A Map Matching Algorithm based on Modified Hidden Markov Model considering Time Series Dependency over Larger Time Span

Ha Yoon Song*, Jae Ho Lee

Department of Computer Engineering, Hongik University, 94 Wausan-ro, Mapo-gu, Seoul, South Korea

Abstract

With the advancement of geopositioning systems and mobile devices, much research with geopositioning data are currently ongoing. Along with the research applications, map matching is a technology that infers the actual position of error-prone trajectory data. It is a core preprocessing technique for trajectory data. Among various map matching algorithms, map matching using Hidden Markov Model (HMM) has gained high attention. However, the HMM model simplifies the dependency of time series data excessively, which leads to inferring incorrect matching results for various situations. For example, complex road relationships or movement patterns, such as in urban areas, or serious observation errors and sampling intervals make matching more difficult. In this research, we propose a new algorithm called trendHMM map matching, which complements the assumptions of HMM.

*Corresponding author

Email addresses: hayoon@hongik.ac.kr (Ha Yoon Song), wogh6860@gmail.com (Jae Ho Lee)

This algorithm considers a wider range of dependencies of geopositioning data by incorporating the movements of neighboring data into the matching process. For this purpose, the concept of the window containing adjacent geoposioning data is introduced. Thus trendHMM can utilize relationships among continuous geopositioning data and showed considerable enhancement over HMM-based algorithm. Through experiments, we demonstrated that trendHMM map matching provides more accurate results than the existing HMM map matching for various environments and geopositioning data sets. Our trendHMM algorithm shows up to 17.58% of performance enhancement compared to HMM based one in terms of Route Mismatch Fraction.
Keywords:
Map Matching, Hidden Markov Model, Geopositioning Data, Trajectory Data

1. Introduction

Due to advancements in wireless communication and geopositioning technologies, it has become possible to collect a large amount of geolocation data from various devices such as smartphones and vehicles in different geopositioning systems. The continuous movements of objects can be represented in periodic time series form. However, issues such as low measurement frequency and measurement errors can lead to discrepancies between the actual position and the collected data. Therefore, proper data preprocessing is necessary. During the data preprocessing process, map matching technology is

frequently used to infer the actual path taken by the subject. This approach involves depicting the connectivity between roads as a graph and inferring which edge, or road, the data at a specific point in time corresponds to. Among various map matching algorithms, those using the Hidden Markov Model (HMM) have gained attention. The HMM-based map matching algorithm, first introduced by Newson and Krum [1], has been widely utilized in research due to its performance and robustness. Particularly, it is known for its excellent performance with data having sampling intervals of less than 30 seconds. However, the existing HMM-based map matching approach oversimplifies the problem. One fundamental issue is that it considers the data at time t to be dependent only on the data at time $t-1$. Therefore, the existing HMM-based map matching approach yields inaccurate results for data with large errors or sampling intervals, as well as for areas with complex road networks such as urban areas. To overcome these limitations, there have been studies utilizing high-order HMMs that consider time points earlier than $t-1$, such as [2, 3]. However, the computational cost significantly increases as the dimensionality increases, preventing the extension of the model beyond second order. Therefore, instead of simply increasing dimensions, we considered the dependencies among various data by grouping multiple data points to account for the movement trend in the dataset, without explicitly expanding the dimensionality. In this research, we propose the trendHMM map matching algorithm, which complements the existing HMM-based map matching algorithm by considering a wider range of dependencies. This algorithm takes

into account the "trend," which represents broader movements by grouping the matching data and neighboring data. By considering the trend, we were able to address the issues of dependency and dimensionality increase that the conventional approach had. Through experiments using various geopositioning data, we confirm that the trendHMM map matching algorithm yields more accurate results than the existing HMM map matching approach. Furthermore, we performed a detailed comparison between trendHMM and the well-known HMM-based map matching algorithm, which is a representative map matching algorithm, both with and without preprocessing. This allowed us to make a precise comparison between trendHMM and HMM. The results of this comparison can be found in subsection 4.3.

The key contributions of this research are as follows:

- It expands the dependency between data by reflecting the movement trend of data.
- It demonstrates good performance without the need for data preprocessing, unlike conventional HMM-based algorithms.
- It exhibits similar time and space complexity as conventional HMM-based algorithms, but achieves better performance.

The contents of this research paper are as follows: Section 2 discusses about research results related with our topic. In section 3, a new algorithm developed in this paper, trendHMM, will be explained. As well, we will discuss existing map matching algorithm based on HMM as a basis of our algorithm

along with terminology used in the research paper. Section 4 will show the experimental results, both for HMM-based algorithms and trendHMM algorithm as well as comparison of results of both algorithms. Final section 5 will conclude this research and discuss about possible future researches.

2. Related Works

According to the map matching survey research [4], map matching algorithms can be broadly classified into four categories based on the applied techniques: Similarity Model, Candidate-Evolving Model, Scoring Model, and State-Transition Model. The Similarity Model infers the closest road geometrically and topologically. In other words, it matches the trajectory data to the road that is closest in terms of shape. Since the moving object always travels on the road network and cannot leap from one segment to another, the measured geolocation series closely resembles the actual path on the map. This model generally demonstrates high efficiency, but it exhibits lower accuracy when dealing with data with large sampling intervals or errors, or in complex road scenarios.

There exists another related survey to our study [5]. In this survey, the authors classified and reviewed existing map matching algorithms. The previously mentioned Similarity Model, Candidate-Evolving Model, Scoring Model, State-Transition Model, etc., are also included, and our research falls within the State-Transition Model category. In addition to this, various research results related to map matching include the following.

In a research [6], the Similarity Model is further classified into two subcategories: point-to-curve matching, where the model is applied to each point of the trajectory data, and curve-to-curve matching, where the model is applied to segments formed by grouping the trajectory data. This classification is also mentioned in research results such as [7] and [8]. In point-to-curve matching, each point of the trajectory data is matched with the nearest edge of the road network. On the other hand, in curve-to-curve matching, the trajectory data is grouped to form trajectory (Tr) segments, and then matched with the closest edge. Various models have been proposed based on different definitions of proximity, and notable research in this area include [9] and [10].

The Candidate-Evolving Model maintains a candidate set (also known as particles or hypotheses) during the map matching process. The candidate set is initialized by the first trajectory (Tr) sample, and only the candidates that are closest to the latest observation are kept from the existing candidates. By iterating the algorithm, the number of times each candidate is included in the candidate combination can be calculated. The most frequently included candidates are then used to compute segments and determine the matching path. Prominent models in this category include methods that combine Monte Carlo sampling techniques and Bayesian inference, such as Particle Filter-based approaches [11, 12], and methods that utilize the Multiple Hypothesis Technique (MHT) [13]. These approaches effectively utilize the candidate set to infer the most likely matching path.

The Scoring Model, as described in research such as [14] and [15], does

not rely on a specific model. Instead, it searches for candidate points that maximize a pre-defined scoring function for each Tr segment. In particular, a research shown in [15] achieved lane-level map matching performance using this approach. The road network is partitioned into grid cells. For each timestamp, the candidate grid cells corresponding to the observed values are identified, and the candidate with the highest scoring function is selected. The scoring function is determined using four linear features, such as the proximity between the grid cell and the trajectory sample. By utilizing the scoring function, the Scoring Model evaluates and selects the most suitable candidates for each Tr segment, providing a flexible and effective approach for map matching tasks.

The State-Transition Model constructs a weighted topological graph consisting of all possible paths that the subject can take. In this graph, nodes represent the possible states of the subject at specific moments, and edges represent transitions between states at different timestamps. The model infers the optimal path by determining the path that maximizes the weights. Notable approaches in this category include the conditional random field (CRF) model [16], the weighted graph transition (WGT) model [17, 18], and methods that utilize Hidden Markov Model (HMM). While these algorithms share similarities, they differ in how they calculate the weights. We focuses on the method that utilizes HMM, which is a popular choice for map matching tasks.

