Addressing some issues of map-matching for large-scale, high-frequency GPS data sets

Qi Luo
Industrial & Operations Engineering
University of Michigan, Ann Arbor
1-734-709-8680
luoqi@umich.edu

Joshua Auld*
Computational Transportation Engineer,
Argonne National Laboratory
9700 S Cass Ave Lemont, IL, 60439
1-630-252-5460,
jauld@anl.gov

Vadim Sokolov
Computational Engineer,
Argonne National Laboratory
9700 S Cass Ave Lemont, IL, 60439
1-630-252-6224,
vsokolov@anl.gov

* corresponding author

Paper submitted for presentation and publication at the 95th Annual Meeting of the Transportation Research Board
Submission date: 08/01/2015
Word Count: 5935 (words) + 1500 (5 figures and 1 table) = 7435

Abstract

Data from GPS-enabled vehicles has become more and more widely used by travel-behavior researchers and transportation system modelers. However as the GPS data has measurement and sampling errors it becomes a non-trivial task to infer a map feature associated with a sequence of GPS measurements, especially as maps features may also have inaccuracies. The task of assigning a set of GPS points to a set of map features is called the map matching problem, and the requirement for the assigned features to form a consistent travel route adds additional complexity. The majority of the existing algorithms concentrate on scenarios when sampling rate is low and/or measurement error is high. However, as GPS devices become more accurate and sampling rate becomes higher, a new issue arises, the issue of efficiently of analyzing large scale high frequency GPS data sets. In this paper we analyze a high-accuracy and high-frequency GPS data set collected from instrumented vehicles that participated in Safety Pilot Model Deployment project using a modified version of the Multiple Hypothesis Technique to match network links, with accuracy and computational efficiency being the focus. We build on previous work in several ways: (i) we proposed a speed-up step that significantly reduces the number of candidate paths, (ii) we improved the way GPS trace segments are matched to road network at turns and intersections and(iii) we added new filtering step to identify U-Turn movements. In addition to these improvements, we demonstrate the process of manually fitting the parameters of the algorithm and suggest future direction on how the process of parameter training can be automated.

1. Introduction

Data from GPS-enabled vehicles has become more and more widely used by travel-behavior researchers and transportation system modelers. Such data can be used for multiple purposes, including inference of travel activities [Stopher 2007, Doherty 2001, Wolf 2003, Auld et al. 2009, Auld et al. 2015] and performance of the transportation system [Herrera 2010, Lo 2008, Work 2008]. However as the GPS data has measurement and sampling errors [Pfoser 99], it becomes a non-trivial task to infer a map feature (point or line) associated with a sequence of GPS measurements. The measurement error is the error in estimated coordinates and associated time step. The error in positional GPS measurement can be described by a bivariate normal distribution [Leick 2015]. The typical error in augmented GPS measurements is in the range between eight and two meters [Brakatsoulas 2005]. The sampling error introduces uncertainty of vehicles position in between the measurements. The task of assigning a single GPS point (as in navigation applications) or a sequence of GPS points (as in post-processing applications used in travel modeling) to a set of map features is called the map matching problem. The goal is to infer the actual location on the map that traveler visited based on the noisy latitude, longitude and possibly speed measurements from GPS, and in the case of post-processing, to have the trajectory resolve into a consistent route through the transportation network. Both measurement error and infrequent sampling rate require attention while developing map matching algorithms. As GPS-equipped vehicles become more and more prevalent, there has been an increasing volume of literature on the map matching problem. The majority of the existing algorithms concentrate on scenarios when sampling rate is low and/or measurement error is high. However, as GPS devices become more accurate and sampling rate becomes higher, a new issue arises, the issue of efficiently of analyzing large scale high frequency GPS

data sets. In previous research, most map matching algorithms focused on introducing topological or probabilistic models to increase the accuracy of matching results, which may actually slow down the map matching process in many cases.

