Hidden Markov Map Matching Through Noise and Sparseness

Paul Newson and John Krumm
Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052 USA
+1 425 705 4507, +1 425 703 8283
{pnewson, jckrumm}@microsoft.com

ABSTRACT
The problem of matching measured latitude/longitude points to roads is becoming increasingly important. This paper describes a novel, principled map matching algorithm that uses a Hidden Markov Model (HMM) to find the most likely road route represented by a time-stamped sequence of latitude/longitude pairs. The HMM elegantly accounts for measurement noise and the layout of the road network. We test our algorithm on ground truth data collected from a GPS receiver in a vehicle. Our test shows how the algorithm breaks down as the sampling rate of the GPS is reduced. We also test the effect of increasing amounts of additional measurement noise in order to assess how well our algorithm could deal with the inaccuracies of other location measurement systems, such as those based on WiFi and cell tower multilateration. We provide our GPS data and road network representation as a standard test set for other researchers to use in their map matching work.

Categories and Subject Descriptors
I.5.1 [**Computing Methodologies**]: Pattern Recognition, -- *Models* (Statistical)

General Terms
Algorithms, Measurement.

Keywords
Map matching, road map, location, driving routes.

1. INTRODUCTION
Map matching is the procedure for determining which road a vehicle is on using data from sensors. The sensors almost always include GPS because of its nearly ubiquitous availability. Map matching has been important for many years on in-vehicle navigation systems which must determine which road a vehicle is traversing in real time. More recently, map matching is becoming important as vehicles are used as traffic probes for measuring road speeds and building statistical models of traffic delays. These models, in turn, can be used to find time-optimal driving routes that avoid traffic jams. Data from such traffic probes has been used in the commercial routing engines of Microsoft [6], Dash [7], and Inrix [8]. Map matching is also growing in importance for research in route prediction [11], interpreting GPS traces [1], and activity recognition [14].

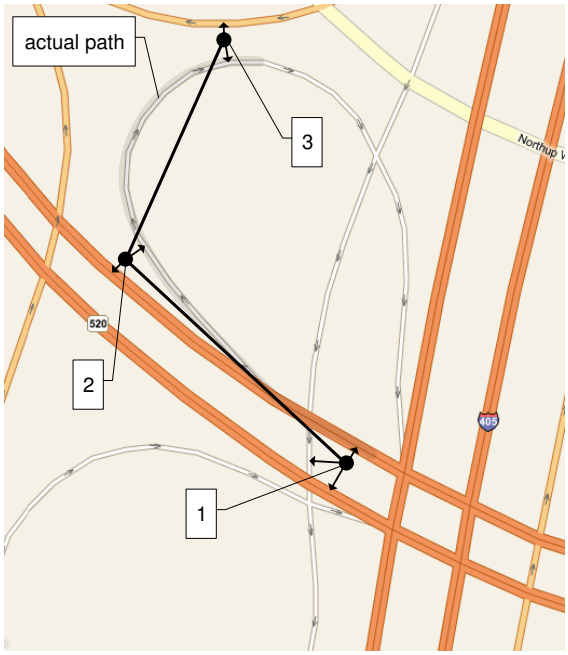

Figure 1: Map matching consists of matching measured locations (black dots) to the road network in order to infer the vehicle's actual path (light gray curve). Merely matching to the nearest road is prone to mistakes.

This paper makes three contributions to the research in map matching. First, it presents a new map matching algorithm based on the Hidden Markov Model (HMM). While the HMM has been used before in map matching, *e.g.* by Hummel [9], our formulation is novel in some important respects, detailed subsequently. We place particular emphasis on maintaining a principled approach to the problem while simultaneously making the algorithm robust to location data that is both geometrically noisy and temporally sparse. Our second contribution is a test of our map matching algorithm where we vary the levels of noise and sparseness of the sensed location data over a 50 mile urban drive. Varying the amount of noise lets us intelligently speculate about how map matching would work with less accurate location

sensors, like multilateration from cell towers and WiFi access points. Varying the sampling rate shows the minimum amount of data required for good map matching. This is important for practitioners who must decide on how often their GPS receivers should sample location data, which affects requirements for memory and bandwidth. Our third contribution is that we make our GPS data, ground truth, and road network representation publicly available for other researchers to use in their map matching work. We believe this is the first time that such a data set has been made publicly available. Until now, all the work on map matching used private data sets for testing, making it impossible to objectively compare results from different algorithms.