HMM (Hidden Markov Model) is one of the most widely used methods in

the State-Transition Model for map matching. HMM focuses on cases where the states in a Markov chain cannot be directly observed but can be inferred from the observed measurements. This assumption aligns well with map matching problems, where each point in the trajectory (Tr) is treated as an observed measurement, and the actual position of the subject is considered as an unobserved state. In HMM, the roads near the measured values are considered potential locations (states) of the subject due to observation errors in the trajectory. The probability of the measured values being observed when the subject is actually located on a road is expressed as emission probability. The probability of the subject transitioning from one candidate location to the next candidate location consecutively is represented as the transition probability. By finding the combination of candidate locations with the highest probability, HMM determines the final matching path.

HMM-based map matching can be broadly classified into online and offline models. The offline model uses the entire trajectory (Tr) to understand the overall relationships within it. It demonstrates robust performance against variations in sampling intervals and observation errors but can be computationally inefficient. The online model, first proposed in [19], performs map matching by incorporating the data collected in real-time and constructing segments. It is utilized in real-time navigation and similar online services, and most online models employ sliding window techniques for the matching process. The algorithm we will present belongs to offline algorithm.

Recently, several algorithms have been proposed to enhance traditional

HMM-based map matching by incorporating a range of factors such as speed, angle, preference, and more, aiming to achieve higher accuracy. For example, research like [20, 21, 22, 23] include factors such as speed limits, road levels, and the difference between the vehicle's heading change and the road segments' heading change in the probability calculations. Additionally, research like [23, 24, 25] take into account driver's travel preference information obtained through heuristics or learning. However, these models assume a perfect road network representation. They assume that the road network includes all roads existing at the time when the trajectory is measured and does not account for any hidden or missing roads. As more factors are considered in map matching, there is a tendency to interpret the data according to the road network as much as possible. Therefore, these models, as shown in [26], may not be suitable for solving the map inference problem of discovering hidden roads when the network itself is incorrect.

Indeed, there have been studies that consider a larger temporal dependency to overcome the Markov property in HMM. Since the movement of the target is a continuous time series, there exist complex spatio-temporal relationships between the current state and previous states. In other words, the Markov property overly simplifies the map matching problem. To address this, [23] proposed a second order HMM-based map matching algorithm, which showed better performance than the first-order approach. However, [23] did not extend the algorithm beyond two dimensions due to computational efficiency issues. Our aim is to provide an algorithm which maintains

the same time complexity as traditional HMM-based map matching but incorporates a larger temporal dependency compared to second order HMM.

There are research results that share similar objectives to our study [27]. It proposes methods to minimize errors in GPS observation data, but its approach to model construction differs from ours. Moreover, its focus on online algorithms is apart from our study. There are also research outcomes that utilize additional equipment over map matching algorithms. In [28], this research demonstrates cases where map matching technology is applied. It introduced radar/INS integrated techniques with map matching techniques to achieve the high level of accuracy required for autonomous navigation.

We proposes a statistical model called trendHMM map matching, which complements traditional HMM map matching. We primarily focused on comparing the most common map matching algorithm, Hidden Markov Model (HMM) map matching, rather than other map matching algorithms shown in subsection 4.3. To consider a larger temporal dependency beyond just the previous timestamp, trendHMM map matching constructs a window by grouping several data points based on the point of interest. Representative points within the window are selected to form a new trajectory Tr_{new} that captures broader movements. During the map matching process of Tr_{new}, the generated weights are additionally considered, reflecting the movements and trends over a wider range. The size of the window in trendHMM can be adjusted to control the temporal dependency, and even with varying window sizes, it provides the same time complexity as the HMM approach. Further-

more, as trendHMM is a purely statistical algorithm without external factors, it is expected to achieve better performance in map inference problems.

3. An Enhanced Version of HMM based Map Matching Algorithm

In this section we have to define terminology used in this paper. Then, we will explain basics of existing HMM based map matching algorithm as a basis of our algorithm, with its limitation and point of advancement. The core of this section is subsection 3.4 which explains trendHMM algorithm on the basis of previous subsections in this section.

3.1. Terminology

Movement data refers to measured geopositioning coordinates, and a trajectory represents the temporal sequence of movement data. Each geopositioning coordinate includes longitude, latitude, and a timestamp, and there may be discrepancies between the measured coordinates and the actual positions due to inherited geopositioning system properties and/or environmental problems. Regarding GPS error is described in [29].

Map matching is a crucial technology for preprocessing trajectories. It involves representing the connectivity between roads in the form of a graph and assigning each movement data point to a specific road segment, thereby inferring the actual location of the object. The concepts used in this manuscript are as follows.

- Geolocation data : It refers to a single geopositioning point. The t-th movement data in the trajectory is denoted as $g(t)$.

- Trajectory (Tr) : It is a time series of movement data. It can be represented as $g(1) \rightarrow g(2) \rightarrow ... \rightarrow g(n)$, and can be written as Tr_n. Then the number of geolocation data in Tr_n, i.e., cardinality of Tr_n is n.

- Road Network : It is represented as a graph to depict the connectivity relationship between roads, denoted as $G(V, E)$. Here, V represents the set of nodes, and E represents the set of edges.

- Route : It is a time series composed of edges from the Road Network that match the trajectory, denoted as Tr.

3.2. Basics on Map Matching based on HMM

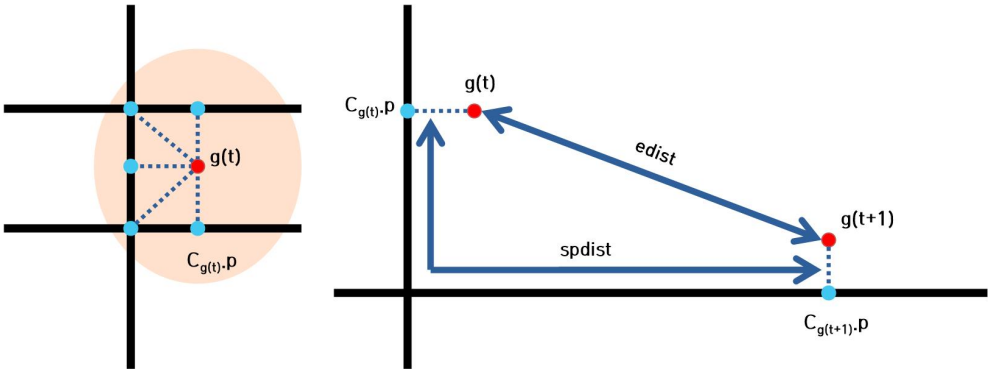
The purpose of map matching is to match each movement data in the trajectory, denoted as Tr, with a specific edge in the road network. The HMM proceeds by modeling the edge where the object is actually located at that point as a hidden state, and modeling the measured location point as an observed value.

First, based on a given mobility dataset, roads within a radius of r are selected as candidates for the data point. $C_{g(t)} \in E$ represents a candidate for the data point $g(t)$. The emission probabilities and transition probabilities are calculated using the shortest distance between the data point and candidate edges, as well as the points $C_{g(t)}.p$ and $C_{g(t+1)}.p$ located at the shortest

distance. The emission probability represents the probability of observing a measurement given the actual road position. The emission probability EP for the candidate $C_{g(t)}$ is defined by equation 1.

$$EP(C_{g(t)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp^{\frac{-dist^2}{2\sigma}} \tag{1}$$

In Equation 1, $dist$ represents the distance between $g(t)$ and $C_{g(t)}.p$, and EP follows a Gaussian distribution with zero mean for $dist$.



(a) Example of $C_{g(t).p}$ (b) Example of $spdist, edist$ calculated by $g(t), g(t+1)$ and $C_{g(t)}.p, C_{g(t+1)}.p$

Figure 1: Typical Operation of Map Matching Algorithm based on HMM

The transition probability represents the probability of transitioning from $C_{g(t)}$ to $C_{g(t+1)}$ in reality. As shown in Figure 1, let $edist$ be the distance between $g(t)$ and $g(t+1)$, and let $spdist$ be the shortest distance traveled along the road between $C_{g(t)}.p$ and $C_{g(t+1)}.p$. The transition probability TP is calculated as Equation 2, and it follows an exponential distribution with respect to the difference between $spdist$ and $edist$.

$$TP(C_{g(t)}, C_{g(t+1)}) = \frac{1}{\beta} \exp^{-\frac{|edist - spdist|}{\beta}} \quad (2)$$

The actual path inferred by the algorithm is the combination of candidates that maximizes Equation 4 with respect to Tr. Equation 3 represents the weights that will be used to compute emission and transition probabilities for each candidate of the data. The weights are larger for candidates that are closer to the data and have smaller differences between *edist* and . This can be efficiently computed using the Viterbi algorithm [30]. The algorithm based on the Viterbi algorithm is described in Algorithm 1.