Our work builds on the previously developed methodology by [Pyo 2001, Marchal 2005, Schuessler 2009]. We use a similar approach that applies Multiple Hypothesis Technique (MHT) to map matching problem. As in [Schuessler 2009] our algorithm is applied to a large-scale GPS data set and matched to detailed road network, with accuracy and computational efficiency being the main drivers. In improve on the previous work in several ways: (i) we proposed a speed-up step that significantly reduces the number of candidate paths, (ii) we improved the way curved road segments are matched to GPS data (iii) we added new filtering step to identify U-Turn movements. In addition to improving the previously suggested algorithms, we demonstrate the process of manually fitting the parameters of the algorithm and suggest future direction on how the process of parameter training can be automated. Our further contribution is the development of a test data set based on the data from Ann Arbor. The data set includes the map, GPS data and match between GPS and map performed manually [GitHub 2015]. This data can be used by researchers to validate their algorithms in the future. We also demonstrate our cross-validation procedure that was used to estimate the out-of-sample performance of our map-matching algorithm.

2. Related Work

This section reviews the existing map matching algorithms, including those that we used as a basis for algorithm proposed in this paper. A typical categorization of the map-matching algorithms includes four groups: (i) geometric [Greenfeld 2009], (ii) topological [Chen 2003, Wolfson 2004], probabilistic [Ochieng 2003], and (iv) more advanced techniques [Obradovic 2006, Quddus 2006, Pink 2008]. A good overview of map-matching algorithm developed prior to 2007 is given in [Quddus 2007]. Geometric methods make use of the shapes of the links and traces and the geometric relations between links and shapes. It doesn't consider the connectivity of links, neither the logical order of trace points. The primary advantage of such methods is that they are the fastest method of matching. However, it can cause errors for deviating trips, and may form illogical trips in which links are not joined. The topological method improves the matching results by considering the connectivity and contiguity of links in the network. Use of a weighing criteria and is very important for topological analysis. The advantage here is that matching results will be logical as links are connected. However, a downside is that if one link is mismatched, it also affects matching connected links, which will be a consecutive error, and can lead to large errors in the final path. Finally, the probabilistic method defines an elliptical or rectangular confidence region around a position. This method is very effective for online map matching, for example, navigation systems, especially when the host vehicle is approaching an intersection. It also suitable for sparse GPS data. However, in most cases, this method requires the most memory to process.

Both geometric and topological algorithms rely on matching the point to the closest map feature, taking the feature geometry into consideration (i.e. road segment curvature). The most naive approach is to

match each point individually to a road segment by minimizing the Euclidian distance. Those methods are somewhat effective and can be easily implemented. In fact, many GIS packages would have this as a standard feature. This approach would work relatively well for grid networks, but would frequently mismatch GPS points close to a road intersection, by incorrectly matching to an adjacent road segment. Another problem is that it sometimes incorrectly matches to wrong side of the road and resulting trajectory. There are several ways to improve this naive approach, one is to incorporate road topology. This will allow to avoid matching to a wrong side of the road and when frequency is high (i.e. we have at least one sample from each road segment) it allows to avoid incorrect matching to adjacent segment. When a given GPS point to a road segment we might still end up with several road segment candidates and further improvements are needed to choose between those candidates. One way to improve is to use heading data, if available and match to the link with the tangent that yields the smallest angle.

Further improvement comes from considering the entire sequence of GPS measurements at one and relying on measuring distances between curves (GPS trace vs. network path). The geometric match between GPS trajectory and road segments is done via minimizing the Fréchet distance, which is a measure of similarity between curves, so that the order of the curve points is taken into account. The goal is to find the path on the map that is the closest, in Fréchet distance sense to the GPS path. Topological methods in addition account for connectivity between road segments and would avoid building a path that is topologically infeasible. A topological method, similar to the Fréchet distance minimization approach was proposed by Li 2013. In this paper authors consider matching a sequence of GPS measurements to a path on the network via minimizing the number of so-called k-segments. Each k-segment is a collection of road segments that present a typical math on a map. This algorithm was motivated by an observations that a small number of sub-paths on the network cover a large fraction of typical trips. Probabilistic methods mostly rely on the notion of filtering and assume some additional information about the state of the vehicle is available of analysis, such as heading or speed. Filtering algorithms (i.e. Kalman filter) allows to assimilate the data into an analytical model of the object motion. For example if we know that a trajectory was generated by a vehicle, we know certain physical constraints associated with how fast vehicle can accelerate/decelerate or perform a turn maneuver. Incorporating those constraints allows to filter out certain measurements that violate them. Further, the filtering algorithms performs the best in real-time applications. Goh 2012 proposed an online map-matching algorithms based on the Hidden Markov Model (HMM) that is robust to noise and sampling errors. In [Goh 2012] authors combined spatial, temporal and topological information using machine learning. Another probabilistic approach was suggested in Chen 2012, where not only the network path was inferred by also a transport mode. The method relies on additional measurements from Bluetooth and accelerometer.

