An incremental map-matching algorithm based on Hidden Markov Model *

Piotr Szwed and Kamil Pekala

AGH University of Science and Technology
pszwed@agh.edu.pl, kamilkp@gmail.com

Abstract. Map-matching algorithms aim at establishing a vehicle location on a road segment based on positioning data from a variety of sensors: GPS receivers, WiFi or cellular radios. They are integral part of various Intelligent Transportation Systems (ITS) including fleet management, vehicle tracking, navigation services, traffic monitoring and congestion detection. Our work was motivated by an idea of developing an algorithm that can be both utilized for tracking individual vehicles and for monitoring traffic in real-time. We propose a new incremental map-matching algorithm that constructs of a sequence of Hidden-Markov Models (HMMs). Starting from an initial HMM, the next models are developed by alternating operations: expansion and contraction. In the later, the map-matched trace is output. We discuss results of initial experiments conducted for 20 GPS traces, which to test algorithm robustness, were modified by introduction of noise and/or downsampled.

Keywords: GPS, map-matching, Hidden Markov Model, Viterbi

1 Introduction

Map-matching algorithms aim at establishing a vehicle location on a road segment based on positioning data from a variety of sensors: GPS receivers, WiFi or cellular radios, odometers and others. As all sensors used as input may yield uncertain data, map-matching involves making decision on to which location at several candidate road segments the vehicle should be assigned. The decision can be based on a current sensor reading or on a history comprising a number of past data.

Map-matching is an integral part of various Intelligent Transportation Systems (ITS) including fleet management, vehicle tracking, navigation services, traffic monitoring and congestion detection. Such systems experience growing

* This is the draft version of the paper presented at the **Artificial Intelligence and Soft Computing - 13th International Conference, ICAISC 2014**, Zakopane, Poland, June 1-5, 2014. The paper was published in **Artificial Intelligence and Soft Computing – Lecture Notes in Computer Science**, Volume 8468, 2014 and is available at: http://link.springer.com/chapter/10.1007/978-3-319-07176-3_51

popularity due to proliferation of smartphone devices capable of receiving positioning data and transferring them over cellular networks. The type of map-matching algorithm that they internally use depends on particular application.

Our work was motivated by an idea of developing an algorithm that can be both utilized for tracking individual vehicles and for monitoring traffic in real-time. Such algorithm must be *incremental*, i.e. should update the information upon arrival of new sensor reading, as opposed to *global*, when a closed sequence of readings is analyzed. In this paper we propose a new incremental map-matching algorithm, which in order to determine the vehicle trajectory constructs a sequence of Hidden-Markov Models (HMMs). In our approach a HMM state corresponds to a road segment and a sensor reading to an observation in HMM. Starting from an initial HMM, the next models are developed by alternating operations: expansion (new states are added to the model) and contraction (dead ends are deleted, the graph root is moved forward along the detected path and a part of trajectory is output). We report results of initial experiments conducted for 20 GPS traces, which to test algorithm robustness, were modified by introduction of artificial noise and/or downsampled.

The paper is organized as follows: the next Section 2 discusses various types of map-matching algorithms. It is followed by Section 3, in which a model of road network is described. The next Section 4 introduces HMM model and provides the algorithm description. Conducted experiments are reported in Section 5 and finally Section 6 gives concluding remarks.

2 Related works

More then thirty map-matching algorithms are surveyed by Quddus et. al in [1]. Authors divided them into four groups: geometric, topological, probabilistic and advanced.

Algorithms employing geometric analysis take into account shapes of road segments only, while ignoring, how they are connected. The simplest approach consists in finding the closest map node (a segment endpoint) to the current GPS reading (point-to-point matching). Another option is to find the closest road segment (point-to-curve matching) [2] or to match pairs of points from the vehicle trajectory to the road segments [3]. All those algorithms are very fast, however, they are sensitive to map data (in particular to the density of nodes) and may yield vehicle trajectories, which are not consistent with the connections within the road network [4].

Topological map-matching algorithms utilize information about connections between road segments. This removes leaps between map links that can be observed for algorithms based only on geometrical information. Another features that can be considered are turn angles and also a vehicle state (heading, velocity) [3, 5].

