Online map-matching based on Hidden Markov model for real-time traffic sensing applications

Online map-matching based on Hidden Markov model for real-time traffic sensing applications

C.Y. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, P. Jaillet

***Abstract*— In many Intelligent Transportation System (ITS) applications that crowd-source data from probe vehicles, a crucial step is to accurately map the GPS trajectories to the road network in real time. This process, known as map-matching, often needs to account for noise and sparseness of the data because (1) highly precise GPS traces are rarely available, and (2) dense trajectories are costly for live transmission and storage.**

We propose an online map-matching algorithm based on the Hidden Markov Model (HMM) that is robust to noise and sparseness. We focused on two improvements over existing HMM-based algorithms: (1) the use of an optimal localizing strategy, the variable sliding window (VSW) method, that guarantees the online solution quality under uncertain future inputs, and (2) the novel combination of spatial, temporal and topological information using machine learning. We evaluated the accuracy of our algorithm using field test data collected on bus routes covering urban and rural areas. Furthermore, we also investigated the relationships between accuracy and output delays in processing live input streams.

In our tests on field test data, VSW outperformed the traditional localizing method in terms of both accuracy and output delay. Our results suggest that it is viable for low-latency applications such as traffic sensing.

I. Introduction

Real-time sensor data collected by massive vehicle fleets, such as the taxis and buses in urban areas, provides vital inputs for applications such as traffic sensing [1], traffic incident detection [2], travel time prediction [3], fleet management [4] and route recommendations [5], [6]. The usefulness of these systems depends on the reliability of the data extracted by available map-matching algorithms, which project the GPS trajectories to their corresponding road segments on a digital map.

Information such as time stamps, locations and speeds are commonly recorded by the probe vehicles. However, the prohibitive amount of storage and bandwidth necessary for handling these large volumes of data has led to the practice of collecting sparse samples, with intervals ranging from tens of seconds to several minutes [7]. Furthermore, the sensors installed on these probes are known to be prone to various errors [7], such as imprecise location and speed measurements, repeated transmissions and time stamp mismatches. In real-time applications such as traffic sensing, it is also desirable to perform map-matching on-the-go while future data points are yet to be revealed, i.e. the algorithm has to be online. A reliable online map-matching algorithm thus has to account for these issues and guarantees outputs which are high in *accuracy* and *timeliness*.

Map-matching algorithms can be characterized as either *global* or *incremental/online*. Global algorithms batch-process the entire input trajectory before generating the solution. Incremental/online algorithms employ localizing strategies that divide the input trajectory into smaller segments and process them sequentially, sometimes resulting in a suboptimal solution. Techniques that have been applied in map-matching include geometrical analysis [12], belief function theory [13], Extended Kalman Filter [14] and Hidden Markov Model (HMM) [11], [15], [16]. The strengths and limitations of these methods have been reviewed in [9]. In particular, HMM-based algorithms and their variants [10], [17] have been adopted for their abilities to concurrently evaluate multiple hypotheses of the actual mapping in order to find the eventual maximum likelihood solution. These methods have been proven to be tolerant against highly noisy measurements, such as the location fingerprints from GSM towers [16], and their accuracies degenerate with increasing temporal sparseness of the trajectory [10], [15], [17].

Our proposed Online HMM (OHMM) map-matching algorithm is inspired by recent methods based on HMM [10], [15]–[17]. We addressed two specific issues that have not been focused in previous related works: (i) the need for an online algorithm that manages the trade-off between accuracy and output delay, and (ii) the fusion of multiple scoring functions to estimate the transition probability.

Most existing incremental/online algorithms use simple localizing strategies such as fixed sliding window and fixed-depth recursive look-ahead. The sliding window method simply divides the trajectory into fixed-sized input sequences and handles them independently. A larger window size leads to better accuracy [10], [17] but longer output delay, and vice versa. The recursive look-ahead method delays the decision of each point by a fixed number of steps to evaluate future path alternatives [18]. Both of these methods, while simple to implement, can lead to suboptimal solutions and long output delays. This is clearly undesirable when the map-matching algorithm is operating on live input streams and is required to generate outputs within a short time window. Motivated by

C. Y. Goh, J. Dauwels, N. Mitrovic and M. T. Asif are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (phone: (65) 85716607; e-mail: {gohc0048, jdauwels, nikola001, muhammad09}@ntu.edu.sg).