Algorithm 1 An algorithm of HMM map matching with Viterbi

Input: CDS, array of sets of candidate site objects ($C_{g(t)}$) for all geolocations.

Function `forward(`CDS`):`

```
L ⟵ length of CDS;
 for each C_g(0) in CDS[0] do
    C_g(0).prob ⟵ 0;
end
for i ← 0 to L − 2 do
    for each C_g(i) in CDS[i] do
        j ⟵ i + 1;
         for each C_g(j) in CDS[j] do
            W ⟵ TP(C_g(i), C_g(j)) + EP(C_g(j));
             if C_g(j).prob < C_g(i).prob + W then
                C_g(j).prob ⟵ C_g(i).prob + W;
                 C_g(j).prev ⟵ C_g(i);
            end
        end
    end
end
```

End Function

Input: CDS, array of candidate site sets for all geolocations.

Output: *result*, list of street numbers that the algorithm matches to each geolocation.

Function `backward(`CDS`):`

```
result ⟵ emptylist;
 L ⟵ length of CDS;
 temp ⟵ The candidate object in CDS[L − 1] with the highest prob;
 while temp is not null do
    insert temp.edge to result;
     temp ⟵ temp.prev;
end
reverse result's order;
 return result;
```

End Function

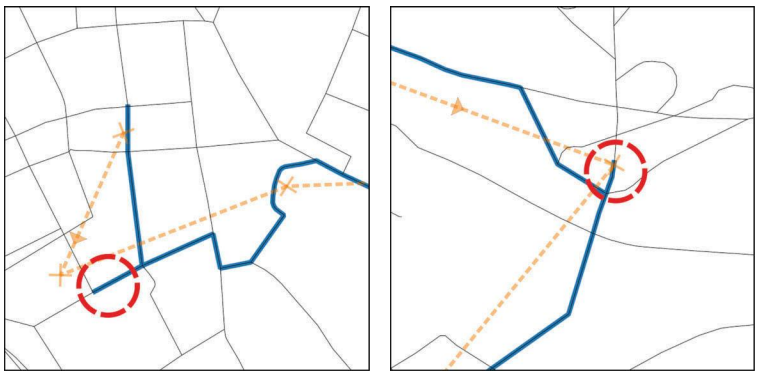
In Algorithm 1, the Viterbi algorithm is divided into two stages: forward and backward. The input parameter CDS is an array of candidate sets for each geolocation data in Tr. In other words, $CDS[t]$ represents the set of candidate objects $C_{g(t)}$ for $g(t)$. Each candidate object $C_{g(t)}$ has instance variables: *edge*, *prob*, and *prev*, which are denoted as $C_{g(t)}.variable$ in the algorithm. $C_{g(t)}.edge$ represents the road that the candidate object points to. $C_{g(t)}.prob$ stores the maximum *score* (Equation 4) from $g(1)$ to $g(t)$ in Tr. Therefore, for a given $C_{g(t)}$, the following equation (Equation 5) holds true. $C_{g(t)}.prev$ refers to the t-1 candidate object that maximizes the value in Equation 5.

$$W_{hmm}(C_{g(t)}, C_{g(t+1)}) = \log ep(C_{g(t+1)}) + \log tp(C_{g(t)}, C_{g(t+1)}) \tag{3}$$

$$score(Tr) = \sum_{t=1}^{n-1} W_{hmm}(C_{g(t)}, C_{g(t+1)}) \tag{4}$$

In Algorithm 1, the forward step is a function that computes the probabilities of all candidate objects. After executing the forward step, the candidate object in $CDS[L-1]$ with the highest prob value corresponds to the maximum value of $score(Tr)$ in Equation 4. The backward step is a function that returns the matched path. Starting from the candidate object with the highest prob value in $CDS[L-1]$, it follows the *prev* pointers to return the matching result of the path.

$$C_{g(t)}.prob = MAX(C_{g(t-1)}.prob + W_{hmm}(C_{g(t)})) \tag{5}$$

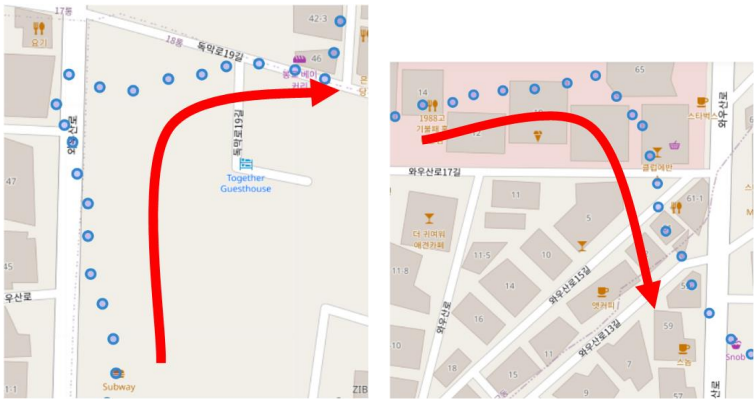


(a) A case with large sampling intervals (b) A case with reverse movement

Figure 2: Example of Erroneous Matching by HMM based Map Matching Algorithm

3.3. Limitation of Map Matching based on HMM

HMM assumes that the observation at time t depends only on the previous time step $t-1$, which is known as the Markov Property. However, this assumption oversimplifies the map matching process and can lead to incorrect results. Figure 2 illustrates incorrect matchings. The area inside the red circle shows incorrect matching, where the path deviates from the correct route. In Figure 2, (a) represents a case with large sampling intervals and significant data errors. While considering multiple time steps in the map matching process could help mitigate the impact of errors, HMM only considers the immediate previous time step, leading to ambiguous results. And (b) depicts a situation that can occur in densely populated urban areas. Similarly, due to data errors, a reverse movement can occur.



(a) A movement trend of vehicle (b) A movement trend of pedestrian

Figure 3: Raw Trend Examples: geolocation data (blue dot) and movement trend (red line)

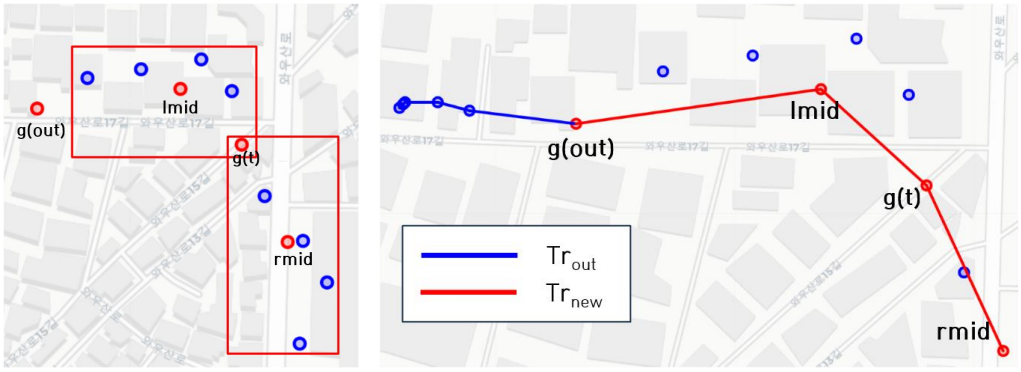
3.4. Proposed Algorithm: trendHMM Map Matching Algorithm