Many positioning devices are capable of delivering a circular or elliptic confidence region associated with each position reading, e.g. the location API for Android devices defines the *accuracy* parameter. The circle radius can be about

10 meters for GPS [6] and 50 m for cellular networks. However, in dense urban area with street canyons and in the presence of trees, the GPS accuracy can degrade substantially. The confidence region can be also estimated using dead reckoning. The idea behind probabilistic algorithms is to select in the match mapping process only those road segments that intersect with the confidence region. If several candidates are found, only one of them with the highest probability is chosen. Such approach was discussed in [7]. An enhanced algorithm that employs such approach at junctions was described in [8].

Advanced algorithms usually combine both topological and probabilistic information, while applying various techniques to assign road links to GPS readings. Kim et al. used Kalman Filter [9] to establish vehicle location along a link after performing point-to-curve matching [10]. Fuzzy Sugeno rules for road segment selection were used in [11, 12]. Gustafsson et al. reported an approach based on particle filter [13], Yang et al. applied Dempster-Shafers evidence theory while determining weights in point-to-curve matching.

Several map-matching algorithms are path-oriented, i.e. they maintain a set of candidate paths. In the algorithm developed by Marchal et al. [14] they ware stored in a collection being sorted according to a path score based on distance to the GPS trace. An idea of using a tree like structure representing a set of candidate paths was proposed by Wu et al. [15]. Both algorithms are *incremantal*, i.e. they update the path representation on arrival of a new GPS reading. On the other hand, if a full GPS trace is initially known a *global* approach based on calculation of Fréchet distance can be applied [16].

Hidden Markov Model (HMM) [17] is a Markov process comprising a number of hidden (unobserved) states. Transitions between states can occur with a certain probabilities. Each state is assigned with a set of observations. One of them is to be output, as the state is reached. For a given state conditional probabilities of observations occurrence (*emission probabilities*) sum up to 1. A problem that can be elegantly formulated with HMM is the *decoding* problem: it consists in finding the most probable sequence of transitions between hidden states that would produce a given sequence of observations. Such sequence can be efficiently determined with the well-known Viterbi algorithm [18].

There are at least four implementations of global map-matching algorithms [19–22] that employ HMM approach. In all of them hidden states correspond to projections of vehicle positions on road segments and observations to location data obtained mainly from GPS sensors. Transition probabilities are established based on links connectivity and/or dead reckoning, whereas emission probabilities assume Gaussian distribution of GPS noise.

In this paper we have taken the similar approach. The main difference that should be emphasized is that our algorithm is incremental, i.e. it does not build a single HMM model for a given GPS trace to be analyzed afterwards with the Vitrebi algorithm, but updates the HMM model on each input and in some situations only applies the Viterbi algorithms. Moreover, the algorithms can be a basis for developing real-time services like vehicle tracking and traffic estimation.

3 Road network model

The used road network model is defined as a directed graph $G = (V, E, I)$, where $V \in \mathbb{R}^2$ are graph nodes described by two coordinates: longitude and latitude, $E \in V \times V$ are straight road segments linking two nodes and $I \subset E \times E$ specifies inhibited maneuvers at road junctions. If $((v_1, v_2), (v_2, v_3)) \in I$, then a path containing the sequence (v_1, v_2, v_3) is forbidden according to traffic regulations.

We assume that road links are represented by straight segments. If a road has a curved geometry, e.g. appears on a map as an arc, it can be approximated a sequence of connected segments. This is a typical approach for many map sources, e.g. OSM [23]. Moreover, as performing various geometrical operations we are actually interested in undirected arcs, we define a function $S\colon E \to 2^V$ that maps an edge (v_1, v_2), $v_1 \neq v_2$ onto a a set $\{v_1, v_2\}$ comprising exactly two vertices. We will extend this function to the whole set E, hence $S(E) = \bigcup_{e \in E} S(e)$.

For a given segment $s = \{v_b, v_e\}$ and a point g, we define the projection $p(g, s)$ of g onto s as a point g' belonging to the segment s that minimizes the distance:

$$p(g, s) = \underset{g' = v_b + t(v_e - v_b) \wedge t \in [0,1]}{\arg\min} d(g, g'), \tag{1}$$

where $d(g, g')$ is a distance between g and g' given by the *haversine* formula.

The projection point $p(g, s)$ calculated according to formula (1) can be either an orthogonal projection on a segment or one of its end points (see Fig. 1).

