SAE 2007 World Congress, April 16-19, 2007, Detroit, MI

Paper Number 2007-01-1102

Map Matching with Travel Time Constraints

John Krumm
Microsoft Research

Julie Letchner
University of Washington

Eric Horvitz
Microsoft Research

ABSTRACT

Map matching determines which road a vehicle is on based on inaccurate measured locations, such as GPS points. Simple algorithms, such as nearest road matching, fail often. We introduce a new algorithm that finds a sequence of road segments which simultaneously match the measured locations *and* which are traversable in the time intervals associated with the measurements. The time constraint, implemented with a hidden Markov model, greatly reduces the errors made by nearest road matching. We trained and tested the new algorithm on data taken from a large pool of real drivers.

INTRODUCTION

Automotive navigation systems use map matching to compute the location of a vehicle on a road based on noisy sensors, usually including GPS. This is a challenge, because neither location sensing nor mapped road positions are always accurate enough for a simple algorithm like nearest road matching to succeed. An example of these inaccuracies is illustrated in Figure 1. This shows three GPS points measured along a longer trip. The center GPS point is several meters from any mapped road, due either to inaccuracy in the map, inaccuracy in the GPS measurement, or some of both. The goal of map matching is to determine the road that was actually being driven in spite of these inaccuracies. Our algorithm computes the gray route as the one that most likely represents the vehicle's true path on the map.

The most frequent application of map matching is as a prerequisite for giving real time directions, because it determines the vehicle's position in the road network. The navigation system's internal router then uses this position to compute directions to the desired destination. Map matching is also important for post hoc analysis of GPS tracks, when the goal is to discover which roads were driven. This is our main motivation, although our algorithm applies to real time matching as well.

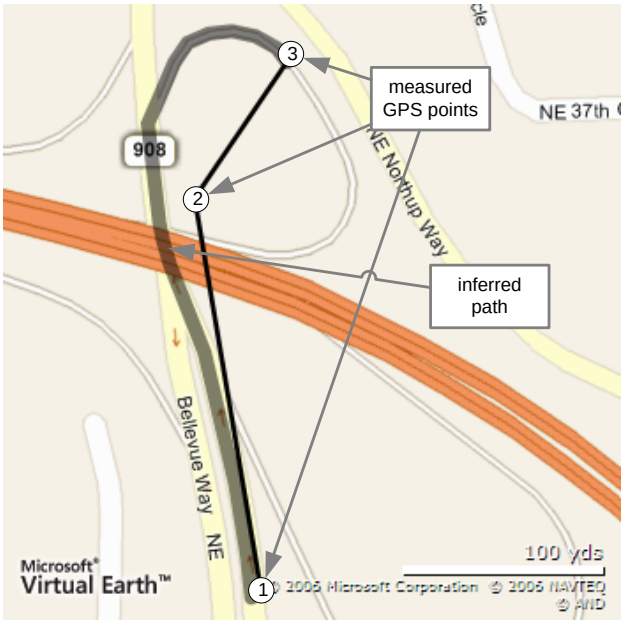

Figure 1: Our algorithm takes inaccurate GPS points and matches them to nearby roads.

Map matching is not only for pure GPS data. Any form of measured location and orientation data could serve as input, including odometry, compass readings, triangulation from Wi-Fi base stations or cell towers, and combinations of these. Although we concentrate on pure GPS data in this paper, our algorithm is applicable to other, multiple sensors whose location measurements can be modeled probabilistically.

The key innovation in our algorithm is that it matches roads based not only on the location measurements, but additionally on the time stamps of the measurements. The time constraint means that the sequence of

Figure 3: Nearest road matching fails in this example. The black line connects the measured GPS points. When the path crosses the north/south street, nearest road matching matches it, ignoring the path's continuity and implying an eventual U-turn, shown in gray. Our algorithm gives the white route.

