Eddy: An Error-bounded Delay-bounded Real-time Map Matching Algorithm using HMM and Online Viterbi Decoder

Guanfeng Wang and Roger Zimmermann
School of Computing, National University of Singapore
Singapore 117417
{wanggf,rogerz}@comp.nus.edu.sg

ABSTRACT

Real-time map matching is a fundamental but challenging problem with various applications in Geographic Information Systems (GIS), Intelligent Transportation Systems (ITS) and beyond. It aims to align a sequence of measured latitude/longitude positions with the road network on a digital map in real-time. There exist a number of statistical matching approaches that unfortunately either process trajectory data offline or provide an online solution without an infimum analysis. Here we propose a novel statistics-based online map matching algorithm called *Eddy* with a solid error- and delay-bound analysis. More specifically, Eddy employs a Hidden Markov Model (HMM) to represent the spatio-temporal data as state chains, which elucidates the road network's topology, observation noises and their underlying relations. After modeling, we shape the decoding phase as a ski-rental problem, and an improved online-version Viterbi decoding algorithm is proposed to find the most likely sequence of hidden states (road routes) in real-time. We reduce the candidate routes search range during the decoding for efficiency reasons. Moreover, our deterministic decoder trades off latency for expected accuracy dynamically, without having to choose a fixed window size beforehand. We also provide the competitive analysis and the proof that our online algorithm is error-bounded (with a competitive ratio of 2) and latency-bounded. Our experimental results show that the proposed algorithm outperforms widely used existing approaches on both accuracy and latency.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Spatial databases and GIS*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Pattern matching*

Keywords

GIS; trajectory data; map matching; Hidden Markov Model; online Viterbi decoding; competitive analysis; real-time system

SIGSPATIAL '14, November 4–7, 2014, Dallas/Fort Worth, TX, USA.
ACM 978-1-4503-3131-9/14/11.
http://dx.doi.org/10.1145/2666310.2666383.

1. INTRODUCTION

Recently, an increasing number of real-time positioning data are collected – a trend that is driven by the ubiquitous availability of localization sensors attached to moving objects such as vehicles and pedestrians. This type of data is continuously acquired and effectively utilized by a broad range of applications. Locations not only provide a pair of longitude/latitude coordinates, but also indicate the spatial context of the moving objects or mobile devices if a surrounding geographic information database is available. Systems including Geographic Information System (GIS), Intelligent Transportation System (ITS) and Location-based Services (LBS) have widely employed such context to customize profile settings, optimize complex operations, etc. To better interpret these useful contexts, a map matching algorithm that integrates the positioning data (from GPS or other sensors) with the spatial road network data plays a fundamental role [20].

The input of a typical map matching algorithm is a temporal sequence of location points, *i.e.*, a trajectory. In practice, most raw location information provided from sensors is not highly accurate or not easily interpretable due to two main reasons: (a) the inherent errors and noise generated by the localization sensors, and (b) the sampling methods employed by the embedded systems [16]. The accuracy issue of various sensors, such as GPS, WiFi and cellular signal measurements (e.g., GSM), has been extensively studied. Generally, GPS offers good accuracy to the level of around 10 meters and is available world-wide. Other techniques are feasible in urban environments, but their accuracy deteriorates in rural areas [7]. Although the standard deviation of GPS location inaccuracy can be quite low, serious deviations are observed due to varying surrounding environmental conditions, *e.g.*, tree covers, high buildings, and other problems [6]. In addition, the use of low-cost, consumer-grade sensors in current mobile devices or vehicles is another inevitable reason for the accuracy degradation. Therefore, a map matching algorithm is desirable to help improve the positioning accuracy if the respective digital map is reliable, and to associate the coordinates with the surrounding spatial entities seamlessly.

The map matching problem is illustrated in Figure 1. The red dots such as r_1, r_2 and r_3 are the measured raw location coordinates. The task of map matching is to find the true roads that the moving object is on. As illustrated, it could be a challenging problem since either the green-dots trajectory (p'_1, p'_2 and p'_3) or the blue-dots trajectory ($\hat{p}_1, \hat{p}_2$ and $\hat{p}_3$) can be the actual driving path, and it is impossible to tell by only analyzing separate samples. A number of statistics-

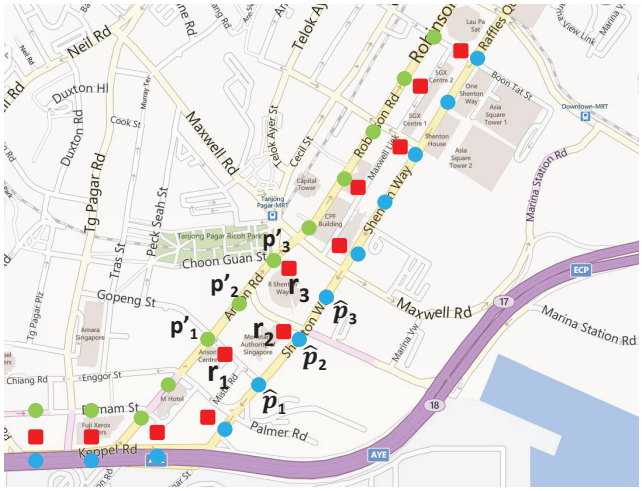

Figure 1: Illustration of the map matching problem.

based map matching algorithms were proposed and developed in recent years. They employed statistical models to solve various map matching problems and distinctly showed the ability to cope with noisy GPS measurements effectively. Particularly, algorithms based on a Hidden Markov Model (HMM) and its variants have been adopted due to their capabilities of concurrently evaluating multiple hypotheses during the mapping procedure [16, 27, 18, 8]. They have also been proven to be tolerant against highly noisy observations, *e.g.*, the location fingerprints from GSM towers [23], and the accuracy degradation owing to the increase of a trajectory's temporal sparseness [18, 16].

In the context of Markov information sources and hidden Markov models, the Viterbi algorithm, a dynamic programming algorithm, is widely used for decoding such models. This algorithm finds the most likely sequence of hidden states for the given observation sequence [25]. It computes a forward pass over the input sequence to compute probabilities, followed by a reverse pass to compute the optimal state sequence. Therefore, all the data must be obtained before any of the hidden states can be inferred. The result of the underlying state chain is called a *Viterbi path*. However, when applied to a real-time or an interactive system, one noticeable disadvantage of the Viterbi algorithm is that the optimal state sequence cannot be computed until the entire input has been observed.

For latency-sensitive applications such as route navigation and traffic incident detection, it is unacceptable to receive map matching results, *e.g.*, on which road arc the truck is driving, after the whole itinerary is finished. In HMM-based map matching, the key input and output of a traditional Viterbi decoder are the location observations (*e.g.*, GPS measurements) and the most likely road trajectory of a moving object. Conceptually, the input observation stream could be extremely long, or even infinite, which leads to a significantly longer latency than a timely response that systems may require. Therefore, the traditional Viterbi decoder is not suited for real-time applications where there are strong latency constraints.