2. THE MAP MATCHING PROBLEM

The map matching problem is illustrated in Figure 1. There are three measured locations in sequence shown as black dots. The problem is to find which roads the vehicle was on. The most obvious algorithm is to simply match each point with the nearest road. Due to measurement noise, however, this algorithm is prone to error. In the illustration, the actual path is obvious, but the 2nd and 3rd point would be mismatched if they were associated with the nearest road. Even using modern GPS receivers, we have observed gross outliers and extended sequences of of erroneous points, likely due to urban canyons and other terrestrial features that affect GPS signals. Because of problems like this, modern map matching takes into account sequences of points before deciding on a match. In the example, there is really only one reasonable path on the road network that could have produced the observed measurements.

In our work, like most other map matching work, the raw input data consists of vehicle locations measured by GPS, as shown in Figure 2. Each measured point consists of a time-stamped latitude/longitude pair. The roads are also represented in the conventional way, as a graph of nodes and edges. The nodes are at intersections, dead ends, and road name changes, and the edges represent road segments between the nodes. Some edges are directional to indicate one-way roads. Each node has an associated latitude/longitude to indicate its location, and each edge has a polyline of latitude/longitude pairs to represent its geometry.

Since point-by-point, nearest road matching often fails, researchers have developed methods that match several points at once. One way to do this is to create a (possibly smoothed) curve from the location measurements and attempt to find matching roads with similar geometry. As an example, White *et al.* [16] present four algorithms, starting with the simple, nearest match scheme. Their second algorithm adds orientation information to the nearest match approach, comparing the measured heading to the angle of the road. Their third algorithm evolves the second algorithm to include connectivity constraints, and their fourth algorithm does curve matching. They were surprised to discover that their most sophisticated algorithm, the fourth one, was outperformed by the simpler second algorithm when tested on a total of about 17 km of driving data. Another purely geometric approach comes from Greenfeld [5], whose algorithm builds up a topologically feasible path through the road network. Matches are determined by a similarity measure that weights errors based on distance and orientation. The algorithm was found to perform

Figure 2: This is the GPS data we used for testing in the Seattle, Washington, USA area. The trip starts in the upper right near Marymoor Park. It consists of 7531 GPS points sampled at 1 Hz, and it covers about 80 kilometers (50 miles) over about 2 hours.

flawlessly, even though the GPS data was collected while Selective Availability was turned on, leading to noisier location measurements than are available now. Kim and Kim [10] look at a way to measure how much each GPS point belongs to any given road, taking into account its distance from the road, the shape of the road segment, and the continuity of the path. The measure is used in a fuzzy matching scheme with learned parameters to optimize performance. One of the most sophisticated geometric matching algorithms is from Brakatsoulas *et al.* [3]. Their algorithm uses variations of the Fréchet distance to match the curve of the GPS trace to candidate paths in the road network. They tested their algorithms on 45 routes in Athens, Greece. Alt et al. [2]give a generalization of the Fréchet for matching curves.

One potential problem with purely geometric approaches is their sensitivity to measurement noise and sampling rate. Clearly, connecting the dots of a set of noisy measurements sampled at a slow rate would not match well with the road geometry, especially direction information. Hidden Markov Models (HMM) solve this problem by explicitly modeling the connectivity of the roads and considering many different path hypotheses simultaneously. One of the earliest applications of the HMM to map matching is from Lamb and Thiébaux [13] who use a combination of a Kalman filter and HMM. Several Kalman filters track the vehicle along different hypothesized paths, and the HMM chooses between them. Other work from Hummel [9] and Krumm *et al.* [12] use an HMM to balance the measurement noise and path probabilities. We will compare and contrast this work with ours subsequently after we explain the details of our algorithm.

3. HMM MAP MATCHING

As illustrated in Figure 1, the key problem in map matching is the tradeoff between the roads suggested by the location data and the feasibility of the path. While the location data is important as the sole indicator of the path, naively matching each noisy point to the nearest road will result in extremely unreasonable paths involving strange U-turns, inefficient looping, and overall bizarre driving behavior. To avoid unreasonable paths, we can introduce knowledge of the connectivity of the road network to help pull the solution away from clearly bizarre behavior. The Hidden Markov Model is an algorithm that can smoothly integrate noisy data and path constraints in a principled way.