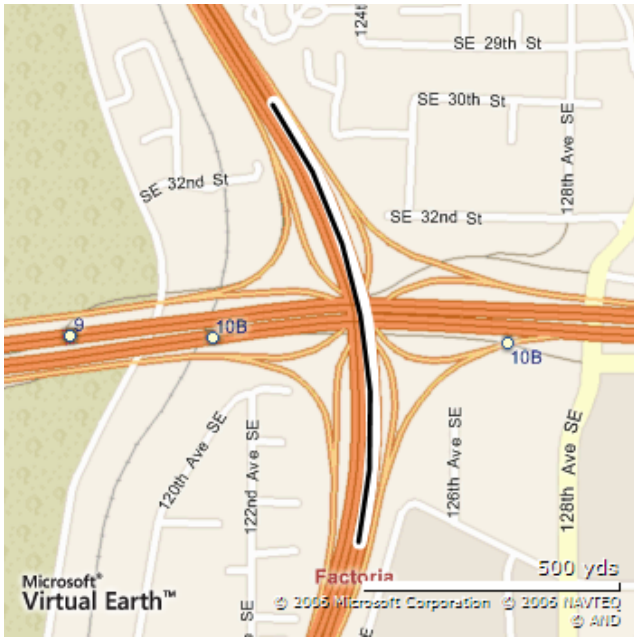

Figure 2: There are many possible candidate sets of roads to match against in this example, making curve-to-curve matching expensive. Based on the GPS points connected by the black line, our algorithm finds the white route that cuts through all the other candidates.

matched roads must be reachable from each other in the time intervals computed from the measurements' time stamps. This additional constraint greatly improves the accuracy of map matching over a simple nearest road algorithm.

OUR ALGORITHM IN CONTEXT WITH PREVIOUS WORK

The most obvious way to match location measurements to road data is simple nearest road matching. In this algorithm, a set of candidate nearby road segments is chosen. For each of these candidates, the algorithm computes the distance between the measured point and the nearest point on the segment. The road segment with the smallest computed distance is declared the matched road. This algorithm often fails because it neglects to account for continuity of the driven path. An example of this failing is shown in Figure 3, where the measured GPS locations are connected by a black line. When the vehicle crossed the intersection, the nearest road algorithm matched to the north/south cross street, despite the fact that the vehicle continued to travel across the intersection on the same east/west street as before. Taken to its logical conclusion, the nearest road match means that the vehicle would have had to make a U-turn at the nearest opportunity (shown in gray) to get back to the original street. Our algorithm instead infers the route as the white line, preserving continuity.

The nearest road algorithm and others are reviewed in [1]. One step up in sophistication from the nearest road algorithm is curve-to-curve matching. Here, a set of location measurements is matched to candidate routes, and the minimum error route is chosen as the best match. This helps fix the continuity problem, but the number of possible route candidates to match against can grow quickly when there are many nearby, branching roads, as in Figure 2, giving an overwhelming number of candidates to check.

One of the more sophisticated classes of map matching algorithms accounts explicitly for the road topology. In [1], their version of the topology algorithm limits the next match in time to roads that are reachable from the current match. Hummel[2] uses a hidden Markov model to eliminate transitions between unconnected road segments. This is similar to our algorithm, except that we use a probabilistic travel time constraint instead of pure topology. This constraint implicitly accounts for topology and also disallows temporally unlikely paths. For instance, in Figure 4, the GPS data seems to indicate an excursion onto a side road, which would have to be followed by a U-turn to get back to the subsequent GPS points. In evaluating possible paths, our algorithm finds one whose computed traversal time approximates the actual traversal time, thus picking a topologically consistent solution that accounts for the elapsed time between measured points as well as the locations of the measured points. We compute candidate travel times with a standard route planner.

In matching location measurements to roads, we compute a compromise between a path that matches the location measurements and a path that is feasible with respect to the measurements' time stamps. This requires