Meanwhile, accuracy is also another crucial factor for most location-based applications. To shorten the mapping delay, a system has the freedom to match raw location measurements greedily, mapping each sample immediately as an extreme case, without waiting for enough future observations. However, it is undesirable to give up the accuracy increase gained by map matching techniques or even worse, pick an incorrect road path as output. The risk of selecting a false road may cause serious issues in real system such as incident detection. Any inaccurate output also raises the expected monetary cost in some enterprise services, *e.g.*, logistics truck monitoring, fleet scheduling and others. Thus, an intelligent algorithm which understands and wisely practises the balance between accuracy and latency is desirable.

Here we propose Eddy, a novel real-time HMM-based map matching system by using our advanced online decoding algorithm. We take the accuracy-latency tradeoff into design consideration. Our algorithm chooses a dynamic window to wait for enough future input samples before outputting the matching result. The dynamic window is selected automatically based on the current location sample's states probability distribution and at the same time, the matching output is generated with sufficient confidence. Our contributions in this work include:

- An improved real-time HMM-based map matching system is presented which is novel with respect to its tradeoff analysis and dynamic window selection algorithm during the decoding phase.
- A competitive analysis and proof illustrating that our online decoding algorithm is error-bounded (with competitive ratio of 2) and latency-bounded.
- Accuracy evaluation and latency comparison between our map matching system results and existing online decoding algorithm outputs.

The rest of this paper is organized as follows. Section 2 provides a survey of related work. The preliminaries and formal definitions related to HMM-based map matching are presented in Section 3. Section 4 contains a brief summary of existing online Viterbi decoders. The detailed description of the proposed online decoding algorithm and its competitive analysis are also reported in this section. Section 5 presents the experimental evaluation and provides illustrations of map matching accuracy and latency improvement. Finally, Section 6 concludes the study and discusses open issues.

2. RELATED WORK

A number of map matching algorithms have been proposed by researchers using different techniques such as geometric analysis, topological analysis, probabilistic theory and so forth. The geometry-based map matching algorithms utilize the shape of the spatial road network without considering its connectivity [2, 26], so that the map matching result is greatly affected by measurement errors. The topology-based map matching algorithms make use of the geometry as well as the connectivity and contiguity of the road arcs in the road network [10, 21, 5]. They leverage the topological information to reduce the candidate matches for each location sample, and develop a weighting system to measure the similarities between the geometry of a portion of the trajectory and candidate road arcs to find the most likely road arcs. However, this category of algorithms is very sensitive to an increase of the sampling interval. A graph-based map matching algorithm considers the entire trajectory as a pure

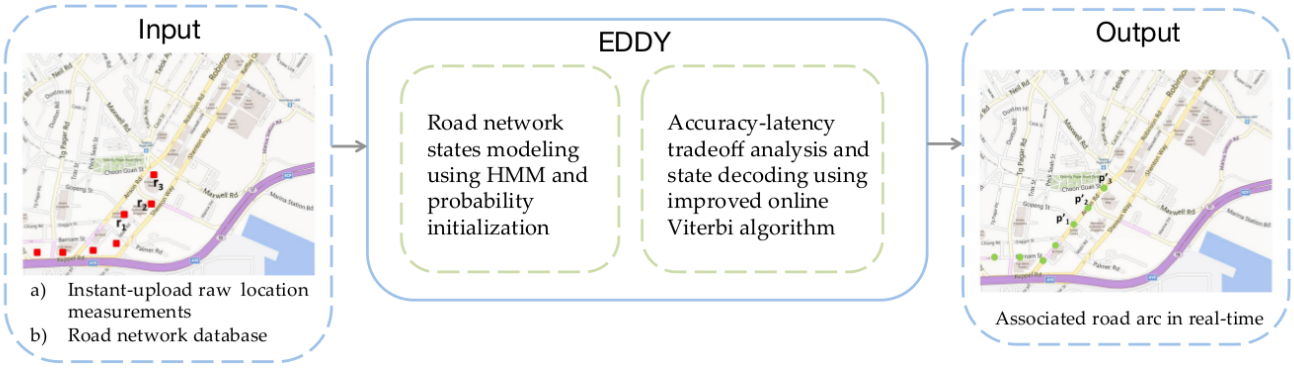

Figure 2: System overview of Eddy.

graphical curve and looks for a path in the road network that is as close as possible to the curve [1, 5]. However, it is usually a global matching procedure and has difficulty to generate arcs in real-time. A comprehensive review of 35 map matching algorithms for navigation applications since 1989 is presented by Quddus *et al.* [20].

Statistics-based map matching algorithms take advantage of statistical models, such as Kalman Filter [12, 19], particle filters [14], HMM [3, 18, 24], *etc.*, to solve various map matching problems. These algorithms are able to cope with noisy location measurements effectively. However, only a few studies have focused on the real-time decoding issue of the HMM model. Goh *et al.* proposed a variable sliding window scheme to provide an online solution while the delay bound of road arc generation is not guaranteed [9]. Additionally, the tradeoff relation between the accuracy and latency from online decoding strategies has not been extensively studied yet.

3. HMM-BASED MAP MATCHING

We first give the preliminaries and the formal definitions of the map matching problem using HMM.

Definition 1 *(Road Network)* : A road network $G(V, E)$ represents a finite street system which consists of a set of one-way or two-way road curves, called *road arcs*, in 2D Euclidean space. Each road arc e_i ($e_i \in E$) is assumed to be piecewise linear and can be characterized by a finite sequence of points $A^i = (a_1^i, a_2^i, ..., a_m^i)$. The end points here a_1^i and a_m^i are nodes and belong to the vertex set V. Other points in the middle are referred to as shape points and each e_i has some properties such as speed constraints.

Definition 2 *(Location Trajectory)* : A location trajectory $L = \{l_1, l_2, \ldots, l_n\}$ is a sequence of measurements from localization sensors (such as GPS) according to the time sequence $T = \{t_1, t_2, \ldots, t_n\}$. Each position measurement l_i consists of a coordinate, *i.e.*, longitude x_i and latitude y_i. We further denote the ground truth of the position sequence data as $G_l = \{g_1, g_2, \ldots, g_n\}$ and their belonging road arcs $G_e = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$, $G_e \in E$.

Definition 3 *(Match Point)* : The match point m_i^j of a location measurement sample point l_i on a road arc e_j is the point that $m_i^j = argmin_{\forall m_k^j \in A^j} dist(m_k^j, l_i)$, where $dist(m_k^j, l_i)$ returns the great circle distance between l_i and any point on A^j, including end points and shape points.

Problem Statement : Given the road network $G(V, E)$, and the trajectory information L and T, find the most likely path $P = \{p_1, p_2, \ldots, p_n\}$, where $a_m^{i-1} = a_1^i$ and $P \subset E$, which is a subset of connected road arcs from G, along with each p_i's mapping output time $T' = \{t_1', t_2', \ldots, t_n'\}$.

Figure 2 illustrates the system overview of Eddy. It takes the location measurements and road network databases as input. The positioning data should be instantly uploaded since we focus on the latency-sensitive applications and services in this study. Eddy results in a real-time streaming of road arcs with guaranteed accuracy and latency level.