A. Oran is with the Center for Future Urban Mobility, Singapore-MIT Alliance for Research and Technology (SMART), Singapore 117543 (e-mail: aoran@smart.mit.edu).

P. Jaillet is with the Laboratory for Information and Decision Systems, Department of Electrical Engineering and Computer Science, and Operations Research Center, MIT, Cambridge, MA, 02139 USA (e-mail: jaillet@mit.edu).

the needs of real-time applications, we propose an optimal localizing strategy that uses a *variable sliding window* (VSW) to divide the inputs into smaller sub-problems and provably finds the global optimal solution. Our approach is conceptually similar to the online Viterbi algorithm in [19], [20]. Furthermore, we also developed a suboptimal variant of VSW that provides worst-case guarantee on the delay performance. In Section V, we will compare the performance of these two methods using the *fixed sliding window* (FSW) approach as the benchmark.

Secondly, we derived a probabilistic scoring mechanism that incorporates various sensor data and topological information in the modeling of emission and transition probabilities in HMM: (i) GPS coordinates (ii) vehicle speed (iii) speed limit (iv) road width (v) inferred vehicle heading directions, and (vi) topological constraints. We used Support Vector Machine (SVM) to learn the transition probability function instead of choosing a model a priori [10], [15], [17] and then estimating its parameters. The main advantage of this approach is that it provides a data-driven framework for integrating multiple transition scoring functions.

We evaluated the performance of our methods using field test data collected on 4 bus routes covering both rural and urban areas in Singapore. The performance was measured in terms of both accuracy and output delay criteria. Our findings show that when operating on real-time input streams, the proposed algorithms achieved optimal or near-optimal solutions with practically low output delays.

This paper is organized as follows. Section II formulates the map-matching problem in terms of HMM. Our method is described in Section III. Our experimental setup is explained in Section IV. The results are presented in Section V. Finally, Section VI summarizes our contributions and discusses the rooms for future works.

II. The Map-matching Problem

A. Problem Definition

Definition 1: A *trajectory*, $T = (t_n | n = 1, \dots, N)$, is a sequence of N data points collected by a vehicle. Each *trajectory point*, t_n is specified by its longitude (t_n.lon), latitude (t_n.lat), speed (t_n.v) and time stamp (t_n.t).

Definition 2: A *segment*, $r = (p_m | m = 1, \dots, M)$, is an M-point polyline representing a road segment curve. It consists of a series of line segments connecting the vertices $p_1, \dots p_M$ in order, where each vertex p_m is specified by its longitude and latitude. A segment is also defined by its road width (r.w), speed limit (r.v) and permissibility of bi-directional travel (r.d $\in$ {True, False}).

Definition 3: A *digital map*, $G = \{r_k | k = 1, \dots, K\}$ is a set of K segments representing a road network.

Given a trajectory T, the goal of map-matching is to find the correspondence between each trajectory point in T to a segment in G.

B. The HMM Approach

In HMM-based map-matching algorithms, candidate paths are sequentially generated and evaluated on the basis of their likelihoods. When a new trajectory point is encountered, past hypotheses of the solution are extended to account for the new observation. Among all candidates in the last stage, the *surviving path* with the highest joint probability is then selected as the final solution.

For every trajectory point, we first identify a set of candidate road segments from which the data was most likely collected. Each of these candidates is represented as a *hidden state* in the Markov chain and has an *emission probability*, which is the likelihood of observing the GPS point conditional on the candidate segment being the true match. Intuitively, we would assign a higher probability to a segment if the point were found nearer to it. Then, we compute the *transition probability* for every pair of adjacent hidden states in the chain such that the probability of the latter is dependent only on the former, hence obeying the Markov assumption. Our goal is to find the maximum likelihood path over the Markov chain that has the highest joint emission and transmission probabilities. The process is illustrated in Fig. 1.

