An Algorithm for Map Matching Given Incomplete Road Data

Jan-Henrik Haunert
Universität Würzburg
jan.haunert@uni-wuerzburg.de

Benedikt Budig
Universität Würzburg
info@benedikt-budig.de

ABSTRACT

We consider the problem of matching a GPS trajectory with a road data set in which some roads are missing. To solve this problem, we extend a map-matching algorithm by Newson and Krumm (Proc. ACM GIS 2009, pp. 336–343) that is based on a hidden Markov model and a discrete set of candidate matches for each point of the trajectory. We introduce an additional off-road candidate for each point of the trajectory. The output path becomes determined by selecting one candidate for each point of the trajectory and connecting the selected candidates via shortest paths, which preferably lie in the road network but, if off-road candidates become selected, may also include off-road sections. We discuss experiments with GPS tracks of pedestrians.

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*

General Terms

Algorithms

Keywords

GIS, map matching, dynamic programming, hidden Markov model

1. MOTIVATION

Many users record trajectories (that is, sequences of points with time stamps) with smart phones or small computing devices. Usually, the positioning is done with GPS, thus we term a point of a trajectory a *GPS point*, but also other sensors may be used. *Map matching* is the problem of finding the path in the road network corresponding to the trajectory. In this paper, we address *off-line map matching*, meaning that we require the entire trajectory as input.

Preprint of paper published in Proc. ACM GIS'12.

While several authors have focused on off-line map-matching algorithms for low-quality GPS trajectories [5, 3] the road data is usually assumed to be complete and accurate. In the following situations, however, the road data is incomplete:

- Some roads have not been mapped yet.
- Minor roads have not been included in the data or have been removed by map generalization.
- The road data has been acquired for a certain means of transportation (e.g., cars) but the user who collected the GPS data used another means (e.g., a bicycle).
- The road data contains topological errors.

Map-matching algorithms often search for a path in the road network that has a favorable global property, for example, a path having minimum distance to the trajectory (according to some distance function). Requiring the output path to be contained in the road network, however, is prone to failure if the road network is incomplete. We therefore suggest a new map-matching algorithm that introduces new (off-road) edges whenever they are needed to avoid that the output path deviates too much from the input trajectory.

When the road data is complete, map matching can be done based on shortest paths between candidate matches [5, 3, 2]. We discuss this approach in Sect. 2 and extend it to handle incomplete road data in Sect. 3. We then present our experiments (Sect. 4) and conclude the paper (Sect. 5).

2. RELATED WORK

For an overview on the literature on map matching we refer to Quddus et al. [7]. For off-line map matching on incomplete road data, Pereira et al. [6] have proposed to use genetic algorithms, that is, a general-purpose heuristic that is typically applied to NP-hard optimization problems. In contrast, we will model the map-matching problem such that it can be solved both efficiently and exactly.

Lou et al. [3] as well as Newson and Krumm [5], whose approach we extend in this paper, assume that the road network is represented as a straight-line graph $G = (V, E)$ and the output path P is contained in G. The path P may start (or end) at a vertex of G or in the interior of an edge. Furthermore, the following assumption is made.

Assumption 1. *Between two consecutive measurements of GPS points, a user travels on a path of minimum cost (termed a* shortest *path) in the road network.*

We here define the cost of a path in G based on a weight $w(e)$ for each edge $e \in E$ that represents the cost of traveling one geometric unit on e. This cost may represent geometric distances, but also other measures can be applied.

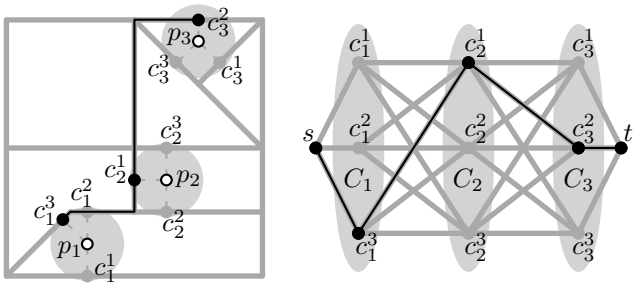


Figure 1: The graph G (left, dark gray) with the candidate matches for the trajectory (p_1, p_2, p_3) and the candidate graph G_{cand} (right). The paths drawn black in G and in G_{cand} correspond to each other.