Our system consists of two parts. We first present the road arc traveling problem as a hidden states transition model. Based on the framework of HMM, the random variable e_t and l_t are a hidden state and an observation at time t, respectively (see the state transition flow in Figure 3). In the context of the map matching problem, we model every road arc e_i as a hidden state and each location measurement l_t as an observation emitted by the hidden state. Two types of arrows in this figure (horizontal and vertical arrows) indicate two important parameters in the model. The horizontal arrow represents the *transition probability* between two consecutive hidden states. It quantifies the likeliness that a vehicle is moving from road e_{t-1} to road e_t. Each vertical arrow represents the *emission probability* between the hidden state and the observation. It represents how likely the measurement l_t can be observed if the vehicle is driving on a certain road arc.

The second part is the online Viterbi decoding algorithm (see the trellis in Figure 3) which is improved based on our quantitative accuracy-latency tradeoff analysis. During the decoding phase, candidate arc paths are sequentially generated and evaluated on the basis of their likelihoods. Our goal is to find the maximum likelihood path over the Markov chain that has the highest joint emission/transmission probabilities and still holds the latency bound.

Formally, the map matching problem is modeled by transition, emission and initial probability :

$$\lambda = (\mathcal{T}, \mathcal{M}, \pi)$$

The state set is E and the observation set is L. In our model, the initial probability π_i of being in state e_i is defined as the emission probability at this state. The emission probability $\mathcal{M}_i(l_t)$ of observation l_t from state e_i is obtained by modeling the positioning measurement noise as a Gaussian

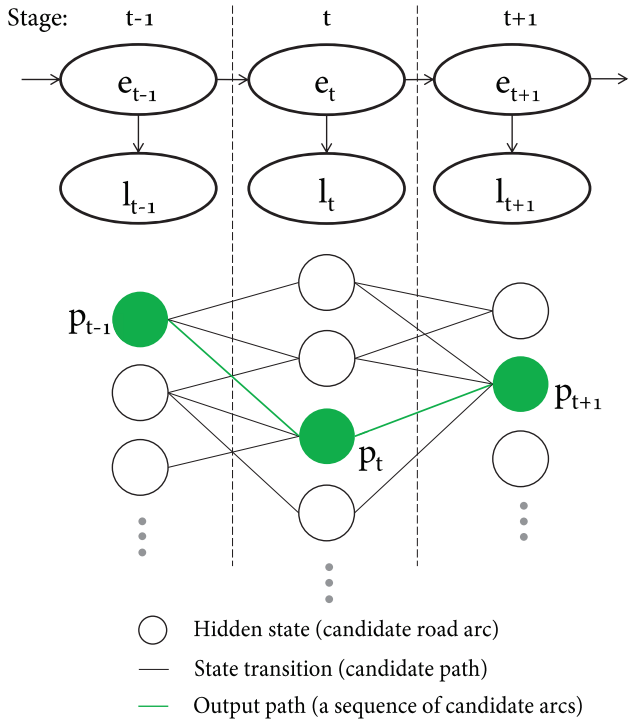

Figure 3: Illustration of state transition flow and Viterbi decoding algorithm.

distribution [17]:

$$\mathcal{M}_i(l_t) = \mathbb{P}(l_t|p_t = e_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{dist(e_i,l_t)^2}{2\sigma^2}}$$

where σ is the standard deviation of the positioning measurements. For example, when the input location observations are a sequence of GPS collected points, we use a standard deviation of 10 meters to estimate the noise distribution [13]. $dist(e_i, l_t)$ represents the shortest distance from l_t to the candidate road arc e_i, which is the great circle distance on the surface of the earth between l_t and its corresponding match point m_t^i.

We also utilize the distance differences between the observation pairs and match point pairs to estimate the transition probabilities based on the study from Newson and Krumm [18]. Given two measurements l_{t-1}, l_t and their match points m_{t-1}^i, m_t^j, the transition probability of moving from e_i to e_j is:

$$\mathcal{T}_t^{ij} = \mathbb{P}(p_t = e_j|p_{t-1} = e_i) = \beta e^{-\beta\|d_l - d_m\|}$$

where d_l is the great circle distance between two location measurements and d_m is the shortest route distance from m_{t-1}^i to m_t^j.

Within a dynamic window size, this model is later decoded by our improved online algorithm and outputs $p_t = \{e_k, e_{k+1}, \dots, e_i\}$, where $\{e_k, e_{k+1}, \dots\}$ is the route path between e_{i-1} and e_i determined by the selected state transition path. This subset of candidate road arcs are generated as the most likely path for given observation l_t. It guarantees that the output paths are connected. In the following descriptions, we omit the $\{e_k, e_{k+1}, \dots\}$ part in equations while we actually keep track of these connecting paths in the real system.

4. IMPROVED ONLINE DECODING

The aim of decoding is to discover the hidden state sequence that is most likely to have produced a given observation sequence. In the context of map matching, our algorithm needs to find the road arc sequence that is most likely to generate the collected location measurements. The traditional Viterbi decoder is a trellis algorithm (see Figure 3) defined as:

$$\delta_t(i) = \max_{p_1 p_2 \cdots p_{t-1}} \mathbb{P}\{p_1, p_2, \dots, p_{t-1}, p_t = e_i, l_1, l_2, \dots, l_t - 1|\lambda\}$$

which gives the highest probability that partial observation sequence and state sequence up to time step t can have, when the current state is i. The initialization and recursion step of the decoding phase are defined as:

$$\delta_1(i) = \pi_i \mathcal{M}_i(l_1)$$

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i)\mathcal{T}_t^{ij}]\mathcal{M}_j(l_t)$$

where N is the cardinality of candidate state set S, $S \subset E$. Usually the scale of the road network in modeling, $card(E)$, is relatively large, which leads to inefficiency in decoding. Eddy narrows down the set of candidate states within S to accelerate the processing. We will elaborate on the details of downsizing later.

In each time step, we normalize the probability distribution to ensure $\sum_{j=1}^{N} \delta_t(j) = 1$. The backtracking pointer of the selected hidden state in each step is as follows:

$$\psi_t(j) = \arg\max_{1 \le i \le N} [\delta_{t-1}(i)\mathcal{T}_t^{ij}]$$

It terminates when the last observation is received and decoded by this procedure. The optimal path can be obtained by backtracking from the last matching result:

$$p_T = \arg\max_{1 \le i \le N} [\delta_T(i)]$$

$$p_t = \psi_{t+1}(p_{t+1})$$

However, this traditional type of decoder is not suited for real-time systems since the optimal state sequence cannot be computed until the entire input has been observed. Thus, some HMM-based frameworks have proposed several localizing strategies to fulfill the online output functionality. We first briefly summarize two widely used online decoding techniques and their limitations.

Fixed Segment/Sliding-Window

One simple and straightforward approach is to divide the trajectory into fixed-sized sequences and handle them independently. Given a desired latency D_d, the system simply fixes the segment size or window size as $\omega \le D_d$ and applies the Viterbi decoder to each segment/window to bound the maximum system delay.