Formally, we denote the emission probability as $p(t|h)$, which is the probability of observing a trajectory point t given the hidden state (segment) h. The transition probability from hidden state s to hidden state h is $f(s,h)$. Given an N-point trajectory, $T = (t_n | n = 1, \dots, N)$, the maximum likelihood sequence of hidden states, $S^* = (s_n \in \mathcal{A}_n | n = 1,\dots,N$, satisfies the following recurrence relation,

$$V_{n,h} = p(t_n|h) \max_{s \in \mathcal{A}_{n-1}} \{f(s,h) V_{n-1,s}\}. \quad (1)$$

Here $V_{0,h} = p(t_0|h)$ and $\mathcal{A}_n$ denotes the set of hidden states at stage n. Then, we can find S^* starting from the last element, $s_N = \arg\max_{h \in \mathcal{A}_N} \{V_{N,h}\}$, working backwards to find the sequence $s_{N-1}, \dots, s_1$ of maximum joint probability. We will present an online algorithm for finding S^* in Section III-E.

III. OHMM Map-matching Algorithm

A. Basic Flow of the Algorithm

- For each trajectory point, find all candidate segments within a radius of 50m around it. The reasons for imposing this threshold are two-fold: (1) to discard all candidates with very low emission probabilities

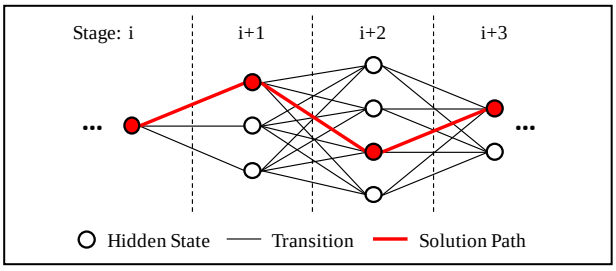

Fig. 1. In the Markov chain, each vertex (hidden state) has an emission probability and the weight of each edge (transition) is the transition probability between its connecting vertices.

(below the range of 10^{-4}), and (2) to avoid penalty in execution speed as a result of excess candidates.

- Emission probability is computed for each candidate segment (hidden state), whereas transition probability is assigned to every edge incident on the hidden state.
- The VSW algorithm performs backtracking on the updated Markov chain and gives the partial solution, if available. Otherwise, the output is delayed by one stage.
- The above process is repeated for the next trajectory point. The algorithm terminates when the last point is reached.

B. Emission Probability

For each candidate segment r found in the vicinity of a trajectory point t, we model its observation probability with a 1D Gaussian function as follows,

$$p(\text{observation}) = \frac{1}{2w}\int_{-w}^{w} \frac{1}{\sqrt{2\pi{\sigma_g}^2}} e^{-\frac{(l-d)^2}{2{\sigma_g}^2}}\, dl. \qquad (2)$$

Here w is the half-width of road segment r ($0.5 \cdot r.\text{w}$), d is the *point-to-curve* great-circle distance between t and r, and σ_g is the estimated standard deviation of GPS error. While the GPS error has been known to exhibit non-Gaussian distribution, we adopted this model for its ease of implementation and proven effectiveness in previous works [10], [15], [17]. Our approach is different in that we also account for the road width. This will allow finer distinction between the road segments, especially at junctions.

In addition, we incur a *speeding penalty factor* based on the assumption that drivers are unlikely to largely exceed the speed limits. The aim is to help distinguish between closely spaced parallel roads, possibly with different speed limits, that branch out from the same junction. We note that in these circumstances, position measurements alone are inadequate for differentiating the segments because the recorded trajectory points may fall in between them. We define the penalty function S as follows,

$$S(v_t, v_r) = \frac{v_r}{\max(0, v_t - v_r) + v_r}. \qquad (3)$$

Here v_t is the recorded speed of t ($t.\text{v}$) and v_r is the speed limit of segment r ($r.\text{v}$). If the speed limit were obeyed, then $\max(0, v_t - v_r) = 0$ (no cost will be incurred). Combining (2) and (3), we define the emission probability, $p(t|r)$ as follows,

$$p(t|r) = S(v_t, v_r)p(\text{observation}). \qquad (4)$$

C. Transition Probability

Let i, j denote a pair of candidate segments attributed to two consecutive trajectory points, t_n and t_{n+1}, respectively, where $i \in \mathcal{A}_n$ and $j \in \mathcal{A}_{n+1}$. We define the *interpolated path*, $P_{i\to j}$ as the sequence of segments that are most likely taken by the vehicle when travelling from i to j. Assuming that the shortest route is chosen by the driver (which is likely if the path distance were short), we can find the interpolated path using the A* path-finding algorithm [21]. For a given sequence of K segments, $(r_1, \ldots, r_K)$ in $P_{i\to j}$, we devise two scoring functions as follows,