The algorithm proposed in this paper is an adaption of the algorithm developed in [Schuessler 2009], which is in turn build on the method from Marchal 2005. These computationally efficient algorithms build on the Multiple Hypothesis Technique (MHT), which is first introduced by Pyo et al. in 2001 to match large-scale GPS data sets on a high-resolution road network. In Pyo et al. (2001) authors used error regions and extended the existing paths by the links within the error region, while Marchal et al. (2005) adopt a topological search algorithm. In this method, several routes candidates are kept in

following the sequence of GPS points, developed and scored to find the best candidate, and only determined the best path at the end of the sequence. The method of Marchal et al. (2005) adopted a topological search algorithm and it proves to be more efficient for larger scale map matching problems. A limit on the number of candidates kept in memory can ensure computational feasibility.

3. Data Sources

In this paper we analyze the high-accuracy and high-frequency GPS data set collected from instrumented vehicles that participated in Safety Pilot Model Deployment project that was conducted by the University of Michigan Transportation Research Institute [UMTRI, n.d.]. The data was collected from in-vehicle GPS on over 2,000 vehicles at a high frequency (10 Hz). A typical trip data set collected during the peak time (most tracking users are driving on road) on a single day consists of 50,000,000 - 70,000,000 points. We have used two sample days at the peak data collection time period to develop and test the algorithm. The points are matched to a road network covering the Ann Arbor-Ypsilanti, Michigan area. The road network is constructed from open street map network which has been run through the POLARIS network editor [Argonne, n.d.], a tool for creating transportation network models from a variety of input data sources. The editor tool allows for the creation of a consistent street network with correct network topology which is not included in the OSM data. The tool also allowed us to incorporate network elements from the SEMCOG regional model into the network. The full area network, with an example trip, is shown in Figure 1.

