service trajectory prediction challenge, ecml-pkdd 2015."`https://archive.ics.uci.edu/ml/datasets/Taxi+Service+Trajectory+-+Prediction+Challenge,+ECML+PKDD+2015`, 2015. Accessed: 2022- 05-31

Thapana Boonchoo is currently a lecturer of Department of Computer Science, Faculty of Science and Technology, Thammasat University, Thailand. He received a Ph.D. degree in Computer Science from Institute of Computing Technology, Chinese Academy of Sciences, China, 2020, M.S. degree in Computer Science from Tsinghua University, China, in 2016, and B.S. degree in Computer Science from Thammasat University, Thailand, in 2012. His research interests include data clustering algorithms, representation learning, machine learning, and data mining.