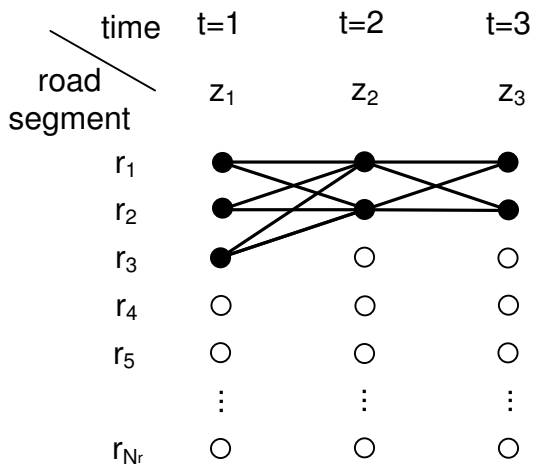

Figure 3: For each measurement z_t, the HMM considers all the road segments r_i as well as all the transitions between the road segments.

The HMM models processes that involve a path through many possible states, where some state transitions are more likely than others and where the state measurements are uncertain. In speech understanding, HMMs are used to model the time sequence of spoken phonemes. The model fits well, because some phoneme-to-phoneme transitions are more likely than others, and because classifying each individual phoneme from microphone measurements is not 100% accurate.

In our map matching algorithm, the states of the HMM are the individual road segments, and the state measurements are the noisy vehicle location measurements. The goal is to match each location measurement with the proper road segment. This state representation naturally fits the HMM, because transitions between road segments are governed by the connectivity of the road network.

More formally, the discrete states of the HMM are the N_r road segments, r_i , $i = 1 \ldots N_r$. In our representation, distinct road segments run between intersections. For each 2D latitude/longitude location measurement z_t, the goal is to find the road segment that the vehicle was actually on. Figure 3 shows an illustration of the HMM for the map matching problem illustrated in Figure 1. Here, each vertical slice represents a point in time corresponding to a location measurement z_t for the three times $t = 1, 2, 3$. At $t = 1$ there are three roads near z_1, shown as three black dots in the first column. There is a feasible driving path, possibly very circuitous, from each of the nearest points on these three roads to points on the two roads near z_2 at $t = 2$, and similarly for $t = 3$. The goal of our algorithm is to find the most probable path through the lattice by picking one road segment for each t. This path should be sensitive to both the measurements and the reasonability of the paths between the road segments. This tradeoff is made based on the probabilities governing the measurements and probabilities governing the transitions between the road choices at each time, which we describe next.

We note that our algorithm, as presented, solves map matching as a batch problem, after all the data has been collected. We speculate that a sliding window version of our algorithm would work well for real time map matching, say, on a vehicle's navigation system.

3.1 Measurement Probabilities

Measurement probabilities (also called emission probabilities) give the likelihood that a measurement resulted from a given state, based on that measurement alone. For map matching, given a location measurement z_t, there is an emission probability for each road segment r_i, $p(z_t|r_i)$. This gives the likelihood that the measurement z_t would be observed if the vehicle were actually on road segment r_i. Our intuition is that road segments farther from the measurement are less likely to have produced the measurement. For a given z_t and r_i, we denote the closest point on the road segment as $x_{t,i}$. An example of this notation is shown in Figure 4. The great circle distance on the surface of the earth between the measured point and the candidate match is $\|z_t - x_{t,i}\|_{great\ circle}$. For the correct match, this difference is due to GPS noise. Based on previous work [15], we can model GPS noise as zero-mean Gaussian, meaning that

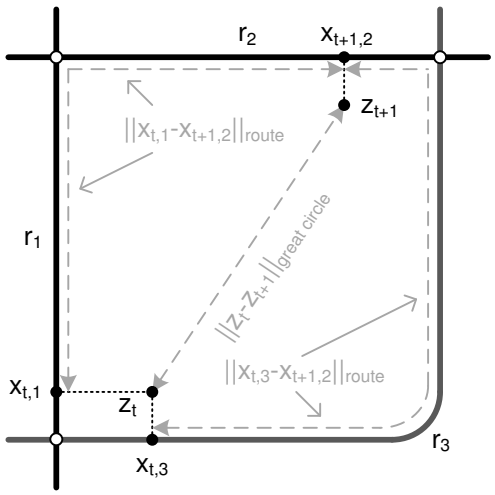

Figure 4: This shows an example of our notation. There are three road segments, r_1, r_2, and r_3, and two measured points, z_t and z_{t+1}. The first measured point, z_t, has candidate road matches at $x_{t,1}$ and $x_{t,3}$. Each match candidate results in a route to $x_{t+1,2}$, which is a match candidate for the second measured point, z_{t+1}. These two routes have their own lengths, as does the great circle path between the two measured points. Our data shows that the route distance and great circle distance are closer together for correct matches than for incorrect matches.

$$p(z_t|r_i) = \frac{1}{\sqrt{2\pi}\sigma_z} e^{-0.5\left(\frac{\|z_t - x_{t,i}\|_{great\ circle}}{\sigma_z}\right)^2} \quad (1)$$

Here σ_z is the standard deviation of GPS measurements, which we estimate in Section 5.2 (Parameter Estimation). While we know that GPS errors are not strictly Gaussian, this assumption proved effective in our map matching algorithm.