For algorithms using a fixed segment (FS), the decoder waits for a segment-length of observations before decoding on the time slice from t to $t + \omega - 1$. In the sliding window method (FSW), the decoder considers only one new observation and moves the window forward one step a time. In the example illustrated in Figure 4, given $\omega = 2$, FS first

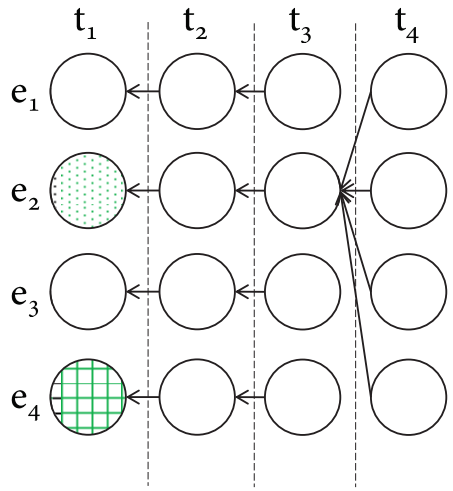

Figure 4: One example of the online Viterbi decoding process.

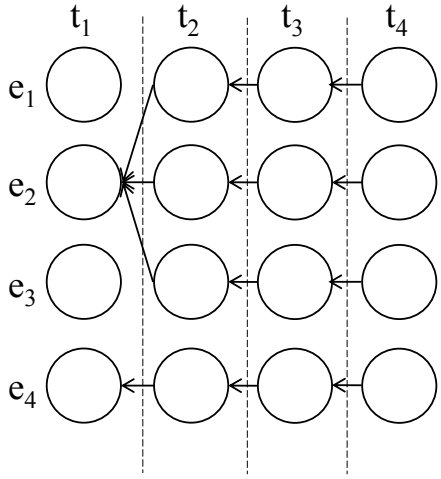

Figure 5: Illustration of the state probability recalculation after future location observations are received.

reads measurements from t_1, t_2 in order, and generates the output path, say $P = \{e_4, e_4\}$. The next observation input of FS is from t_3. Thus, the matching output delays for the observations within the same segment are different. FS outputs the result for l_1 with 1 time step delay (to wait l_2 to fill the segment size) while the matching delay of l_2 is 0 (we do not add the client-server transmission time and matching processing time into the delay calculation since we focus on the decoding delay in this study). Differently, given $\omega = 2$, FSW takes l_3 as input right after generate the matching result of l_1, by sliding the window from $[t_1, t_2]$ to $[t_2, t_3]$. The matching delay for all observations from this method is constant (except for the last ω location measurements since there is no future room to slide forward).

Usually, a larger window size leads to a more accurate matching result but a longer output delay, and vice versa [16, 27, 22]. In the previous example, given $\omega = 3$, the FS/FSW decoder can generate an arbitrary path for segment/window $[t_1, t_3]$. One plausible path could be $P' = \{e_4, e_4, e_4\}$, illustrated as a lattice-pattern circle sequence. However, when the decoder receives l_4 , it would have enough knowledge to recognize that P' is not a possible output. Although this example is only an undesirable case and may not be triggered frequently in real scenarios, it illustrates the tradeoff we need to carefully deal with between the accuracy and latency. This accuracy degradation occurring from the sub-optimal path generation may arise due to certain pre-defined segment- or window-size settings. Therefore, a decoding algorithm which can intelligently choose a dynamic window size is preferred.

Convergence State Discovery

Some HMM-based applications adopted another technique named Convergence State Discovery (CSD, also called fusion point finding), which is capable of finding the optimal path before the entire trajectory is received [4, 9]. The basic idea in this algorithm is to delay the label generation until encountering a converging state like e_2 at t_3 in Figure 4. When CSD reads the input observation l_4 and calculates related probabilities, it sees that all backtracking pointers point to the same state, e_2. It is easy to prove that all future surviving paths will contain the same sub-path before this convergence state. Thus at time t_4, CSD can output the matching result for observations l_1, l_2 and l_3, $P = \{e_2, e_2, e_2\}$ (the dot-filled circle chain).

This algorithm has the advantage of holding the promise that the generated output path is identical to the result from the original Viterbi decoder. However, a serious issue that this algorithm may encounter is the absence of a fusion point in some real problems or some pre-defined probability normalization rules. The matching delay is prolonged if the convergence state comes late, and may persist to the end of the observation sequence if no such point exists. In other words, CSD is not delay-bounded and in the worst case degenerates to the original Viterbi algorithm. Therefore, for most latency-sensitive applications, this decoding algorithm is unfit.

4.1 Improved Online Decoder based on Ski-rental Model

To better interpret the tradeoff between the map matching accuracy and latency, we model the online decoding phase as a ski-rental problem in this study. The ski-rental model, also known as "rent or buy" dilemma, is one of the fundamental problems in online algorithms. This problem was first abstracted by Karlin *et al.* and used in a communication minimization algorithm [11]. In a classic ski rental problem, a skier may rent skis for R per day or buy them for B dollars. At the end of any day, the skier may break his legs along with the skis, or in some other way irrevocably finish skiing. The goal is to develop an online strategy minimizing the cost spent on skiing, where the cost is compared to the cost of an optimal offline strategy for the same input. The worst-case ratio between these two amounts is called *competitive ratio*.

Inspired by one of its variants, "Multislope Ski Rental" [15], we use a generalized model with a inconstant buying price B_t that changes over time in our case. Obviously, the total cost of skiing is

$$\mathcal{C}_s = B_{\hat{t}} + R \times \hat{t} \tag{1}$$

where the skier decides to buy the skis in the evening of the $\hat{t}^{th}$ day.

Similarly, in our scenario, we model the accuracy penalty and latency penalty as the buying price and rental rate, respectively. We need to decide whether to stay in the current decoding state and pay a certain amount of latency cost per time unit, or output the present matching result and pay some large accuracy penalties but with no further delay penalty. Without loss of generality, we assume the location observation l_0 measured at t_0 has been matched to the road network and l_1 from t_1 is under the decoding phase currently. The future information up to $\hat{t}$ is observed and transferred to the decoding system to help the joint probability computation. Moreover, the decoder decides to output the matched result p_1 at time $\hat{t}$. Straightforwardly, the delay of decoding l_1 is $\hat{t} - t_1$, which is similar to the rental rate that a skier has to pay before a buying decision. Meanwhile, to better estimate the accuracy of the matching roads, we leverage the probability distribution $\delta_{t_1,\hat{t}}(j)$ which indicates the likelihood of each state e_j being the matching road. Notably, this is different from $\delta_{t_1}(j)$ since the system involves future information into the inference chain. We first calculate $\delta_{t_1}(j)$ considering that the matching result p_0 for the observation l_0 has been generated already,

$$\delta_{t_1}(j) = \max_{1 \le i \le N} [\delta_{t_0}(i) \mathcal{T}_{t_1}^{ij}] \mathcal{M}_j(l_1)$$

$$\delta_{t_0}(i) = \begin{cases} 1, & \text{if } p_0 \in e_i. \\ 0, & \text{otherwise.} \end{cases}$$

where the distribution of δ_{t_0} is determined. With all the future observations we waited and received from t_1 to $\hat{t}$, we afterwards obtain,

$$\delta_{t_1,\hat{t}}(j) = \sum_{i=1}^{N} \delta_{\hat{t}}(i)$$

$$\text{if} \quad \psi_{t_1,\hat{t}}(i) = j$$

where $\psi_{t_1,\hat{t}}(i)$ is the backtracking function from time frame $\hat{t}$ to t_1

$$\psi_{t_1,\hat{t}}(i) = \psi_{t_1}(\psi_{t_2}(...\psi_{\hat{t}-2}(\psi_{\hat{t}-1}(i))))$$

