Fast Viterbi Map Matching with Tunable Weight Functions

Hong Wei
Shanghai Key Lab of Scalable Computing and Systems
Shanghai Jiao Tong University
keith.collens@sjtu.edu.cn

Yin Wang George Forman
Hewlett-Packard Labs
Palo Alto, CA, USA
{first.last}@hp.com

Yanmin Zhu Haibing Guan
Shanghai Key Lab of Scalable Computing and Systems
Shanghai Jiao Tong University
{yzhu,hbguan}@sjtu.edu.cn

ABSTRACT

This paper describes a map matching program submitted to the ACM SIGSPATIAL Cup 2012. We first summarize existing map matching algorithms into three categories, and compare their performance thoroughly. In general, *global max-weight* methods using the Viterbi dynamic programming algorithm are the most accurate but the accuracy varies at different sampling intervals using different weight functions. Our submission selects a hybrid that improves upon the best two weight functions such that its accuracy is better than both and the performance is robust against varying sampling rates. In addition, we employ many optimization techniques to reduce the overall latency, as the scoring heavily emphasizes on speed. Using the training dataset with manually corrected ground truth, our Java-based program matched all 14,436 samples in 5 seconds on a dual-core 3.3 GHz iCore 3 processor, and achieved 98.9% accuracy.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Applications—*Data mining; Spatial databases and GIS*

General Terms

Algorithms, Experimentation, Performance

Keywords

GPS, map matching, Viterbi, Fréchet distance

1. INTRODUCTION AND SURVEY

The problem of map matching is to locate a sequence of GPS coordinates onto a road map. We denote the input GPS sequence as $z_0, z_1, \ldots, z_n$. The output of map matching is a sequence of estimated locations on road segments, denoted by $x_0, x_1, \ldots, x_n$. Most existing map matching algorithms can be classified into three categories, incremental max-weight [16, 17, 3, 7, 5, 13, 15, 18, 6, 10], global max-weight [9, 8, 12, 14, 11] and global geometrical methods [2, 4]. Both incremental and global max-weight methods consist of the following steps:

1. For each GPS sample z_i, determine a set of candidate locations $\{x_i^0, x_i^1, ...\}$, which are typically the perpendicular projections on road segments $\{y_i^0, y_i^1, ...\}$ within a radius or an error eclipse of z_i.
2. Calculate a weight for each candidate.
3. Output the candidate sequence with the max weight.

Incremental and global methods differ in the last step on how to pick the candidate sequence. Incremental methods calculate the best candidate for each sample one at a time. The calculation is based on either a range of the recent samples, or a summary of all previous samples (e.g., Bayes filter). In contrast, global max-weight methods consider the aggregated weight of an entire candidate sequence, and output the sequence with the maximum aggregated weight at once. Most global max-weight methods employ Hidden Markov Model (HMM), and apply the Viterbi dynamic programming algorithm. We consider the emission and transition probabilities of HMM as the weights for these global methods. Incremental methods can be applied to online map matching since they rely on previous observations only. For offline map matching, however, we show that global methods achieve better accuracy because future observations are often needed to match the current sample correctly.

Weight calculations for both incremental and global methods are based on a common set of features. Table 1 summarizes 15 map matching algorithms. The first 10 rows are incremental methods and the last 5 are global methods. Each algorithm calculates weights using a subset of the seven features listed in the table. The first two features are calculated for each GPS sample independently: Distance measures the great circle distance between a sample z_i and a candidate location x_i^j. Bearing measures the difference between the GPS-measured heading direction of z_i and the direction of the road y_i^j where x_i^j is located. The remainder of the features are calculated using two consecutive samples z_{i-1}, z_i. Connectivity depends on whether the candidate x_{i-1}^k of z_{i-1} can reach the candidate x_i^j of z_i through a path of reasonable length or drive time, considering both map topology and turn restrictions. Shortest-path generalizes connectivity by calculating the length of the shortest path from x_{i-1}^k to x_i^j. Direction measures the angle difference between line segment $\overrightarrow{z_{i-1}, z_i}$ and the road segment y_i^j of candidate x_i^j. Length is the great circle distance between the previous sample z_{i-1} and candidate x_i^j of the current sample z_i. Speed is the length of the shortest path between two candidates divided by the sampling interval. Although seldom used, we

ACM SIGSPATIAL GIS '12, Nov. 6-9, 2012. Redondo Beach, CA, USA

Table 1: Summary of max-weight map matching algorithms.

