Map Matching by Fréchet Distance and Global Weight Optimization

Hong Wei* Yin Wang† George Forman† Yanmin Zhu*

Abstract

The process of *map matching* takes a sequence of possibly noisy GPS coordinates from a vehicle trace and estimates the actual road positions—a crucial first step needed by many GPS applications. There has been a plethora of methods for map matching published, but a thorough comparison has been lacking. Here we provide a unifying framework to help make sense of the many published methods, cataloging the different mathematical formulas they use for weighting and aggregating the features that are common to most. In evaluations using both a low-noise dataset from Seattle and a high-noise dataset from Shanghai, we find that global max-weight and global geometrical map matching methods are the most accurate, but each has its weaknesses. We therefore propose a new map matching algorithm that integrates Fréchet distance with global weight optimization. Our new algorithm is more accurate across all sampling intervals. Furthermore, it requires very little tuning and the performance is robust across datasets of different characteristics.

1 Introduction

Global Positioning System (GPS) receivers are integrated into navigation devices, vehicle telematics systems, and smart phones. Due to their inherent measurement error, map matching is a necessary step to pinpoint the correct location on the road network. Many GPS related applications require map matched data, e.g., traffic monitoring [13, 31, 33], event detection [17], and road quality assessment [10].

Despite the large amount of work on map matching, it is unclear how different algorithms compare, especially under different circumstances. Many research works evaluate only their own proposed algorithm, and such results are incomparable since the datasets are different. Some comparative analysis exists but is limited. Quddus et al. [23,25] evaluated several early map matching algorithms that are typically designed for in-device online matching. Recent offline algorithms achieve better accuracy through global optimization. Brakatsoulas et al. [8] compared an online map matching algorithm with an algorithm that minimizes the Fréchet distance. Their evaluation was based on the same Fréchet distance measure and therefore favors the algorithm that minimizes it. Both Quddus and Brakatsoulas studied only high frequency samples, although much GPS data is only available with sparse sampling, e.g., fleet management systems [6, 15, 16, 28]. Lou et al. [18] compared their proposed offline map matching algorithm with an online algorithm and an algorithm that minimizes average Fréchet distance. However, it is unclear which online algorithm was chosen, and their notion of *average* Fréchet distance was not defined.

In this paper, we propose a novel framework that encompasses most existing map matching algorithms. These algorithms determine the most probable match for a sample by calculating a weight for each candidate road location. This weight integrates a common set of factors, such as the distance between sample and road segment, the alignment between measured bearing and road direction, and the shortest path between consecutive matches. The output route is the candidate sequence with the highest score. These algorithms differ primarily in the way they combine these factors. We call this category of methods *max-weight* methods, which can be further divided into *incremental max-weight* and *global max-weight* methods. Incremental methods determine the output for each point before advancing to the next, which is suitable for online map matching, and global methods consider the whole trace in determining the output. The other major category consists of *global geometric* methods, which determine the best matched path solely by geometric measures such as *Fréchet distance*.

Our framework enables us to compare the essential differences among the various mathematical formulas used by the different algorithms, using a uniform software harness. We evaluate representative algorithms in all three categories using a low-noise dataset from Seattle [1] and a high-noise dataset we collected from Shanghai [2]. Our experiments show that global methods are more accurate than incremental methods because future observations can often help determine previous locations. On the other hand, global max-weight and global geometric methods have their respective strengths and weaknesses. Global max-weight methods are accurate and robust even with long GPS sampling intervals, but they require substantial tuning and are sensitive to dif-

*Shanghai Jiao Tong Univ., {keith.collens,yzhu}@sjtu.edu.cn
†HP Labs, Palo Alto, {yin.wang,george.forman}@hp.com

ferent data characteristics. Global geometric methods require no tuning and the matching results are easy to understand, but the performance is poor when there are alternative paths with the same optimal geometric measurements, which is often the case with long sampling intervals or high-noise data.

Based on these observations, we propose to integrate global weight optimization into map matching based on Fréchet distance. Our algorithm first determines the minimum Fréchet distance for an input GPS trace, and then chooses the maximum weight path among all minimum distance paths. Similar to most global max-weight algorithms, we use dynamic programming to avoid the explicit enumeration of all candidate paths, which can be exponential in the length of the input trace. Our experiments show that our proposed algorithm is much more accurate than both global max-weight and global geometric algorithms across all sampling intervals. Global max-weight algorithms can be tuned to perform on par with our algorithm, but the optimal parameters are different for different datasets.

The contributions of this paper are the following.

- A comprehensive survey of existing map matching algorithms and a novel framework to incorporate most of them.
- The implementation and comparison of top performing algorithms in all three categories of map matching methods.
- A new offline map matching algorithm that combines geometric algorithms with max-weight algorithms for better accuracy and robustness under different data characteristics.
- Experiments using both low-noise and high-noise datasets. The map matching literature is dominated by low-noise GPS data, despite that real-world GPS data is often noisy, especially in cities. We are the first to collect a high-noise GPS dataset with manually annotated ground truth, which we make available to the public for future research [2].

Section 2 surveys map matching algorithms and motivates our solution. Section 3 describes the design and implementation of our map matching algorithm. Section 4 shows experimental results, and Section 5 concludes the paper.

2 Map Matching Survey

The input to a map matching algorithm is a sequence of GPS samples, $z_0, z_1, ..., z_n$, and a map of the road network. The map is typically represented by a set of roads, and each road is represented as a polyline, i.e., a sequence of line segments. The output of the algorithm is a sequence of estimated locations of the vehicle on specific road segments, denoted as $x_0, x_1, ...x_n$.