Another required probability is the initial state probabilities π_i, $i = 1 \ldots N_r$, which in the case of map matching gives the probability of the vehicle's first road over all the roads at the beginning of the drive. While some HMM formulations assign a uniform distribution to π_i, assuming no measurements have been taken, we start at the first measurement and have $\pi_i = p(z_1|r_i)$, *i.e.* using the first measurement z_1.

In practice, we do not consider matching to road segments that are quite distant from the measurement. In our algorithm, we set to zero any measurement probability from a road segment that is more than 200 meters away from z_t. This helps reduce the number of candidate matches that our algorithm has to consider, decreasing its running time. This is illustrated in Figure 3 with the unfilled circles representing road segments that are too far away to consider.

3.2 Transition Probabilities

Each measurement z_t has a list of possible road matches, as does the next measurement z_{t+1}. Transition probabilities give the probability of a vehicle moving between the candidate road matches at these two times. Intuitively, some transitions will be very unlikely, such as those requiring a complicated set of maneuvers. Practically, we favor transitions whose driving distance is about the same as the great circle distance between the measurements.

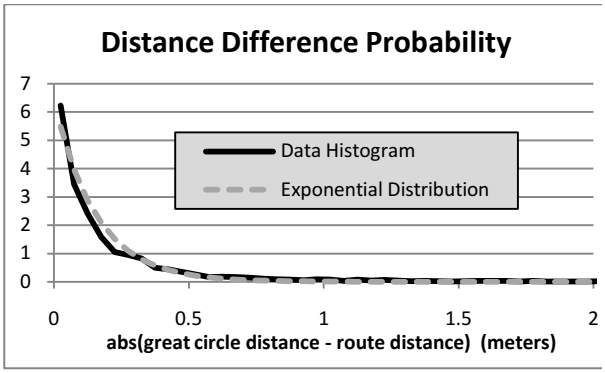

Figure 5: The histogram of $\left|\|z_t - z_{t+1}\|_{great\ circle} - \|x_{t,i} - x_{t+1,j}\|_{route}\right|$ follows an exponential probability distribution.

Specifically, for a measurement z_t and candidate road segment r_i, the latitude/longitude point on the road segment nearest the measurement is $x_{t,i}$. For the next measurement z_{t+1} and candidate road segment r_j, the corresponding point is $x_{t+1,j}$. We compute the driving distance between these two points using a conventional route planner configured to give the route with the shortest distance. We note that the correct pair of matched points typically results in a very short route, because the matched points on the road come from closely spaced GPS points. We refer to this driving distance as the "route distance", notated as $\|x_{t,i} - x_{t+1,j}\|_{route}$. We compare the route distance to the great circle distance between the measured points, $\|z_t - z_{t+1}\|_{great\ circle}$. Figure 4 shows an example of these distances.

Our intuition, confirmed by experiment, is that these two distances will be about the same for correct matches. This is because the relatively short distance traveled on the road(s) between a pair of correct matches will be about the same as the distance between the measured GPS points. We confirmed this by looking at the ground truth matched roads, which we detail in Section 5 (GROUND TRUTH DATA). We computed a histogram of the absolute values of the differences between the great circle distances and the route distances from the correct matches, shown in Figure 5. This histogram fits well to an exponential probability distribution given by Equation (2):

$$p(d_t) = \frac{1}{\beta} e^{-d_t/\beta} \quad (2)$$

Here

$$d_t = \left|\|z_t - z_{t+1}\|_{great\ circle} - \|x_{t,i^*} - x_{t+1,j^*}\|_{route}\right| \quad (3)$$

where i^* and j^* indicate the ground truth road segments of the route that we describe in Section 5. We estimate the value of β in Section 5.2 (Parameter Estimation).

3.3 Optimal Path

With measurement probabilities from Equation (1) and transition probabilities from Equation (2), we used the Viterbi algorithm to compute the best path through the HMM lattice. The Viterbi algorithm uses dynamic programming to quickly find the path through the lattice that maximizes the product of the measurement probabilities and transition probabilities. This gave us an

inference of the correct road segment for each location measurement.