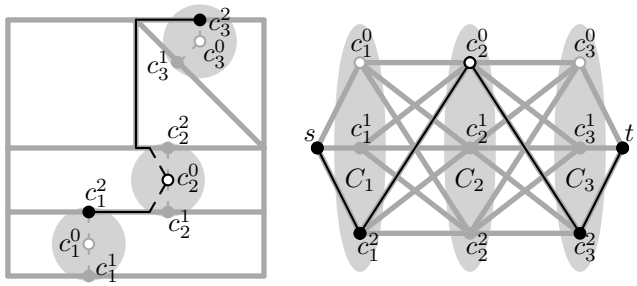


Figure 2: A road network G (left) with three GPS points and their candidate matches, including one off-road candidate for each GPS point, and the corresponding candidate graph G_{cand} (right). The two black paths correspond to each other.

For each point p_i of the trajectory $T = (p_1, \dots, p_n)$, Lou et al. [3] and Newson and Krumm [5] define a discrete set $C_i = \{c_i^1, \dots, c_i^{k_i}\}$ of candidate matches, where k_i is the cardinality of C_i. Let D_i be the disk with a user-set radius r centered at GPS point p_i. For each edge $e \in E$ that intersects D_i, the set C_i is defined to contain the point on e that is closest to p_i. Hence, the maximal size k of a set C_i is in $\mathcal{O}(|E|)$. The output path is generated by selecting one candidate match for each GPS point and, for each two consecutive GPS points, connecting the two selected matches via a shortest path in G, see Fig. 1 (left).

To find a path that matches the trajectory best, Newson and Krumm [5] use a hidden Markov model (HMM). This generally describes a sequence of states of a system that is being observed [8]. In our application, the set of possible states (that is, candidate matches) is discrete and the set of possible observations (that is, GPS points) continuous. The state sequence is defined based on a Markov process, that is, the state at a certain time only depends on the previous state and a *transition probability* $\Pr(t \mid s \text{ was before})$ for each two states s, t (that is, the probability that, given s, the system will next enter t). For each time step, an observation is given and, for each state s and each possible observation o, the *observation probability density* $f(o \mid s)$ is known (that is, the probability density that, given s, o will be observed). Using Bayes' rule, the probability of a state sequence $S = (s_1, \dots, s_n)$ under a given sequence $O = (o_1, \dots, o_n)$ of observations becomes

$$\Pr(S \mid O) = f(O \mid S) \cdot \Pr(S)/f(O)\,, \tag{1}$$

where $f(O \mid S) = f(o_1 \mid s_1) \cdot \ldots \cdot f(o_n \mid s_n)$ and $\Pr(S) = \Pr(s_1) \cdot \Pr(s_2 \mid s_1 \text{ was before}) \cdot \ldots \cdot \Pr(s_n \mid s_{n-1} \text{ was before})$. If not stated otherwise, we assume that the a-priori probability $\Pr(s_1)$ of s_1 is the same for each state s_1.

To find the state sequence that is most likely, a directed acyclic graph $G_{\text{cand}} = (C, A)$ is defined, see Fig. 1 (right). This graph contains a vertex c_i^v for each time step i and each possible state v at time i plus two vertices s and t. Furthermore, G_{cand} contains an arc from s to every vertex in C_1, an arc from every vertex in C_n to t, and, for $i = 1, \dots, n-1$, an arc between every vertex in C_i and every vertex in C_{i+1}. The arc weights are defined by

$$W(sc_1^v) = \Pr(c_1^v) \cdot f(o_1 \mid c_1^v) \tag{2}$$

$$W(c_{i-1}^u c_i^v) = \Pr(c_i^v \mid c_{i-1}^u \text{ was before}) \cdot f(o_i \mid c_i^v) \tag{3}$$

$$W(c_n^u t) = 1. \tag{4}$$

With this definition, the product of arc weights of an s-t-path in G_{cand} (that is, a state sequence S) is equal to $\Pr(S) \cdot f(O \mid S) = \Pr(S \mid O) \cdot f(O)$. Therefore, and since $f(O)$ is constant for all state sequences, a state sequence S that maximizes the product of arc weights maximizes $\Pr(S \mid O)$ and thus is the most likely explanation for the sequence O of observations. Such a path can be found in $\mathcal{O}(|C| + |A|)$ time with a dynamic-programming approach, which in the context of HMMs is known as the *Viterbi algorithm* [8].