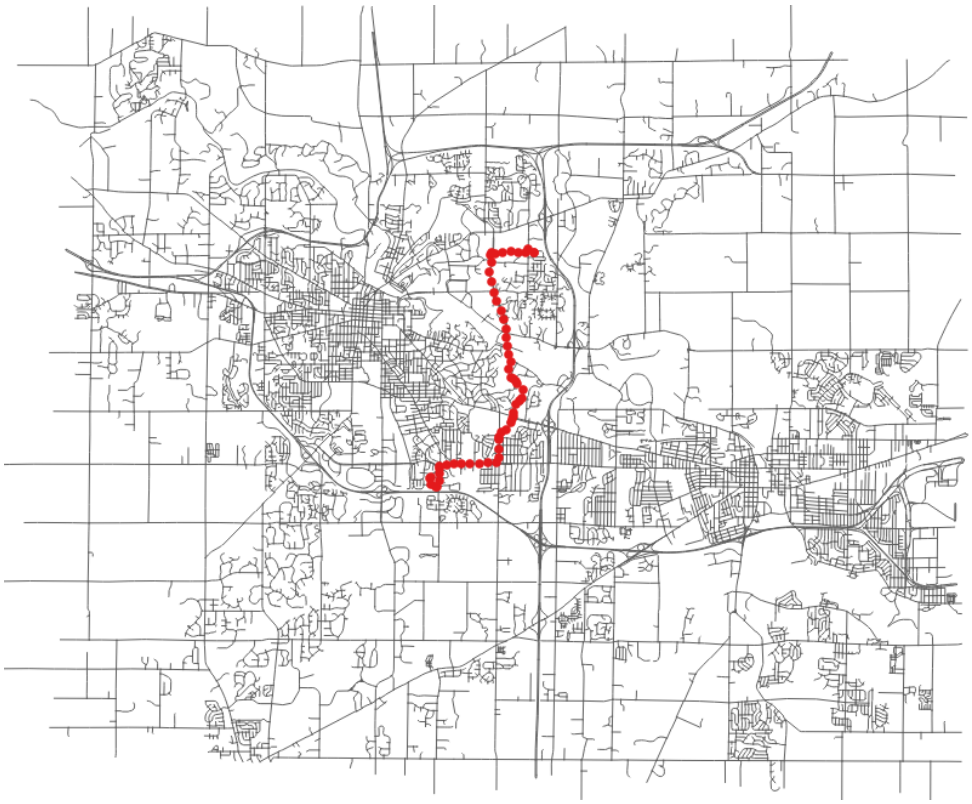

Figure 1. Ann Arbor area street network with example trip

4. Methodology

The basic methodology is an adaptation of the process developed by Schuessler and Axhausen [2009] where we add additional functionality for identifying turn movements, especially for U-turns, and for truncating the set of candidate paths. The implementation of the algorithm, along with pre-processing and post-processing were implemented using Python and SpatiaLite, which is a spatial extension to open source database management system SQLite. SpatiaLite provides vector geodatabase functionality for storing and computing on spatial data. The workflow used can be divided in six steps:

1. Pre-process: trimming out irrelevant points, reduce the number of processing points, and grouping points to each person's trips.
2. Initialization: determining the starting links and assign candidate links to each point
3. Path-building: tracing the points sequence to add new connected links for each route, scoring each route when adding new routes to the pool, and remove routes with worst performance to save memory.
4. Post-process: trimming out irregular links in candidate routes through topological criteria.
5. Selection: selecting the best route as the final result

4.1. Pre-processing of GPS points

During the pre-process step four types of irrelevant or erroneous points are removed from the original data set. First, we remove points out of the current road network range (i.e. trips out of the immediate area). The area of study (shown in Figure 1) was smaller than the envelope that encompasses all of the GPS measurements. We used SpatiaLite capabilities to remove the points that are outside of the study area, defined by a polygon. Second, we removed repeated measurements that are usually associated with a stopped vehicle. For that we identified measurements with nearly the same coordinates and zero or nearly zero travel speed. The stop segments are identified and the points are collapsed into a beginning and ending point of the stop. Third, we remove obvious outliers, so that non-smooth traces are avoided. Points that have large heading change are treated as outliers, because they would violate the laws that govern vehicle dynamics. Finally, GPS measurements were removed that were not matched to any of the road segment using a simple geometric match with a 50 meters threshold, which is well outside the error range of the GPS signal. After performing this steps roughly half of the points would be removed from the data set, primarily due to the stop and out-of-area filters. Additionally, a GPS signal stability check is also used. If the time difference or the distance between two contiguous points is too large, then it means the GPS device may lose signals during the trip. If so, because the main purpose is to match points to links forming consistent trips, breaking trips into two at the gap will mitigate this problem. The gaps can be filled in later using either shortest-path routing or other means depending on heuristic rules using gap distance and gap time to determine what the gap represents (e.g. signal loss, stopping at an activity).

4.2. Initialization of remaining GPS points

Initialization is the first step of the main map-matching algorithm. At this stage we generate multiple candidate routes for a trip (thus the name multiple hypothesis). Since topological methods are very sensitive to the choice of a starting link for the route, choosing a best geometric match becomes infeasible, it will potentially lead the rest of map matching to a nonsense route. Therefore, at least 10

closest links to the starting point are included and each link is assigned to be the first link of a candidate route. The candidate links are found as sequences of searches within circles with increasing radius. The search area is expanded until a set of 10 candidate links is identified. The set of initial links and associated paths is shown in Figure 2. In our implementation we create a new instance of class Path and initialize it with the first candidate links.