Measure 1: The *distance discrepancy function*, T, measures the discrepancy between the sensor-deduced travelling distance and the interpolated path length,

$$T\left(d_{i\to j}, D_{i\to j}\right) = \frac{|d_{i\to j} - D_{i\to j}|}{D_{i\to j}}, \qquad (5)$$

where $d_{i\to j} = \bar{v}_{i\to j}\Delta t$ is the distance travelled by the vehicle over time interval Δt at average speed of $\bar{v}_{i\to j}$, whereas $D_{i\to j}$ is the path length of $P_{i\to j}$. Measure 1 above evaluates the feasibility of the hypothetical path $P_{i\to j}$ by comparing its length to the *Deduced Reckoning* (DR) estimate. The difference is assumed to be close to zero if $P_{i\to j}$ were the true path.

Measure 2: The *momentum change function*, M, measures the average momentum change incurred by the vehicle for every road segment taken in $P_{i\to j}$,

$$M(\boldsymbol{v_0}, \boldsymbol{v_1}, l_1, \ldots, \boldsymbol{v_K}, l_K) = \frac{\sum_i^K l_i \|\boldsymbol{v_i} - \boldsymbol{v_{i-1}}\|}{\bar{v}_{i\to j}\sum_i^K l_i}, \qquad (6)$$

where $(\boldsymbol{v_1}, \ldots, \boldsymbol{v_K})$ are the velocity vectors of the vehicle for each segment in $P_{i\to j}$, and $(l_1, \ldots, l_K)$ are the corresponding segment lengths. We assume that the vector magnitudes change linearly with time from $|\boldsymbol{v_1}|$ to $|\boldsymbol{v_N}|$, whereas their directions are parallel with the segment curves. Note that the additional parameter, $\boldsymbol{v_o}$ is the *initial velocity vector* which is inherited from the *terminal velocity* of the previous transition. By similar logic, we will use $\boldsymbol{v_K}$, the current terminal velocity as the initial velocity in the next transition, and so on. Fig. 2 illustrates this concept. Measure 2 above can be described as a 'smoothing factor' that penalizes infeasible transitions consisting of many abrupt turns.

The scoring functions T and M introduced above offer two different 'measures of fitness' for the transition $i \to j$. This suggests that the transition probability, $p(i \to j)$, can be derived by fusing these two measures together. We trained a

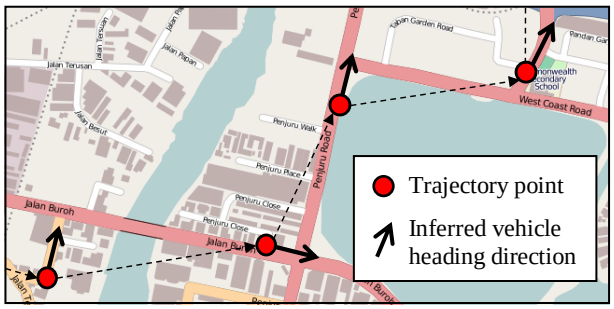

Fig. 2. At every stage, the vehicle heading direction is inferred based on the terminating direction of the last point and the historical path leading to the current point.

Support Vector Machine (SVM) classifier using instances that are labeled as either a correct or incorrect transition, where the feature vector consists of the component scores given by Measure 1 and Measure 2. With this classification approach, $p(i \rightarrow j)$ is the probability that an input score combination belongs to the 'correct transition' class. We will describe the training process with more details in Section IV-B.