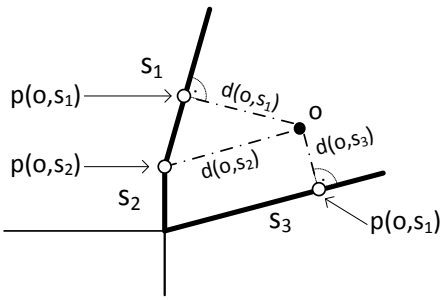

Fig. 1. Projections of a GPS point o and distances to road segments s_1, s_2 and s_3

4 Map matching algorithm

The map-matching algorithm comprises three basic operations organized into a pipeline (see Fig. 2). Firstly, an input GPS trace is smoothed with Kalman filter. This allows for compensating noise and removing outliers from the trace.

In the next processing step it is checked, whether the distance between two consecutive samples is small enough to match the map scale (or more precisely lengths of typical road links). If the distance is too large, the required number of intermediate samples is generated by applying simple linear interpolation. Finally, the input trace after the two preprocessing steps is interpreted with the proper map-matching algorithm based on Hidden Markov Model (HMM). Due to limited capacity, in this section we will focus on this step only.

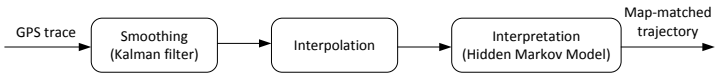

Fig. 2. Processing steps of the map-matching algorithm

4.1 Hidden Markov Model

While constructing Hidden Markov Models the approach similar to [21, 22] was taken. A state in HMM describe both a road segment and a projection of a GPS fix on the segment calculated according to formula (1). Thus, each state tuple $(s, p, i) \in Q$ has the following components: e - a road segment, p - a projection point belonging to the segment $S(e)$ and i - a sequence number.

In the assumed model observations O correspond to data obtained from GPS sensor, i.e. they are tuples (x, y, t), whose elements are longitude, latitude and time respectively.

Below we give the definition of Hidden Markov Model reflecting adaptation introduced to support the map matching problem.

Definition 1 (Hidden Markov Model). *Hidden Markov Model is a tuple* $\lambda = (Q, A, O, P_t, P_o, q_0)$*, where*

- Q *is a set of states,* $Q \subset E \times \mathbb{R}^2 \times \mathbb{N}$
- $A \subset Q \times Q$ *is a set of arcs,*
- O *is a set of observations,* $O \subset \mathbb{R}^2 \times \mathbb{R}$
- $P_t \colon A \to (0, 1]$ *is a function that assigns a probability to a transition between states.*
- $P_o \colon Q \times O \to [0, 1]$ *is an emission probability function satisfying* $\forall q \in Q \colon \sum_{o \in O} P_o(q, o) = 1$*.*
- q_0 *is an initial (root) state.*

Two states $q_1 = (e_1, p_1, i_1)$ and $q_2 = (e_2, p_2, i_2)$, where $e_1 = (v_{11}, v_{12})$ and $e_2 = (v_{21}, v_{22})$, can be connected with an arc $a = (q_1, q_2)$, if $e_1 = e_2$ or there exists a path in a graph $\pi = v_{11}, \ldots v_{22}$ linking endpoints of road segments. Currently, in most cases we consider sequences of length 3, i.e. two consecutive segments having common endpoints. Longer sequences can be calculated to

handle special situations requiring reinitialization of algorithm. This assumption imposes the requirement that observations (locations obtained form a GPS sensor should be dense enough to be assigned to consecutive segments. If a segment was missed, then the map matching algorithm would probably get lost. The interpolation step (see Fig. 2) was introduced to satisfy this requirement and achieve real-time performance.

In order to calculate a transition probability for an arc a linking states q_1 and q_2 a weight function $\theta(a)\colon A \to [0,1]$ is used. Basically, it assigns 1 if q_1 and q_2 can be connected by a path, however if the possible path violates traffic rules or physical constraints (e.g. speed greater than 250 km/h) a small value (0.1) is used. Finally, the weights assigned to outgoing arcs for a given state q are normalized applying the formula (2) to give the probabilities.

$$\begin{gathered} P_t(a) = \tfrac{1}{Z_t}\theta(a), \\ \text{where } Z_t = \sum_{\substack{a_i\colon a_i=(q,q_i)\in A \\ a=(q,q_a)}} \theta(a_i). \end{gathered} \tag{2}$$