In our map-matching application, an observation is given by a GPS point $p_i \in \mathbb{R}^2$. Lou et al. [3] as well as Newson and Krumm [5] assume that a GPS measurement follows a normal distribution around the true position. Furthermore, by analyzing ground-truth data, Newson and Krumm [5] have found out that, for two consecutive GPS points p_{i-1}, p_i, the absolute difference between the Euclidean distance $\delta_i := d_{\text{E}}(p_{i-1}, p_i)$ and the geometric length $d_{\text{G}}(u, v)$ of the shortest path between the two corresponding correct matches u and v follows an exponential probability distribution. Based on this result, they express the transition probability densities, which implies that a shortest path between all pairs of candidate matches that belong to two consecutive GPS points needs to be found. This requires $\mathcal{O}(kn)$ runs of Dijkstra's algorithm [1] and thus $\mathcal{O}(kn \cdot |E| + kn \cdot |V| \log |V|)$ time in total.

3. OUR ALGORITHM

Our idea for handling incomplete data is to allow each GPS point p_i to become matched with a special (off-road) candidate c_i^0, see Fig. 2. We fix the position of c_i^0 at GPS point p_i. The set C_i of candidate matches for p_i becomes $C_i = \{c_i^0, c_i^1, \dots c_i^{k_i}\}$, where k_i is now the number of *on-road* candidates for p_i. We make the following assumption.

Assumption 2. *Between two consecutive measurements of GPS points, a user changes at most once between a road contained and a road not contained in the data or vice versa.*

To define the path P between two candidate matches c_i^u and c_{i+1}^v, we distinguish three cases.

(C1) If $u \neq 0$ and $v \neq 0$, that is, if both c_i^u and c_{i+1}^v are *on-road* candidates, Assumption 2 implies that we have to remain in the road network (we must not leave and re-enter it). In this case, we can apply Assumption 1, which implies that P is a minimum-cost path in G.

(C2) If $u = v = 0$, that is, if both c_i^u and c_{i+1}^v are *off-road* candidates, we simply define P as the straight-line segment connecting c_i^u and c_{i+1}^v.

(C3) If exactly one of the candidates c_i^u and c_{i+1}^v is an off-road candidate, Assumption 2 implies that P is the concatenation of a connected off-road path and a connected on-road path. The off-road path should be geometrically short and the on-road path should be short with respect to the edge weights. Defining a cost w' for traveling one geometric unit outside of the road network allows us to define P as a minimum-cost combination of an off-road and an on-road path.

Note that in case (C3) the off-road section of P is always a straight-line segment s between a point p in G and a GPS point, where p either is a vertex of G or lies on an edge e of G. In the latter case, the segment s and the edge e form a *critical angle* that only depends on w' and $w(e)$ [4].

To define the arc weights in G_{cand}, we extend the original HMM-based approach. We assume that we have two types of observations, namely the GPS points $p_1, \ldots, p_n$ and, for $i = 2, 3, \ldots, n$, the (derived) distance δ_i between p_{i-1} and p_i. The probability density $f(p_i \mid c_i^u)$ of GPS point p_i is defined with a normal distribution centered at c_i^u and, since Newson and Krumm [5] found out that δ_i follows an exponential probability distribution that is maximal for $\delta_i = d_{\text{G}}(c_{i-1}^u, c_i^v)$, we define a parameter $\beta \in \mathbb{R}^+$ and

$$f(\delta_i \mid c_{i-1}^u, c_i^v) = \frac{1}{2\beta} e^{-\frac{|\delta_i - d_{\text{G}}(c_{i-1}^u, c_i^v)|}{\beta}}. \tag{5}$$

To express our a-priori knowledge of the fact that the path of the user is very likely to be contained in the road data and that a missing link is rather an exception, we define non-constant transition probabilities. If GPS point p_{i-1} corresponds to an *on-road* candidate c_{i-1}^u, we assume that the probability of each transition to an on-road candidate c_i^v is by a constant factor of φ higher than the probability of a transition to the off-road candidate c_i^0. Similarly, given that GPS point p_{i-1} corresponds to the *off-road* candidate c_{i-1}^0, we assume that the probability of each transition to an on-road candidate c_i^v is by a constant factor of ψ higher than the probability of a transition to c_i^0. Formally, this means

$$\Pr(c_i^v \mid c_{i-1}^u \text{ was before}) = \begin{cases} \frac{1}{\varphi k_i + 1} & \text{if } u \neq 0, v = 0 \\ \frac{\varphi}{\varphi k_i + 1} & \text{if } u \neq 0, v \neq 0 \\ \frac{1}{\psi k_i + 1} & \text{if } u = 0, v = 0 \\ \frac{\psi}{\psi k_i + 1} & \text{if } u = 0, v \neq 0 \end{cases}. \tag{6}$$

To keep our model simple, we define $\Pr(c_1^u)$ to be equal to the transition probability of moving from a (hypothetical) on-road candidate to c_1^u.