Having explained our basic algorithm, we can now compare it to two similar algorithms in the research literature. Hummel [9] used an HMM for map matching. Her measurement probabilities used the same Gaussian GPS noise assumption as ours, but she also added a term for the heading mismatch between the vehicle and the road. We did not use heading data, mostly because we are assuming we have no compass data from the vehicle. While heading can be computed from measured GPS points, this can be very inaccurate when there is a long interval between GPS measurements, which is a condition we investigated as part of our research. Our transition probabilities are different from Hummel's in that ours take into account the empirical difference between great circle distances and route distances, while Hummel uses a simpler model that only accounts for roads immediately adjacent to the current match candidate.

Compared to the work of Krumm *et al.* [12], our main difference is in the transition probabilities. While we look at route distance differences, Krumm *et al.* looked at route time differences. Time differences are much more sensitive to traffic conditions, so are likely less reliable than distance differences. They are also sensitive to the peculiarities of the given route-finding algorithm, with particular values to assess the time cost of turns and stops at intersections. Also, in this paper's algorithm, we test against ground truth, test against inaccurate and subsampled GPS data, make our data available for other researchers, and give several practical implementation details, described in the next section.

4. ALGORITHM PARTICULARS

The HMM formulation described above represents a principled approach to balancing the effects of measurement noise and routing behavior. In practice, the algorithm can be made to work better and faster with some simple enhancements that we discovered by working with real GPS data.

4.1 Preprocessing

Before the GPS points are used to construct the HMM, we move through the points in time sequence, removing points that are within $2\sigma_z$ of the previous included point. The justification for this step is that until we see a point that is at least $2\sigma_z$ away from its temporal predecessor, our confidence is low that the apparent movement is due to actual vehicle movement and not noise. This has the benefit of reducing the number of steps in the HMM for high sample rate data, which speeds processing. For our ground truth data described in Section 5, this step eliminated about 38.9% of the original data.

4.2 HMM Breaks

There are various conditions that induce a break in the HMM lattice where all the transition probabilities from one time step to the next are zero. We detail these conditions below, after which we give a simple method to work around these breaks, effectively removing unmatchable data.

The conditions that lead to a break are:

- **Route Localization.** In a pure implementation of the algorithm, every road in the road network would be considered as a potential match candidate, and a complete route would be calculated between every one of these match candidates. As mentioned in section 3.1, to avoid an unreasonable amount of computation, we set to zero the measurement probability of any road segment more than 200 meters away from the GPS point. From an implementation perspective, this means we simply can ignore their existence and not add them to the HMM. It is possible that there are GPS points in our data set that have no match candidates within 200 meters, which causes our implementation to have no match candidates for a particular time step in the HMM. This situation may arise when the vehicle is travelling on a road surface or parking structure that is not marked on the map. It may also arise if the GPS is experiencing particularly high noise, which we have seen when the vehicle enters a tunnel or an urban canyon.
- **Low Probability Routes.** Assuming a connected road network, it should be possible to find a route between any two road segments on the map. However, once the route distance $\left\|x_{t,i} - x_{t+1,j}\right\|_{route}$ becomes much larger than the great circle distance $\|z_t - z_{t+1}\|_{great\ circle}$, the transition probability $p(d_t)$ corresponding to that route becomes very small. This can happen when the routes become circuitous and strange. Rather than continue seeking for a route that is obviously incorrect, we terminate the search for a route when $\left\|x_{t,i} - x_{t+1,j}\right\|_{route}$ becomes greater than $\|z_t - z_{t+1}\|_{great\ circle}$ by 2000 meters or more, and assign a probability of zero.
- **GPS Outliers.** There are two more tests for reasonableness that we apply to any route calculated as a transition between HMM states. Since we know we are tracking ordinary vehicles on public streets, if a calculated route would require the vehicle to exceed a speed of 50 m/s (112 miles per hour, 180 kilometers per hour), or travel in excess of three times the posted speed limit, we consider the route to be unreasonable, and set its probability to zero.

In the pure HMM model, which considers all match candidates and all routes, however unlikely they may be, the Viterbi algorithm will always be able to find a complete optimal path through the HMM. However, because of the simplifications described above, which limit the number of match candidates and routes that will be considered, it is possible that there is no complete path through the HMM.