Emission probability P_o is computed for a subset of states in HMM Q_H and an observation o. For a given HMM state $q = (e, p, i)$, where $p = (x_p, y_p)$ is the vehicle position, its GPS observations o can be distributed on XY plane around the point p. Until there is no bias, e.g. related to satellite visibility, the applied distribution should have its mean at the point p and decrease with growing distance between points $d(p, o)$. We have assumed 2-dimensional normal distribution given by (3).

$$P(x,y) = \frac{1}{D} e^{-k\left((x-x_p)^2+(y-y_p)^2\right)}. \tag{3}$$

The D normalizing factor is given as $D = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} P(x,y)\,dx\,dy$. For k the value 0.01 was taken, what corresponds to a noise giving translations of GPS readings by 10m. In such case $D \approx 314.0$. As the map data used in experiments used longitude and latitude coordinates, we applied, however, a modified version of (3), in which Euclidean distance was replaced by the haversine formula.

4.2 Trace interpretation algorithm

The algorithm takes at input a sequence of GPS readings (observations) $\omega = (o_i\colon i = 1, n)$ and constructs a sequence of Hidden Markov Models $\Lambda = (\lambda_i\colon 0 = 1, n)$. Basically, it contains two stages: *initialization*, during which the first model λ_1 is built and *processing* that is repeated for successive observations.

Initialization. This stage involves determining a set of possible states (road segments), to which the initial vehicle position might be assigned. The algorithm examines all road segments in a supplied part of the map and chooses only these, whose distances to the measured point are less or equal than a certain threshold r (e.g. 35 meters). At that point the construction a sequence of HMMs, which can be perceived as a trajectory tree, begins. The tree root is set to a fictional

state from which the vehicle might have moved to any of the states belonging to initial Hidden Markov Model λ_1. The steps of this stage are listed in Algorithm 1.

Algorithm 1 Initialization

1. For a given observation o_1 calculate a set of road segments, whose distance to o_1 is less or equal r, where r is a certain threshold: $H = \{e \in E \colon d(o_1, p(o_1, S(e))) \leq r\}$.
2. Assign $Q_1 \leftarrow \{q_0\} \cup \{(e_i, p(o_1, S(e_i), 1) \colon e_i \in H\}$; q_0 is a fictitious state (a tree root).
3. Connect states with arcs $A_1 \leftarrow \{q_0\} \times Q_H$, where $Q_H = Q_1 \setminus \{q_0\}$.
4. Assign to each arc $a \in A_1$ equal transition probability $P_{t1}(a) = \frac{1}{|Q_H|}$.
5. For each element in Q_H calculate emission probability according to formula (3).

Processing. This step is being applied repeatedly for all, but the first GPS observations. Each i-th iteration comprises two phases: *expansion* and *contraction*, during which a new HMM model λ_i is constructed. The expansion phase consists in adding new states and transitions to previous model λ_{i-1}. Its steps are given by Algorithm 2.

Algorithm 2 Expansion

1. Select the set of states in λ_{i-1} that was added in the previous iteration (heading states) $Q_H = (e, p, k) \in Q \colon k = i - 1$.
2. Establish a set of edges E_R that are physically reachable from Q_H. As discussed in Section 4.1, currently, only neighbor edges are considered.
3. Calculate a subset $E_{RD} \subset E_R$ comprising those edges, which are placed at a distance less than or equal to a certain threshold r:
   $E_{RD} = \{e \in E_R \colon d(o_i, S(e)) \leq r\}$.
4. Insert edges from E_{RD} as new states into the model λ_i. Hence, $Q_i \leftarrow Q_{i-1} \cup \{(e, p(e, S(e), i) \colon e \in E_{RD}\}$.
5. Link new states with edges: $A_i \leftarrow A_{i-1} \cup Q_H \times (Q_i \setminus Q_{i-1})$
6. Establish transition and emission probabilities according to (2) and (3).

The contraction phase has two goals: firstly orphan nodes without successors are removed, what keeps the detection model compact, secondly the HMM root is moved forward and a next part of the trajectory is output. Operations conducted during this phase are summarized in Algorithm 3.

One of the contraction operation, namely *join handling* requires some comments. A state q_J is a join, if it has two different predecessors:

$$\exists q_a, q_b \in Q_{i-1} \colon (q_a, q_J), (q_a, q_J) \in A \wedge q_a \neq q_b.$$

Algorithm 3 Contraction

1. Remove from Q_i all states with the timestamp less then i, i.e. assign: $Q_i \leftarrow Q_i \setminus \{(e, p, k) \colon k < i\}$ and update A_i accordingly.
2. Handle joins.
3. Update the root state q_r:
   a If q_r has exactly one successor q_j in A, output q_r as a next element of the vehicle trajectory. Otherwise, STOP.
   b Assign: $q_r \leftarrow q_j$ and go to a.

