Sparse Map-Matching in Public Transit Networks with Turn Restrictions

Hannah Bast
University of Freiburg
Freiburg, Germany
bast@cs.uni-freiburg.de

Patrick Brosi
University of Freiburg
Freiburg, Germany
brosi@cs.uni-freiburg.de

ABSTRACT

We investigate the following map-matching problem: given a sequence of stations taken by a public transit vehicle and given the underlying network, find the most likely geographical course taken by that vehicle. We provide a new algorithm and tool, which is based on a hidden Markov model and takes characteristics of transit networks into account. Our tool can be useful for the visualization of transit lines in map services, for transit data providers, and for an on-line matching of live passenger GPS data to a public transit vehicle. We evaluate our tool on real-world data, and compare it against two baselines. The shapes produced by our tool are very close to the true shapes. We have made our software publicly available, enabling full reproducibility of our results.

CCS CONCEPTS

• **Information systems** → **Geographic information systems**;
• **Theory of computation** → **Routing and network design problems**;

KEYWORDS

Public Transit, Map-Matching, Schedule Data, GTFS

ACM Reference Format:
Hannah Bast and Patrick Brosi. 2018. Sparse Map-Matching in Public Transit Networks with Turn Restrictions. In *26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '18), November 6–9, 2018, Seattle, WA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3274895.3274957

1 INTRODUCTION

Most map services offer an optional transit layer showing an area's transit lines on top of the regular map data. Ideally, the actual geographical course of these lines is shown. However, for many areas they are shown only as straight-line connections (Figure 1). In principle, the geographical course of a transit line is part of GTFS (General Transit Feed Specification), the de facto standard format for transit data [4]. However, this *shape* data, as it is called in GTFS, may be missing for some feeds.

The goal of this paper is to automatically compute accurate shape data from the following information: the sequence of the (possibly imprecise) station positions, the geography of the underlying network and optional metadata about the network. These shapes can then be used as a substitute for missing or inaccurate shape data in a GTFS feed.

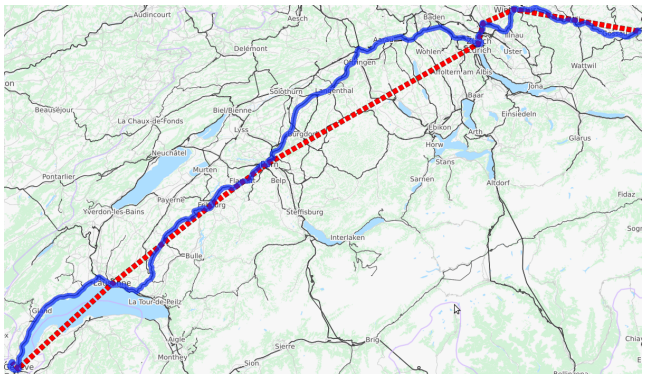


Figure 1: Red: Straight-line paths between stops of a Swiss train, taken from schedule data. Blue: correct geographical path found by *pfaedle*.

1.1 Related Work

Our work is closely related to map-matching of GPS trajectories (given a sequence of positions, determine the most likely path through a given network). Like our approach, recent work is based on a hidden Markov model (see [8] for an overview).

The main difference between our work and existing approaches is the notion of sparseness. For the latter, a sampling rate > 30 seconds may already be considered sparse [6]. In our case, we frequently have to match data where the average distance between sample points may be up to several hundred kilometers. Another difference lies in the variable type of the underlying network. While previous work was mostly limited to road networks, map-matching on rail networks is commonly required in a public transit scenario.

Only little applied work on map-matching of schedule data exists. In [2], the authors describe a greedy approach based on an iterative shortest-path search. A global approach based on the construction of a "pseudo-graph" which resembles a hidden Markov model was described in [7] and evaluated against the schedule data of Zurich on an OpenStreetMap (OSM) network. (Inter-hop) turn restrictions and a broader use of OSM attributes were left as an open problem.

Our solution to inter-hop turn restrictions is based on previous work in [3]. An edge-based Dijkstra's algorithm was described to improve time and space efficiency of routing in road networks.