The main concepts of trendHMM are as follows. The movement data $g(t)$ at the time t is dependent on a wider range of movement data as well as the time $t-1$. In particular, neighboring data points such as $g(t-2)$, $g(t-1)$, $g(t+1)$, $g(t+2)$ have a closer relationship with $g(t)$ compared to other data points. Although these neighboring data points may exhibit different characteristics in terms of direction, speed, and other movement-related features, they share a common underlying movement pattern. These unique movement patterns exhibited by neighboring data points are defined as movement trend. Figure 3 provides an example of movement trend. The measured geopositioning coordinates (blue dots) exhibit variations due to observation errors and have different movement-related characteristics. However, they share a common movement trend (red line). The actual location of $g(t)$ is significantly

influenced not only by the immediate previous data point, $g(t-1)$ but also by the wider movement trend. To address this, we proposes a trendHMM map matching algorithm that supplements the traditional HMM model with movement trends. We introduces a weight W_{trend} that considers the movement trend in addition to the existing weight W_{hmm} combined into equation 4 when calculating the map matching score. In other words, trendHMM aims to infer the path by maximizing the following equation 6, which takes both the traditional HMM score and the movement trend weight into account.

$$trend\_score(Tr) = \sum_{t=1}^{t=n-1} (W_{hmm}(C_{g(t)}, C_{g(t+1)}) + W_{trend}(C_{g(t+1)})) \quad (6)$$

W_{trend} assigns a higher value to candidates that fit the movement trend. The specific method for calculating W_{trend} for candidate $C_{g(t)}$ is as follows. Firstly, create a window by grouping W neighboring data points around $g(t)$ as the reference. This window is constructed to capture the relevant movement trend, as illustrated in Figure 4(a). Within the window, including $g(t)$, calculate the centroid ($lmid$) of the data points from previous time steps. After obtaining the centroid $lmid$ for the data points preceding $g(t)$, the same procedure is applied to the subsequent data points to obtain the centroid $rmid$. Then, as illustrated in Figure 4 (b), we connect $g(out)$, $lmid$, $g(t)$, and $rmid$ to create a new trajectory Tr_{new}. $g(out)$ represents the data point outside the window, and its corresponding out value is determined as $max(0, t-w+1)$, where w is window size. By considering the centroids,



(a) Example of using window (red box) to create Tr_{new}, which is composed by $g(out), lmid, g(t), rmid$ (red dot).

(b) Example of Tr_{new} and Tr_{out}

Figure 4: Examples of TrendHMM Algorithm Operations

Tr_{new} incorporates the movement trend more robustly, even in the presence of errors. In Figure 4, Tr_{out} represents a trajectory constructed from the original trajectory up to $g(out)$. Finally, the generated Tr_{new} and Tr_{out} are used to calculate $W_{trend}(C_{g(t)})$ according to Equation 7.

$$W_{trend}(C_{g(t)}) = \frac{1}{N} max(trend\_score(Tr_{out}) + score(Tr_{new})) \tag{7}$$

The equation 7 represents the maximum value obtained from conducting trendHMM map matching on Tr_{out} and performing traditional HMM map matching on Tr_{new} while fixing the candidate for $g(t)$ as a specific $C_{g(t)}$. This can be efficiently computed using the Viterbi algorithm. It's important to note that equation 7 incorporates the influence of the original Tr_{out}. The trend score $trend\_score(Tr_{out})$ is calculated by the Viterbi algorithm before

determining $W_{trend}(C_{g(t)})$. The normalization parameter N is used to ensure that the magnitudes of each W_{hmm} and W_{trend} in equation 6 are equal. The value of N can be determined from the equations. The total number of weights for $trend\_score(Tr_{out})$ is $2 \times out$ from Equation 6, and the total number of weights for $score(Tr_{new})$ is 3 from Equation 5. Therefore, the value of N can be set to $2out + 3$ for the calculation of equation 7.

The trendHMM algorithm using the Viterbi algorithm is described in Algorithm 2. In Algorithm 2, trendForward is an algorithm that modifies the forward step to align with trendHMM. The function addTrendWeight incorporates the trend into the calculation according to the trendHMM algorithm. Afterwards, the backward step of Algorithm 1 is performed to return the final matching path.

The algorithm proposed in this research has the same time complexity as traditional HMM-based map matching. Let's denote the average number of candidate points for each geolocation coordinate as C, and the number of geolocation coordinates in Tr as T. The time complexity of traditional HMM-based map matching, determined by the Viterbi algorithm, is $O(TC^2)$. For trendHMM, each geolocation requires an additional map matching for Tr_{new}, which consists of 4 data points. This means that 3 additional computations are performed for each geolocation compared to the traditional HMM. However, since C and T remain the same, the overall time complexity of trendHMM remains $O((3T+T)C^2) = O(4TC^2) = O(TC^2)$. Therefore, we can conclude that the time complexity of both algorithms are the same.

Algorithm 2 An algorithm of trendHMM with map matching Viterbi

Input: CDS, array of candidate site object (CD) sets for all geolocations.
Function `addTrendWeight(`t, CDS`):`

make candidates array $trendList$ with $g(out), lmid, g(t), rmid$ as shown in Figure 4;
calculate $W_{trend}(C_{g(t)})$ using equation 7 with $trendList$ and add to $C_{g(t)}.prob$;

End Function
Input: CDS, array of candidate site object (CD) sets for all geolocations.
Function `trendForward(`CDS`):`

$L \longleftarrow$ length of CDS;
for *each* $C_{g(0)}$ *in* $CDS[0]$ **do**
  $CD.prob \longleftarrow 0$;
end
for $i \leftarrow 0$ **to** $L-2$ **do**
  addTrendWeight(i, CDS)
  for *each* $C_{g(i)}$ *in* $CDS[i]$ **do**
    $j \longleftarrow i+1$
    for *each* $C_{g(j)}$ *in* $CDS[j]$ **do**
      $W \longleftarrow TP(C_{g(i)}, C_{g(j)}) + EP(C_{g(j)})$;
      if $C_{g(j)}.prob < C_{g(i)}.prob + W$ **then**
        $C_{g(j)}.prob \longleftarrow C_{g(i)}.prob + W$;
        $C_{g(j)}.prev \longleftarrow C_{g(i)}$;
      end
    end
  end
end
addTrendWeight(L-1);

For the spatial complexity, both HMM and trendHMM have the same spatial complexity, which is $O(L)$, where L represents the number of candidate spatial objects for the next step. Here, L is a variable meaning the length of trajectory. Regardless of window size, two major points are generated for map matching one point. Algorithm 2 generates two more CDS during the process. Thus, we can conclude the space complexity of trendHMM algorithm is $O(L + 2L) = O(L)$.

4. Experiments

4.1. Data and Experiment Setup

The dataset used in the experiments consists of a large-scale real-world dataset comprising 100 geopositioning tracks, with locations spanning various locations worldwide.The dataset is publicly available in [31]. Each track is represented by a map or a route that accurately matches the map. Additionally, some tracks are labeled with features that may pose challenges for map matching algorithms.

- u-turns: the vehicle turned 180° and reversed the direction of travel
- hives: large numbers of points packed in a small area
- loops: the vehicle was traveling in circles
- gaps: temporal gaps existing in the track

- severe congruence issues: situations where the map and the track are incongruent or dissimilar

In addition, in the data sets, there are 19 tracks formed by 19 high-quality geopositioning data points without connected segments. The length of the tracks ranges from 5 to 100 kilometers, and the dataset contains a total of 247,251 points with a sampling rate of 1Hz. The average length and duration of the 100 tracks are 26.8 km and 4950.7 seconds, respectively. More detailed information can be found in the dataset documentation [31].

We subsampled the original datasets in order to create five different datasets concentrating on sampling intervals rather than one second of original datasets: 10 seconds, 20 seconds, 30 seconds, 60 seconds, and 120 seconds. Additionally, we generated preprocessed datasets using the Douglas-Peucker algorithm [32] on each dataset.

The matching accuracy is measured by the Route Mismatch Fraction (RMF) proposed by Newson and Krumm [1]. The measured values include the total length of false positive road segments, denoted as d_+, and the total length of false negative road segments, denoted as d_-. If we consider d_0 as the sum of d_+, the RMF quantifies the map matching error as the ratio $(d_+ + d_-)/d_0$. A smaller RMF value indicates that the map matching results are more similar to the actual path.