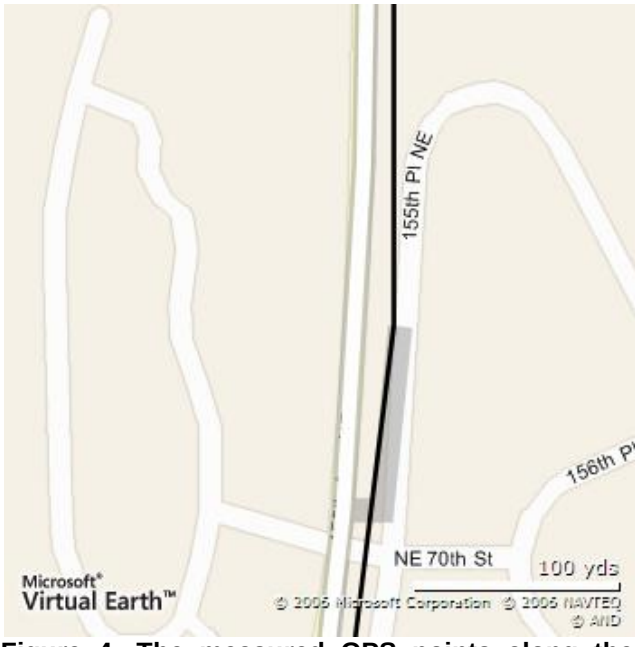

Figure 4: The measured GPS points along the black line indicate an excursion and U-turn along a side road, giving the gray route. Our algorithm ignores this distraction and returns the white route, because the excursion would take more time than the time-stamped GPS points indicate.

a principled notion of the error in GPS measurements and the error in our estimates of path traversal times. We optimize over these two sources of error with a hidden Markov model (HMM). The HMM solution is a path that simultaneously stays close to the location measurements and whose road segments are traversable in the time it actually took to drive the path.

The HMM is based on probability distributions representing GPS error and trip time estimation error. We compute these distributions based on a large survey of drivers carrying GPS receivers, detailed in the next section.

MULTIPERSON LOCATION SURVEY

We trained and tested our algorithm on GPS data from 187 volunteer drivers in the Microsoft Multiperson Location Survey (MSMLS)[3]. These subjects volunteered to place one of our 55 GPS receivers in their vehicle for two weeks (and occasionally longer) as they drove normally. Nearly all the subjects live in the Seattle, WA USA area, and they include employees of our institution and their family members. The GPS receivers were Geko 201 models, capable of recording up to 10,000 time-stamped (latitude, longitude) coordinates. Each subject was given a cable to supply GPS power from the vehicle's cigarette lighter. Using a simple hardware modification, we altered the GPS receivers so they would automatically turn on whenever power was supplied. This meant that the drivers did not have to remember to turn the receivers on or off, and could instead just set the receiver on the dashboard and neglect it for the entire survey period. Because some vehicles' cigarette lighters are powered even when the vehicle is off, we used a mode on the GPS receivers that only recorded points when the receiver is in motion, eliminating the accumulation of points when the vehicles were parked.

We gathered a total of 1,351,669 (latitude, longitude) points for an average of 7,228 points per person. The points were separated by a median distance of 64.4 meters and 6 seconds. We also gathered demographic data from each subject: 72% were male, 75% had a domestic partner, 37% had children, and the average age of drivers was 37.

LOCATION ACCURACY

Our computed paths are a compromise between the measured GPS points and temporal feasibility. In order to trade off these two factors, we must know how much error is associated with each one. For GPS, we assume that all the receivers in our study have similar error characteristics, since they are all identical models. Furthermore, there is likely some error in the mapped locations of the roads, which contributes to the deviation between the measured location and the location on the map. We modeled the error in (latitude, longitude) as a 2D, circular Gaussian with zero mean[4]. This means we must have an estimate of the standard deviation, σ_g, of the distance between the measured location and the actual point on the map. We computed this by assuming that, for *most* of the GPS points, the nearest road was actually the correct road. This is borne out by our informal observations of nearest road matching. With this assumption, we computed σ_g using a robust estimator of standard deviation, the median absolute distance (MAD)[5].

Detailing this procedure, we represent the measured location points as 2D vectors $\underline{x}_g^{(i)}$ for $i = 1 \ldots N$, with N equal to the number of measured GPS points in our database. For each measured GPS point, we find the nearest on-road point, $\underline{x}_r^{(i)}$. If we assume that the actual location was the nearest on-road point, then the combination of GPS error and map error is the distance between $\underline{x}_g^{(i)}$ and $\underline{x}_r^{(i)}$: $d^{(i)}$. Since these are (latitude, longitude) coordinates, we compute the distance using the Haversine formula. Our road network data came from NAVTEQ™, accessed through an API developed at our institution for an upcoming map product. In this dataset, roads segment splits generally occur wherever there is a junction such as an intersection, exit, or entrance.