In examining our data, we found that it typically follows a pattern of long stretches of reasonably easy-to-match location measurements, interspersed with occasional, short sequences that are difficult to match, leading to the breaks described above. Our first solution to this problem was to manually remove the offending points, which was effective, but tedious. We automated this process in the following way. When a break is detected between time step t and time step $t+1$, we remove measured points z_t and z_{t+1} from the model, and check to see if the break has been healed. The break is considered healed if the measured points at $t-1$ and $t+2$ lead to a reconnection in the HMM after rechecking the points with the bulleted conditions above. If the break is still present, we continue to remove the points on either side of the break until either the break is healed, or the break is more than 180 seconds long. If the break exceeds this threshold, we split the data into separate trips and do map matching on each one separately. If the original data has a sampling period greater

than 180 seconds, we omit this heuristic and do not try to fix these breaks.

5. GROUND TRUTH DATA

The data for our test route was collected by driving a known, planned route in a vehicle containing a commercially available consumer grade GPS device with a logging feature that records the current latitude and longitude once per second. The device uses the SiRF Star III GPS chipset and is enabled with WAAS. A map of the route is shown in Figure 2. This route is about 80 kilometers (50 miles) long, and it took about 2 hours to drive. It resulted in 7531 time stamped latitude/longitude pairs. This high fidelity position data was then processed using a tool that applies the map match algorithm and displays the result graphically so it can be inspected and corrected if the algorithm has made an error. The result of this "hand match" constitutes our ground truth data. The scope of corrections available in the graphical tool is limited to choosing alternate HMM states or transitions. Note that the ground truth data is correct inasmuch that it represents the correct path taken by the vehicle through the road network. The exact location of the vehicle in the road network corresponding to each GPS point in the ground truth data is unknowable, and therefore the point selected as the match to each road, while reasonable, should not be considered ground truth. Only the path that was taken by the vehicle is known.

5.1 Degraded Data

We simulate degraded versions of the GPS data by removing points and adding Gaussian random noise. We simulated sampling periods of 2, 5, 10, 20, 30, 45, 60, 90, 120, 180, 240, 300, 360, 420, 480, 540, and 600 seconds in addition to the original 1-second data from our logger. We simulated random Gaussian noise with standard deviations of 10, 15, 20, 30, 40, 50, 75, and 100 meters. We also included test data with no noise added. Note that to simulate a particular noise level, we had to account for the fact that there is already some random noise in the original data. Fortunately, Gaussian noise is additive in the sense of the equations below, so we could simulate any amount of additional noise with knowledge of the original σ_z from the sensor.

$$\begin{aligned} X_1 &\sim N(\mu_1, \sigma_1^2) \\ X_2 &\sim N(\mu_2, \sigma_2^2) \\ (X_1 + X_2) &\sim N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2) \end{aligned} \qquad (4)$$

In our case, since we assume zero-mean noise, $\mu_1 = \mu_2 = 0$.

5.2 Parameter Estimation

Our HMM needs two probability-related parameters. One is σ_z, which is the standard deviation of Gaussian GPS noise. We estimate this starting with our z_t measurements. For each of these, we know the index $i^*$of the correctly matched road r_{i^*} from our manually matched ground truth data. Using the notation presented in Section 3, the point on r_{i^*} nearest z_t is x_{t,i^*}. If we assume this was the actual location of the vehicle, then $\|z_t - x_{t,i^*}\|_{great\ circle}$ is an estimate of the magnitude of the GPS error. The standard deviation of these values is our estimate of the GPS noise, σ_z. We estimated σ_z using the median absolute deviation (MAD), which is a robust estimator of standard deviation:

$$\sigma_z = 1.4826\ \text{median}_t\left(\left\|z_t - x_{t,i^*}\right\|_{great\ circle}\right) \qquad (5)$$

For our test data, this value was $\sigma_z = 4.07$ meters, which is a reasonable value for GPS noise.

The other probability parameter we need is β from the exponential distribution in Equation (2). This describes the difference between route distances and great circle distances. We estimated β with a robust estimator suggested by Gather and Schultze [4]:

$$\beta = \frac{1}{\ln(2)}\text{median}_t\left(\left|\|z_t - z_{t+1}\|_{great\ circle} - \|x_{t,i^*} - x_{t+1,j^*}\|_{route}\right|\right) \qquad (6)$$

Note that in Equations (5) and (6) we use x_{t,i^*} and x_{t+1,j^*}. The stared i^* and j^* indicate the ground truth road segment that we found by manually matching the measured GPS points.

The parameters σ_z and β are the two, basic, adjustable parameters for our map matching algorithm, and they explicitly represent the tradeoff between our trust in the location measurements and candidate routes. A larger value of σ_z, which measures noise in the location measurements, represents less trust in the location measurements. A larger value of β, which measures the difference between great circle distances and route distances, represents more tolerance of non-direct routes. In our work, we estimate these two parameters directly from the data. An alternative would be to find the values of σ_z and β that optimize performance of the algorithm. We leave this for future work.