Similar to Equations (2)–(4), we now define the arc weights

$$W(sc_1^v) = \Pr(c_1^v) \cdot f(p_1 \mid c_1^v) \tag{7}$$

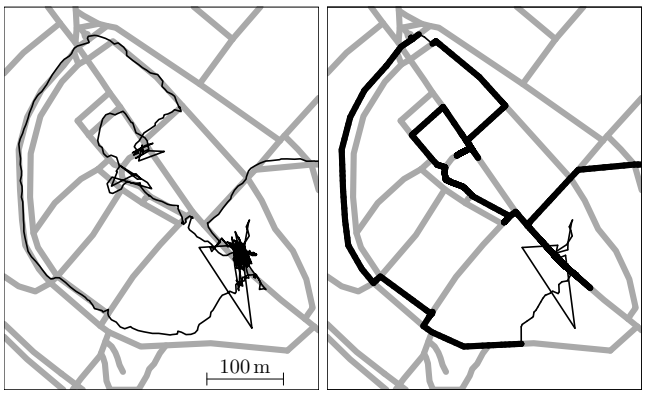
$$W(c_{i-1}^u c_i^v) = \Pr(c_i^v \mid c_{i-1}^u \text{ was before}) \cdot f(p_i \mid c_i^v) \cdot f(\delta_i \mid c_{i-1}^u, c_i^v) \tag{8}$$

$$W(c_n^u t) = 1\,. \tag{9}$$

With this setting, an s-t path in G_{cand} that maximizes the product of arc weights is indeed a most likely state sequence.

To compute the paths in G associated with the arcs in G_{cand}, we execute Dijkstra's algorithm [1] for each candidate match c_i^u with $i < n$ as source. That is, we expand a shortest-path tree from c_i^u until it contains all candidate matches in C_{i+1}. To compute shortest paths to or from off-road candidates, we could augment G, for each edge $e \in E$, with at most two off-road edges that form a critical angle with e and, for each vertex in V, with one off-road edge. Then, we could execute Dijkstra's algorithm on the augmented graph. Note however, that explicitly adding all possible off-road edges can be avoided.

With our extension the asymptotic running time of the method by Newson and Krumm [5] remains unchanged. To speed up our method, we use a heuristic that has also been used by Newson and Krumm. That is, if the paths become so long that $f(\delta_i \mid c_{i-1}^u, c_i^v)$ for every unfinished target c_i^v would fall below a certain threshold, we terminate the expansion of the shortest-path tree from c_{i-1}^u and set $f(\delta_i \mid c_{i-1}^u, c_i^v) = 0$.



(a) roads (gray) and trajectory (black) (b) output path (black, off-road parts thin)

Figure 3: A sample from our test data.

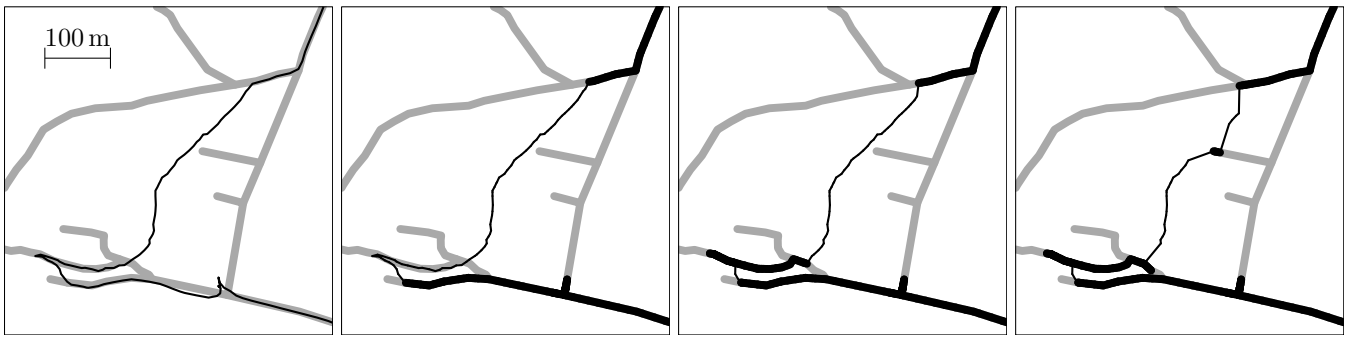
4. EXPERIMENTAL RESULTS

We implemented our algorithm in Java and tested it with OpenStreetMap data that was provided on-line[1] and trajectories that we recorded with a GPS receiver of type Garmin Edge® 710 during four hikes in the surroundings of Würzburg, Germany. In total, the GPS receiver yielded 7799 GPS points over a distance of 65 km. Figure 3(a) shows the end of Track2 in a small medieval town, where the GPS receiver produced several outliers when we visited a restaurant. Nevertheless, we did not filter the trajectories.