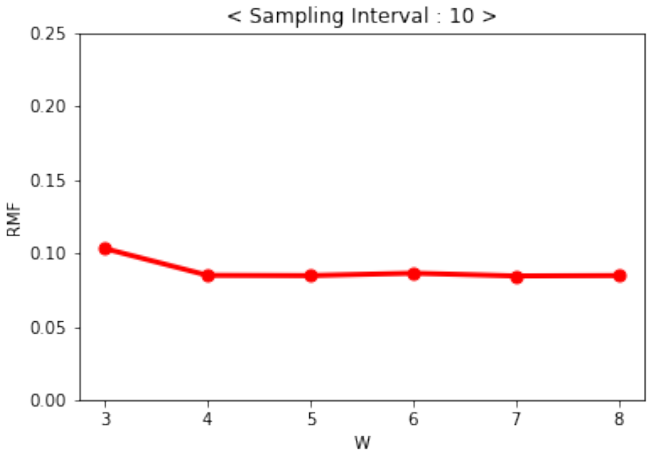
Next, we evaluate the efficiency of trendHMM for geolocation data with various sampling intervals. To compare trendHMM with the traditional HMM-based method, we derive the RMF values for each approach. Throughout the

experiments, we set the geopositioning measurement standard deviation σ to 10 $meters$ in Equation 1 for both methods. The scaling factor β in Equation 2, used for estimating transition probabilities, is also set to 10 $meters$, but the qualitative results are maintained within the range reported in [24]. For efficiency purposes, we construct candidates by searching for road segments within a radius of 200 $meters$ from each geopositioning point. All methods are implemented in Python, and the experiments are conducted on two Intel Xeon CPU E5-2630 v2 2.60GHz CPUs with 64GB RAM.

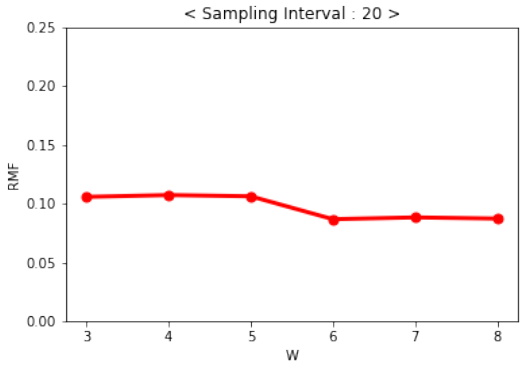
4.2. Parameter Estimation

The trendHMM algorithm proposed in this research requires an additional estimation of the window size, denoted as W. According to the algorithm in Section 3.4, as the value of W increases, the distance and time interval between the considered movement data points also increase. In other words, the range of temporal dependencies expands.

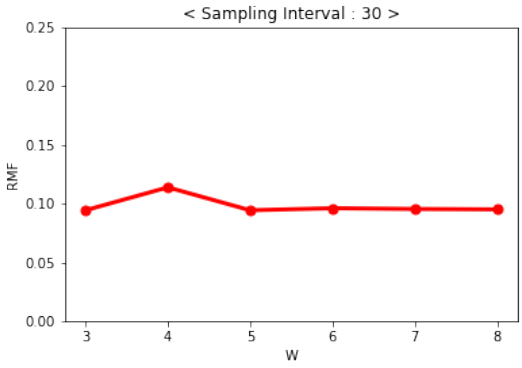
In this experiment, we conducted experiments by varying the value of W as shown in Figure 5 to select an appropriate W value for the data generated with different sampling intervals (10 sec, 20 sec, 30 sec, 60 sec, 120 sec). Overall, we observed a trend where a higher W value tends to result in lower RMF values when the sampling interval is smaller. However, when the sampling interval is larger, we noticed that a higher W value leads to lower performance. To accurately estimate the precise location of the movement data $g(t)$, a narrow range of movement trend composed of data points



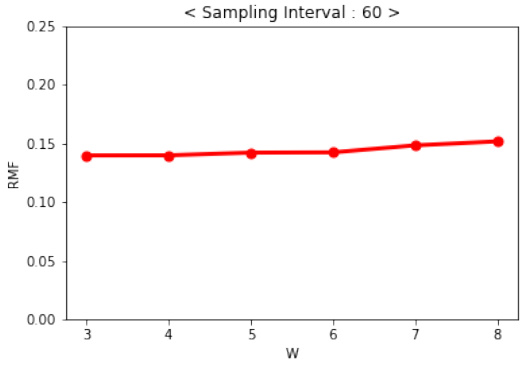
(a) Sampling Interval : 10



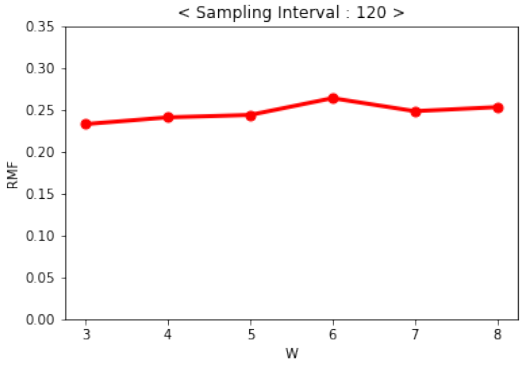
(b) Sampling Interval : 20



(c) Sampling Interval : 30



(d) Sampling Interval : 60



(e) Sampling Interval : 120

Figure 5: RMF Values with respect to Window Size W

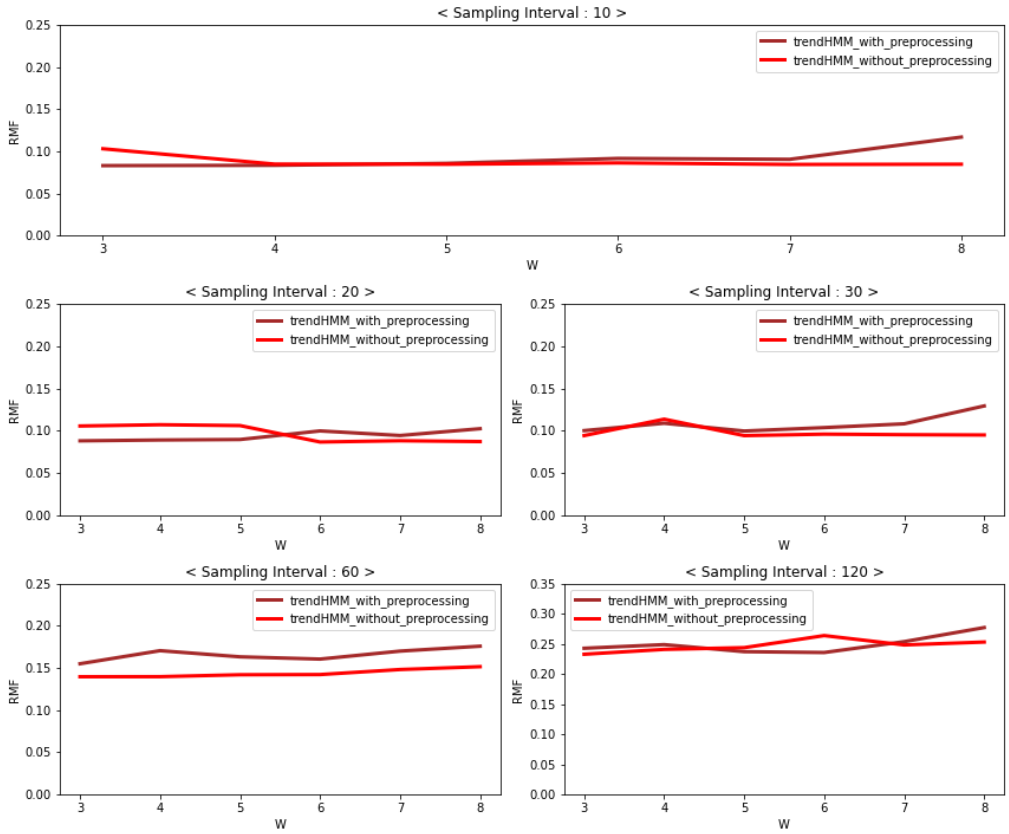
with close temporal and spatial proximity is required. However, with larger sampling intervals, although approximate location estimation is still possible, the lack of precision compared to smaller sampling intervals results in lower performance as the value of W increases.