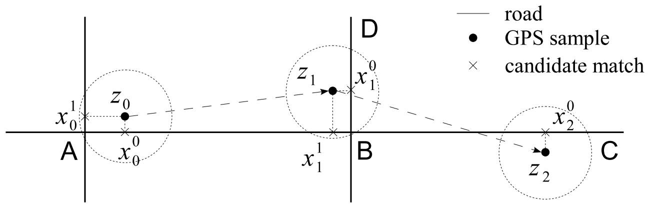


Figure 1: Max-weight map matching

Most existing map matching algorithms can be classified into three categories, incremental max-weight [3, 7, 11, 12, 14, 20, 24, 27, 30, 32], global max-weight [18, 19, 21, 22, 26] and global geometrical methods [4, 8]. 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; see Figure 1 for an example.
2. Calculate a weight for each candidate.
3. Output the candidate sequence with the maximum 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 compute an aggregated weight for each candidate sequence in entirety, and they output the sequence with the maximum. Most global max-weight methods employ a Hidden Markov Model (HMM), solving with the Viterbi dynamic programming algorithm. The emission and transition probabilities of the HMM provide the weights for 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 previous samples 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

Table 1: Summary of incremental and global 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 [30]	d	$\Delta\theta$	1 or 0					threshold	
Yang05 [3]	d			l				rules	
Blazquez06 [7]	d						$v - \frac{v_i + v_{i-1}}{2}$	threshold	
Li07 [14]	d	$\Delta\theta$						tie-breaker	
Greenfeld02 [11]	$C - w_d d^{n_d}$				$w_\alpha cos(\Delta\alpha)^{n_\alpha}$			sum	
Quddus03 [24]	$w_d \frac{1}{d}$	$w_\theta cos(\Delta\theta)$						sum	
Velaga09 [27]	$w_d \left(1 - \frac{d}{80}\right)$	$w_\theta cos(\Delta\theta)$	w_c or $-w_c$					sum	
Zheng11 [32]	$w_d \left(1 - \frac{d}{200}\right)$	$w_\theta \lvert cos(\Delta\theta)\rvert$		$w_l \left(1 - \frac{\lvert l - l_0\rvert}{1000}\right)$	$w_\alpha \lvert cos(\Delta\alpha)\rvert$			sum	
Griffin11 [12]	d	$\Delta\theta$		$\frac{l_0}{l}$				decision tree	
Mazhelis10 [20]	$\frac{1}{d + \delta}$				$1 - 2k\frac{\Delta\alpha}{\pi}$	$\frac{1}{(d' + \xi)^2}$		Bayes filter	
Marchal05 [19]	d								sum
Lou09 [18]	$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{d^2}{2\sigma^2}}$			$\frac{l_0}{l}$			$\frac{v' \cdot v}{\lVert v'\rVert \, \lVert v\rVert}$	multiply	sum
Pink08 [22]	Mahalanobis		$\frac{1}{n+1}$ or 0					multiply	multiply
Vtrack09 [26]	$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{d^2}{2\sigma^2}}$		ε or 0					multiply	multiply
Newson09 [21]	$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{d^2}{2\sigma^2}}$			$\frac{1}{\beta} e^{-\frac{l - l_0}{\beta}}$				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 $d = \|z_i - x_i^j\|_{great\ circle}$, bearing $\Delta\theta = |bearing(z_i) - bearing(y_i^j)|$, connectivity n is the number of roads connected to y_i^j, shortest-path $l = shortestPath(x_{i-1}^k, x_i^j)$, $l_0 = \|x_{i-1}^k - x_i^j\|_{great\ circle}$, direction $\Delta\alpha = |bearing(\overrightarrow{z_{i-1} z_i}) - bearing(y_i^j)|$, length $d' = \|z_{i-1} - x_i^j\|_{great\ circle}$, 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.

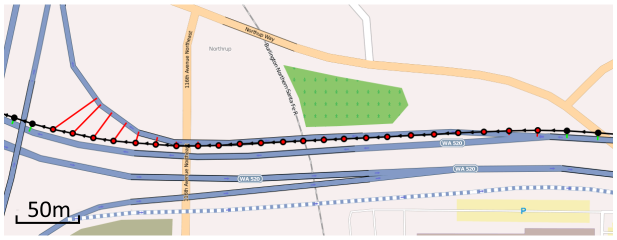
remainder of the features are calculated using two consecutive samples z_{i-1}, z_i. *Connectivity* depends on whether the candidate x_{i-1}^k of z_{i-1} can reach the candidate x_i^j of z_i through a path of reasonable length or drive time, considering both map topology and turn restrictions. *Shortest-path* 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 list these last two features in the table for completeness.

In contrast to max-weight methods, global geometric map matching finds the optimal path on the map by geometric similarity measures, e.g., Fréchet distance; these methods do not calculate a 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 [4, 8].

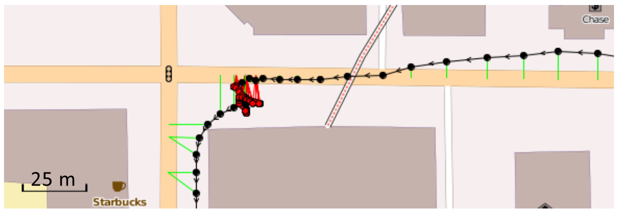
2.1 Incremental Max-Weight Map Matching.

Incremental max-weight algorithms calculate the matched location for each input sample one at a time. Most incremental algorithms in Table 1 calculate the current match using only the current and the previous sample, as well as the previous matched location. The only exception is Mazhelis10 [20], which uses a Bayes filter. With each input sample, it updates the current belief state, which is a probability distribution over the set of candidate locations. The output match is the location with the maximum probability.

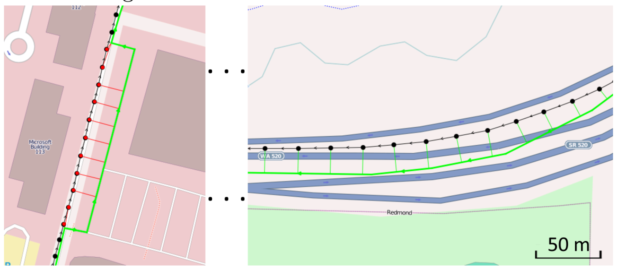
For the example in Figure 1, incremental methods usually pick location x_1^0 as the best match for GPS sample z_1 since it is much closer to the sample than x_1^1



(a) Incremental methods perform poorly at Y-split.



(b) Max-weight methods have difficulty differentiating backward drifting noise from U-turn.



(c) Large Fréchet distance permits "alternative" paths. The green path is the output matched path. It is misaligned with the background raster map in the right-hand snippet probably because of distortion due to map projection.

Figure 2: Examples to illustrate weaknesses of each category of algorithms. Green line segments mark correct matches, and red segments mark wrong matches.