5.3 Public Data Availability

Our GPS data, ground truth, and relevant road network are available on a public Web page[1]. We made this data available to facilitate the fair comparison of map matching algorithms. We believe this is the only public data set in existence for map matching.

6. RESULTS

We ran our algorithm on the test route shown in Figure 2. This 50-mile route was sampled at 1 Hz, giving 7531 time stamped latitude/longitude pairs. After removing points as described in Section 4.1 (Preprocessing), there were 4605 remaining. The result of running our algorithm is a road segment match for each point except for about 100 that were discarded due to breaks, as described in Section 4.2 (HMM Breaks).

We quantified the accuracy of the map matching by comparing the ground truth route to the route determined by our algorithm. In

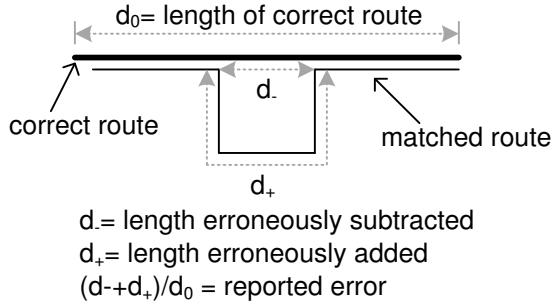

Figure 6: This illustrates how we measured the error between the correct route and the route from map matching.

[1] http://research.microsoft.com/en-us/um/people/jckrumm/MapMatchingData/data.htm

particular, we sum the lengths of incorrect road added to and subtracted from the correct route. We divide this sum by the length of the correct route to compute the fraction of incorrect route, which is the error value we report. This is shown in Figure 6. We chose this way to quantify accuracy over these other candidates:

- **Locations on Road**. This accuracy measure says that the matched point should be in the same location as the actual vehicle. Since we measured the vehicle's location with inherently noisy GPS, we do not know its actual location.
- **Road Segment**. This accuracy measure says that the matched point should be on the same road segment as the actual vehicle. While the correct road segment is easier to guess than the correct location, it is still ambiguous at intersections, where a noisy measurement could match to any of the roads converging at that point.

Our map matching algorithm gave exactly the same route as our ground truth in our test, which means it worked perfectly at a one second sampling period and with GPS accuracy location measurements.

We are interested in the performance of our algorithm with degraded input data, as described in Section 5.1 (Degraded Data). We degraded the data by subsampling and adding noise. Subsampling is interesting because it shows how robust our algorithm would be if the location sensor were to collect data at a slower rate. If the algorithm works well at lower sampling rates, this can lead to savings in bandwidth and storage for institutions that collect data with the intent to match it to roads. Figure 7 shows how our results degrade with subsampling. We note that the error is only 0.11% even when the sampling period grows to 30 seconds.

Added noise is interesting, because it gives an idea of how the algorithm would perform if the location sensor were less accurate, such as multilateration from WiFi access points or cell towers. The plot in Figure 8 shows how well our algorithm performs with added noise at different sampling periods. Surprisingly, it is more sensitive to noise with a 1 second sampling period than at longer periods. This is likely because frequent, noisy points tend to pull the route rather violently in different directions. At longer sampling periods, the algorithm shows robustness to measurement noise as high as 50 meters standard deviation, which is roughly the accuracy of WiFi-based multilateration.

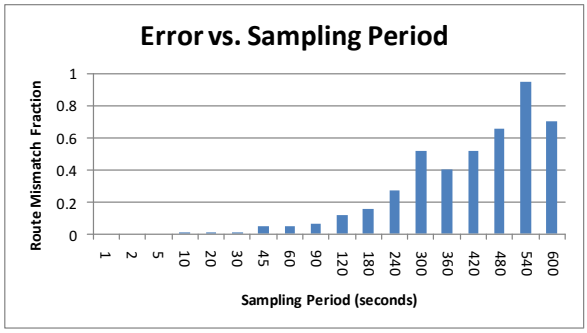

Figure 7: Our algorithm's performance is still quite good when the GPS samples are 30 seconds apart.