Such situation is illustrated in Fig. 3, where states q_2 and q_J are examples of joins. Presence of a join in HMM indicates that during the map matching process vehicle positions were assigned to parallel roads that finally joined at a certain point. Hence, the algorithm faces the problem of selecting the most probable among at least two competing paths. This is achieved with a dedicated procedure that searches the closest parent node q_F, from which (1) *all* paths led to q_J and (2) states belonging to them are reachable only from q_F. If such state exist, the Viterbi algorithm is applied to the subgraph between q_F and q_J and most probable path is kept in the λ_i. States lying beyond the computed path are removed.

The subgraph between q_F and q_2 in Fig. 3 does not satisfy the conditions given above, as there exists a path from q_F to q_3 that is not closed. Similarly, the subgraph between q_1 and q_J cannot be accepted, as q_2 is a join for a path that does not start in q_1. The subgraph between q_F and q_J satisfies the given conditions and, after applying the Viterbi algorithm, can be replaced by a single path between these nodes.

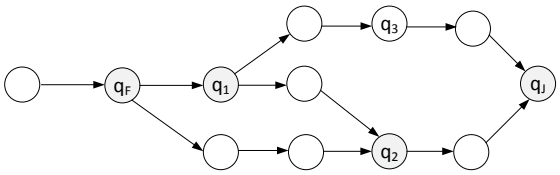

Fig. 3. Example of submodel of HMM, to which Viterbi algorithm is applied to get rid off joins

Handling special situations. An exceptional situation in expansion phase occurs if the set E_{RD} established in step 3 of Algorithm 2 is empty. We may conclude then, that the map matching algorithm got lost. There may be several reasons of such situations. It may stem from a noise that was not sufficiently removed by the Kalman filter. The other reason can be that observations are not dense enough to be matched to neighbor map segments. Such effect can be

observed at curved roads, e.g. highway links or roundabouts, which are approximated by a number of short straight lines. Basically, we handle this issue by performing reinitialization using Algoritm 1 and obtaining a new model λ_{i0}.

Further processing depends on application of the map matching component within an ITS. If we are particularly interested in reconstructing the trajectory of a tracked vehicle, models λ_{i-1} and λ_{i0} are merged by adding links that are obtained by applying locally A* shortest path algorithm. If the goal of map matching is to calculate traffic parameters based on GPS readings, some tracking errors can be accepted. For such applications λ_{i-1} model is processed with Viterbi algorithm to get the most probable path and the whole matching process restarts from λ_{i0}.

Another specific situation is, when it is known that the sequence of observations ω is finite and ends with o_n. Then for the last model λ_n the most probable trajectory is computed with Viterbi algorithm and the algorithm stops.

5 Experiments

The algorithm was tested on the map of Kraków in Poland. The map originated from OpenStreetMap project [23]. The input dataset was represented by 20 GPS traces, which were recorded during several car trips throughout Kraków with EasyTrials GPS[1] software running on iPhone 5. The total length of traces used in experiments was 148.46km. Both input and map-matched trajectories were stored in GPX format that is supported by JOSM, the OpenStreetMap editor.

The first phase of experiments consisted in determining parameters of a discrete Kalman filter used in the preprocessing phase. It was designed in form of two distinct second order filters processing separately noise for longitude and latitude components. The state variables corresponded, hence, to position and velocity along one of the axes. Initial parameters for both filters were identical. They were determined empirically using the Matlab software for calculations and visualization. The best effects were achieved (and thus those were applied in the final algorithm) with the following set of parameters (see [9] for notation details): process noise covariance matrix $Q = \begin{bmatrix} 1 & 0 \\ 0 & 0.05 \end{bmatrix}$, measurement noise covariance matrix $R = 4.5$ and initial process noise covariance matrix $P = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$. It should be mentioned that the selected parameters reflect features of the used sensor, i.e. this installed in iPhone 5. They may differ for other device. The results of of trace smoothing with the designed filter are shown in Fig. 4. An artificial noise introduced into the original trace yields the zig-zag line, which is smoothed (the black bold line).