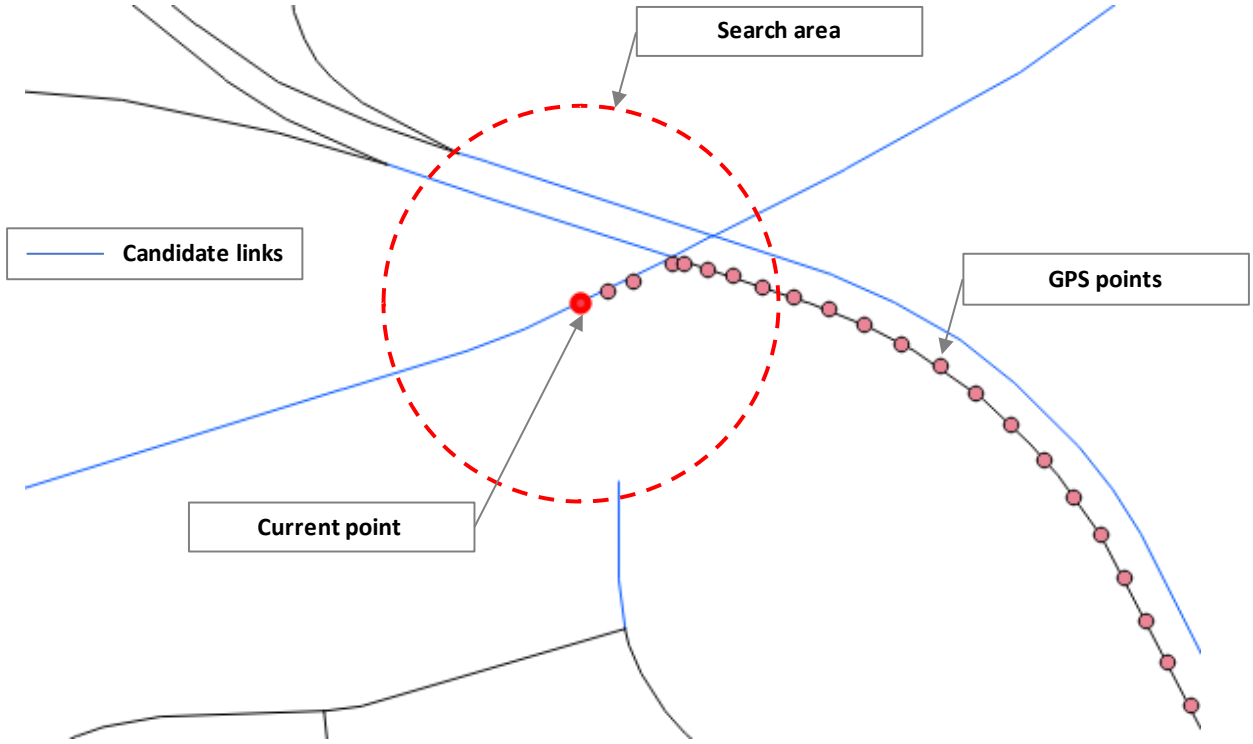

Figure 2. Example of candidate link identification at a point

After identifying the candidate link for the starting point, the set of closest links for each remaining point is also identified. In this case the candidate link for each point are all links within a (fairly conservative) distance of 50 meters. The candidate links for the starting point and remaining points are identified using SpatiaLite intersect functionality, where the links are intersected with the search circle making the process highly efficient. In the process of identifying the candidate links, the projection of each point onto each of its candidate links (or the distance to the nearest end node if no projection) is also stored and used for future analysis.

4.3. Path-building

In the next development step we develop routes by following the sequence of GPS measurements. The algorithm starts with the initial set of candidate paths defined previously at the start point and then for each successive point updates each path by determining if the point is on the current link in each path, or if a transition to a new link has started, growing the candidate path set. The process continues until the end of the GPS trace is reached.

The algorithm has seven primary steps while checking each point *p* in the set of all points (*P*), including:

1. Check for single candidate point -> trim candidate path set (*R)* if true
2. For each path *r* in *R*:
    a. Check if point *p* is past the end of the last link *l* in *r*, if not, go to next path
    b. If *p* is past end of *l*, get the set of candidate links, *C*, at *p*
    c. Trim the candidates using the network topology, i.e. only include candidates (l_c) that are topologically connected to *l*
    d. Add a new path for each viable link ($r + l_c$) to *R*
    e. Remove *r*, from *R*
3. Rescore all paths in *R*
4. Drop highest scoring paths until the size of *R* is below path set size threshold (10 paths).