D.Online Viterbi Algorithm

Our goal is to find the global map-matching solution using an incremental method. This means that the algorithm needs to make irreversible, online decisions along the Markov chain without knowledge of future inputs, while ensuring that the partial solutions, when combined, results in the global optimal solution. To achieve this, we apply online dynamic programming to solve the recurrence relations in (1). The key insight is that when the current surviving paths converge at some point (*convergence point*) in the Markov chain, all future surviving paths will contain the same sub-path up to the convergence point. The relevant proofs are available in [19] and [20].

We formulate the pseudocode of our OHMM algorithm as follows. Algorithm 1 (*MapMatchOHMM*) processes the trajectory points incrementally and in every stage, it outputs the partial solution returned by Algorithm 2, if available. Otherwise, it gives an empty output and incurs one delay stage. Algorithm 2 (*OnlineViterbiDecode*) checks if there exists any convergence point in the solution chain and returns the maximum likelihood subsequence up to that point, if any.

It is convenient to describe the working principles of Algorithm 1 and 2 in terms of sliding window. The window expands forward as new trajectory points are processed and shrinks from behind when a convergence point is found anywhere in the Markov chain covered by the window. Note that the sizes of the sliding windows can vary according to the structure of the state space, hence the name variable sliding window. Fig. 3 illustrates the working principles of VSW.

However, one disadvantage of VSW is that there is no guarantee of the worst-case window size; hence the *output delays* can be arbitrarily large in extreme cases. We modified Algorithm 1 by setting an upper bound on the window size such that when the threshold is reached, the algorithm will output the maximum likelihood solution up to the current stage. We will label this modified approach the *bounded variable sliding window* (BVSW) method. But unlike VSW, this approach may lead to suboptimal solutions.

Algorithm 1: MapMatchOHMM

Input: trajectory, $T = (t_1, \dots, t_N)$
Output: matching road segments, $S = (s_1, \dots, s_N)$
Let *score*[ ] store the joint probability up to each state;
Let *pre*[ ] store the parent of each state;
for $i = 1$ to $N - 1$ do
  Find the set of segments R nearest to t_i
  Find the set of segments L nearest to t_{i+1}
  for each l in L do
    Compute $p(l|t_{i+1})$
    for each r in R do
      if $i = 1$ then /* initialize the scores */
        Compute $p(r|t_i)$
        $score[r] = p(r|t_i)$
      Compute $p(r \rightarrow l)$ and infer heading directions
    $score[l] = p(l|t_{i+1}) \cdot \max_{r \in R}\{score[r] \cdot p(r \rightarrow l)\}$
    $c = \arg\max_{r \in R}\{score[r] \cdot p(r \rightarrow l)\}$
    $pre[l] = c$
  if $i < N - 1$ then /* output partial solution, if any */
    output *OnlineViterbiDecode*(R, pre)
  else /* terminate match */
    $e = \arg\max_{l \in L}\{score[l]\}$
    output *OnlineViterbiDecode*(e, pre)

Algorithm 2: OnlineViterbiDecode

Input: *R*, *pre*
Output: *sol*
Let *sol* [ ] denote the partial solution;
c = *findConvergencePoint*(*R*, *pre*)
while *c* is not *NULL* do
  sol.add(*c*)
  c = *pre*[*c*]
remove all *s* from *pre*[ ] where *pre*[*s*] is in *sol*
return *sol*.reverse()

IV. Experimental Setup

A. Field Test Data

Using GPS-enabled smart phones, we collected ground truth data on 4 pre-determined bus routes in Singapore as shown in Fig. 4. Since we are concerned with the *path accuracy* (which will be defined in Section VI-C) of map-matching results, knowledge of the actual test path (ground truth path) is enough for us to validate the algorithm. To enable comparisons of its performances under different environmental settings, we picked 4 routes that cover both the rural and urban areas in Singapore. The rural routes (R1 and R2) involve fewer turns and mostly consist of straight courses through open areas, such as expressways. The urban routes (U1 and U2), besides being more highly branched,