	GPS sample		segment of two consecutive GPS samples					aggregation	
method	distance	bearing	connectivity	shortest-path	direction	length	speed	sample	sequence
White00 [16]	d	$\Delta\theta$	1 or 0					threshold	
Yang05 [17]	d			l				rules	
Blazquez06 [3]	d						$v - \frac{v_i + v_{i-1}}{2}$	threshold	
Li07 [7]	d	$\Delta\theta$						tie-breaker	
Greenfeld02 [5]	$C - w_d d^{n_d}$				$w_\alpha cos(\Delta\alpha)^{n_\alpha}$			sum	
Quddus03 [13]	$w_d \frac{1}{d}$	$w_\theta cos(\Delta\theta)$						sum	
Velaga09 [15]	$w_d \left(1 - \frac{d}{80}\right)$	$w_\theta cos(\Delta\theta)$	w_c or $-w_c$					sum	
Zheng11 [18]	$w_d \left(1 - \frac{d}{200}\right)$	$w_\theta \lvert cos(\Delta\theta)\rvert$		$w_l \left(1 - \frac{\lvert l - l_0\rvert}{1000}\right)$	$w_\alpha \lvert cos(\Delta\alpha)\rvert$			sum	
Griffin11 [6]	d	$\Delta\theta$		$\frac{l_0}{l}$				decision tree	
Mazhelis10 [10]	$\frac{1}{d+\delta}$				$1 - 2k\frac{\Delta\alpha}{\pi}$	$\frac{1}{(d' + \xi)^2}$		Bayes filter	
Marchal05 [9]	d								sum
Lou09 [8]	$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{d^2}{2\sigma^2}}$			$\frac{l_0}{l}$			$\frac{v' \cdot v}{\lVert v'\rVert \, \lVert v\rVert}$	multiply	sum
Pink08 [12]	Mahalanobis		$\frac{1}{n+1}$ or 0					multiply	multiply
Vtrack09 [14]	$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{d^2}{2\sigma^2}}$		ε or 0					multiply	multiply
Newson09 [11]	$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{d^2}{2\sigma^2}}$			$\frac{1}{\beta} e^{-\frac{l - l_0}{\beta}}$				multiply	multiply

z_i is a GPS sample from the given trace, and a matching candidate x_i^j of z_i is the perpendicular projection of z_i on a road segment y_i^j. Distance $d = \lVert z_i - x_i^j \rVert_{great\ circle}$, bearing $\Delta\theta = \lvert bearing(z_i) - bearing(y_i^j)\rvert$, connectivity n is the number of roads connected to y_i^j, shortest-path $l = shortestPath(x_{i-1}^k, x_i^j)$, $l_0 = \lVert x_{i-1}^k - x_i^j \rVert_{great\ circle}$, direction $\Delta\alpha = \lvert bearing(\overrightarrow{z_{i-1} z_i}) - bearing(y_i^j)\rvert$, length $d' = \lVert z_{i-1} - x_i^j \rVert_{great\ circle}$, speed $v_i = speed(z_i)$, $v = l/t_{interval}$, v' is the speed limit vector of the road, and the remaining variables represent constants or tunable parameters.

list these last two features in the table for completeness.

In addition to max-weight methods, global geometric map matching finds the optimal path on map by geometric similarity measures, e.g., Fréchet distance, and do not calculate the candidate set for each GPS sample. A popular intuitive definition of the Fréchet distance between two curves is the minimum length of a leash required to connect a dog and its owner, constrained on two separate paths, as they walk without backtracking along their respective curves from one endpoint to the other. Map matching algorithms based on Fréchet distance find the path on the map that has a minimum Fréchet distance to the GPS trace [2, 4]. We compare the performance of representative algorithms in each of the category in Section 2, and discuss our Viterbi map matching algorithm in Section 3. Section 4 concludes the paper.

2. COMPARATIVE STUDY

We implemented a map matching framework with tunable weight functions to compare the performance of different algorithms in Table 1. For incremental methods, our algorithm first picks the candidate with the maximum weight for the initial sample, and then iteratively calculates the next max-weight candidate based on the current best candidate. For global methods, we use the Viterbi algorithm. Since the training dataset [1] does not have vehicle bearing or speed information, our weight calculation is based on the distance feature d and shortest-path feature l (which generalizes the connectivity feature) only.