4.3. Experimental Results

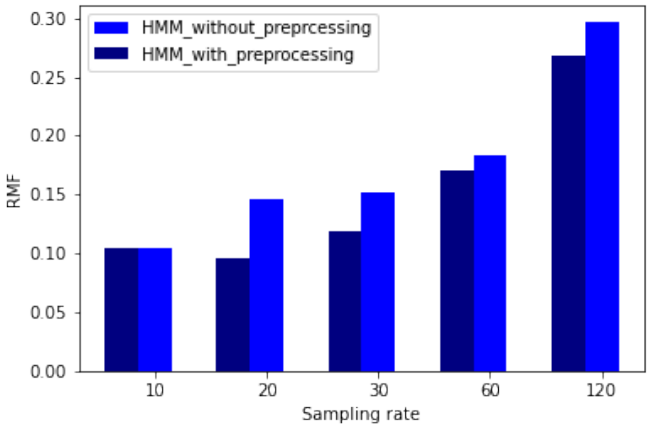
The experiments were conducted based on two main criteria: trendHMM vs. HMM and the presence or absence of data preprocessing presented by Douglas-Peucker algorithm [32]. In all experiments, the parameter value *epsilon* for the Douglas-Peucker algorithm was fixed at 0.001 for data preprocessing. The radius r for the $g(t)$ data was set to 0.002, and the number of neighboring data points to be grouped, denoted as w, was set to one of $\{3, 4, 5, 6, 7, 8\}$.

- *epsilon* : The value of the parameter *epsilon* for the Douglas-Peucker algorithm is 0.001.
- r : The value of the radius r for the $g(t)$ data is 0.002.
- w : The number of neighboring data points to be grouped, denoted as w, is a value from integer set $\{3, 4, 5, 6, 7, 8\}$.

Firstly, we compared the results of trendHMM and HMM with and without data preprocessing. Each of the basic algorithms was applied in the same manner, with the only difference being the inclusion or exclusion of

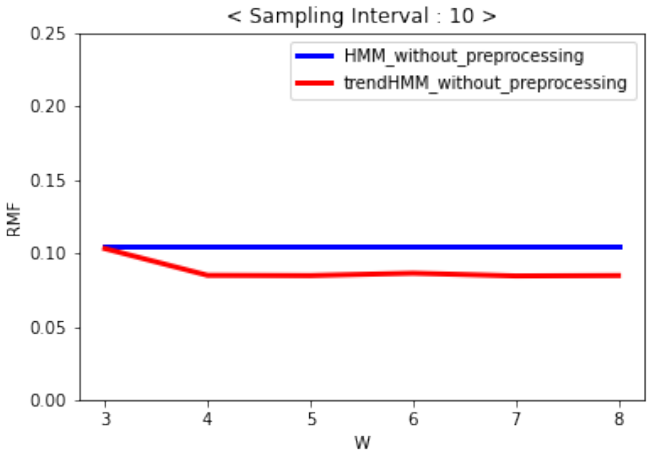


(a) The RMF values of trendHMM with and without Data Preprocessing

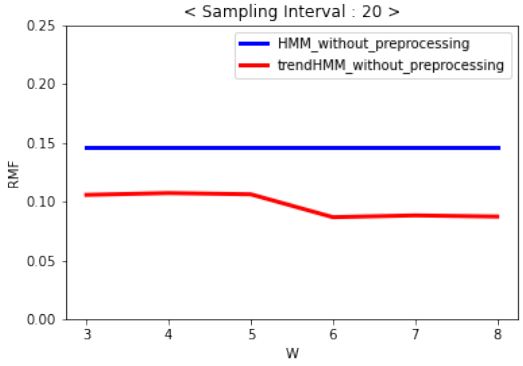


(b) The RMF values of HMM with and without Data Preprocessing

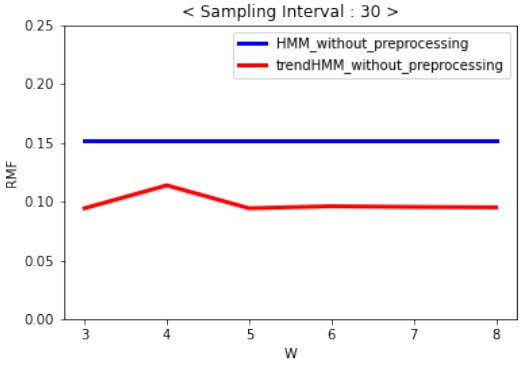
Figure 6: The RMF values of trendHMM and HMM with and without Data Preprocessing



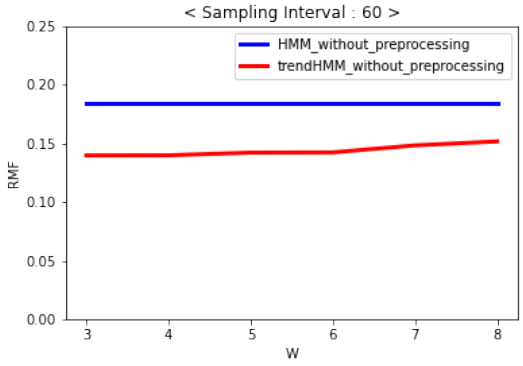
(a) Sampling Interval : 10



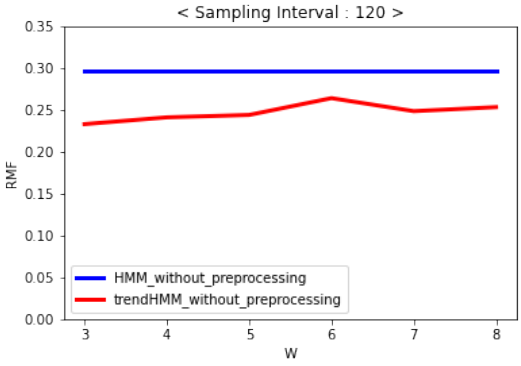
(b) Sampling Interval : 20



(c) Sampling Interval : 30

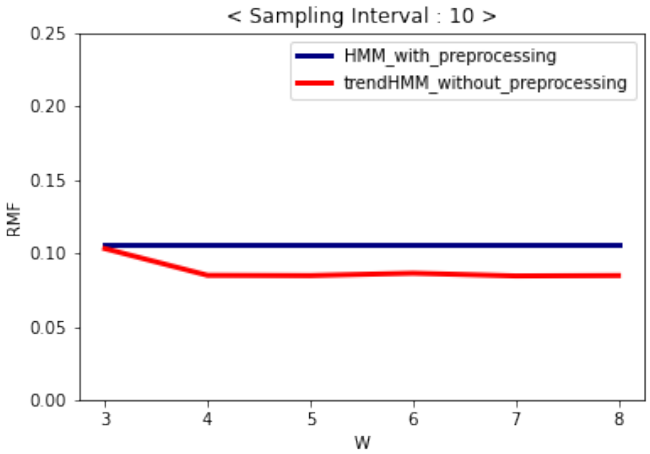


(d) Sampling Interval : 60

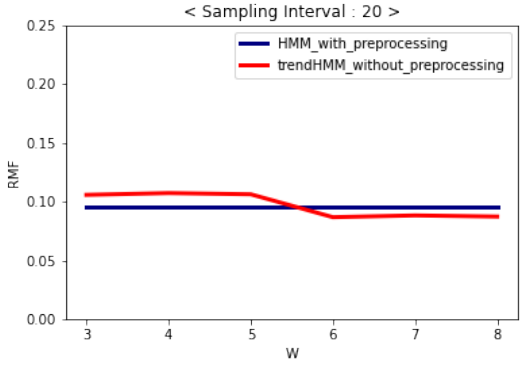


(e) Sampling Interval : 120

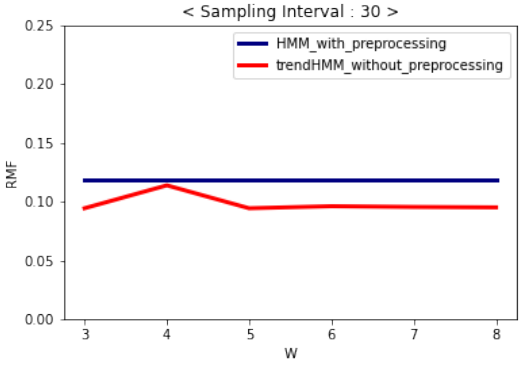
Figure 7: The RMF values for trendHMM and HMM, all without Preprocessing