(assuming roads AB and BD are connected and there is no left-turn restriction). Figure 2a shows a real and more serious case with our Seattle dataset. Due to noise, the series of red samples are much closer to the upper ramp than the lower freeway. Most incremental methods will continue to match these samples to the ramp until it is beyond some distance and then snap to the correct road. Even though Bayes filters or particle filters may attribute some small probability mass to the correct freeway, they too will exhibit this myopic, forward-only behavior.

2.2 Global Max-Weight Map Matching. Most global max-weight algorithms assume HMM models. Using the Viterbi dynamic programming algorithm, the optimal output sequence $(x_0, x_1, \ldots, x_n)$ has an intuitive explanation. For example, Newson09 [21] calculates the max-weight candidate sequence as

$$\text{(2.1)} \qquad \underset{(x_0,x_1,\ldots,x_n)}{\arg\max} \prod_{i=0}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{d_i^2}{2\sigma^2}} \frac{1}{\beta} e^{-\frac{l_i - l_{i,0}}{\beta}}$$

which is equivalent to

$$\text{(2.2)} \qquad \underset{(x_0,x_1,\ldots,x_n)}{\arg\min} \sum_{i=0}^{n} d_i^2 + \alpha(l_i - l_{i,0})$$

where $\alpha = 2\sigma^2/\beta$ is a coefficient to be estimated. Given an input GPS trace, $l_{i,0}$ is a constant. Therefore, Newson09 essentially calculates the candidate sequence whose weighted summation of d_i^2 and l_i is the smallest. Comparing with incremental max-weight methods, global max-weight methods assign weights to each candidate the same way, but the output sequence is the one with maximum aggregated weight. For the example in Figure 1, global max-weight methods can match z_1 to x_1^1 because the overall sequence has higher weight even though x_1^0 may have a higher weight than x_1^1 locally.

Both incremental and global max-weight methods ignore the geometric shape and continuity of the drive path, which can hurt accuracy under noise conditions. Figure 2b shows an example using the Seattle dataset. The vehicle was likely waiting for a left turn at the intersection. The GPS samples drift backward before they move forward again. Max-weight methods that consider either connectivity or shortest-path usually characterize this pattern as a double U-turn because the candidate locations of the backward drifting samples are behind their predecessors. Some max-weight methods include a noise filtering parameter that consider samples too close to their predecessors as noise and match them to the same location [21], but it is difficult to determine the optimal threshold. The matching result in Figure 2b already employed a threshold of 5 m to filter out random noise when the vehicle is stopped, but it is not large enough. On the other hand, this threshold is already large enough to induce another type of error: The last and the 3rd last sample in the figure were considered noise and therefore assigned to the matched locations of their predecessors.

2.3 Global Geometrical Map Matching. Matching a GPS sample sequence to a path based on geometric similarity first appeared in [30], called *curve-to-curve* matching. The algorithm is based on an ad hoc distance measure of two curves. Most global geometric map matching methods in the literature find the matched path that minimizes the Fréchet distance to the GPS trace.

Formally, the Fréchet distance between two curves $f, g : [0, 1] \to \mathbb{R}^2$ is defined as

$$\delta_F(f, g) := \inf_{\alpha,\beta:[0,1]\to[0,1]} \max_{t\in[0,1]} \|f(\alpha(t)) - g(\beta(t))\|$$

where α and β are continuous and non-decreasing time-warping functions with $\alpha(0) = \beta(0) = 0$ and

$\alpha(1) = \beta(1) = 1$. A polynomial algorithm to determine the Fréchet distance between two polylines was first proposed in [5]. An extension of the algorithm finds a path in a planar map that has the minimum Fréchet distance to a given polyline [4].

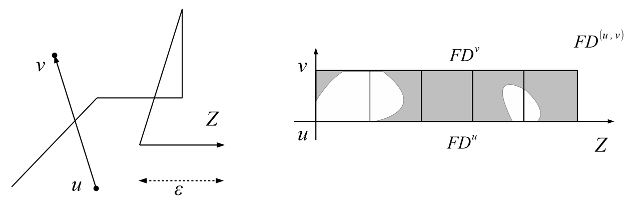
The Fréchet distance is defined by the maximum distance between two curves. In the application of map matching, this is often at the location where the GPS noise is the highest, or the map is the most inaccurate. If this maximum distance is larger than the distance between two alternative paths at other locations, the output path can pick either alternative, and the Fréchet distance is the same. The standard algorithm does not have any preference for alternative paths, and its accuracy suffers from random choices. Figure 2c shows an example from the Seattle dataset, where the Fréchet distance is determined by the offset shown in the right-hand snippet, around 25 m. Judging by an aerial image, the error was due to having an imprecise map. Regardless of the cause, with this large distance the snippet shown on the left mistakenly picks an alternative parallel road that is within 25 m from the ground truth path. This alternative path problem is most serious at long sampling intervals, where the vehicle can make a few turns in-between two samples, and therefore the GPS trajectory has a large Fréchet distance to the ground truth path.

Variants of Fréchet distance measurements have been proposed for map matching, such as weak Fréchet distance and average Fréchet distance [8]. These definitions do not address the alternative path problem. Subsequent work on Fréchet distance based map matching focuses on computation speed optimization [9, 29].

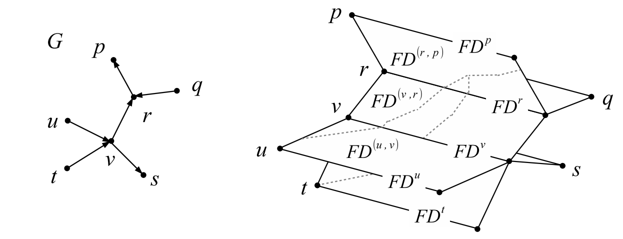
3 Our Algorithm

Based on the weaknesses discussed in Section 2 for each category of methods, we propose to incorporate global weight optimization into map matching based on Fréchet distance. We first introduce the Fréchet distance algorithm that our map matching algorithm augments. We refer readers to [4] for a more complete discussion. Compared to the original algorithm, our implementation simplifies the principal subroutine; see Appendix A in our supplementary file.