The MAD gives a valid estimate of the standard deviation of a set of values even if up to half those values are outliers. This is why, even if up to half the nearest road points are wrong, we can still compute a reasonable estimate of σ_g. The MAD formula is

$$\sigma_g = 1.4826 \cdot \text{median}\left|d^{(i)} - \text{median}\left(d^{(i)}\right)\right| \quad (1)$$

Here, since we assume that GPS error has zero mean, we replace $\text{median}\left(d^{(i)}\right)$ with zero. The factor of 1.4826 makes the estimate consistent for Gaussian distributions. We computed $\sigma_g = 7.6386$ meters.

The measured location accuracy σ_g is ultimately used to compute the probabilities of sets of candidate road matches for each measured point. In particular, for each measured location $\underline{x}_g^{(i)}$, we search for the 10 nearest road segments in a radius of 200 meters. The number of road segments we actually find can be less than 10, and we call this number R_i, corresponding to the i^{th} location measurement. We aim to compute the observation probability $P\left(r_{i,j}\left|\underline{x}_g^{(i)}\right.\right)$, which is the probability that $r_{i,j}$, $j = 1 \ldots R_i$, is the correct road segment out of the R_i candidate roads given that the measured location was $\underline{x}_g^{(i)}$. We can compute this with Bayes rule:

$$P\left(r_{i,j}\left|\underline{x}_g^{(i)}\right.\right) = \frac{p\left(r_{i,j}\left|\underline{x}_g^{(i)}\right.\right)P\left(r_{i,j}\right)}{\sum_{k=1}^{R_i} p\left(r_{i,k}\left|\underline{x}_g^{(i)}\right.\right)P\left(r_{i,k}\right)} \quad (2)$$

The likelihood $p\left(r_{i,j}\left|\underline{x}_g^{(i)}\right.\right)$ is a zero-mean, one-dimensional Gaussian with standard deviation σ_g evaluated at $d_{i,j}$, which is the distance between the measured location $\underline{x}_g^{(i)}$ and the nearest point on road candidate $r_{i,j}$. $P\left(r_{i,j}\right)$ is simply $1/R_i$, an uninformative prior probability reflecting the fact that we have no upfront bias about which is the correct road to match. Simplifying Equation (2) with this likelihood and prior, we get

$$P\left(r_{i,j}\left|\underline{x}_g^{(i)}\right.\right) = \frac{\exp\left[-0.5\left(\frac{d_{i,j}}{\sigma_g}\right)^2\right]}{\sum_{k=1}^{R_i}\exp\left[-0.5\left(\frac{d_{i,k}}{\sigma_g}\right)^2\right]} \quad (3)$$

In summary, this gives the observation probability that road $r_{i,j}$ is the correct match for location measurement $\underline{x}_g^{(i)}$. If we neglected all other factors, maximizing this probability at every location measurement would result in simply matching to the nearest on-road point. We know this method frequently fails (*e.g.* Figure 3 and Figure 4), so we introduce a temporal constraint in the next section.

TEMPORAL ACCURACY

Counteracting the tendency to match each measured point to the nearest road, we introduce a constraint on the traversal time between measured points. In looking at the next measured location in a sequence, two candidate road segment matches can be close together in distance but relatively far apart in time. One example is the opposing lanes on a highway: while the lanes can be physically adjacent, legally moving from one to the other requires a time-consuming U-turn. Paying attention to the temporal consistency of a set of candidate matches helps enforce the continuity of driver's path and smoothes over errors in location measurements. Our goal in this section is to assess the accuracy of the computed traversal time between candidate road segments. This implicitly governs how much weight we should assign to the temporal constraint *vs*. the measurement error discussed in the previous section.

Each location measurement comes with an accurate time stamp. In addition, we can predict the traversal time between any two points on any two road segments using a conventional route planner. Our route planner gives traversal time estimates based on the assumption that the driver will take the minimum time route. The route planner computes the traversal time from the speed limits and lengths of the road segment(s) between the two points. The computed paths are quite short, as we are considering road segment matches based on sequential location measurements, which are separated by a median distance of only 64.4 meters.