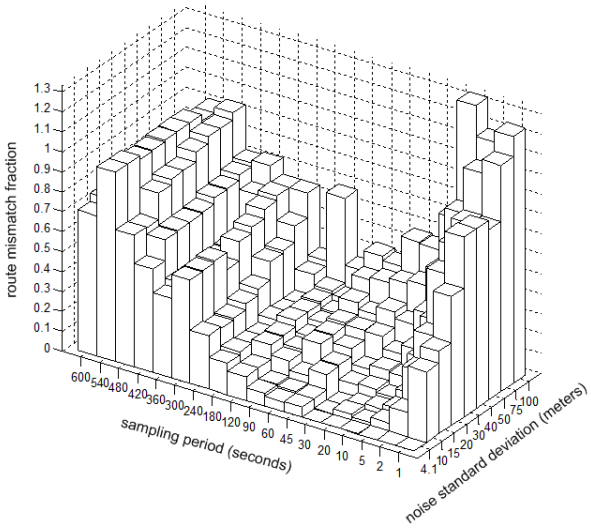

Figure 8: This shows how well our map matching algorithm performs with different sampling periods and noise on the location measurements. A lower value is better.

To our knowledge, these are the first reported tests of these kind for a map matching algorithm. We believe tests like this are important to assess when the algorithm breaks down, which in turn guides choices for how to sense the data.

7. CONCLUSIONS

As map matching becomes increasingly important for probing traffic and driving behavior, it is important to have principled, well-characterized map matching algorithms. We have presented a new algorithm based on the HMM that explicitly accounts for measurement noise and the feasible routes through the road network. We tested the algorithm on an 80-kilometer (50 mile) drive. Compared to manually matching the data, our algorithm performed perfectly. We also tested how the accuracy of our algorithm degrades when the location sampling rate decreases and when the measurement noise increases. Significantly, we found that even with 30 seconds between measured locations, the accuracy of our algorithm was barely degraded. We believe this is the first reported test of this kind for a map matching algorithm. Finally, we made our test data, ground truth data, and road network publicly available for other researchers to develop, test, and compare their own map matching algorithms.

REFERENCES

1. Agapie, E., et al., Seeing Our Signals: Combining Location Traces and Web-Based Models for Personal Discovery, in International Conference On Mobile Systems, Applications And Services (MobiSys 2009). 2008: Napa Valley, California, USA. p. 6-10.
2. Alt, H., et al., Matching Planar Maps. Journal of Algorithms, 2003. 49: p. 262–283.
3. Brakatsoulas, S., et al., On Map-Matching Vehicle Tracking Data, in 31st International Conference on Very Large Databases (VLDB 2005). 2005: Trondheim, Norway p. 853-864.

4. Gather, U. and V. Schultze, Robust Estimation of Scale of an Exponential Distribution. Statistica Neerlandica, 2001. 53(3): p. 327-341.
5. Greenfeld, J.S., Matching GPS Observations to Locations on a Digital Map, in 81th Annual Meeting of the Transportation Research Board. 2002: Washington, DC, USA.
6. http://www.bing.com/maps/. [cited 2009].
7. http://www.dash.net/. [cited 2009].
8. http://www.inrix.com/. [cited 2009].
9. Hummel, B., Map Matching for Vehicle Guidance, in Dynamic and Mobile GIS: Investigating Space and Time, J. Drummond and R. Billen, Editors. 2006, CRC Press: Florida.
10. Kim, S. and J.-H. Kim, Adaptive Fuzzy-Network-Based C-Measure Map-Matching Algorithm for Car Navigation System. IEEE Transactions on Industrial Electronics, 2001. 48(2): p. 432-441.
11. Krumm, J., A Markov Model for Driver Turn Prediction, in Society of Automotive Engineers (SAE) 2008 World Congress. 2008: Detroit, Michigan, USA.
12. Krumm, J., J. Letchner, and E. Horvitz, Map Matching with Travel Time Constraints, in Society of Automotive Engineers (SAE) 2007 World Congress. 2007: Detroit, Michigan, USA.
13. Lamb, P. and S. Thiebaux, Avoiding Explicit Map-Matching in Vehicle Location, in 6th World Conference on Intelligent Transportation Systems (ITS-99). 1999: Toronto, Canada.
14. Patterson, D.J., et al., Inferring High-Level Behavior from Low-Level Sensors, in Fifth Internation Conference on Ubiquitous Computing (UbiComp 2003). 2003, Springer. p. 73-89.
15. VanDiggelen, F., GNSS Accuracy: Lies, Damn Lies, and Statistics, in GPS World. 2007. p. 26-32.
16. White, C.E., D. Bernstein, and A.L. Kornhauser, Some map matching algorithms for personal navigation assitants. Transportation Reserach Part C: Emerging Technologies, 2000. 8(1-6): p. 91-108.