Given a constant ε, the *free space* of two curves $f, g : [0,1] \to \mathbb{R}^2$ is defined as $F_\varepsilon(f,g) := \{(s,t) \in [0,1]^2 \mid \|f(s) - g(t)\| \leq \varepsilon\}$. Region $[0,1]^2$ is therefore partitioned into free space and non-free space, called the *free space diagram*. For example, the free space of two line segments is a full or partial ellipse, and the free space diagram of two polylines of m and n segments is a $m \times n$ segment-segment free space diagram. It has been shown that $\delta_F(f,g) \leq \varepsilon$ if and only



(a) Free space diagram $FD^{(u,v)}$ for edge (u,v) and GPS trace Z.



(b) Free space surface consists of free space diagrams glued together according to the topology of G. Grey dashed paths are monotone paths in the free space, which may not be unique.

Figure 3: Free space surface illustration, based on the example in [4]

if there exists a path within $F_\varepsilon(f,g)$ from the lower left corner $(0,0)$ to the upper right corner $(1,1)$, which is monotone in both coordinates [5]. This path induces functions α and β in the definition of $\delta_F(f,g)$. Constructing the free space diagram and determining the existence of a monotone path for two polylines takes polynomial time. The Fréchet distance of two polylines is calculated by a parametric or binary search that finds the minimum ε.

The definition of the free space between two polylines generalizes to the free space between a planar graph $G = (V, E)$ of the road network and a polyline $Z = (z_0, \ldots, z_n)$ of GPS samples. With slight abuse of definition, we consider the free space diagram between a vertex $v \in V$ and Z. The result is a one-dimensional free space line, denoted as FD^v. The free space diagram between an edge $(u,v) \in E$ and Z is a $1 \times n$ segment-segment diagram, denoted as $FD^{(u,v)}$; see Figure 3a for an example. Notice that the free space diagrams of all edges adjacent to a vertex v share the free space line FD^v. We therefore construct free space diagrams $FD^{(u,v)}$ for all $(u,v) \in E$, and "glue" them together along shared free space lines, according to the topology of G. The resulting three-dimensional structure is called a *free space surface* of graph G and polyline Z. Figure 3b gives an example graph G and its free space surface; the trace is omitted for simplicity. There is a path in G with Fréchet distance at most ε to Z if and only if there is a monotone path on the free space surface [4]. Finding the minimum Fréchet distance to Z is achieved by parametric or binary search.

When there are multiple monotone paths on the free space surface, we apply a Viterbi-like dynamic

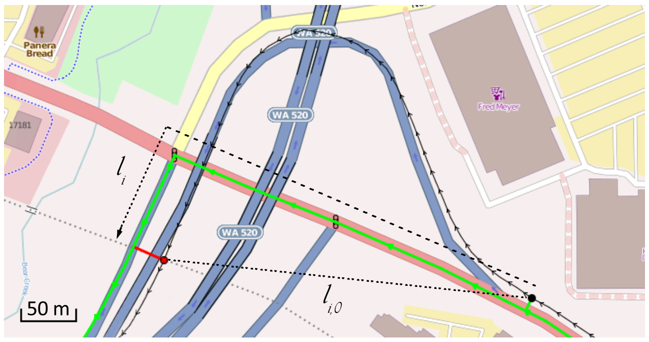


Figure 4: A problem with long sampling intervals.

programming algorithm to assign weights to these paths and output the one with the maximum weight. More specifically, for each free space line segment on the free space surface, corresponding to either an edge in G or a segment in Z, we store the intermediate value that is the maximum accumulated weight a monotone path ending at the segment can achieve. For the purpose of path reconstruction, we also store a pointer that indicates the immediate free space line segment where the max-weight path comes from. After the forward calculation, we find the max-weight line segment at the end of the free space surface, and backtrack to recover the path. Appendix A in our supplementary file includes the implementation details.

Finally, our algorithm allows a choice for the tunable weight function to use when scoring the paths of minimum Fréchet distance. Using the weight function of either Newson09 or Lou09, our algorithm outperforms them, respectively. However, Newson09 is not robust against varying sampling intervals, and Lou09 is generally less accurate. The example map matching error in Figure 4 illustrates how Newson09 degrades at longer sampling intervals. The term $\alpha(l_i - l_{i,0})$ in (2.2) increases when the sampling interval is longer. Since α is a constant, whenever there is a long delay between consecutive samples, Newson09 has a bias for "shorter" connecting paths over "closer" matches to the actual GPS points, leading to the mistaken output shown by the green path.

Therefore, we designed a weight function that improves upon Newson09 and Lou09:

$$\arg\min\,(x_0, x_1, \ldots, x_n) \sum_{i=0}^{n} t_i d_i^2 + \alpha l_i \tag{3.3}$$

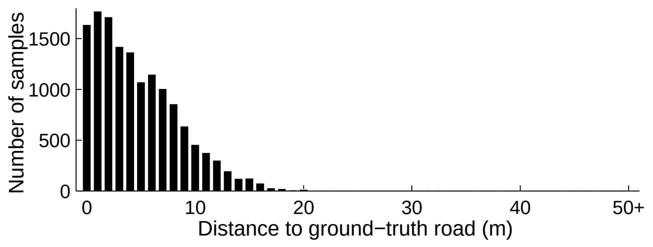
where t_i is the time interval between z_i and z_{i-1}. The rationale of our design is the following. The summation of l_i for a candidate sequence is exactly the length of the shortest path that links it, which does not change with the sampling interval. In addition, the expected value of the summation of $t_i d_i^2$ does not depend on the sampling interval either. Therefore, the ratio between these two terms remains the same for differing sampling intervals, and a constant α should perform consistently across all sampling intervals. Since (3.3) takes into account the time elapsed between consecutive samples, our algorithm is also robust against input traces with variable sampling rates or with occasionally missing GPS points, such as within long tunnels. Our experiments use this new weight function. Notice that at a 1 s sampling interval, (3.3) is equivalent to (2.2) for the same α because $t_i = 1$ and the summation of $l_{i,0}$ is a constant. We applied (3.3) to the Viterbi algorithm alone and its accuracy is on par with Newson09 and as robust as Lou09 across all sampling intervals; i.e., roughly the maximum of the two.