Figure 3, below shows the process for updating a path *r* at point *P*, graphically.

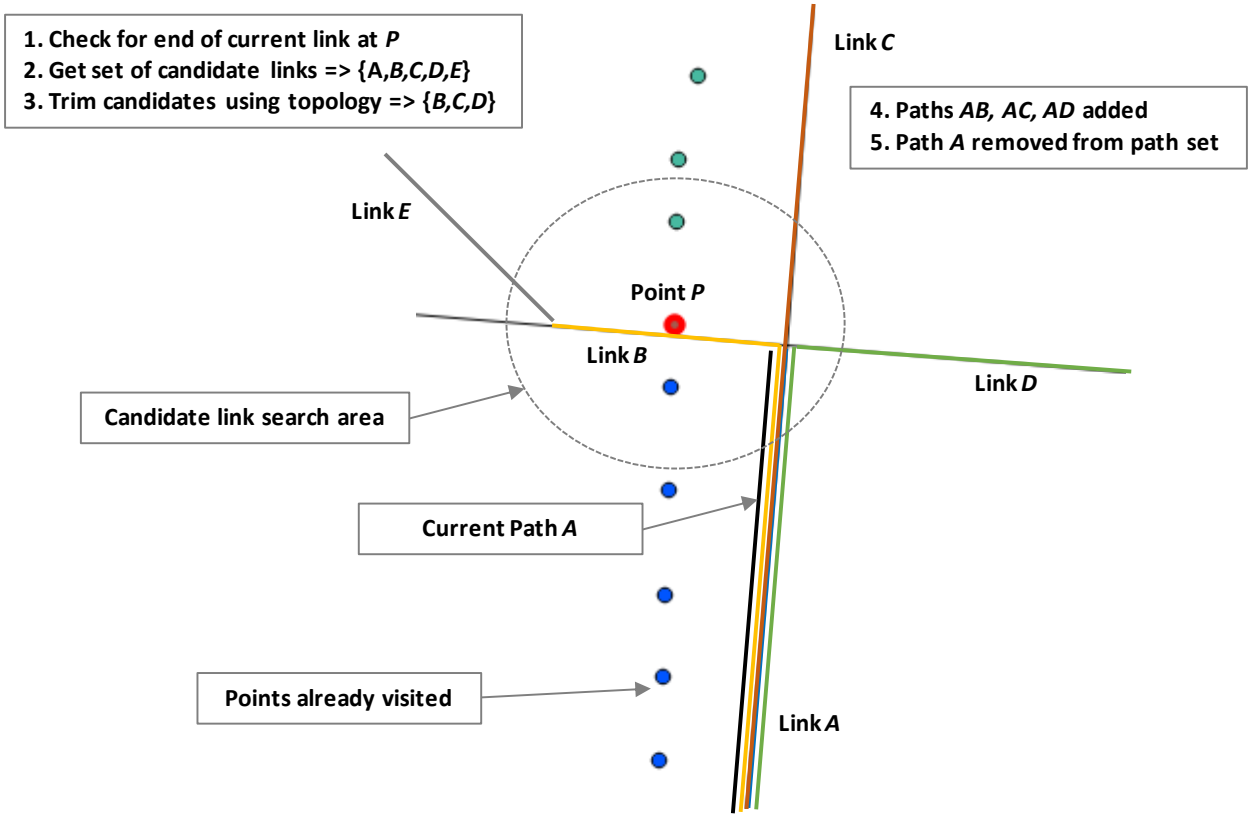

Figure 3. Path-building process

In the figure, the point *P*, is shown to be past the end of link *A*, which is the first link in the path. Since it is past the link end, the set of candidate links at *p* is identified as {*A,B,C,D,E*}. However, A is excluded as it is the same as the current link, and *E* is excluded as there is no topological connection between the current last link *A* and *E*. Now, the new candidate path set becomes {*A, AB*, *AC, AD*}, and finally, path *A* is removed making the candidate set {*AB*, *AC, AD*}, which are then rescored with addition of the new point. This process will then continue for subsequent point *P+1*, except now the same process will be followed for each path in the new candidate set. In the example, points will continue to be added to path *AB* since the link end will not be triggered, however the score will grow rapidly as the distance from

the points to *B* increases. Path *AC* will also continue to add points, with a much lower score as this is the path the points actually follow. And path *AD* will trigger the link end at point *P+1* as this point does not project onto link *D*. This link end will follow the above process to add the new paths *ADA, ADB, ADC, ADE*, replacing path *AD*. However, these paths will be higher scoring than the more direct candidates and will eventually be trimmed out.