Analyzing the traces manually we have observed that invalid paths was obtained (i.e. broken and impossible to repair with A*) in about 30% of the cases, in which the algorithm was forced to perform reinitialization. Moreover, reitialization usually takes more time than the normal algorithm processing step. Thus,

[1] `http://www.easytrailsgps.com/`

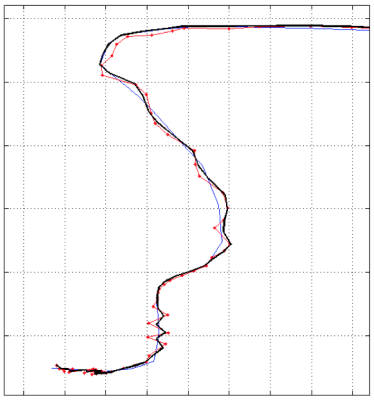

Fig. 4. Example of application of the implemented discrete Kalman filter

to avoid manual annotating of GPS trials, we decided to use the number of reinitialization as a quality metrics.

We have tested the algorithm by feeding the collected data in four forms: original, modified by artificially introduced random noise (magnitude between 0 and 20 meters added to each sample), half sampled (H-S) and half-sampled with the noise. The obtained values are gathered in Table 1. Each table row shows test results for a particular GPS trace. Subcolums marked with *RI* give number of algorithm reinitializations in the selected mode, *RIS* denotes average number of reinitalizations per sample.

It is clear and not surprising that the best results were achieved by running the tests with the original input data. However, for applied half-sampling the number of reinitializations was practically identical. This effect can be probably attributed to the interpolation. The worse indicator value was obtained for noisy data. Nevertheless, all obtained values are fairly good. In the normal mode the reinitialization occurred once per 6.18km and in about 70% of cases it was possible to recover from errors.

Only initial results of performance tests can now be reported. The algorithm implemented in C# language and published as a RESTfull web service was capable of processing 20 simultaneous feeds with 50 times speed-up, i.e. time intervals between subsequent send operations were 50 times smaller then differences between sample timestamps. This corresponds to 1000 mobile sensors feeding real-time data simultaneously.

6 Conclusions

This paper presents a new map matching algorithm based on Hidden Markov Model. Although the idea of applying HMM to map matching was reported already in a few articles, in works by Krumm et. al [20] and Newson and Krumm [21] only a global algorithm analyzing the whole path was described. The article

Table 1. Test results

No	Length (km)	Samples	Original		Noise		H-S		H-S & Noise	
			RI	RIS	RI	RIS	RI	RIS	RI	RIS
1	9.45	256	0	0	3	0.012	3	0.012	3	0.012
2	8.26	248	3	0.012	10	0.04	2	0.008	1	0.004
3	7.78	261	2	0.008	5	0.019	1	0.004	2	0.008
4	7.67	267	0	0	6	0.022	3	0.011	5	0.019
5	9.67	233	3	0.013	0	0	0	0	0	0
6	6.40	209	0	0	0	0	0	0	0	0
7	5.59	108	0	0	0	0	0	0	1	0.009
8	9.03	259	0	0	10	0.039	0	0	3	0.012
9	7.05	216	1	0.005	1	0.005	1	0.005	0	0
10	7.94	248	2	0.008	4	0.016	1	0.004	0	0
11	7.19	190	0	0	1	0.005	1	0.005	10	0.053
12	11.22	273	1	0.004	7	0.026	1	0.004	2	0.007
13	4.19	118	0	0	0	0	1	0.008	1	0.008
14	5.96	192	2	0.01	2	0.01	0	0	2	0.01
15	9.03	271	0	0	2	0.007	0	0	0	0
16	6.45	242	1	0.004	3	0.012	2	0.008	3	0.012
17	7.95	228	4	0.018	7	0.031	3	0.013	3	0.013
18	7.41	192	1	0.005	3	0.016	2	0.01	2	0.01
19	7.06	283	4	0.014	7	0.025	2	0.007	6	0.021
20	3.16	188	0	0	2	0.011	0	0	1	0.005
Total	148.47	4482	24	0.005	73	0.016	23	0.005	45	0.010

by Thiagarajan et al. gives yet less details [22]. The paper makes two contributions. Firstly we describe an incremental algorithm that in each iteration updates the HMM model by expanding it with new states corresponding to road segments and contracting to output a certain part of the vehicle trajectory. In most cases the structure of obtained HMM forms a tree similar to that, proposed by Wu et al. [15]. However, our model accepts parallel roads. Our second contribution is the report on performed tests showing that the developed algorithm with applied filtering and interpolation operations is robust enough to handle noisy and downsampled data.