In evaluating the probability of a transition between two road candidates, we compare the actual time spent driving between the two points against the estimated driving time between the two points. Any deviation is attributed to a combination of the natural variation of driving times and the error in the traversal time estimate. The goal of this section is to develop a formula representing the accuracy of our route planner's traversal time estimates. We assume that the deviations have a Gaussian distribution, and we estimate the parameters of this Gaussian using a robust estimator as in the previous section.

Referring to our database of time-stamped location measurements, we look at adjacent measurements in time, $\underline{x}_g^{(i)}$ and $\underline{x}_g^{(i+1)}$. Because we are using robust estimators, we will again assume that the nearest on-road matches are correct. From these matches, we compute the estimated traversal time and then compute the deviation from the actual traversal time. The median of these errors is our robust estimate for the mean of the

measurement $\underline{x}_g^{(i)}$			measurement $\underline{x}_g^{(i+1)}$	
road match candidates	observation probabilities	transition probabilities	road match candidates	observation probabilities
$r_{i,1}$	$p\left(r_{i,1}\middle\|\underline{x}_g^{(i)}\right)$	$\alpha_{3,1}^{(i,i+1)}$	$r_{i+1,1}$	$p\left(r_{i+1,1}\middle\|\underline{x}_g^{(i+1)}\right)$
$r_{i,2}$	$p\left(r_{i,2}\middle\|\underline{x}_g^{(i)}\right)$	$\alpha_{3,2}^{(i,i+1)}$	$r_{i+1,2}$	$p\left(r_{i+1,2}\middle\|\underline{x}_g^{(i+1)}\right)$
$r_{i,3}$	$p\left(r_{i,3}\middle\|\underline{x}_g^{(i)}\right)$	$\alpha_{3,R_{i+1}}^{(i,i+1)}$	$r_{i+1,3}$	$p\left(r_{i+1,3}\middle\|\underline{x}_g^{(i+1)}\right)$
⋮			⋮	
r_{i,R_i}	$p\left(r_{i,R_i}\middle\|\underline{x}_g^{(i)}\right)$	$\alpha_{R_i,R_{i+1}}^{(i,i+1)}$	$r_{i+1,R_{i+1}}$	$p\left(r_{i+1,R_i}\middle\|\underline{x}_g^{(i+1)}\right)$

Table 1: This shows the roles of the observation and transition probabilities associated with two sequential location measurements and their respective road candidates.

Gaussian error distribution, and the MAD is our estimate for the standard deviation. Based on our data, these estimates are $\mu_t = -0.5690$ seconds and $\sigma_t = 2.7725$ seconds. The negative mean indicates that the traversal time computation is underestimating the actual traversal time. This could be due to traffic conditions, for which the route planner does not account.

The transition probably we need to compute is $\alpha_{j,k}^{(i,i+1)}$, which represents the probability of the driver transitioning from road candidate $r_{i,j}$ corresponding to measured point $\underline{x}_g^{(i)}$ to road candidate $r_{i+1,k}$ corresponding to measured point $\underline{x}_g^{(i+1)}$. If the actual and estimated traversal times are approximately equal, this transition probability should be higher than if the times are far apart. The temporal error between the actual traversal time and the estimated traversal time is $\Delta t_{j,k}^{(i,i+1)}$. The transition probabilities using the Gaussian assumption are

$$\alpha_{j,k}^{(i,i+1)} = \frac{\exp\left[-0.5\left(\frac{\Delta t_{j,k}^{(i,i+1)} - \mu_t}{\sigma_t}\right)^2\right]}{\sum_{l=1}^{R_{i+1}} \exp\left[-0.5\left(\frac{\Delta t_{j,l}^{(i,i+1)} - \mu_t}{\sigma_t}\right)^2\right]} \qquad (4)$$

Intuitively, this says that the transition probability between two road candidates decreases with increasing deviation between the measured time interval and the estimated time interval for the candidates in question. The denominator normalizes the transition probabilities to one. Referring to Figure 3, the transition probability would be small leading into the excursion off the correct road to the U-turn, because this would take much longer than the elapsed time between the measured points.