For our evaluation, we build on metrics for map-matching quality previously described in the literature. A similarity metric between curves based on the average Fréchet distance was first used as a

SIGSPATIAL '18, November 6–9, 2018, Seattle, WA, USA

ACM ISBN 978-1-4503-5889-7/18/11.
https://doi.org/10.1145/3274895.3274957

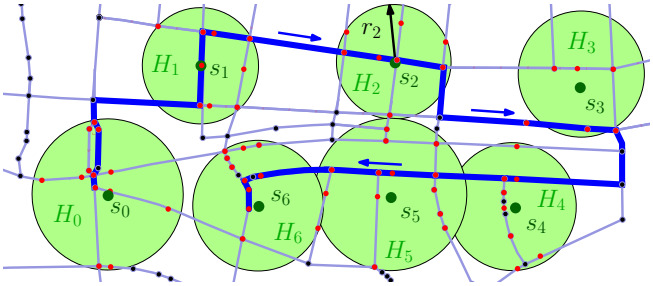


Figure 2: Station positions $s_0, ..., s_6$ from a GTFS feed (possibly imprecise) of a bus route through a road network. Red: candidate nodes within a radius r_i around each s_i. Blue: the correct path of the bus.

quality metric for map-matching results in [1]. In [5], two additional metrics A_N and A_L based on the number and length of incorrectly matched segments were introduced. We use A_N in Section 5.

2 BASIC APPROACH

Let us denote by $S = (s_0, s_1, ..., s_n)$ the sequence of station positions, and by $G = (V, E)$ the graph of the underlying network. This section describes how we find the node candidates for each s_i (Section 2.1), how we find the most likely sequence of node candidates using a hidden Markov model (Section 2.2), and how we obtain the most likely path through G between those selected candidates (Section 2.2.3).

2.1 Finding Node Candidates

In a first step, we collect sets of possible node candidates $H_i \subseteq V$ for each s_i. We identify all edges $e = (u, v) \in E$ that lie within a radius r_i around s_i. At the projection of s_i on e, we insert a candidate node. H_i is the set of these candidate nodes. In our experiments, we set all $r_i = r_t$, where t is the vehicle type pertaining to S (see Table 2).

2.2 Optimal Candidate Sequences

Given the candidate nodes H_i for each sample point s_i, we now want to find the most likely candidate $h_i \in H_i$ for each s_i.

A simple approach is to greedily compute the $h_0, ..., h_n$ one after the other: having computed h_i, take h_{i+1} as that node from H_{i+1} with the shortest path from h_i. This algorithm may be refined by searching the shortest path from *any* node candidate $h_{i'} \in H_i$ to any $h_{i+1} \in H_{i+1}$. Then, an additional step to connect the hop segments (h_{i-1}, h_i) and $(h_{i'}, h_{i+1})$ is necessary to ensure a continuous path. A simple solution is to just run another shortest-path search between h_i and $h_{i'}$ and fall back to a straight line if no such path exists.

A better approach to map-matching is to use a hidden Markov model (HMM). The station positions s_i now act as *observations*, and the node candidates H_i as *hidden states*. For each H_i and each candidate node $h_k^i \in H_i$, an emission probability distribution $\Pr(s_i|h_i^k)$ is defined which gives the likelihood that h_i^k is a matching candidate given that we observed s_i. Additionally, for each h_i^k, the probability that the next state will be some $h_{i+1}^{k'}$ is given by $\Pr(h_i^k \to h_{i+1}^{k'})$. To incorporate many weight functions into our approach, we model our transition probability based on a set $W_\mathcal{T}$ of weight functions

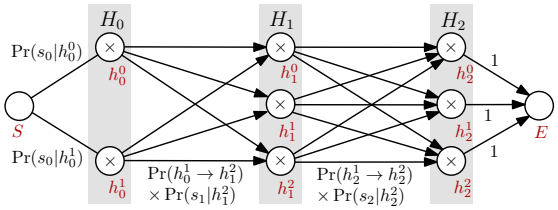


Figure 3: Hidden Markov model for finding the most likely node candidate sequence through 3 layers H_0, H_1 and H_2.