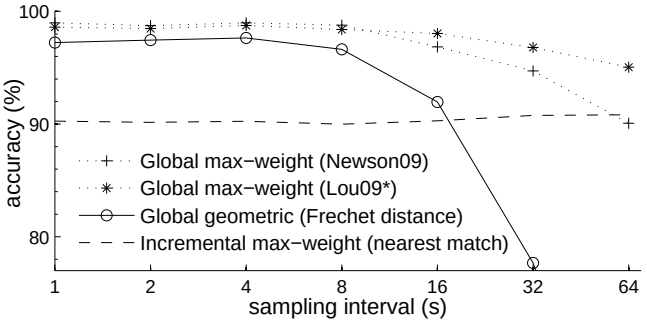
We discover that all incremental methods perform poorly on the training dataset, even worse than simply matching each sample to the nearest road.[1] Y-splits at freeway ramps contribute to the low accuracy. Figure 1b shows an example. Red samples are incorrectly matched to the upper ramp due to both location proximity and road connectivity until the distance between a sample and the ramp is beyond the threshold of 50 m. Without examining future samples, it is impossible to match these red samples correctly. Nearest-match performs slightly better in this case because it ignores connectivity and therefore matches the last five red samples correctly to the freeway below. We use nearest-match as a representative incremental matching algorithm for our performance comparison.

For global methods, we pick Newson09 and a variant of Lou09 since the features they use to calculate weights (probabilities) are more general than other global methods. We take the logarithm of the probabilities to simplify the calculation and to avoid the underflow of floating point when the probabilities are too small. For example, Newson09 calculates the max-weight candidate sequence as

$$\underset{(x_0, x_1, \ldots, x_n)}{\operatorname{argmax}} \prod_{i=0}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{d_i^2}{2\sigma^2}} \frac{1}{\beta} e^{-\frac{l_i - l_{i,0}}{\beta}} \tag{1}$$

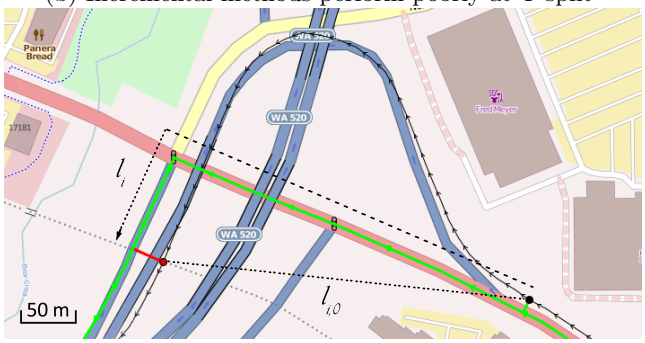
which is equivalent to

[1]We ignore the directionality of identical-shape polylines that represent opposite directions of two-way roads.

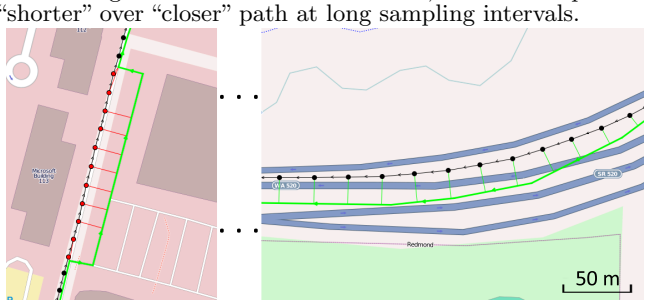


(a) Accuracy at different sampling intervals

(b) Incremental methods perform poorly at Y-split



(c) $l_i - l_{i,0}$ in (2) increases when the sampling interval becomes longer. Since α is a constant, Newson09 prefers "shorter" over "closer" path at long sampling intervals.



(d) Since Fréchet distance considers only the maximum, "alternative" paths can be selected if the maximum distance is large (due to either noise or a long sampling interval).

Figure 1: Performance comparison results

$$\underset{(x_0,x_1,\ldots,x_n)}{\operatorname{argmin}} \sum_{i=0}^{n} d_i^2 + \alpha(l_i - l_{i,0}) \tag{2}$$

where $\alpha = 2\sigma^2/\beta$ is a coefficient to be estimated. Our variant of Lou09, denoted as Lou09*, simply replaces the term $l_i - l_{i,0}$ in (2) with $-l_{i,0}/l_i$. We choose this variant because it is more consistent with the HMM formulism, and is more accurate in our experiments using the training dataset.