Thus, $\delta_{t_1,\hat{t}}(j)$ is the sum of $\delta_{\hat{t}}(i)$ where e_j at time step t_1 and e_i at time step $\hat{t}$ are on the same candidate path connected by standard Viterbi backtracking pointers. As illustrated in Figure 5, the probability that e_1 is the output matching result is the sum of $\delta_{t_4}(e_1)$, $\delta_{t_4}(e_2)$ and $\delta_{t_4}(e_3)$ when computing at time t_4. For each $e_j \in S$, $\delta_{t_1,\hat{t}}(j)$ presents the probability that l_1 should be matched to e_j after future observations up to $\hat{t}$ are considered into the HMM framework.

Intuitively, if only one state is calculated with a significantly high probability and the other states' likelihoods are near zero, we can deduce confidently that this state is the matching road and generate this road arc as the output label. To better describe the distribution characteristics and incorporate this into our decoding procedure, we use the information entropy of $\delta_{t_1,\hat{t}}(j)$ as a proxy of the accuracy penalty.

$$\mathcal{H}(t_1, \hat{t}) = -\sum_{j=1}^{N} \delta_{t_1,\hat{t}}(j) \; log \; \delta_{t_1,\hat{t}}(j)$$

The entropy $\mathcal{H}(t_1, \hat{t})$ is a logarithmic measure of the number of states with significant probability of being occupied, which indicates the degree of *uncertainty* at time step t_1 after receiving future observations up to $\hat{t}$. According to the definition of entropy function, the larger the value $\mathcal{H}$ is, the higher the uncertainty of this outcome state could be. The highest entropy outcome is achieved when $\delta_{t_1,\hat{t}}(j)$ is evenly distributed among all candidate states. On the other hand, if $\mathcal{H}$ is close enough to zero, it means that one state is extremely outstanding within the candidate space. This plays the same role as the buying price B_t in the ski-rental model. Therefore, in accordance with Equation (1), we derive our objective cost function as the sum of the accuracy and delay penalties,

$$\mathcal{C}(t_1, \hat{t}) = \mathcal{H}(t_1, \hat{t}) + \gamma(\hat{t} - t_1)$$

where γ is the parameter to control the tradeoff between accuracy gain and delay cost. If the real-time system is extremely sensitive to the latency, a larger value of γ should be chosen. By contrast, if the monetary cost of false road matching is expensive, a small γ should be considered to penalize more on the accuracy part.

Similar to the ski-rental model, whose ultimate target is to determine the buying date, here we need to provide a strategy to decide at which $\hat{t}$ we should stop delaying and output the matching result $\arg\max_j[\delta_{t_1,\hat{t}}(j)]$. Thus, our online system needs to choose an appropriate label generation time $\hat{t}$ to minimize the cost $\mathcal{C}$.

Clearly, the delay cost accumulates linearly like a monotonically increasing function. If the accuracy penalty $\mathcal{H}(t_1, \hat{t})$ changes arbitrarily over time, its sum $\mathcal{C}$ is difficult to be minimized or even analysed. Thus, here we assume that given t_1, $\mathcal{H}$ is a monotonically decreasing function of variable $\hat{t}$. The physical meaning of this assumption is that we believe the uncertainty of the state outcome at a certain time step would decrease as a growing number of future observations are analysed within the decoding procedure. We will show in experimental section that our assumption is reasonable across the entire test dataset.

Thereby, we need to minimize the sum of a decreasing function and an increasing function. In the ski-rental model, the *break-even* algorithm is known as the best deterministic algorithm for this set of problems [11]. We adopt a similar idea and choose the time point $\hat{t}$ when $\mathcal{H}(t_1, \hat{t})$ is equal or less than the value of $\gamma(\hat{t} - t_1)$, to output the matching road result. The intuition behind this algorithm is to adaptively adjust the window size based on the uncertainty of the state matching. If the uncertainty degree is high, the algorithm should extend the window size to absorb more future location observations before generating the road arc label. Conversely, if the initial $\mathcal{H}$ value is low enough or the function $\mathcal{H}$ drops rapidly, the window should become smaller and the matching output will be generated soon.

The pseudo-code for general cases is detailed in Algorithm 1. The "+" operator on line 10 means to attach a new output to the global sequence. P and T' can be implemented as a *pipe* with capacity of 1, so that once a new output p_i is generated, it can be consumed by an upstream real-time application immediately, and the latency is exactly $t'_i - t_i$.

4.2 Accuracy and Latency Analysis

To better illustrate the advantage of our improved online decoding algorithm, here we present a theoretical competitive and upper-bound analysis for accuracy and latency, respectively. First we provide the competitive ratio of our

Algorithm 1: IMPROVED ONLINE VITERBI DECODING

Input: A location trajectory $L = \{l_1, l_2, \ldots, l_n\}$, and its according time sequence $T = \{t_1, t_2, \ldots, t_n\}$, both of which could be infinite. A set of candidate road arcs (hidden states) $E = \{e_1, e_2, \ldots, e_N\}$.

Output: A sequence of path $P = \{p_1, p_2, \ldots, p_n\}$, where $P \subset E$, and each p_i's mapping output time $T' = \{t'_1, t'_2, \ldots, t'_n\}$.

```
1  P ← {∅}, T' ← {∅}
2  t̂ ← t_1
3  foreach t_i ∈ T do
4      if t_i ≥ t̂ then
5          t̂ ← t̂ + 1
6      while t_i < t̂ do
7          if H(t_i, t̂) ≤ γ(t̂ − t_i) then
8              t'_i ← t̂
9              p_i ← arg max_{1≤j≤N} [δ_{t_i,t'_i}(j)]
10             P ← P + p_i, T' ← T' + t'_i
11             t_k ← t_i + 1
12             while t_k ≤ t̂ do
13                 foreach e_i ∈ E do
14                     update δ_{t_k}(e_i)
15                     update φ_{t_k}(e_i)
16             leave loop
17         else
18             t̂ ← t̂ + 1
19             foreach e_i ∈ E do
20                 update δ_{t_i,t̂}(e_i)
```