$w_\tau^1, w_\tau^2, ..., w_\tau^m$, all following an exponential distribution:

$$\Pr(h_i \to h_{i+1}) = \prod_{j=1}^{|W_\mathcal{T}|} e^{-\lambda_j w_\tau^j(h_i, h_{i+1})}. \tag{1}$$

Emission probabilities are based on weight functions $w_\epsilon \in W_\mathcal{E}$:

$$\Pr(s_i|h_i) = \prod_{j=1}^{|W_\mathcal{E}|} e^{-\beta_j w_\epsilon^j(s_{i+1}, h_{i+1})}. \tag{2}$$

The goal is then to find the hidden state sequence $h_0, h_1, ..., h_n$ with the maximum likelihood

$$\max_{h_0,...,h_n} \prod_{i=0}^{n-1} \Pr(h_i \to h_{i+1}) \cdot \Pr(s_{i+1}|h_{i+1}) \tag{3}$$

$$= \min_{h_0,...,h_n} \sum_{i=0}^{n-1} \Bigg[\underbrace{\sum_{j=1}^{W_\mathcal{T}} \lambda_j w_\tau^j(h_i, h_{i+1})}_{=c_\tau(h_i, h_{i+1})} + \underbrace{\sum_{j=1}^{W_\mathcal{E}} \beta_j w_\epsilon^j(s_{i+1}, h_{i+1})}_{=c_\epsilon(s_{i+1}, h_{i+1})} \Bigg] \tag{4}$$

which can be found by doing a shortest-path search on the transition DAG with edge costs $c(h_i, h_{i+1}) = c_\tau(h_i, h_{i+1}) + c_\epsilon(s_{i+1}, h_{i+1})$.

2.2.1 Transition Weights. In the basic version of our approach, $c_\tau(h_i, h_{i+1})$ consists of two transition weights $w_\tau(h_i, h_{i+1})$: (1) The network distance in meters $w_\tau^d(h_i, h_{i+1})$. (2) The number of stations passed without stop $w_\tau^s(h_i, h_{i+1})$. Both costs are weighted by λ_d and λ_s respectively and calculated by finding the path through G from h_i to h_{i+1} that optimizes the transition cost function

$$c_\tau(h_i, h_{i+1}) = \lambda_d w_\tau^d(h_i, h_{i+1}) + \lambda_s w_\tau^s(h_i, h_{i+1}) \tag{5}$$

which can be found by doing a shortest path search (we use an ordinary A^* algorithm) through G. The intuition behind w_τ^d is that vehicles will most likely use the next node candidate h_{i+1} that is *nearest* (in terms of network distance) to h_i. With w_τ^s, we avoid paths that pass through many stations without stopping there.

2.2.2 Emission Weights. In the basic version, our approach uses a single emission weight $w_\epsilon^d(s_i, h_i)$: the great-circle distance between s_i and node candidate h_i. The emission cost function is

$$c_\epsilon(s_i, h_i) = \beta_d w_\epsilon^d(s_i, h_i). \tag{6}$$

We describe additional weights based on attribute similarities between the input geo data and the schedule data in Section 4.

2.2.3 Path Guessing. As we computed the optimal paths between layers with respect to our w_τ during the calculation of the transition weights (Sect. 2.2.1), connecting the optimal node candidate sequence is trivial. We can just combine the shortest paths found by our A^* runs into the final result path P.

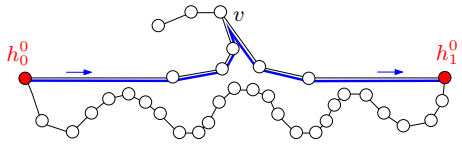


Figure 4: Blue: most likely transition from h_0^0 to h_1^0 with distance-based cost. For trains, the lower path is more likely.

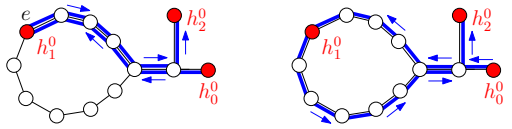


Figure 5: Map-matching without inter-hop turn restrictions (left), and with inter-hop turn restrictions (right).

3 TURN RESTRICTIONS

Turn restrictions are valuable information for calculating the transition probability. A transition from a candidate node h_0^0 to h_1^0 may be impossible in the real world because traffic rules forbid making a turn required to reach h_1^0. We would therefore like to add a transition weight w_τ^t that punishes those turns.

In rail networks, we have to respect turn restrictions resulting from physical limitations. For example, a train does not usually make a 180° turn on open track (Fig. 4). To address this, we would like to add a transition weight w_τ^f punishing these full turns.