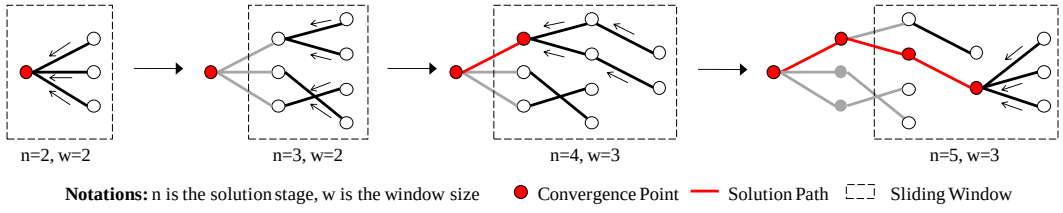

Fig. 3. The VSW method performs backtracking on the surviving paths and finds the convergence point, if any.

Fig. 4. The 4 selected bus routes cover both the rural and urban regions in Singapore.

cover city blocks which are densely packed with high-rise buildings. The lengths of R1, R2, U1 and U2 are 36.3km, 11.3km, 27.3km and 32.5km, respectively. Furthermore, to simulate trajectories of varying sampling frequencies, we sub-sampled our original data (recorded once every 1–3 seconds) at sampling intervals ranging from 10 seconds to 5 minutes.

B. Training and Parameter Estimation

The parameter σ_g needs to be estimated for the emission probability in (2) and SVM training is required for the transition probability.

We estimate σ_g by analyzing the perturbations of our ground truth data. For every trajectory point, we compute the great-circle distance of the point from the center of its nearest road segment. Then, the standard deviation is calculated based on the Median Absolute Deviation (MAD) of the distances,

$$\sigma_g = 1.4826 \,\mathrm{median_i}\left(\left|d_i - \mathrm{median_j}\left(d_j\right)\right|\right). \qquad (7)$$

Here d_i denotes the perpendicular distance between an individual trajectory point and its matching segment. The distribution of distances allows us to estimate the one-dimensional perturbations of trajectory points around the ground truth path. Note that in (7), the MAD is scaled by a constant factor of 1.4826 because we assumed that the GPS measurement error is normally distributed. We adopted the MAD approach for its resiliency against outliers in the data set. The same method for estimating the standard deviation has been adopted in [7], [15]. Based on the entire data set, we obtained $\sigma_g = 6.86$m.

To infer the transition probability, we trained a SVM classifier using 3,000 labeled instances where each corresponds to either a correct transition (class labeled '1') or an incorrect transition (class labeled '0'). Each instance is a 2D feature vector consisting of the score values computed with (5) and (6) and both components are scaled to [0, 1]. The scaling function is $(1+x)^{-1}$, where x is either component of the feature. Using a grid search on the parameter space and 5-fold cross validation, we found the best combination of parameters to be $C = 0.25$ and $g = 0.5$, where C is the soft margin parameter and g is the Radial Basis Function (RBF) kernel parameter. The training result is shown in Fig. 5.

C. Performance Evaluation

We will assess our algorithm using two performance metrics: accuracy and output delay.

Accuracy is defined as the fraction of correctly matched trajectory points in the ground truth path. A correct match is registered when the trajectory point is mapped to any road segment contained in the ground truth path. This measure of accuracy avoids penalizing 'boundary cases' where the points are located right in the middle of road junctions. We note that it is impractical to precisely determine the road segment from which every trajectory point was collected for two reasons: (i) 'boundary cases' can be attributed to the road segments on either exit of the junction, and (ii) possible miscalibration of the digital map. Therefore, the path accuracy measure is a more suitable assessment criterion.

Output delay is the average output latency incurred by the algorithm for each trajectory point. It is quantified by the number of seconds elapsed before a matching result is obtained.

Tests were conducted as follows:

- We performed map-matching on test trajectories of varying sampling intervals, ranging from 3 seconds to 5 minutes, for both the rural and urban test routes. For each route category, we aggregate the results obtained for the two test routes.
- We compared 3 localizing strategies in terms of accuracy and output delay: VSW, BVSW and FSW. For BVSW and FWS, different window sizes were tested. For every window size w, we aggregate the results for the whole set of test data (4 test routes with sampling intervals between 10 seconds to 5 minutes).