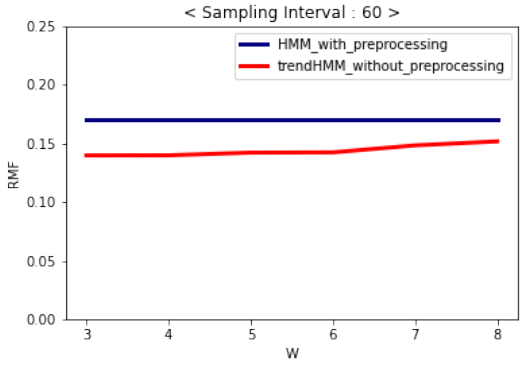
(a) Sampling Interval : 10



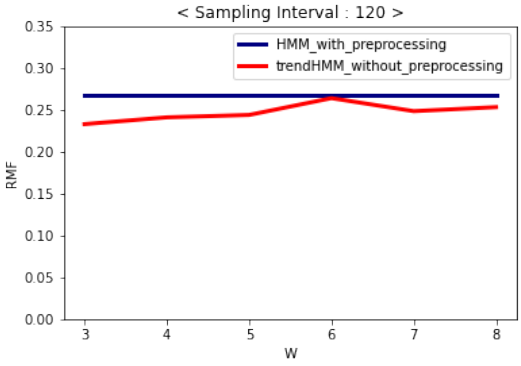
(b) Sampling Interval : 20



(c) Sampling Interval : 30



(d) Sampling Interval : 60



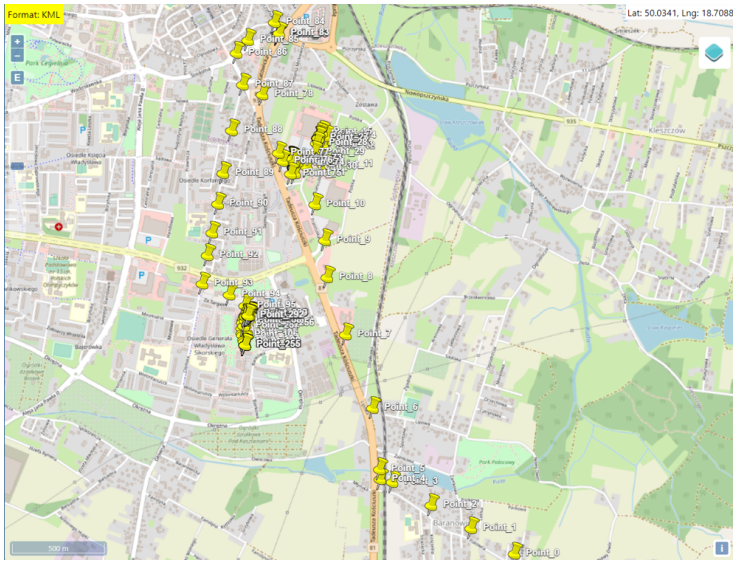
(e) Sampling Interval : 120

Figure 8: The RMF values for trendHMM without Preprocessing versus HMM with Preprocessing

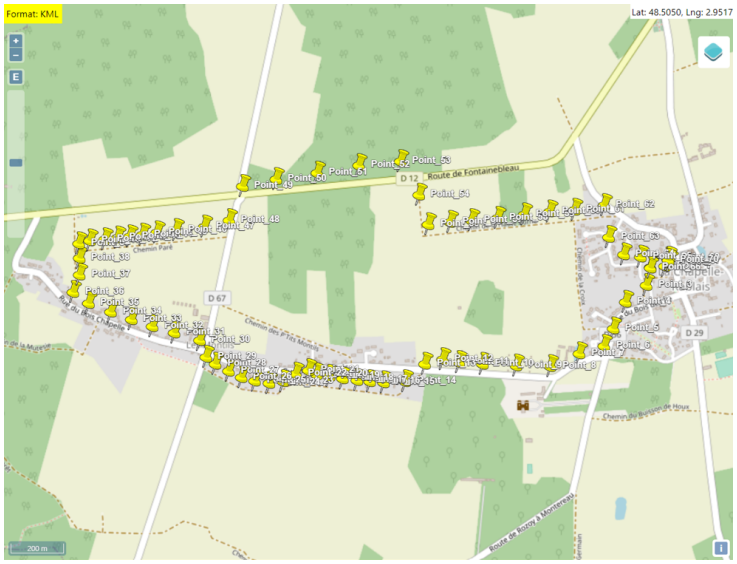
the data preprocessing step. In Figure 6, (a) shows that without preprocessing has better overall performance than with preprocessing in trendHMM, and (b) shows that with preprocessing is better in HMM. In Figure 6 (a), the red line representing "without_preprocessing" generally exhibits lower RMF values than the brown line representing "with_preprocessing." This means that trendHMM without preprocessing performs better compared to trendHMM with preprocessing. On the other hand, in Figure 6 (b), for HMM, "with_preprocessing" shows lower RMF values than "without_preprocessing." In other words, in the case of trendHMM, results without preprocessing are generally better, however, in the case of HMM, it was confirmed that the results with preprocessing are better. Next, we compared the non-preprocessed trendHMM with each HMM. Figure 7 shows a comparison with the HMM without preprocessing, and Figure 8 shows a comparison with the HMM with preprocessing. Overall, it can be confirmed that trendHMM without preprocessing shows superior performance than the cases of HMM with or without preprocessing.

However, in cases of a sampling interval of 20 sec and W values of $[3, 4, 5]$, trendHMM without preprocessing showed lower performance compared to the HMM with preprocessing. To investigate the cause, we plotted some of the data that resulted in lower performance on the map. Figure 9 shows that it corresponds to cases where the trajectory stays in one location for a long time.

Subfigure (a) represents the map with the highest error, and subfigure



(a) Map with the highest error.



(b) Map with the second highest error.

Figure 9: A Mapping of the Geopositioning Dataset representing Unexpected Results

Sampling Interval(s)	**10**	**20**	**30**	**60**	**120**
trendHMM_with _preprocessing	0.0918	**0.0938**	0.1083	0.1658	0.2491
trendHMM_without _preprocessing	**0.0880**	0.0968	**0.0980**	**0.1438**	**0.2469**
HMM_with _preprocessing	0.1053	0.0956	0.1189	0.1707	0.2678
HMM_without _preprocessing	0.1049	0.1463	0.1523	0.1846	0.2969

Table 1: Comparison of RMF Values w.r.t. Various Sampling Intervals

(b) represents the map with the second-highest error. This suggests that the lower performance in certain situations may be due to factors such as the data preprocessing and variations caused by the window size W. It is likely that some inaccuracies occurred under specific conditions. However, trendHMM with preprocessing at the corresponding sampling interval showed higher performance than HMM of each case.

According to Table 1, it can be observed that among all the models, trendHMM without preprocessing exhibits the highest performance. Additionally, when comparing trendHMM and HMM as shown in Table 2, irrespective of preprocessing, it can be observed that trendHMM shows improved

Sampling Interval(s)	**10**	**20**	**30**	**60**	**120**
trendHMM	0.0880	0.0938	0.0980	0.1438	0.2469
HMM	0.1049	0.0956	0.1189	0.1707	0.2678
Performance Enhancement	**16.11%**	**1.88%**	**17.58%**	**15.76%**	**7.80%**

Table 2: RMF values comparison: trendHMM versus HMM with their best RMF Values

performance. In general, the best RMF value of trendHMM was extracted from trendHMM without preprocessing, and the best RMF value of HMM was extracted from HMM with preprocessing.