Figure 1a shows the results of representative algorithms in each category at different sampling intervals. For both global max-weight algorithms, we search for match candidate in a 50 m radius circle. The coefficient α in (2) is tuned and set to the value where the algorithm is most accurate with the training data at 1 s sampling interval. Overall, nearest-match achieves around 90% accuracy across all sampling intervals. Newson09 is the most accurate at sampling intervals below 8 s, afterwards Lou09* tops out. The example in Figure 1c explains the reason of the degraded performance of Newson09. Fréchet distance based map matching performs reasonably well at lower sampling intervals, but its accuracy decreases rapidly beyond 8 s. Figure 1d shows the major cause with an example.

3. OUR ALGORITHM AND RESULTS

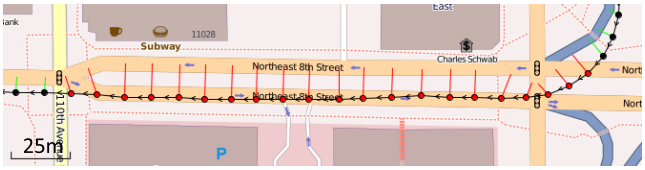
Accuracy improvement We choose the global max-weight method using the Viterbi algorithm because it is the most accurate according to our comparative study in Section 2. Observing that Newson09 performs better at low sampling intervals while Lou09* performs better at long sampling intervals, we design a weight function that improves upon the two:

$$\underset{(x_0,x_1,\ldots,x_n)}{\operatorname{argmin}} \sum_{i=0}^{n} t_i d_i^2 + \alpha l_i \tag{3}$$

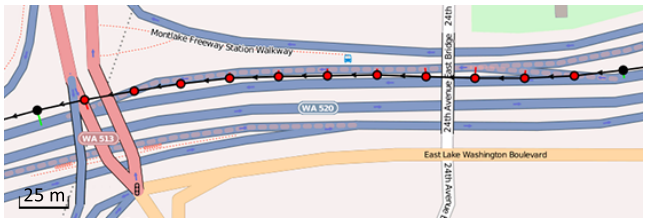
where t_i is the time interval between z_i and z_{i-1}. The rationale of our design is the following. The summation of l_i for a candidate sequence is exactly the length of the shortest path that links it, which does not change with the sampling interval. In addition, the expected value of the summation of $t_i d_i^2$ does not depend on the sampling interval either. Therefore, the ratio between these two terms remains the same, and a constant α should perform consistently across all sampling intervals. Since (3) takes into account the interval of each individual sample, our algorithm is also robust against input traces with variable sampling rates. Finally, at 1 s sampling interval, (3) is equivalent to (2) for the same α because $t_i = 1$ and the summation of $l_{i,0}$ is a constant.

Parsing speedup We implemented our map matching framework in Java. Without any optimization, parsing the map alone takes more than 30 seconds. We increase the parsing speed using the following techniques. First and foremost, our parser reads the traces first and bypasses roads outside of the bounding box that encloses all traces. For roads inside the bounding box, we built our own string scanner instead of `split()` to parse each input line, and implemented a quick double parser that parses numerical characters only and ignores special cases such as `NaN`. Parallel parsing of each input line is essential for our speed optimization, which scales linearly with the number of cores. We set the number of threads to the return value of `availableProcessers()`.

Viterbi speedup In addition to parsing, we improve the speed of the Viterbi map matching algorithm using the following techniques. For our comparative study of existing map matching algorithms, we applied Azimuthal Equidistant projection which preserves the great circle distance better than other projection methods, but the computation cost is high. For the competition, we use only a linear projection, which does not affect the accuracy of the training dataset because it covers a relatively small area. For road indexing, we use R*-tree from the JTS library. The bottleneck of our Viterbi map matching is the shortest-path calculation for every consecutive pair of match candidates. Assume there are n candidates for sample z_i and m candidates for



(a) Red samples match to the lower road in ground truth, which is a one-way road in reverse direction.



(b) The vehicle remained on the lower freeway in ground truth, but the GPS trace is closer to the upper ramp.

Figure 2: Tricky cases

sample z_{i+1}, a total of nm shortest paths need to be calculated. We adapt the Dijkstra algorithm to calculate all shortest paths from each candidate of z_i to all m candidates of z_{i+1} at once. Therefore we only need to apply the algorithm n times. We did not implement all-pair shortest path algorithm such as Floyd-Warshall. However, the Dijkstra algorithm allows us to limit the shortest-path search range from the source, which we set to be the sampling interval multiplied by 50 m/s. This greatly reduces the computation time because often only the correct candidate pairs can be reached within the speed limit. We also limit the candidate search radius to 30 m for the competition to reduce the size of candidate sets, because the largest distance between a sample and its matched road is around 25 m for the training dataset. Similar to parallel parsing, our program processes each input trace in parallel. We did not implement parallel matching for an individual trace, so there is no performance gain if the input is in one big file.