V. Results

Fig. 6 shows the comparison of map-matching accuracy between rural and urban test routes. The results indicate that accuracy for rural routes is better than urban routes by a margin of about 5%, except at sampling intervals larger than 4 minutes. At intervals of less than 1 minute, the accuracy for both routes is above 0.9. In both cases, the accuracy deteriorates with increasing sampling intervals.

In Fig. 7 and Fig. 8, the dotted line represents the optimal result achieved using the VSW localizing strategy. The BVSW method converges to the optimal accuracy of 0.921 at $w = 8$ and above. In all cases, the FSW method gave consistently lower accuracy and did not converge to the optimal solution even at $w = 20$.

In Fig. 8, the average output delay for VSW is 82s. Compared to FSW, it achieves substantially lower latencies without trading off optimality of the solution. Using BVSW, there was no significant advantage in the delay performance at window sizes of 4 and above. This suggests that most decision points in the Markov chain occurred before the

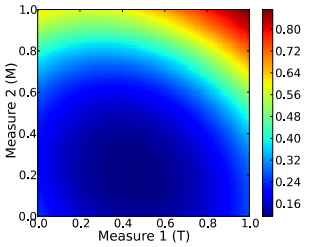

Fig. 5. Transition probability function derived from SVM training

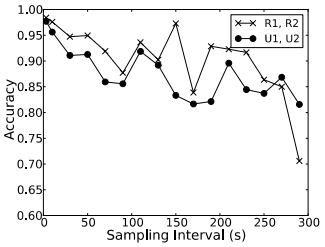

Fig. 6. Comparison of accuracy between rural and urban test routes

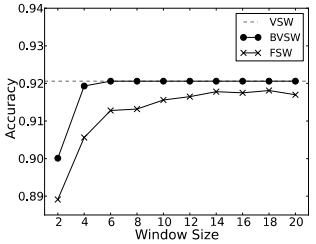

Fig. 7. Accuracy for VSW, BVSW and FSW, aggregated over all test routes and sampling intervals

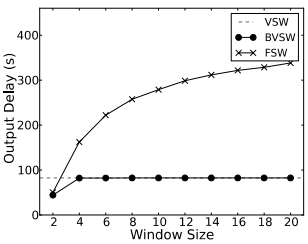

Fig. 8. Output delay for VSW, BVSW and FSW, averaged over all test routes and sampling intervals

window bound was reached. In the case of FSW, the delay increases proportionately with window size but the accuracy gains diminish to nearly zero after a certain threshold point.

VI. Conclusions & Future Work

In this paper, we described an online algorithm for map-matching and analyzed its performance on ground truth data. We devised the VSW and BVSW methods for finding the online solutions. Both outperformed the traditional FSW localizing strategy used in previous HMM-based algorithms in terms of accuracy and output delay. We also developed a data-driven approach for inferring the transition probability which fuses sensor measurements and topological information in the map-matching process. Altogether, these methods provide a general framework for designing online HMM-based map-matching algorithms which are suitable for real-time applications using floating car data. Other variants of the algorithm may incorporate additional sensor data, such as acceleration and altitude measurements, in estimating the emission and transition probabilities.

For future work, we can explore the design of map-matching algorithms with dynamic parameters that detect and adapt to different environmental settings, such as in urban or rural areas where GPS accuracies may vary. Sensor information, such as the dilution of precision (DOP) values for GPS measurements, may prove useful in achieving this goal. Furthermore, we suggest better methods [22] for interpolating the trajectory points rather than assuming the shortest paths between them. A better approximation of the actual traveled paths may improve map-matching accuracy.

Acknowledgment

The research described in this project was funded in part by the Singapore National Research Foundation (NRF) through the Singapore-MIT Alliance for Research and Technology (SMART) Center for Future Mobility (FM).

References

[1] Lim Z., Zhu Y., Zhu H. & Li M., "Compressive sensing approach to urban traffic sensing," *Distributed Computing Systems (ICDCS)*, 2011 31st International Conference, pp.889-898, 20-24 June 2011.

[2] Li M., Zhang Y. & Wang W., "Analysis of congestion points based on probe car data," *Intelligent Transportation Systems ITSC '09*, 12th International IEEE Conference, pp.1-5, 4-7 Oct. 2009.