The most challenging part of developing routes is to determine if the point has reached the end of current link and to identify a transition from one link to another. The main cause of the difficulty is that often minor roads or connector roads (parking lots) are not represented on the map, and that exact street configurations at intersections are not included. A vehicle is driving to/from a non-public road or a parking lot will deviate from the current link earlier, which should be counted as the end of the link. This is a very common situation especially at the end of the trip when the vehicle starts to park in an off-street parking space. Another complication is caused by U-turn maneuvers. U-turn is a common problem because the real route will jump from the current link to its unconnected neighbor link without reaching the physical end of the link (as seen in Figure 4).

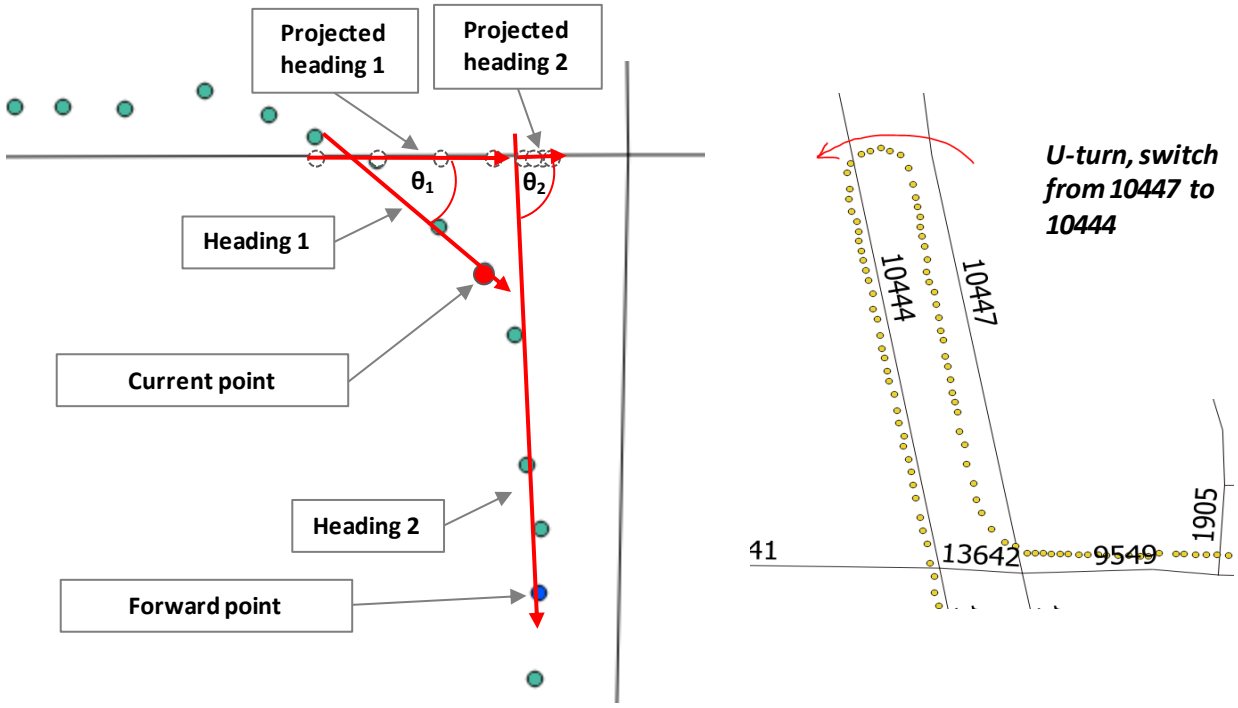

Figure 4. Identification of link ends (or start of U-turns) by heading changes

The original algorithms proposed in Ochieng 2003 and Marchal 2005 addressed all of the issues mentioned above, except for the u-turn case, via three filtering steps. We address the u-turn case by adding a fourth step to link end identification algorithm. The steps include:

1. Check if the distance point is projected onto the link. Here we use the pre-calculated results from the initialization step where the projections of the point onto each candidate link were stored

using the Line_Locate_Point function of SpatiaLite. The function returns a value from 0.0 to 1.0, which represents the fraction of total length along the line the projected point is located. If the point projects off the link, it will return 0.0 or 1.0, and so either value indicates reaching a link end.