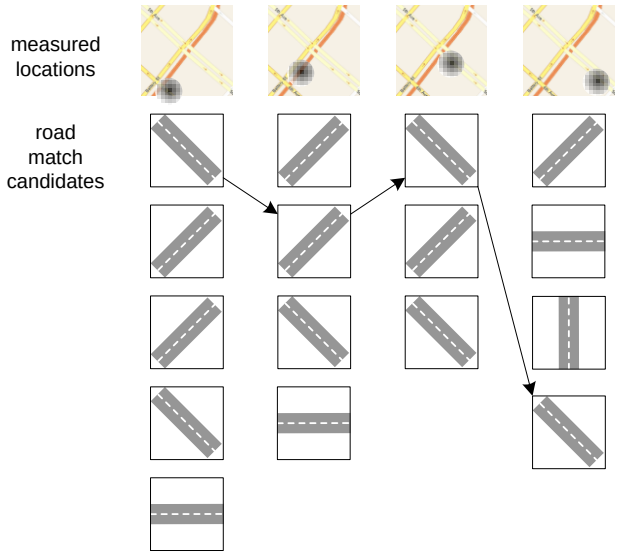

Figure 5: The hidden Markov model has a set of possible road segments for each location measurement. The Viterbi algorithm finds the optimal sequence of road segments considering both the measurement errors and transition probabilities.

An illustration of the various probabilities is shown in Table 1. Here there are two location measurements, each with an associated set of candidate roads. Each candidate has an observation probability computed from Equation (3). There is a transition probability between each of the candidates associated with the first measurement and those associated with the second measurement, computed from Equation (4).

MAP MATCHING WITH TRAVEL TIME CONSTRAINTS

The problem now is to find a sequence of road segment candidates that give an optimal compromise between the measured locations and the traversal times between the candidates. For each measured location, we have a list of up to 10 nearby, candidate on-road points, each on a different road segment, as illustrated in Figure 5. Associated with each of these on-road points is an observation probability based on our estimated GPS error distribution. For every pair of sequential location measurements, we have transition probabilities between each of the candidate road segments, based on the difference between the actual traversal time and computed traversal time.

The observation and transition probabilities fit well with a hidden Markov model (HMM)[6]. An HMM models a time series as a sequence of discrete-time, noisy measurements of a set of discrete states. In our case,

Figure 6: The GPS points are connected by a black line. Our algorithm gives the path in white, while the naive nearest road algorithm gives the path in gray.

the noisy measures are the location measurements, and the discrete states are the road segments. In fact, the terminology we have used in the previous two sections fits exactly with conventional HMM terminology: observation probabilities and transition probabilities. The only necessary element to add is a probability distribution over the initial set of road segments corresponding to the initial location measurement. Lacking any prior knowledge, we made the initial distribution uniform, *i.e.* $1/R_0$. One slight difference between a conventional HMM and ours is that the state space associated with each observation in our HMM changes from observation to observation, depending on which road segments are nearby.

The solution to the map matching problem is to find the path through the states that maximizes the probability of the sequence of road segments with respect to the observations and transition probabilities. The Viterbi algorithm uses dynamic programming methods to efficiently accomplish this, resulting in path through the candidate road segments, as illustrated in Figure 5.

The results we show in the next section come from applying Viterbi to the entire sequence of location measurements for each subject. In a real time scenario such as a vehicle's navigation system, Viterbi would be applied to location measurements up to and including the most recent, perhaps starting at the beginning of the trip or starting at some preset time into the past. In our experience, looking ahead in time, as we effectively do in our computations, only slightly improves the results.

Given a set of location measurements for a driver, the recipe for our algorithm is

1. For each location measurement, search for the nearest road segments and compute the nearest point on each. Stop searching at 10 road segments or when the distance from the location measurements exceeds 200 meters.

2. For each of the points on the nearby road segments, compute the observation probability using Equation (3).

3. For each location measurement, compute the transition probabilities from its nearby road segments to those of the next location measurement using Equation (4).

4. Apply the Viterbi algorithm to the observation probabilities and transition probabilities to compute the maximum probability sequence of road segments.