Traditional HMM-based map matching only considers very short-term dependencies by utilizing the immediate past data. However, trendHMM has the advantage of being able to adjust the time dependencies more flexibly by using the window size W. Additionally, trendHMM achieves high performance even without data preprocessing, while HMM requires preprocessing as a crucial step. Therefore, trendHMM has the advantage of reducing the costs associated with preprocessing, making it more efficient compared to HMM.

5. Conclusion and Future Research

We proposed the trendHMM map matching method as an enhancement to the existing HMM-based map matching. The trendHMM overcomes the limitations of HMM by considering a wider range of movement patterns, while maintaining the same time complexity as traditional methods. Through experiments with diverse sampling intervals, trendHMM demonstrated superior performance. Notably, trendHMM achieved high accuracy without the need for data preprocessing, whereas HMM required preprocessing as a crucial step. Therefore, it can be concluded that trendHMM can reduce the preprocessing costs associated with traditional HMM approaches.

The trendHMM is a method that purely relies on statistical modeling

without considering external factors such as speed, heading, or road preferences. Therefore, it is expected to be more suitable for addressing the map inference problem of identifying unmapped roads. Future research is planned to further explore and address this aspect. In the trendHMM, a fixed value of W was used for extracting movement trends. If W is too small, it may not properly capture the movement trend. On the other hand, if it is too large, it may incorporate irrelevant and extensive movement trend unrelated to the current motion. Therefore, a method for dynamically determining an appropriate W value is needed. In future research, we plan to explore methods for self-adaptive window size determination using factors such as the angle and speed of movement data.

6. Data Availability

Code and data used in the experiment can be found at the following github repository. `https://github.com/Jaeho-99/map_matching`.

7. Author Contribution

- Ha Yoon Song, Ph.D.: Conceived and designed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.
- Jae Ho Lee: Performed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.

8. Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] P. Newson, 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, 2009, pp. 336–343.

[2] X. Fu, J. Zhang, Y. Zhang, An online map matching algorithm based on second-order hidden markov model, Journal of Advanced Transportation 2021 (2021).

[3] V. Salnikov, M. T. Schaub, R. Lambiotte, Using higher-order markov models to reveal flow-based communities in networks, Scientific reports 6 (2016) 1–13.

[4] P. Chao, Y. Xu, W. Hua, X. Zhou, A survey on map-matching algorithms, in: Australasian Database Conference, Springer, 2020, pp. 121–133.

[5] L. Jiang, C. Chen, C. Chen, H. Huang, B. Guo, From driving tra-

jectories to driving paths: a survey on map-matching algorithms, CCF Transactions on Pervasive Computing and Interaction 4 (2022) 252–267.

[6] M. A. Quddus, W. Y. Ochieng, R. B. Noland, Current map-matching algorithms for transport applications: State-of-the art and future research directions, Transportation research part c: Emerging technologies 15 (2007) 312–328.

[7] H. Wei, Y. Wang, G. Forman, Y. Zhu, Map matching by fréchet distance and global weight optimization, Technical Paper, Departement of Computer Science and Engineering 19 (2013).

[8] L. Zhu, J. R. Holden, J. D. Gonder, Trajectory segmentation map-matching approach for large-scale, high-resolution gps data, Transportation Research Record 2645 (2017) 67–75.

[9] K. P. Sharma, R. C. Pooniaa, S. Sunda, Map matching algorithm: curve simplification for frechet distance computing and precise navigation on road network using rtklib, Cluster Computing 22 (2019) 13351–13359.

[10] G. Cui, W. Bian, X. Wang, Hidden markov map matching based on trajectory segmentation with heading homogeneity, GeoInformatica 25 (2021) 179–206.

[11] X. Wang, W. Ni, An improved particle filter and its application to an ins/gps integrated navigation system in a serious noisy scenario, Measurement Science and Technology 27 (2016) 095005.

[12] P. Bonnifait, J. Laneurit, C. Fouque, G. Dherbomez, Multi-hypothesis map-matching using particle filtering, in: 16th World Congress for ITS Systems and Services, 2009, pp. 1–8.

[13] S. Taguchi, S. Koide, T. Yoshimura, Online map matching with route prediction, IEEE Transactions on Intelligent Transportation Systems 20 (2018) 338–347.

[14] M. Quddus, S. Washington, Shortest path and vehicle trajectory aided map-matching for low frequency gps data, Transportation Research Part C: Emerging Technologies 55 (2015) 328–339.

[15] M. Sharath, N. R. Velaga, M. A. Quddus, A dynamic two-dimensional (d2d) weight-based map-matching algorithm, Transportation Research Part C: Emerging Technologies 98 (2019) 409–432.

[16] T. Hunter, P. Abbeel, A. Bayen, The path inference filter: model-based low-latency map matching of probe vehicle data, IEEE Transactions on Intelligent Transportation Systems 15 (2013) 507–529.

[17] G. Hu, J. Shao, F. Liu, Y. Wang, H. T. Shen, If-matching: Towards accurate map-matching with information fusion, IEEE Transactions on Knowledge and Data Engineering 29 (2016) 114–127.

[18] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, Y. Huang, Map-matching for low-sampling-rate gps trajectories, in: Proceedings of the 17th ACM

SIGSPATIAL international conference on advances in geographic information systems, 2009, pp. 352–361.

[19] C. Y. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, 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, IEEE, 2012, pp. 776–781.

[20] R. Song, W. Lu, W. Sun, Y. Huang, C. Chen, Quick map matching using multi-core cpus, in: Proceedings of the 20th International Conference on Advances in Geographic Information Systems, 2012, pp. 605–608.

[21] M. M. Atia, A. R. Hilal, C. Stellings, E. Hartwell, J. Toonstra, W. B. Miners, O. A. Basir, A low-cost lane-determination system using gnss/imu fusion and hmm-based multistage map matching, IEEE Transactions on Intelligent Transportation Systems 18 (2017) 3027–3037.

[22] R. Mohamed, H. Aly, M. Youssef, Accurate real-time map matching for challenging environments, IEEE Transactions on Intelligent Transportation Systems 18 (2016) 847–857.

[23] X. Fu, J. Zhang, Y. Zhang, An online map matching algorithm based on second-order hidden markov model, Journal of Advanced Transportation 2021 (2021).

[24] T. Osogami, R. Raymond, Map matching with inverse reinforcement learning., in: IJCAI, 2013, pp. 2547–2553.

[25] G. R. Jagadeesh, T. Srikanthan, Online map-matching of noisy and sparse location data with hidden markov and route choice models, IEEE Transactions on Intelligent Transportation Systems 18 (2017) 2423–2434.

[26] Y. Wang, X. Liu, H. Wei, G. Forman, C. Chen, Y. Zhu, Crowdatlas: Self-updating maps for cloud and personal use, in: Proceeding of the 11th annual international conference on Mobile systems, applications, and services, 2013, pp. 27–40.

[27] H. Hu, S. Qian, J. Ouyang, J. Cao, H. Han, J. Wang, Y. Chen, Amm: An adaptive online map matching algorithm, IEEE Transactions on Intelligent Transportation Systems (2023).

[28] M. Elkholy, M. Elsheikh, N. El-Sheimy, Radar/ins integration and map matching for land vehicle navigation in urban environments, Sensors 23 (2023) 5119.

[29] Official U.S. government information about the Global Positioning System (GPS) and related topics, `https://www.gps.gov/systems/gps/performance/accuracy/`, 2022. Online, accessed 17-06-2023.

[30] G. D. Forney, The viterbi algorithm, Proceedings of the IEEE 61 (1973) 268–278.

[31] M. Kubička, A. Cela, P. Moulin, H. Mounier, S.-I. Niculescu, Dataset

for testing and training of map-matching algorithms, in: 2015 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2015, pp. 1088–1093.

[32] A. Saalfeld, Topologically consistent line simplification with the douglas-peucker algorithm, Cartography and Geographic Information Science 26 (1999) 7–18.