decoder, which is the worst-case ratio between the cost of the solution found by our algorithm and the cost introduced by an optimal solution. Assume for a given l_i received at t_i, Eddy generates the according road arc label at time t. Two situations need to be considered when analyzing the worst case — one is that the actual optimal output time step T_o is earlier than t, and the other is $T_o > t$. The cost of the optimal solution is $\mathcal{H}(t_i, T_o) + \gamma(T_o - t_i)$. If $T_o < t$, it indicates that, even with more measurements adopted, the cost decrease from the accuracy penalty $\mathcal{H}$ does not make up for the cost increase caused by the latency penalty. In other words, the concentration expectation of the state distribution based on future observations is not achieved. The worst case in this situation is that $\mathcal{H}(t_i, t_i) = \mathcal{H}(t_i, t) + \epsilon$ where ϵ is a real number approaching zero (it cannot be zero since $\mathcal{H}$ is a monotonically decreasing function), and the optimal output is $T_o = t_i$. The optimal solution outputs the map matching result immediately since the future observations benefit nothing to the decoding process in order to involve no latency penalty to the cost function,

$$\mathcal{C}(t_i, T_o) = \mathcal{C}(t_i, t_i) = \mathcal{H}(t_i, t_i)$$

Since our algorithm generates a road arc result at t, not $t-1$, we have

$$\mathcal{H}(t_i, t) < \gamma(t - t_i)$$

$$\mathcal{H}(t_i, t-1) > \gamma(t - 1 - t_i)$$

Also, $\mathcal{H}(t_i, t) - \mathcal{H}(t_i, t-1) < \epsilon < \gamma$, so we obtain

$$\gamma(t - 1 - t_i) < \mathcal{H}(t_i, t) < \mathcal{H}(t_i, t_i)$$

Thus, the cost of our method is,

$$\begin{aligned}\mathcal{C}(t_i, t) &= \mathcal{H}(t_i, t) + \gamma(t - t_i) \\ &= \mathcal{H}(t_i, t_i) + \gamma(t - 1 - t_i) + \epsilon + \gamma \\ &< \mathcal{C}(t_i, T_o) + \mathcal{C}(t_i, T_o) + \epsilon + \gamma \\ &= 2\mathcal{C}(t_i, T_o) + \epsilon + \gamma\end{aligned}$$

If $T_o > t$, the worst case is that $T_o = t+1$ and $\mathcal{H}(t_i, T_o) = 0$ because this is the lowest value pair for both two penalties and all other cases would achieve a higher $\mathcal{C}(t_i, T_o)$. Thus the cost of the optimal solution is,

$$\begin{aligned}\mathcal{C}(t_i, T_o) &= \mathcal{C}(t_i, t+1) \\ &= \mathcal{H}(t_i, t+1) + \gamma(t + 1 - t_i) \\ &= 0 + \gamma(t - t_i) + \gamma \\ &> \frac{\gamma(t - t_i) + \mathcal{H}(t_i, t)}{2} + \gamma \\ &> \frac{\mathcal{C}(t_i, t)}{2}\end{aligned}$$

Thereby, we proved that the cost of our algorithm $\mathcal{C}(t_i, t)$ is no more than 2 times of the cost introduced by all the other solutions plus a constant, and thus our improved online decoder is a 2-competitive algorithm.

Next, we illustrate that our improved online decoding is latency-bounded. Assume at time t, the algorithm has not generated the road arc output for a given measurement l_i. Since we adopt the break-even condition, we have $\mathcal{H}(t_i, t) > \gamma(t - t_i)$. In addition, $\mathcal{H}$ is a monotonically decreasing function and of course $t > t_i$ because we cannot perform map matching without receiving the measurement. Thus, we have $\mathcal{H}(t_i, t) < \mathcal{H}(t_i, t_i)$. Clearly, by the transitive property of inequalities, we obtain $\gamma(t - t_i) < \mathcal{H}(t_i, t_i)$. Therefore, the upper-bound of map matching delay of l_i is $\mathcal{H}(t_i, t_i)/\gamma + t_i$, which is only determined by the characteristic of distribution δ_{t_i}. The matching process of every incoming observation would terminate for sure even if the entire measurement input is infinite.

4.3 Candidate State Space Reduction

To make the decoding process more efficient, we narrow the range of candidate states $card(S)$ in our HMM model. Due to the fact that the vehicles usually drive at a limited speed during the time interval between two consecutive sample measuring locations, it is very likely that all candidate road arcs of the current location observation fall into a small area around the previous sample point. Therefore, we employ the radial search method proposed by Fang and Zimmermann [8], to find the candidate road arcs of a location measurement point instead of using the traditional range query.

5. EXPERIMENTAL EVALUATION

To evaluate our Eddy system, we implemented the other two online Viterbi decoding strategies, FS and FSW, as comparisons. As previously described, the CSD strategy always generates the optimal solution (identical to the offline decoder's result) but does not guarantee any delay upper-bound, which usually involves a long latency (in the order of minutes) and is not applicable to real-time services [9].

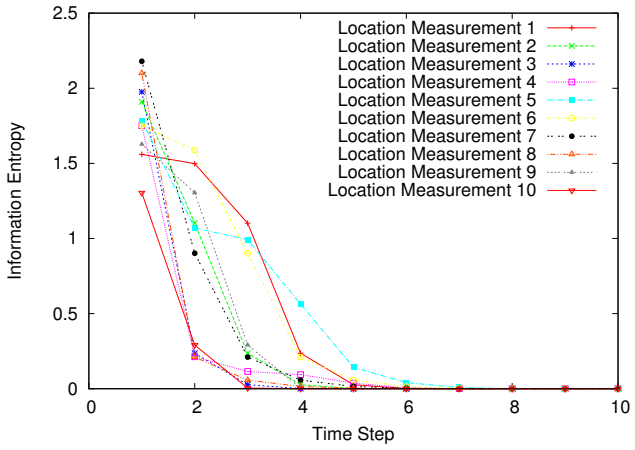

Figure 6: Information entropy trends of 10 example location measurements.

Thus we did not compare our algorithm with CSD in this study.

In our experiments, we adopt the public real-world dataset collected in Seattle provided by Newson and Krumm [18], including the relevant road network, GPS trajectory data, and ground truth. The road network comprises more than 150,000 road arcs. The raw GPS trajectory data is a 50-mile route in Seattle which is sampled at 1 Hz and took about 2 hours to drive, giving 7,531 time-stamped latitude/longitude pairs. The ground truth contains a sequence of road arcs with the directions in which the vehicle actually travelled. Since it is impossible for us to know the exact actual location of the vehicle in the road network corresponding to each GPS sample point, only the path taken by the vehicle is viewed as the ground truth. We also adopt the underlying HMM model parameters, σ and β, which have been tested and verified in their study [18].

We focus on two evaluation aspects, accuracy and latency, in these experiments. First, we compute the actual trends of information entropy $\mathcal{H}$ for all the location measurement points from the dataset. We show that our assumption is reasonable that it is a monotonically decreasing function of variable $\hat{t}$. Afterwards, we apply our method and two baseline algorithms to the whole dataset to compute and visualize the tradeoff between accuracy and latency.

Our improved online decoding algorithm and the other two comparison methods are all implemented in C# and connected with a lightweight in-memory database, SQLite. Since we focus on the road arc label generation delay instead of thye real processing time, this database is completely stored and processed in RAM.