MAP MATCHING RESULTS

We ran our algorithm on all our GPS data, resulting in a road segment associated with each GPS point. As part of our algorithm, we compute the road segment that is nearest each measured location, effectively giving results from the naive nearest road map matching algorithm. We display our results by mapping the sequence of computed road segments. In some cases, the computed road segments for sequential location measurements are not connected. This can be due to widely separated location measurements or to topologically improbable matches from the naïve nearest road algorithm, *e.g*. jumping into an opposing lane of traffic on a highway. When we encountered gaps, we filled them by planning a route between the disconnected segments.

A typical example of our results is shown in Figure 6. Here the measured locations are connected by a black line, showing the driver crossing a bridge over a highway. The result of our map matching algorithm is shown as the thicker white path, which in this case corresponds well to the measured locations. The nearest road algorithm gives the gray path, which includes the segments necessary to fill in the gaps. This algorithm mistakenly matched one of the points on the bridge to the highway underneath. But in order to switch from the bridge to the highway, the driver would need to find the nearest highway onramp. Once this mistaken match is satisfied, the driver must pick up the next matched road segment back on the overpass road, requiring an exit from the highway and a loop back. Looping onto and off the highway is clearly wrong. Our algorithm rejected this hypothesis because it would take much longer than the measured time stamps indicate. Although the matched road segments are not the ones nearest the location measurements, they are consistent with the combination of location measurements and time stamps.

We give several more results in Figures 7-13, using the same drawing convention as Figure 6. The results in

these figures are grouped by the type of distraction (*e.g*. crossover roads, parallel roads), showing how our algorithm avoids the pitfalls made by the nearest road algorithm. While we have no way of knowing for certain the ground truth for our GPS data, it is clear in these figures that our algorithm is giving a result much closer to the truth than the naïve algorithm.

CONCLUSIONS AND FUTURE WORK

Our map matching algorithm accurately matches measured locations to roads. It is based on finding an optimal compromise between measured locations and traversal times, implicitly accounting for the topology of the road network and speed limits. This tradeoff is accomplished with a hidden Markov model. We compute the parameters of the model in a principled way using data taken from a large set of actual drivers. Tests show many situations where our algorithm successfully ignores complex distractions to find the correct path.

Future work on this algorithm should include a more careful characterization of the travel time constraint. While our algorithm is based on expected travel times, it may be more accurate to enforce only a minimum travel time, accounting for the fact that drivers may slow, stop, or park, meaning there is no upper bound on how much time a driver may spend between two candidate road segments.

ACKNOWLEDGMENTS

Thank you to Kelli McGee of Microsoft Research for administering our GPS study.

REFERENCES

1. White, C.E., D. Bernstein, and A.L. Kornhauser, *Some Map Matching Algorithms for Personal Navigation Assistants.* Transportation Research Part C, 2000. **8**: p. 91-108.
2. 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.
3. Krumm, J. and E. Horvitz, *The Microsoft Multiperson Location Survey*. 2005, Microsoft Research (MSR-TR-2005-103): Redmond, WA USA.
4. Diggelen, F.v., *GPS Accuracy: Lies, Damn Lies, and Statistics*, in *GPS World*. 1998. p. 41-45.
5. Rousseeuw, P.J. and C. Croux, *Alternatives to the Median Absolute Deviation.* Journal of the Americal Statistical Association, 1993. **88**(424): p. 1273-1283.
6. Rabiner, L.R., *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.* Proceedings of the IEEE, 1989. **77**(2): p. 257-286.

CONTACTS

John Krumm and Eric Horvitz
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052 USA
{jckrumm, horvitz}@microsoft.com

Julie Letchner
University of Washington
Computer Science and Engineering
Box 352350
Seattle, WA 98195 USA
letchner@cs.washington.edu

Figure 7: Crossover distractions. The measured points along the black line cross over another road. The nearest road route in gray is pulled off the actual route and forced to loop back to the correct road. Our algorithm's path is shown in white.

Figure 8: GPS problems. Our technique gives good results in spite of inaccurate and under-sampled GPS.

Figure 9: Parallel road distraction. Parallel roads or lanes of opposing traffic often distract the nearest road algorithm, while ours stays on a more likely path.

Figure 10: Spur distraction. Our algorithm successfully ignores spurs off the traveled road that distract the nearest road algorithm.