Acknowledgments. This work is supported by the National Centre for Research and Development (NCBiR) under Grant No. O ROB 0021 01/ID 21/2

References

1. Quddus, M.A., Ochieng, W.Y., Noland, R.B.: Current map-matching algorithms for transport applications: State-of-the art and future research directions. Transportation Research Part C: Emerging Technologies **15**(5) (2007) 312–328
2. White, C.E., Bernstein, D., Kornhauser, A.L.: Some map matching algorithms for personal navigation assistants. Transportation Research Part C: Emerging Technologies **8**(1) (2000) 91–108

3. Greenfeld, J.S.: Matching GPS observations to locations on a digital map. In: National Research Council (US). Transportation Research Board. Meeting (81st: 2002: Washington, DC). Preprint CD-ROM. (2002)
4. Quddus, M.A., Ochieng, W.Y., Noland, R.B.: Integrity of map-matching algorithms. Transportation Research Part C: Emerging Technologies **14**(4) (2006) 283–302
5. Quddus, M., Ochieng, W., Zhao, L., Noland, R.: A general map matching algorithm for transport telematics applications. GPS Solutions **7**(3) (2003) 157–167
6. Modsching, M., Kramer, R., ten Hagen, K.: Field trial on GPS accuracy in a medium size city: The influence of built-up. In: 3rd Workshop on Positioning, Navigation and Communication. (2006) 209–218
7. Zhao, Y.: Vehicle location and navigation systems. Artech House ITS series. Artech House (1997)
8. Ochieng, W.Y., Quddus, M., Noland, R.B.: Map-matching in complex urban road networks. Revista Brasileira de Cartografia **2**(55) (2009)
9. Greg Welch, G.B.: An introduction to the Kalman filter (2006) Chapel Hill.
10. Kim, W., Jee, G.I., Lee, J.: Efficient use of digital road map in various positioning for its. In: Position Location and Navigation Symposium, IEEE 2000. (2000) 170–176
11. Syed, S., Cannon, M.: Fuzzy logic-based map matching algorithm for vehicle navigation system in urban canyons. In: proceedings of the Institute of Navigation (ION) national technical meeting, USA. (2004)
12. Fu, M., Li, J., Wang, M.: A hybrid map matching algorithm based on fuzzy comprehensive judgment. In: Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on. (2004) 613–617
13. Gustafsson, F., Gunnarsson, F., Bergman, N., Forssell, U., Jansson, J., Karlsson, R., Nordlund, P.J.: Particle filters for positioning, navigation, and tracking. Signal Processing, IEEE Transactions on **50**(2) (2002) 425–437
14. Marchal, F., Hackney, J., Axhausen, K.: Efficient map-matching of large GPS data sets-tests on a speed monitoring experiment in Zurich. Arbeitsbericht Verkehrs-und Raumplanung **244** (2004)
15. Wu, D., Zhu, T., Lv, W., Gao, X.: A heuristic map-matching algorithm by using vector-based recognition. In: Computing in the Global Information Technology, 2007. ICCGI 2007. International Multi-Conference on. (2007) 18–18
16. Brakatsoulas, S., Pfoser, D., Salas, R., Wenk, C.: On map-matching vehicle tracking data. In: Proceedings of the 31st international conference on Very large data bases, VLDB Endowment (2005) 853–864
17. Rabiner, L., Juang, B.: An introduction to hidden Markov models. ASSP Magazine, IEEE **3**(1) (1986) 4–16
18. Viterbi, A.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. Information Theory, IEEE Transactions on **13**(2) (1967) 260–269
19. Hummel, B.: 10. Innovations in GIS. In: Map Matching for Vehicle Guidance. CRC Press (November 2006)
20. Krumm, J., Letchner, J., Horvitz, E.: Map matching with travel time constraints. In: SAE World Congress. (2007)
21. Newson, P., Krumm, J.: Hidden Markov map matching through noise and sparseness. In: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM (2009) 336–343

22. Thiagarajan, A., Ravindranath, L., LaCurts, K., Madden, S., Balakrishnan, H., Toledo, S., Eriksson, J.: Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, ACM (2009) 85–98
23. OpenStreetMap: OpenStreetMap Wiki. `http://wiki.openstreetmap.org/wiki/Main\_Page` (2013) [Online; accessed Dec 2013].