We weighted the edges of the road network according to their geometric lengths and tested the method with different parameter sets. The parameters in Table 1 overall yielded the best results, see Fig. 3(b) for the results for the sample in Fig. 3(a). The outliers did not prevent our method from matching most parts of the trajectory correctly. Moreover, the part containing the outliers plus two sections where we indeed walked on unmapped paths did not become matched and were correctly classified as off-road sections. In total, our solutions for the four tracks contained 23 off-road sections, which all were correct.

Figure 4 shows results with different values for ψ and φ. In Fig. 4(b), the probability of transitions to off-road candidates was high, thus a small on-road section of the trajectory that lies between two off-road parts has not become matched

[1] download.geofabrik.de/osm/



(a) map and trajectory (b) $\varphi = 6.8$, $\psi = 1.02$ (c) $\varphi = 10$, $\psi = 1.5$ (d) $\varphi = 1000$, $\psi = 150$

Figure 4: A sample where two connections are missing in the map. By choosing φ and ψ it is possible to control to what degree the path becomes attracted by the roads ((b)–(d)). The solution with our default setting (c) reflects the situation best.

description	*symbol*	*value*
radius for selection of candidate points	r	40 m
standard deviation of GPS coordinates	σ	25 m
cost of an off-road edge of one unit	w'	1.5
parameter for probability density of distance measurements	β	20.0
parameter for transition probabilities from off-road candidates	ψ	1.5
parameter for transition probabilities from on-road candidates	φ	10.0

Table 1: Default parameters used in our experiments.

	Track1	Track2	Track3	Track4
number of points	1969	2613	2116	1101
time to solution	14.5 s	51.1 s	7.7 s	47.2 s
number of points	196	261	211	110
time to solution	0.8 s	1.4 s	0.4 s	2.5 s

Table 2: Computation times when using all GPS points (rows 2–3) and every tenth point (rows 4–5). All tests were performed on a Windows PC with 3 GB of RAM and a 3.00 GHz Intel dual-core CPU; s stands for seconds CPU time.

with the road network. In contrast, when using our default setting, the result indeed contains two off-road sections with a short on-road section in between, see Fig. 4(c). Finally, when setting ψ and φ to high values, the trajectory becomes attracted too much by the road data, see Fig. 4(d).

Table 2 shows the running time of our method. We did not tune our implementation to achieve a high efficiency and did not restrict the size of the candidate set of a GPS point. Therefore, our method needed relatively long to process Track2 and Track4, which covered dense urban areas. By reducing the sampling rate of the trajectories we decreased the running time drastically but still obtained good results.

5. CONCLUSION AND FUTURE WORK

We conclude that, overall, our algorithm copes well with edges missing in the road network. In terms of the asymptotic worst-case running time, we have not added complexity to the method by Newson and Krumm [5]. We have added, however, the parameters ψ and φ to define the transition probabilities between off-road and on-road candidates and the parameter w' by which we determine where the path between an on-road and an off-road candidate leaves (or enters) the road network. Future research will deal with automatic parameter training and with on-line map matching.

6. REFERENCES

[1] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[2] J. Eisner, S. Funke, A. Herbst, A. Spillner, and S. Storandt. Algorithms for matching and predicting trajectories. In *Proc. Workshop Algorithm Engineering and Experiments (ALENEX '11)*, pages 84–95. SIAM, 2011.

[3] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate GPS trajectories. In *Proc. 17th ACM SIGSPATIAL Internat. Conf. Advances in Geographic Information Systems (GIS '09)*, pages 352–361. ACM, 2009.

[4] J. S. B. Mitchell and C. H. Papadimitriou. The weighted region problem: finding shortest paths through a weighted planar subdivision. *J. ACM*, 38(1):18–73, 1991.

[5] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. In *Proc. 17th ACM SIGSPATIAL Internat. Conf. Advances in Geographic Information Systems (GIS '09)*, pages 336–343. ACM, 2009.

[6] F. C. Pereira, H. Costa, and N. M. Pereira. An off-line map-matching algorithm for incomplete map databases. *European Transport Research Review*, 1(3):107–124, 2009.

[7] 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*, 15(5):312–328, 2007.

[8] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, pages 4–15, January 1986.