In map-matching, a third problem appears: inter-hop turn restrictions. Consider Figure 5. Even if we respect turn-restrictions when calculating the transition probabilities from h_1^0 to h_2^0, the blue route on the left will be the most likely - but the vehicle has to make an unnecessary full turn.

To address these issues, we transform the network graph $G = (V, E)$ into its edge-to-vertex dual $L(G) = (V', E')$. Each edge $(u, v) \in E$ is represented by $v'_{u,v} \in V'$, and a node $v_j \in V$ is represented by edges $e_{(u,v_j),(v_j,w)} \in E'$ (Fig. 6). Weights in G' now model costs between edges in G, which enables us to directly punish turn restrictions in Dijkstra's algorithm [3].

To finally enable turn restrictions between hops, instead of choosing orientationless node candidates $h_i^k \in H_i \subset V$ for each sample point s_i, we now use *all* outgoing edges of each h_i^k as hidden states in our HMM (Fig. 7). Intuitively, this means we now have hidden states for each possible vehicle orientation at h_i^k.

4 INCORPORATING OSM METADATA

Our approach described so far can be used on arbitrary transit network datasets which contain suitable geographical information. *Pfaedle* uses OpenStreetMap (OSM) data as the default geo data

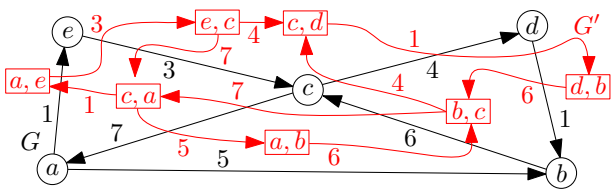


Figure 6: Black: Weighted network graph G. Red: edge-to-vertex dual graph G' of G.

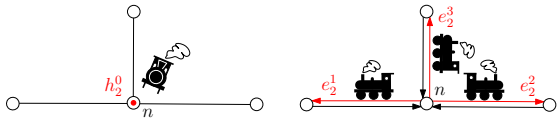


Figure 7: Left: A single hidden state h_2^0 is used to model the orientationless arrival at node n. Right: each outgoing edge of n is used explicitly as a hidden state e_i^k.

input format. The data is filtered per vehicle type using manually compiled rules. In this section, we describe how to incorporate information present in the OSM data into our approach to enhance the overall map-matching result.

4.1 Augmenting Transition Probabilities

The following information contained in the OSM data can naturally be used directly in our approach.

4.1.1 Turn-Restriction Relations. The relation `restriction` contains many types of edge-to-edge turn restrictions. *Pfaedle* uses most of them in the approach described in Section 3 and respects specific exceptions, for example turns that are specifically allowed for public transit vehicles.

4.1.2 One-Way Streets. One way streets can be incorporated by setting the corresponding edge cost in our network graph to ∞.

4.1.3 Edge Levels. We make use of the fine-grained hierarchy of `way` objects in OSM by using 5 different transition weights $w_\tau^{d_0}, ..., w_\tau^{d_5}$ instead of a single w_τ^d in our cost function (Eq. 5). The weight factors $\lambda_{d_0} ... \lambda_{d_5}$ are chosen to punish roads of higher levels.

4.1.4 Public Transit Routes. OSM data already contains extensive information regarding the routes of public transit vehicles in `route` relations. To use this information into our approach, we add a transition cost w_τ^L that is calculated as the sum of the length of all network segments that do not have a matching `route` relation.

4.2 Augmenting Emission Probabilities

In addition to improving the transitions weights with attributes of the OSM data, several attributes can be used to also make the emission weights more precise.