4 Experiments

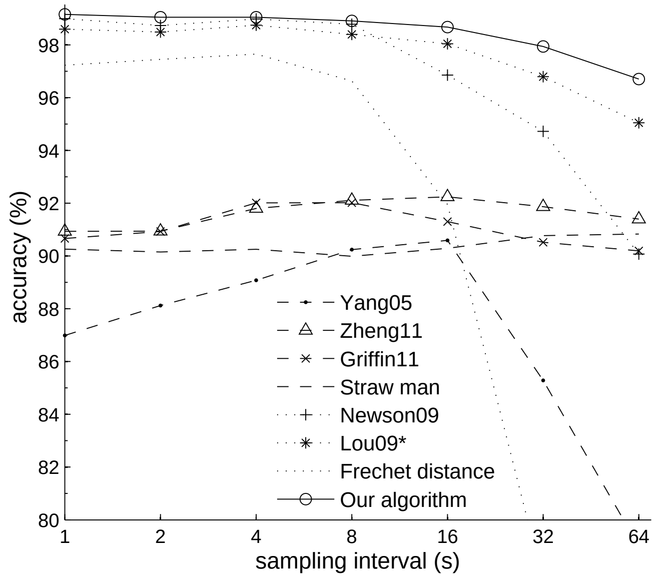
We first discuss our evaluation methodology and then show experimental results for each dataset.

4.1 Evaluation Methodology. We use both Seattle and Shanghai datasets for performance comparison. Both datasets were collected with a one second sampling interval. We subsample the traces to evaluate the performance at longer sampling intervals. For a given sampling interval of m seconds, we generate one subsequence starting with the first sample $(z_0, z_m, z_{2m}, \ldots)$, another subsequence starting with the second sample $(z_1, z_{m+1}, z_{2m+1}, \ldots)$, etc. Thus, there are m sets of traces used for each subsampling experiment. This way we fully utilize our datasets at long sampling intervals and reduce the variance in our results caused by different starting locations.

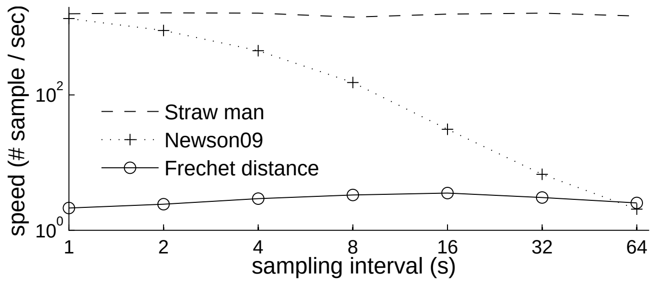
Of the max-weight algorithms listed in Table 1, we focus on those using the two key features: distance and shortest-path. Bearing and speed are not available in the Seattle dataset. Direction and length of line segments drawn between consecutive samples are useless at long sampling intervals. Connectivity can be derived from shortest-path, e.g., threshold filter. Therefore, we compare Yang05, Zheng11, and Griffin11 for incremental methods, and Lou09 and Newson09 for global methods. Lou09 does not use an HMM and aggregates weights by summation rather than multiplication. For our evaluation, we use an HMM variant that essentially replaces $l - l_0$ in (2.2) with l_0/l, denoted as Lou09* to avoid confusion. Our variant is more accurate than the original algorithm in our experiments. For each max-weight algorithm we compare, we tune its constant parameters in the weight function to maximize its accuracy for the Seattle dataset at the one second sampling interval. Then we fix the values for the rest of experiments. In addition to the weight function, there are two other important parameters for all max-weight al-



(a) Noise histogram



(b) Accuracy comparison



(c) Computation speed

Figure 5: Experiments using the Seattle dataset.

gorithms. The first is the search radius to find candidates for each sample. A small radius may miss the ground truth location, and a large radius significantly slows down the algorithm because the number of candidates grows quadratically. Another parameter is the noise filtering threshold discussed in Section 2.2. We discuss these two parameters later in more detail for each dataset.

4.2 Seattle Dataset. Our first dataset is from the SIGSPATIAL Cup 2012, which contains 14,436 samples collected from the Seattle area [1]. Each sample consists of a timestamp and a longitude-latitude pair. The ground truth indicates the ID of the road each sample matches to. We discovered five sections of wrong matches in the ground truth, for a total of 78 samples affected; see Appendix B for discussion. We manually corrected these errors for our experiments. The road map is exported from OpenStreetMap (OSM), which represents roads as polylines. The ground-truth of the dataset matches each sample to a road ID, which typically represents a section of road in-between two adjacent intersections. Figure 5a shows a histogram of the distance between each sample and its ground-truth road. The maximum error is 25.5 m and the median is 4.47. Therefore we set the candidate search radius for max-weight methods to be 50 m, roughly twice the maximum error.

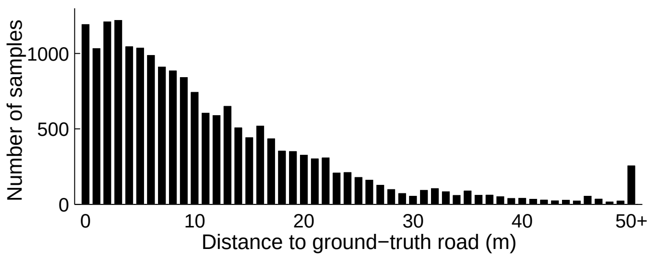
Figure 5b shows the accuracy of each algorithm as we increase the number of seconds between GPS samples. For reference, we include a straw-man algorithm that simply matches each sample to the nearest road. The incremental methods (each shown using long-dashed lines) are not substantially better than the straw-man algorithm, and they are much less accurate than global methods (upper cluster). We found they suffer from numerous freeway Y-splits, such as the one in Figure 2a.

The top curve shows that our algorithm dominated in accuracy across different sampling intervals. The global algorithms each degrade at long sampling intervals. The Fréchet distance algorithm degrades due to the alternative path problem we discussed with respect to Figure 2c. Newson09 degrades because of the weight function design, as we illustrated with Figure 4. The accuracy of max-weight methods is very sensitive to the choice of the noise filtering parameter, as explained in Section 2.2. We tested different values using our Seattle dataset, and used the optimal value 10 m for our experiments. This value is sufficient to classify the red samples in Figure 2b as noise and match them correctly. However, it is too large for normal but slowly moving samples, which are considered noise and matched to the previous road segments. We use the value for the overall best accuracy. Without this noise filtering, the accuracy of both Newson09 and Lou09* drops to only 92% at one second sampling interval. The Fréchet distance algorithm and our algorithm do not need noise filtering.