5.1 Accuracy Penalty Trend

We utilize the radial search method to reduce the candidate set of road arcs, and we set the candidate state size parameter $\alpha = 1.8$ in our experiments, which has been empirically tested earlier [8]. This leads to the property that only a small set of candidate states e_i share the matching probability and thereby the distribution concentrates more quickly than in the case where we use the whole road network as the candidate set. We calculate the information entropy, which is considered as the accuracy penalty proxy in our algorithm, for every location measurement in the scope of the whole trip. For each measurement l_i, we record and update its entropy value changes when future observations $l_{i+1}, l_{i+2}, \ldots, l_n$ are received.

Figure 6 illustrates 10 example trends of the location measurement's information entropy as the time elapses (one new observation received at every time step). As shown, the value of entropy function $\mathcal{H}$ is relatively high when only the current measurement is received and no future observation is incorporated into the model. It indicates the difficulty of generating the matching result immediately. As the time step increases, $\mathcal{H}$ turns to be a monotonically decreasing function as we hypothesized.

If the value of $\mathcal{H}$ increases as the time step moves forwards for a given l_i, we judge that this entropy function is not a monotonically decreasing function, and we also record the time step where the entropy value increases as the increasing point. Among the entire trajectory dataset, we find that 91.53% of the measurements' entropy function is monotonically decreasing. Moreover, in the remaining part of this dataset, 5.52% of functions' increasing point appears after receiving more than 400 future observations. It is very likely that the system has already passed the break-even point before seeing such a large number of future observations. In other words, 97.05% of functions are actually decreasing if the delay of a system is limited to less than 400 seconds, which is a reasonable setting in the context of a real-time system. Additionally, if the real-time system only considers future observations within the range of 50 samples, 100% of $\mathcal{H}$ satisfies our assumption. This result intuitively makes sense because of the underlying logic in that the more future observations are incorporated into the decoding model, the more confidently we can determine which road the vehicle is driving on.

5.2 Error and Delay

To illustrate the tradeoff between the matching accuracy and latency, we apply our system and two comparison algorithms to the Seattle trajectory dataset with different γ values and window sizes w. Different sampling periods are considered in our experiments as well to show the robustness of our algorithm under different location measuring rates. We adjust the γ value from 0.01 to 2 to tune the tradeoff between the road arc mismatch rate and delay time. The parameter w varies according to the change of the location measurement sampling intervals. For example, in order to obtain an accuracy change from no delay at all to a latency of 120 seconds, we tune the w value from 0 to 120 for FSW, and from 1 to 241 for FS, with a sampling period of 1 second. The reason is that FS generates labels for all location observations within the current window at once (when the window is full), so that the location measurements in the second half of the window have lower *effective latency* than the measurements in the first half. Clearly, the road arc label of the last location observation tucked into the window will be matched and generated by FS immediately without any latency no matter how large the window size is. Thus, we consider the average effective latency among the observations within the same window, $(w-1)/2*$(sampling period), as the average latency. Similarly, when the sampling period becomes 10 seconds, we evaluate the w value from 0 to 12 for FSW, and from 1 to 25 for FS, respectively, to compute the mismatch percentage trend from no delay to a latency of 120 seconds.

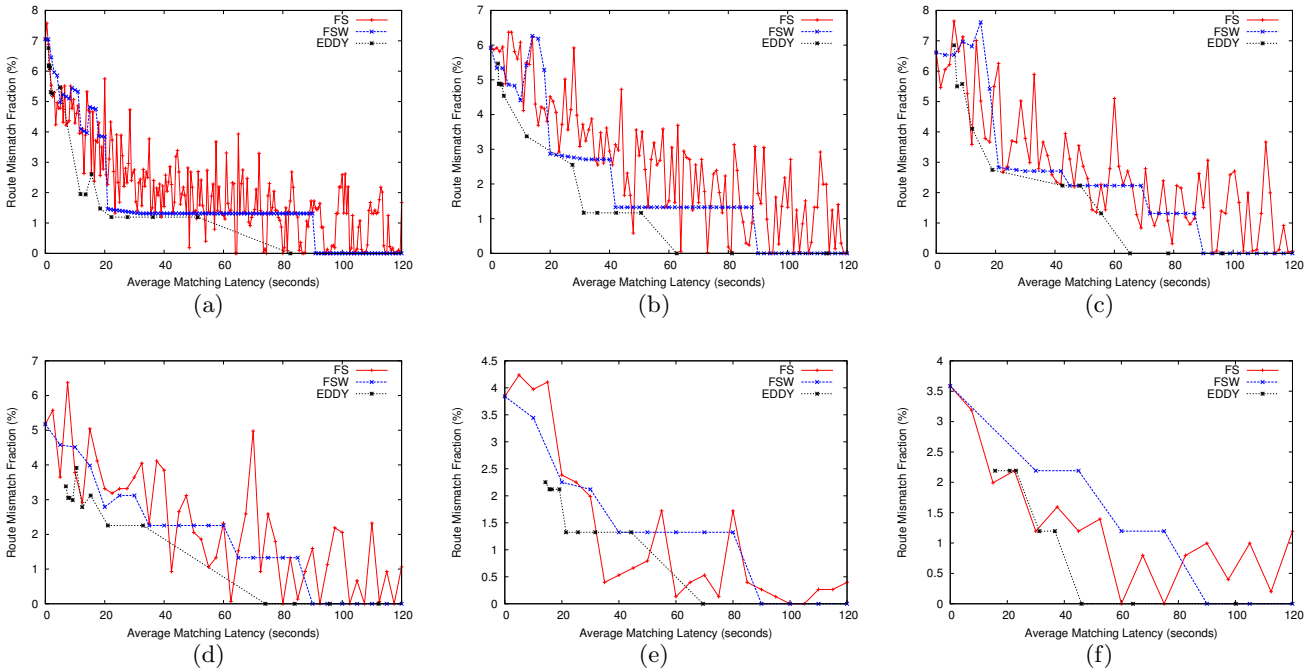

Figure 7: The accuracy (in RMF) and latency (in seconds) of map matching results on different location measurement sampling intervals: 1 observation sample (a) per second, (b) every 2 seconds, (c) every 3 seconds, (d) every 5 seconds, (e) every 10 seconds, (f) every 15 seconds.

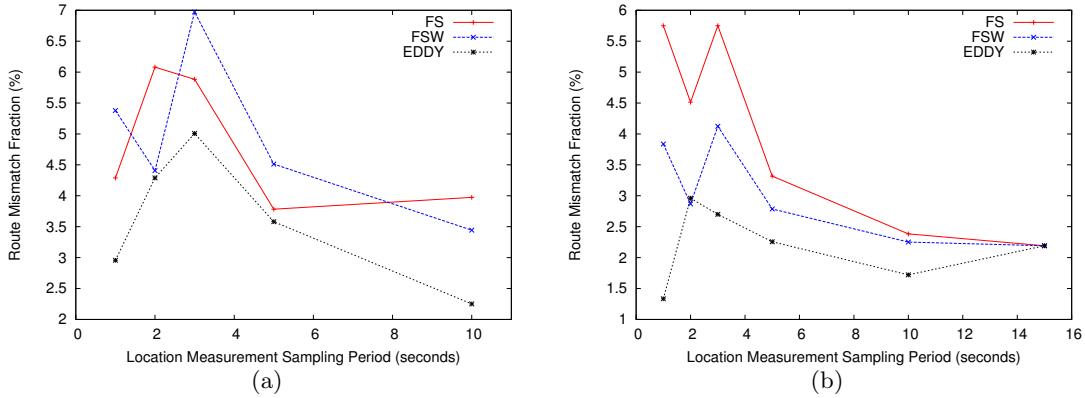

Figure 8: The comparisons of map matching results' accuracy for different location measurement sampling intervals under fixed latency constraints of: (a) 10 seconds and (b) 15 seconds.