[3] De Fabritiis, C., Ragona, R. & Valenti, G., "Traffic estimation and prediction based on real time floating car data," *11th International IEEE Conference on Intelligent Transportation Systems*, 197-203, 2008.

[4] Liao Z., "Real-time taxi dispatching using global positioning systems," *Communications* of the ACM, 46(5), 81-83, 2003.

[5] Zheng Y. & Xie X., "Learning travel recommendations from user-generated GPS traces," *ACM Transactions On Asian Language Information Processing*, 2(1), 9-es, 2011.

[6] Yuan J., Zheng Y., Zhang C., Xie W., Xie X., Sun G. & Huang Y., "T-Drive: Driving directions based on taxi trajectories," *Science And Technology*, 99-108, 2010.

[7] Wang Y., Zhu Y., He Z., Yue Y. & Li Q., "Challenges and opportunities in exploiting large-Scale GPS probe data", HP Laboratories, Technical Report HPL-2011-109, 21 Jul. 2011.

[8] Wenk C., Salas R. & Pfoser D., "Addressing the need for map-matching speed: localizing global curve-matching algorithms," *18th International Conference on Scientific and Statistical Database Management SSDBM06* (pp. 379-388), 2006.

[9] Quddus M., Ochieng W. & Noland R., "Current map-matching algorithms for transport applications: State-of-the art and future research directions," *Transportation Research Part C: Emerging Technologies*, 15(5), 312-328, 2007.

[10] Lou Y., Zhang C., Zheng Y., Xie X., Wang W. & Huang Y., "Map-matching for low-sampling-rate GPS trajectories," Proceedings of the *17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems GIS 09*, (c), 352, 2009.

[11] Pink O. & Hummel B, "A statistical approach to map matching using road network geometry, topology and vehicular motion constraints," *11th International IEEE Conference on Intelligent Transportation Systems*, 862-867, 2008.

[12] Chen D., Driemel A., Guibas L. J. & Wenk C., "Approximate Map Matching with respect to the Frechet Distance," *Computing*, 75-83, 2011.

[13] Nassreddine G., Abdallah F. & Denoeux T., "Map matching algorithm using interval analysis and Dempster-Shafer theory," *Intelligent Vehicles Symposium, 2009 IEEE*, vol., no., pp.494-499, 3-5 Jun 2009.

[14] Obradovic D., Lenz H., & Schupfner M., "Fusion of Map and Sensor Data in a Modern Car Navigation System," *The Journal of VLSI Signal Processing Systems for Signal Image and Video Technology*, 45(1-2), 111-122, 2006.

[15] Newson P. & Krumm J., "Hidden Markov map matching through noise and sparseness," Proceedings of the *17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems GIS 09*, 336, 2009.

[16] Thiagarajan A., Ravindranath L., Balakrishnan H., Madden S. & Girod L, "Accurate, Low-Energy Trajectory Mapping for Mobile Devices," *Artificial Intelligence*, 20–20, 2011.

[17] Yuan J., Zheng Y., Zhang C., Xie X., & Sun G.-Z., "An Interactive-Voting Based Map Matching Algorithm," *Eleventh International Conference on Mobile Data Management*, 43-52, 2010.

[18] Brakatsoulas S., Pfoser D., Salas R. & Wenk, C, "On Map-Matching Vehicle Tracking Data," Proceedings of the *31st International Conference on Very Large Databases* (pp. 853-864), 2005.

[19] Bloit J. & Rodet X, "Short-time Viterbi for online HMM decoding: Evaluation on a real-time phone recognition task," *IEEE International Conference on Acoustics Speech and Signal Processing*, 2121-2124, 2008.

[20] Šrámek R., Brejová B. & Vinař T., "On-line Viterbi Algorithm and Its Relationship to Random Walks", *arXiv:0704.0062v1*, 2007.

[21] Hart, N. Nilsson, & B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on system science and cybernetics*, 4:100-107, 1968.

[22] Leung I. X. Y., Chan S.-Y., Hui P. & Lio P, "Intra-City Urban Network and Traffic Flow Analysis from GPS Mobility Trace," *arXiv:1105.5839v1*, 2011.