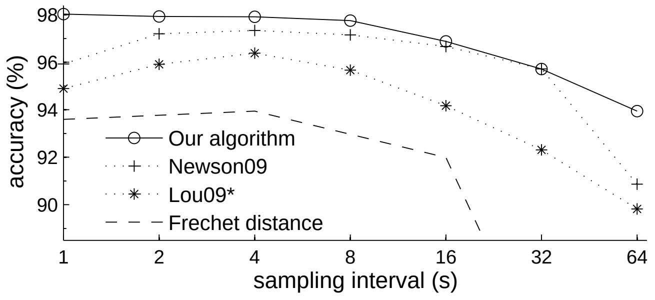
Figure 5c shows the computation speed of representative algorithms from different categories. The figure is only meant to roughly compare their speeds and to show their trends as the GPS sampling interval increases. The exact performance numbers are specific to our implementation. In general, incremental max-weight methods are faster than global max-weight methods, and the latter is in turn faster than global geometric methods. Nearest match identifies the nearest road for each sample independently, and therefore its processing throughput is not affected by the sampling interval. Global max-weight methods slow down linearly as the interval increases, because the shortest-path calculation takes longer as consecutive samples become further apart. Finally, the computation speed of map matching based on Fréchet distance is almost three orders of magnitude slower than nearest match, but its speed is not significantly affected by the sampling interval. The computation time of our algorithm is essentially the summation of Fréchet distance and Newson09, where the former dominates the overall computation. There are various techniques to accelerate Fréchet distance calculation [9, 29]. We focus on map matching accuracy in this paper and do not exploit these optimization techniques.

4.3 Shanghai Dataset. We collected the second dataset ourselves from Shanghai, including a total of 19,080 samples [2]. There is a large concentration of high-rise buildings, elevated roads, and tunnels in Shanghai, which leads to substantial GPS noise. The Shanghai road network is also complex, including narrow streets, tight curves, close parallel roads, 5-way and 6-way intersections, and double-layered roads [28]. All these problems lead to a noisy real-world test dataset that is substantially more challenging than existing public datasets. Figure 6a shows the histogram of GPS noise; the large variation in comparison with the Seattle dataset in Figure 5a is evident. The maximum error is 133 m and the median is 8.97 m. We therefore set the candidate search radius to 250 m, roughly twice the maximum and five times the value for the Seattle dataset.

Under high-noise data, the performance differences among different algorithms are magnified. Incremental methods perform much worse compared with global methods. Nearest match achieves only 79% accuracy, and other incremental methods perform similarly. To simplify the presentation and fit the page limit, we omit the results for incremental methods and show only global methods in Figure 6b. Again, our algorithm performs the best, followed by Newson09, Lou09*, and the Fréchet distance algorithm. Interestingly, both global max-weight methods achieve their best accuracy at a four second interval, because it eliminates some of the random GPS noise, such as we showed in Figure 2b. We used the same noise filtering threshold 10 m for our Shanghai dataset. We found the optimal threshold is 15 m, which brings the accuracy of Newson09 closer to the result of our algorithm. Without the filtering, Newson09 drops to only 83%. In general, the performance of max-weight methods is very sensitive to the choice of various parameters, whose optimal values can vary from dataset to dataset. In contrast, the Fréchet distance calculation of our algorithm is parameter-free. The global weight optimization step requires the tuning of the weight function, which is relatively robust.



(a) Noise histogram



(b) Global methods

Figure 6: Experiments using the Shanghai dataset.

5 Conclusion

Using the framework we developed to make sense of the extensive literature on map matching, we have been able to test and compare a variety of methods empirically using two diverse test datasets. With our understanding of where and how different algorithms fail, we have been able to develop a new algorithm that enjoys the benefits of global max-match methods as well as global geometric methods. In testing, it dominated accuracy across the spectrum of sampling intervals—particularly under higher noise situations. This makes it the leading candidate for use on much real-world GPS data, such as volumes of fleet-management data, where the sampling interval is necessarily lower to make the data and transmission costs manageable. The improvement in accuracy comes at the cost of additional computation effort. Such trade-offs are commonplace in computer science generally, and each application may choose based on the relative value of runtime vs. accuracy, with the optimal decision boundary moving ever toward higher accuracy as computing hardware improves exponentially.

References

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

[2] Shanghai GPS dataset. `http://grid.sjtu.edu.cn/mapmatching/data/`.

[3] 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.

[4] H. Alt and et al. Matching planar maps. *Journal of Algorithms*, 49(2):262–283, 2003.

[5] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geometry Appl.*, 5:75–91, 1995.

[6] A. Biem and et al. IBM Infosphere streams for scalable, real-time, intelligent transportation services. In *ACM SIGMOD*, 2010.

[7] 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.

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

[9] D. Chen and et al. Approximate map matching with respect to the Fréchet distance. In *Workshop on Algorithm Engineering and Experiments*, 2011.

[10] J. Eriksson and et al. The pothole patrol: using a mobile sensor network for road surface monitoring. In *MobiSys*, 2008.

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

[12] T. Griffin and et al. Routing-based map matching for extracting routes from GPS trajectories. In *International Conference on Computing for Geospatial Research & Applications*, 2011.

[13] B. Hoh and et al. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *MobiSys*, 2008.

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

[15] Z. Liao. Real-time taxi dispatching using global positioning systems. *Commun. ACM*, 46(5):81–83, 2003.

[16] K. Liu and et al. Feasibility of using taxi dispatch system as probes for collecting traffic information. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, 13(1):16–27, 2009.

[17] S. Liu and et al. Towards mobility-based clustering. In *ACM KDD*, 2010.

[18] Y. Lou and et al. Map-matching for low-sampling-rate GPS trajectories. In *SIGSPATIAL GIS*, 2009.

[19] F. Marchal and et al. Efficient map matching of large global positioning system data sets. *Transportation Research Record*, 1935(1):93–100, 2005.

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

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

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

[23] M. A. Quddus. *High Integrity Map Matching Algorithms for Advanced Transport Telematics Applications.* PhD thesis, Imperial College, London, 2006.