The matching accuracy is measured by the Route Mismatch Fraction (RMF). This fraction is the total length of a false positive route in P and a false negative route in G_e divided by the length of the original route. We report RMF in percentage for each experiment and a higher RMF result indicates more erroneous road arcs are generated by the online map matching algorithm.

As illustrated in Figure 7, we report the map matching accuracy trend from immediate label generation to a latency of 120 seconds, under different measurement sampling periods. First, all figures show an overall declining trend of road arc mismatch fraction, which is sensible in that less error results are generated if more future location observations are analyzed within the HMM model. Second, the output quality of the FS algorithm is much less stable than the other two. Although the general trend of FS is descending as well, more fluctuations arise when the latency increases. By contrast, FSW and our algorithm are more stable, which means the matching results are confidently expected to be more accurate if more future information is provided. Most importantly, it is also observable that the curve of Eddy is mostly below FS and FSW. It indicates that our Eddy map matching system outputs better results in most cases with respect to two aspects: a) under the same latency constraints, the RMFs of Eddy are mostly the lowest one, and b) under the same accuracy constraints, Eddy is able to achieve the shortest latencies.

Figure 8 illustrates the online map matching accuracy improvements under the same latency constraints, 10 seconds and 20 seconds respectively. As shown, when applying our

algorithm to the location measurement datasets with different sampling rates, our matching result almost always outperforms the other two methods with less error erroneous generations.

Moreover, we also notice from the experiments that the RMF value of Eddy stably reaches 0 much earlier than FS and FSW (under different sampling rates shown in Figure 7). It means that our system is able to achieve a stable 100% accuracy of the road arc generation results with a much shorter latency.

6. CONCLUSIONS

We presented a real-time HMM-based map matching system, Eddy, based on an improved online Viterbi decoding algorithm. Our method analyzes the tradeoff between the map matching accuracy and latency, and incorporates a ski-rental model and its best-known deterministic algorithm to solve the online decoding problem. Therefore, our system is capable of dynamically selecting the window size according to characteristics of the candidate state probability distribution. The experimental results demonstrate the advantages of our approach on both accuracy and latency aspects. In our future work we plan to explore the possibility of involving nondeterministic algorithms into the online decoding phase to yield a better map matching accuracy and a shorter label generation delay.

Acknowledgements

This research has been supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office through the Centre of Social Media Innovations for Communities (COSMIC). We would also like to thank Paul Newson and John Krumm for making their dataset publicly available.

7. REFERENCES

[1] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching Planar Maps. In *14th Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2003.

[2] D. Bernstein and A. Kornhauser. An Introduction to Map Matching for Personal Navigation Assistants. 1998.

[3] R. Billen, E. Joao, and D. Forrest. *Dynamic and Mobile GIS: Investigating Changes in Space and Time*. CRC Press, 2006.

[4] J. Bloit and X. Rodet. Short-time Viterbi for Online HMM Decoding: Evaluation on A Real-time Phone Recognition Task. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008.

[5] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On Map-Matching Vehicle Tracking Data. In *31st International Conference on Very Large Data Bases*, 2005.

[6] S. S. Chawathe. Segment-based Map Matching. In *IEEE Intelligent Vehicles Symposium*, 2007.

[7] I. Constandache, S. Gaonkar, M. Sayler, R. R. Choudhury, and L. Cox. Enloc: Energy-efficient Localization for Mobile Phones. In *IEEE Conference on Computer Communications (INFOCOM)*, 2009.

[8] S. Fang and R. Zimmermann. Enacq: Energy-efficient GPS Trajectory Data Acquisition based on Improved Map Matching. In *19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2011.

[9] C. Y. Goh, J. Dauwels, N. Mitrovic, M. Asif, A. Oran, and P. Jaillet. Online Map-matching based on Hidden Markov Model for Real-time Traffic Sensing Applications. In *15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2012.

[10] J. S. Greenfeld. Matching GPS Observations to Locations on A Digital Map. In *Transportation Research Board 81st Annual Meeting*, 2002.

[11] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive Snoopy Caching. *Algorithmica*, 1988.

[12] W. Kim, G.-I. Jee, and J. Lee. Efficient Use of Digital Road Map in Various Positioning for ITS. In *Position Location and Navigation Symposium*. IEEE, 2000.

[13] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, et al. Place Lab: Device Positioning Using Radio Beacons in the Wild. In *Pervasive Computing*. Springer, 2005.

[14] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. Learning and Inferring Transportation Routines. *Artificial Intelligence*, 2007.

[15] Z. Lotker, B. Patt-Shamir, and D. Rawitz. Rent, Lease or Buy: Randomized Algorithms for Multislope Ski Rental. *SIAM Journal on Discrete Mathematics*, 2012.

[16] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for Low-sampling-rate GPS Trajectories. In *17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2009.

[17] A. Mohamed and K. Schwarz. Adaptive Kalman Filtering for INS/GPS. *Journal of Geodesy*, 1999.

[18] P. Newson and J. Krumm. Hidden Markov Map Matching Through Noise and Sparseness. In *17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2009.

[19] O. Pink and B. Hummel. A Statistical Approach to Map Matching Using Road Network Geometry, Topology and Vehicular Motion Constraints. In *Intelligent Transportation Systems*, 2008.

[20] 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*, 2007.

[21] M. A. Quddus, W. Y. Ochieng, L. Zhao, and R. B. Noland. A General Map Matching Algorithm for Transport Telematics Applications. *GPS Solutions*, 2003.

[22] R. Šrámek, B. Brejová, and T. Vinař. On-line Viterbi Algorithm and Its Relationship to Random Walks. *arXiv:0704.0062*, 2007.

[23] A. Thiagarajan, L. Ravindranath, H. Balakrishnan, S. Madden, L. Girod, et al. Accurate, Low-Energy Trajectory Mapping for Mobile Devices. In *8th USENIX Conference on Networked Systems Design and Implementation*, 2011.

[24] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. VTrack: Accurate, Energy-Aware Road Traffic Delay Estimation Using Mobile Phones. In *7th ACM Conference on Embedded Networked Sensor Systems*, 2009.

[25] A. J. Viterbi. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, 1967.

[26] C. E. White, D. Bernstein, and A. L. Kornhauser. Some Map Matching Algorithms for Personal Navigation Assistants. *Transportation Research Part C: Emerging Technologies*, 2000.

[27] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun. An Interactive-voting based Map Matching Algorithm. In *11th IEEE International Conference on Mobile Data Management (MDM)*, 2010.