Figure 11: Spaghetti cut. So-called "spaghetti" junctions have many distracting roads, but our algorithm cuts through successfully.

Figure 12: Spaghetti bypass. Our algorithm avoids being drawn into complex interchanges when the actual route passes by.

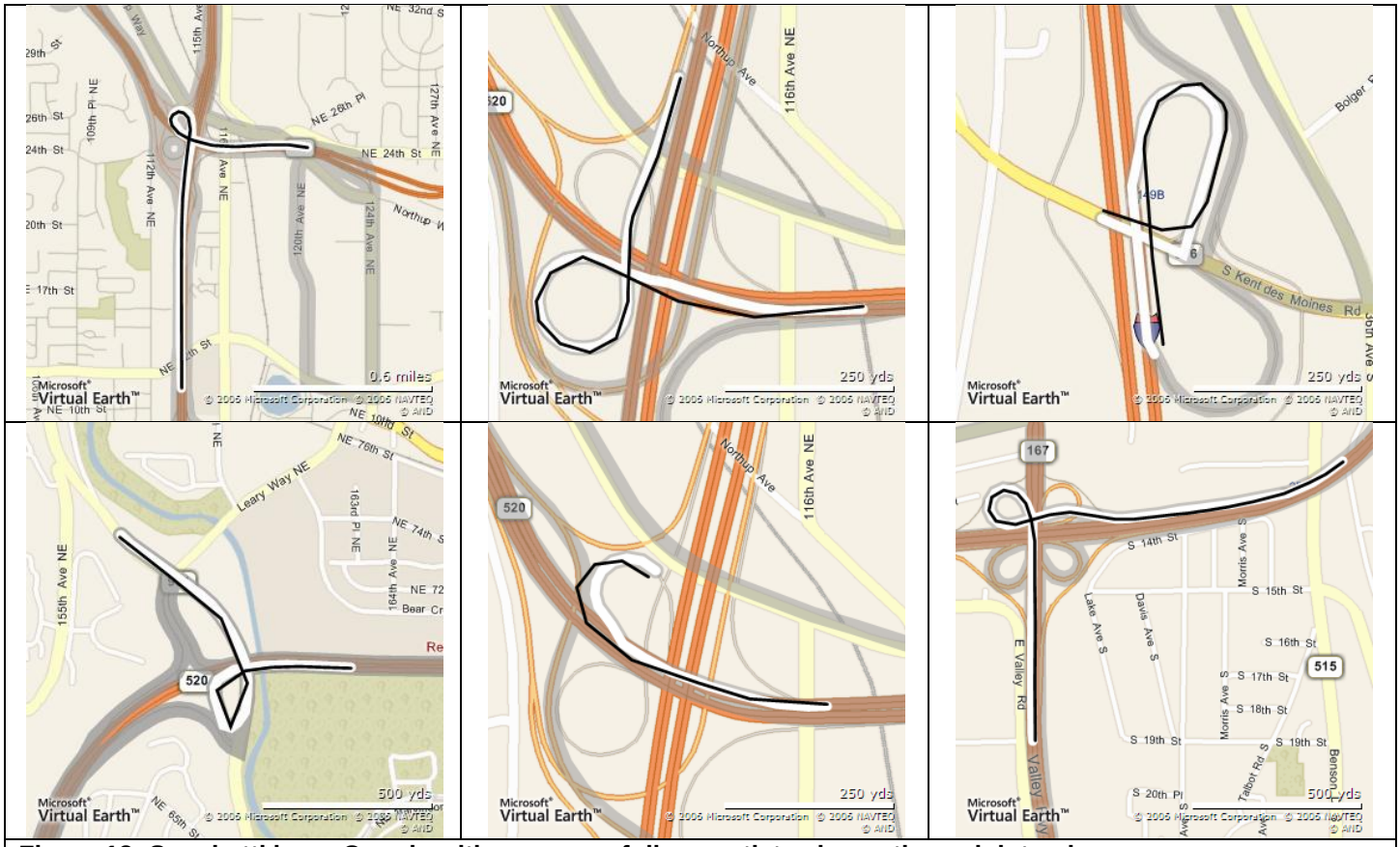

Figure 13: Spaghetti loop. Our algorithm successfully negotiates loops through interchanges.