[24] M. A. Quddus and et al. A general map matching algorithm for transport telematics applications. *GPS Solutions*, 7(3):157–167, 2003.

[25] M. A. Quddus, R. Noland, and W. Ochieng. The effects of navigation sensors and spatial road network data quality on the performance of map matching algorithms. *GeoInformatica*, 13:85–108, 2009.

[26] A. Thiagarajan and et al. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *SenSys*, 2009.

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

[28] Y. Wang and et al. Challenges and opportunities in exploiting large-scale GPS probe data. Technical Report HPL-2011-109, HP Labs, 2011.

[29] C. Wenk and et al. Addressing the need for map-matching speed. In *Scientific and Statistical Database Management Conference*, 2006.

[30] C. E. White and et al. Some map matching algorithms for personal navigation assistants. *Transportation Research Part C*, 8(1–6):91–108, 2000.

[31] J. Yoon, B. Noble, and M. Liu. Surface street traffic estimation. In *MobiSys*, 2007.

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

[33] H. Zhu and et al. Seer: Metropolitan-scale traffic perception based on lossy sensory data. In *INFOCOM*, 2009.

A Implementation

Compared to the original map matching algorithm based on Fréchet distance [4], our algorithm simplifies the principal subroutine and also adds an additional step that finds the max-weight path among all minimum Fréchet distance paths, which we discuss next.

A.1 Fréchet distance calculation. We introduce a few more notions for the convenience of explanation. Given a free space diagram, we call the region of free space *white*, and the region of non-free space *black*. The free space diagram of two line segments is called a *cell*, which contains at most one full or partial white ellipse. Each of the four edges of a cell has at most one white interval. The free space diagram of an edge (u, v) in graph G and a polyline $Z = (z_0, \dots, z_n)$ is a sequence of n cells. In the i-th cell, the lower white interval on FD^u (corresponding to vertex u and line segment (z_{i-1}, z_i)) is denoted as I^u_{i-1}, and the upper white interval on FD^v (corresponding to vertex v and (z_{i-1}, z_i)) is denoted as I^v_{i-1}. The left white interval (corresponding to edge (u, v) and sample z_{i-1}) is denoted as $I^{(u,v)}_{i-1}$, and the right interval (corresponding to (u, v) and z_i) is denoted as $I^{(u,v)}_i$.

Given a free space surface of graph $G = (V, E)$ and polyline $Z = (z_0, \dots, z_n)$, we orient it such that segments corresponding to Z are *horizontal* and segments corresponding to E in G are *vertical*, i.e., I^v_i is horizontal and $I^{(u,v)}_i$ is vertical. The free space surface therefore can be viewed as $n + 1$ vertical copies of G, denoted as $G_0, \dots, G_n$, connected by $|V|$ horizontal lines.

Given a distance ε, the following procedure determines whether there is a path on G with Fréchet distance at most ε to Z.

1. Construct the free space surface and identify the white space by calculating all white intervals of all cells.

2. Calculate the sub white space that is monotone according to the topology of G (following the direction of edges in E) and montone from G_0 to G_n (following the direction of Z).

3. The path exists if and only if there exists a white interval on G_n.

With the above procedure that solves the decision problem, the optimization problem of finding the minimum ε is solved by parametric or binary search. The most challenging step is the calculation of the monotone white space, which is exactly the region reached by all monotone paths starting from white intervals on G_0. Using the analogy of a man walking a dog along two different curves, a path is monotone if and only if neither the man nor the dog moves backward. Algorithm 1 is our simplified procedure that calculates the monotone white space.

Algorithm 1 Calculating the monotone white space

```
Input: A free space surface for trace $(z_0, \dots, z_n)$ and
    graph $G = (V, E)$, with all (non-monotone) white
    intervals calculated
Output: Updated white intervals marking the mono-
    tone white space
 1: for $i = 0, \dots, n - 1$ do
 2:   for all edge $(u, v) \in E$ do
 3:     if vertical interval $I^{(u,v)}_i$ is not empty then
 4:       mark horizontal interval $I^v_i$ as visited
 5:     end if
 6:   end for
 7:   sweep from $G_i$ to $G_{i+1}$ and update horizontal
      intervals not visited yet from already visited
      horizontal intervals, according to the topology of
      $G$, and mark them as visited.
 8:   for all vertex $u \in V$ do
 9:     if horizontal interval $I^u_i$ is not visited then
10:       empty $I^u_i$
11:     end if
12:     for all vertex $v \in V$, $(u, v) \in E$ do
13:       update vertical interval $I^{(u,v)}_{i+1}$ from vertical
          interval $I^{(u,v)}_i$ and horizontal interval $I^u_i$
14:     end for
15:   end for
16: end for
```

The white intervals on G_0 do not need to be updated since monotone paths start from there. For each step from G_i to G_{i+1}, we first find all non-empty vertical intervals on G_i and mark their adjacent horizontal intervals as visited because they can be reached for the vertical intervals directly via monotone paths. We also need to update horizontal intervals that can be reached from horizontal intervals only. We use a sweep line algorithm here to avoid circular updates. Finally, with all horizontal intervals between G_i and G_{i+1} updated, we can update the vertical intervals at G_{i+1} by these horizontal intervals and the vertical intervals at G_i. Unlike a horizontal interval that can be shared by many adjacent cells, a vertical interval corresponds to exactly one cell and therefore can only be reached from either the bottom or the left edge of the cell.

To find the Fréchet distance, we need to calculate all candidate values and apply parametric or binary search. With complex road networks and long GPS traces, there can be trillions of potential values. Both the value calculation and binary search can be very time consuming. Our implementation saves a great deal of time by stopping early, without finding the exact Fréchet distance. Instead, we start with a large enough distance and binary search the space of real numbers. The procedure stops when the remaining interval is shorter than a threshold. We set the threshold to 1 m, since roads are at least 10 m apart in all the maps we have seen. In our experiments, this approximation always produces the same path as the exact algorithm does.

A.2 Finding Max-Weight Path on Free Space Surface. After the binary search finds an approximate minimum Fréchet distance, there can still be multiple monotone paths on the free space surface with equal Fréchet distance to the polyline Z. We apply a Viterbi-like dynamic programming algorithm to compute a weight for each path and return the one with the maximum weight.