Training Results Using a 3.3 GHz iCore 3 processor, our optimized program takes 2 seconds to parse the map, and another 1.5 seconds to match all 14,436 samples. Using sub-sampled training data, the matching time gradually increases with the sampling interval, up to 3 seconds at 30 s interval. When the sampling interval increases, the shortest path calculation takes longer but there are fewer samples to compute. The `.exe` binary conversion adds another 1.5 second overhead. We discovered five sections of incorrectly matched samples in the training data, a total of 78 samples affected. One of them has been discussed in the mailing list, where an intersection in the map is mistakenly disconnected. The others are likely due to mislabeling; see Figure 2a for an example. Using manually corrected ground truth, the accuracy of our Viterbi algorithm is equal to or better than the top performer of Newson09 and Lou09* across all sampling intervals. More specifically, it is 98.9% accurate at 1 s interval, and gradually decreases to 97.0% at 64 s interval.

4. CONCLUSION

We have shown that existing map matching methods can be largely categorized as incremental max-weight, global max-weight, and global geometric methods. Our map matching framework incorporates algorithms in the former two categories through tunable weight functions. Using the training dataset, we have demonstrated that global max-weight methods using the Viterbi algorithm are the most accurate. Improving upon the best two performers, our own Viterbi map matching is more robust against varying sampling intervals. Using manually corrected ground truth, our algorithm achieves 98.9% with the training data. Modeling the driver behavior can lead to further improvements; see Figure 2b for an example.

5. REFERENCES

[1] ACM SIGSPATIAL Cup 2012. `http://depts.washington.edu/giscup/home`.

[2] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. *Journal of Algorithms*, 49(2):262–283, 2003.

[3] C. A. Blazquez and A. P. Vonderohe. Simple map-matching algorithm applied to intelligent winter maintenance vehicle data. *Transportation Research Record*, 1935(1):68–76, 2006.

[4] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *VLDB*, 2005.

[5] J. S. Greenfeld. Matching GPS observations to locations on a digital map. In *Transportation Research Board*, 2002.

[6] T. Griffin, Y. Huang, and S. Seals. Routing-based map matching for extracting routes from GPS trajectories. In *International Conference on Computing for Geospatial Research & Applications*, 2011.

[7] X. Li, M. Li, W. Shu, and M. Wu. A practical map-matching algorithm for GPS-based vehicular networks in Shanghai urban area. In *IET Conference on Wireless, Mobile and Sensor Networks*, 2007.

[8] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate GPS trajectories. In *ACM SIGSPATIAL GIS*, 2009.

[9] F. Marchal, J. Hackney, and K. Axhausen. Efficient map matching of large global positioning system data sets: Tests on speed-monitoring experiment in Zürich. *Transportation Research Record*, 1935(1):93–100, 2005.

[10] O. Mazhelis. Using recursive Bayesian estimation for matching GPS measurements to imperfect road network data. In *Intelligent Transportation Systems*, 2010.

[11] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. In *ACM SIGSPATIAL GIS*, 2009.

[12] O. Pink and B. Hummel. A statistical approach to map matching using road network geometry, topology and vehicular motion constraints. In *Intelligent Transportation Systems*, 2008.

[13] M. A. Quddus, W. Ochieng, L. Zhao, and R. Noland. A general map matching algorithm for transport telematics applications. *GPS Solutions*, 7(3):157–167, 2003.

[14] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *SenSys*, 2009.

[15] N. R. Velaga, M. A. Quddus, and A. L. Bristow. Developing an enhanced weight-based topological map-matching algorithm for intelligent transport systems. *Transportation Research Part C: Emerging Technologies*, 17(6):672–683, 2009.

[16] C. E. White, D. Bernstein, and A. L. Kornhauser. Some map matching algorithms for personal navigation assistants. *Transportation Research Part C: Emerging Technologies*, 8(1–6):91–108, 2000.

[17] J. Yang, S. Kang, and K. Chon. The map matching algorithm of GPS data with relatively long polling time intervals. *Journal of the Eastern Asia Society for Transportation Studies*, 6:2561–2573, 2005.

[18] Y. Zheng and M. A. Quddus. Weight-based shortest path aided map-matching algorithm for low frequency GPS data. In *Transportation Research Board*, 2011.