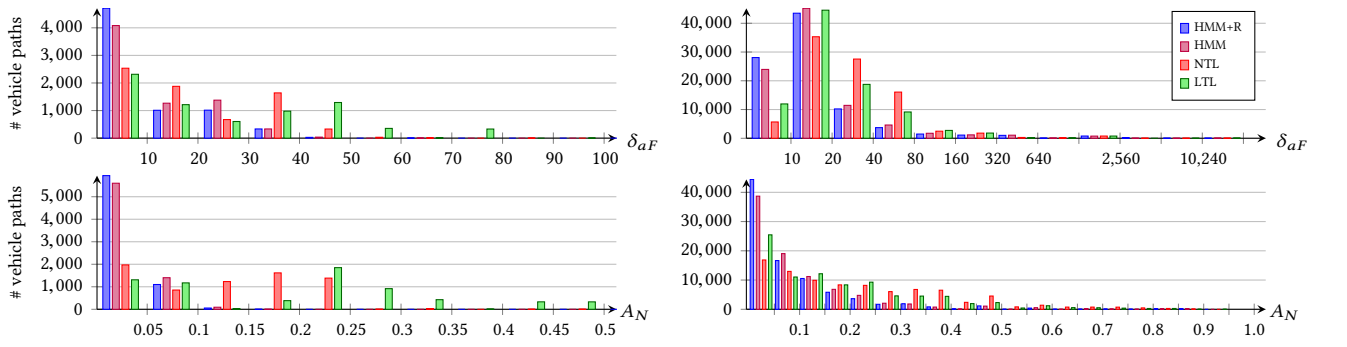
4.2.1 Station Labels. OSM already contains nodes tagged as stations. To use this, we extend our node candidate search by first identifying all nodes within radius r_i around the sample point s_i (with station label $n(s_i)$) that are marked as stations in OSM (with station label $n(h_i^k)$). To measure the similarity between two station labels a and b, we define the following token-subset edit distance:

$$\mathrm{TED}(a, b) = \min \left[\min_{A' \in \mathcal{P}(T(a))} \mathrm{lev}(|A'|, b), \min_{B' \in \mathcal{P}(T(b))} \mathrm{lev}(|B'|, a) \right],$$

where $\mathrm{lev}(a, b)$ is the Levenshtein distance, $T(a)$ is the set of white-space separated tokens in a and $|A'|$ is the space-separated concatenation of tokens in A'. With this, we add an emission weight

$$w_\epsilon^n(s_i, h_i^k) = \frac{TED\left(n(s_i), n(h_i^k)\right)}{\max\left(|n(s_i)|, \left|n(h_i^k)\right|\right)}. \tag{7}$$

4.2.2 Track Numbers. Stations in OSM may also contain a `ref` attribute specifying the track number. To favor node candidates with a matching track, we define a track label function $t(n)$ and add a weight $w_\epsilon^n(s_i, h_i^k)$ which is 1 if $t(h_i^k) = t(s_i)$, or else 0.



(a) Vitoria-Gasteiz (b) Stuttgart

Figure 8: Evaluation results of our approach (HMM / HMM+R) and our two baselines (NTL / LTL).

5 EVALUATION

We evaluated *pfaedle* on the GTFS feeds of Stuttgart (S), Vitoria-Gasteiz (VG), Switzerland (CH), and the Paris/Île-de-France region (P). Table 1 shows the dimensions of these feeds. Table 2 shows the parameter values of *pfaedle*. S and VG were selected because of the high-quality shape data, which we took as ground truth in our evaluation. CH and P were selected because of their size, they do not have good shape data. The shapes computed by *pfaedle* for all four feeds are available online under http://travic.cs.uni-freiburg.de.

Figure 8 shows the results of our evaluation for S and VG. We used two metrics to compare the path P computed by *pfaedle* to the corresponding path Q from the ground truth. The first metric is the average Fréchet Distance δ_{aF} of P and Q (divided into an equal number of segments of length 1 m in P). The second metric is the percentage A_N of hop (between-station) segments with Fréchet Distance $\geq$ 20 m. We evaluated four approaches: our approach using OSM route metadata (HMM+R), our approach without this metadata (HMM), the greedy 1-to-layer approach from Section 2.2 (NTL) and the layer-to-layer approach from the same section (LTL).

Both HMM approaches clearly outperform both baselines. Most paths deviate less than 20 m from the ground truth, which we consider an almost perfect match (below 20 m, inconsistencies between the ground truth and the OSM data become noticeable). Surprisingly, the additional use of OSM metadata leads to only minor improvements (HMM+R vs. HMM). For the Stuttgart data, the spike of NTL and LTL at $10 \leq \delta_{aF} \leq 20$ is caused by the high percentage of regional trains in this dataset. For trains, both baseline approaches work particularly well, because the number of node candidates per station is usually low.