For the purpose of illustration, our weight function uses only distance d and shortest-path l among all features in Table 1. It can be extended to include other features. For generality, we assume the weight function to be maximized is $\sum_{i=0}^{n}(x(d_i)+y(l_i))$, where functions $x(\cdot)$ and $y(\cdot)$ are user specified. Algorithm 2 shows the procedure.

We first set the weight for each white interval $I_0^{(u,v)}$ on G_0, which is calculated from the distance between z_0 and the edge corresponding to $I_0^{(u,v)}$, denoted as $d_0^{(u,v)}$ in Line 2 of the algorithm. The weight is set to negative infinity if $I_0^{(u,v)}$ is an empty interval. The main forward loop from Lines 4 to 12 assigns the maximum weight to each white interval via dynamic programming. Lines 5 to 7 assign the initial weight to a horizontal interval I_{i-1}^{v} by comparing all the adjacent cells of I_{i-1}^{v}, corresponding to adjacent edges of v in G. We add the weight calculated from the length of (u,v) to the cell corresponding to (u,v) since the path has moved from vertex u to v. Line 8 updates the weights of horizontal intervals from other horizontal intervals using the initial weights and following the topology of G, similar to Dijkstra's shortest-path algorithm. This is because the maximum weight may be achieved by a path coming from an adjacent horizontal interval rather than the previous vertical interval. Lines 9 to 11 update a vertical interval on G_i from the parallel vertical interval and the bottom horizontal interval of the cell it belongs to. Finally, Line 13 reconstruct the paths backward, from the max-weight interval on G_n to the interval on G_0, following the pointer stored at each interval, which we omitted in the pseudocode.

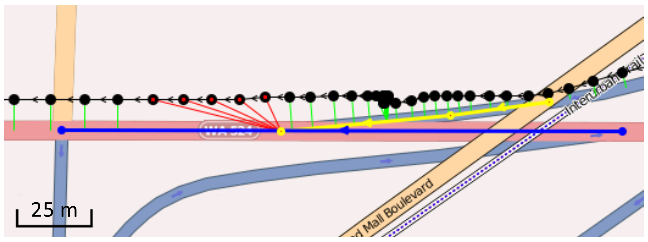
Algorithm 2 Finding the max-weight path on a free space surface via dynamic programming

```
Input: A free space surface for trace $(z_0,\dots,z_n)$ and
    graph $G=(V,E)$, with the monotone white space
    calculated, and the global weight function to maxi-
    mize $\sum_{i=0}^{n}(x(d_i)+y(l_i))$
Output: The max-weight path on the surface
 1: for all edge $(u,v)\in E$ do
 2:     $I_0^{(u,v)}.weight = x(d_0^{(u,v)})$
 3: end for
 4: for $i=1,\dots,n$ do
 5:     for all vertex $v\in V$ do
 6:         $I_{i-1}^{v}.weight =$
            $\max_{\{u|(u,v)\in E\}}\left(I_{i-1}^{(u,v)}.weight + y((u,v).length)\right)$
 7:     end for
 8:     update $I_{i-1}^{v}.weight$ for all $v\in V$ from each other
        based on the topology of $G$ using a procedure
        similar to Dijkstra's shortest-path algorithm
 9:     for all edge $(u,v)\in E$ do
10:         $I_i^{(u,v)}.weight =$
            $\max\left\{I_{i-1}^{u}.weight, I_{i-1}^{(u,v)}.weight\right\} + x(d_i^{(u,v)})$
11:     end for
12: end for
13: return max-weight path reconstruction
```

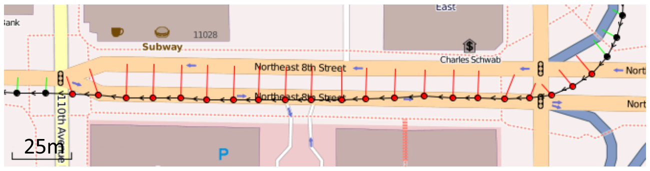
Recovering the matched location for each GPS sample from the reconstructed monotone path is straightforward. A monotone path goes through $G_0,\dots,G_n$. Assume at G_i, the vertical interval the path goes through is $I^(u,v)_i$, then sample z_i is matched to road (u,v). The sequence of white horizontal intervals (corresponding to vertices in G) that the monotone path goes through forms a path in G, which is the highest weight path with Fréchet distance no more than ε to trace Z.

B Ground-truth errors in SIGSPATIAL Cup dataset

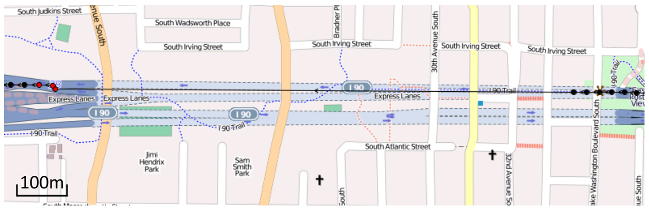
There are five sections of wrong matches in the ground truth of the Seattle dataset from the SIGSPATIAL Cup 2012, for a total of 78 samples affected, which amounts to an 0.5% error penalty for a perfect map matching output. Figure 7a shows an error due to the inaccurate map. In this figure, ramp (yellow) is mistakenly disconnected from the local road (blue) it leads to. This local road contains only one line segment and the ramp leads to the middle of it. Apparently an intersection is missing. For our experiments we split the trace into two parts at the disconnection. The remaining four errors are due to mislabeling in the ground truth. Figure 7b shows an example where



(a) Yellow & blue roads are mistakenly disconnected.



(b) Red samples match to the upper road in actuality, but the dataset assigns them to the lower road in the reverse direction.



(c) Red samples to the left of the tunnel are mistakenly assigned to the lower road, which is disconnected from the upper parallel road where samples before and after are correctly matched.

Figure 7: Ground truth errors in the SIGSPATIAL Cup dataset

the GPS points have been mislabeled as matching the lower road. However, this is a one-way road in reverse direction. The upper road should be the match. Figure 7c shows another ground truth mismatch. The three red samples to the left of the tunnel are matched to a road that is disconnected and parallel to the road continuing to the left where adjacent samples are correctly matched. We manually corrected the ground truth for these errors for our experiments. We have reported the errors to the curators of the SIGSPATIAL Cup data for future correction.