Table 3 shows the running times of our approach HMM+R versus LTL on an Intel Core i7-2600 3.40 GHz machine with 32 GB RAM.

6 CONCLUSIONS AND FUTURE WORK

We presented a novel approach for automatically computing missing shape data in GTFS feeds with very high accuracy. Unlike previous approaches, we take critical characteristics of transit data into account. Our approach is implemented in a publicly available tool, called *pfaedle*. Its running time is sufficiently fast for practical use, but slower than that of simpler baselines.

Our work so far does not use the temporal information available in GTFS feeds. This information may be valuable, however, for filtering out unlikely paths.

Table 1: Dataset dimensions. $|\mathcal{T}|$ is the number of trips in the schedule data, ϕ is the average straight-line distance between stations in the schedule data in meters. $|V|$ is the number of nodes, $|E|$ the number of edges in the OSM data.

	Bus				Tram+Subway				Rail			
	$\|\mathcal{T}\|$	ϕ	$\|V\|$	$\|E\|$	$\|\mathcal{T}\|$	ϕ	$\|V\|$	$\|E\|$	$\|\mathcal{T}\|$	ϕ	$\|V\|$	$\|E\|$
VG	6k	380	5.6k	14.5k	1.3k	400	83	196	—	—	—	—
S	53k	833	156k	409k	14k	594	1.5k	3.4k	23k	6.2k	5.7k	14.8k
CH	395k	2.1k	0.5M	1.4M	60k	2.1k	3.5k	8k	81k	8.7k	153k	0.4M
P	260k	1.1k	346k	0.9M	57.6k	1.1k	14.3k	38.8k	13k	6.3k	14.4k	38.8k

Table 2: Manually chosen parameters for our HMM and HMM+R approaches.

	λ_{d_0}	λ_{d_1}	λ_{d_2}	λ_{d_3}	λ_{d_4}	λ_{d_5}	λ_s	λ_f	λ_t	λ_L	β_d	β_n	β_t	r_t
Rail	1	1.25	1.5	2	2.5	3.5	100	3k	1	1	3	100	2k	200
Subw.	1	1.5	2	2.5	3.5	5	100	2k	1	0.5	3	235	0	100
Tram	1	1.5	2	2.5	3.5	5	100	2k	1	0.5	3	235	0	100
Bus	1	1.25	1.5	1.75	2.25	3	0	500	5k	1.75	2.5	500	0	100

Table 3: Running times of HMM+R vs. LTL. HMM times are similar to HMM+R, NTL times are similar to LTL.

	Bus		Tram		Subway		Rail		∑	
	HMM+R	LTL	HMM+R	LTL	HMM+R	LTL	HMM+R	LTL	**HMM+R**	LTL
VG	1.2s	18ms	9ms	3ms	—	—	—	—	**1.2s**	21ms
S	2m 7.1s		—	—	0.4s	16ms	19s	0.2s	**2.3m**	7.4s
CH	41m	2.4m	4.4s	1s	32ms	7ms	21m	0s	**1h**	3m
P	45m	74s	60ms	8ms	4s	32ms	53s	4s	**46m**	79s

REFERENCES

[1] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. 2005. On map-matching vehicle tracking data. In *VLDB*. 853–864.

[2] Patrick Brosi and Uli Müller. 2015. Automatisierte Erstellung von fahrplanreferenzierten ÖV-Netzen. *AGIT*, 122–129.

[3] Robert Geisberger and Christian Vetter. 2011. Efficient routing in road networks with turn costs. In *SEA*. Springer, 100–111.

[4] GTFS. [n. d.]. General Transit Feed Specification. https://developers.google.com/transit/gtfs.

[5] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. 2009. Map-matching for low-sampling-rate GPS trajectories. In *SIGSPATIAL/GIS*. 352–361.

[6] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *SIGSPATIAL/GIS*. 336–343.

[7] Flavio Poletti, Patrick M Bösch, Francesco Ciari, and Kay W Axhausen. 2017. Public transit route mapping for large-scale multimodal networks. *ISPRS*, 268.

[8] Hong Wei, Yin Wang, George Forman, Yanmin Zhu, and Haibing Guan. 2012. Fast Viterbi map matching with tunable weight functions. In *SIGSPATIAL/GIS*. ACM, 613–616.