2. Check if the GPS points are running at an angle to the link (Figure 4). The angle between the link and the trace is calculated by determining the angle between two heading vectors, composed of the trace points and their projected points on the link. If this angle is bigger than a default value (60 degrees), then this vehicle is may be leaving the current link. In addition, a second check is if the relative angle increases at a forward point ahead in the sequence, or whether the angle grows smaller indicating a temporary deviation from the link. If the angle grows the link end is reached. This step is very time consuming compared to Step 1, so should not be used unless the point sequence is approaching the end of the link as determined by the projected fraction (<0.1 or >0.9). However, for a short link, say less than 200 meters, fraction cannot reflect how close the point is to the intersection, and so the angle is checked at every point.
3. U-turn check is very similar to the angle check, however it is run at multiple forward points to verify that the trajectory is turning back on itself. If the vehicle is in the middle of a link, and the heading angle between two heading vectors, which are composed with certain number of previous and following points, is bigger than a default value (in our case we used 120 degrees), then we try to select the second closest link as the next link.
4. Finally, we check if the distance travelled by the GPS sequence along the current link is longer than the length of that link (as in Marchal [2005]) by a factor of 1.5 or higher. This check is rarely triggered, however, due to the accuracy of GPS and the addition of the angle checking step.

Searching for the next link for current point is based on topological attributes of the previous link and other links in the candidate links set. First, new links should be connected to the previous link, which share the same node. Second, if the new link is a single-direction link, it needs to be a one-way road the vehicle can drive to, which is decided by starting and ending nodes of the links. Lastly, the new link cannot be the same as the previous link.

The process of adding candidate links and paths is similar to the breadth-first search algorithms, that traverses a graph by starting at a root node and exploring all of the neighbors before moving to the next level. In our case we not only traverse the graph but also store all of the candidate paths. At each step where the end of a link is reached the total number of paths will increase from N to $N+C$, where C is the number of candidates links at each point. In order to reduce the number of potential paths, we perform a pruning step, by removing paths with a low likelihood of being a part of a final path as mentioned above. As in Schuessler 2009 we use a scoring system to rank the candidate paths and to cut those with lowest score with certain threshold.

To calculate the score, we define $d(p,l)$ be the distance between link l and point p defined as shown in equation 1 (which has been pre-calculated in the initialization step):

$$d(p,l) = \begin{cases} d_p(p,l) & if\ p_i\ projects\ onto\ l \\ \min\left(d_e(p,l_a), d_p(p,l_a)\right) & otherwise \end{cases} \quad (1)$$

where:

d_p = projected distance of *p* onto *l*

d_e = Euclidean distance from *p* to the *a* or *b* node of link *l*

The score of each route, then, is calculated by:

$$SC_{route} = \sum_{point} \sum_{link} \left(d(p,l)\delta_{pl}\right) \quad (2)$$

Where:

δ_{pl} = 1 when point *p* is assigned to link *l*.

Ideally, the real route should have the minimum score. In this algorithm, the candidate routes limit is set to 10, which is the same as the starting candidate routes number and suitable for high accuracy GPS data. If number of routes is higher than 10, routes with maximum scores are ruled out. When adding new routes, duplicate routes are also deleted.

Finally, any time in the route development process that the set of candidate paths all converge to the same link, all of the paths are evaluated and only the best performing path is kept since the remaining paths are no longer needed as they have no way to outperform the current best path. This significantly reduces the number of path evaluations needed, and happens in areas of the network where the GPS points only have a single candidate link (i.e. within 50 meters), as the topological connection constraint forces all paths to converge.

4.4. Post-processing and selection

The final post-process step starts when the algorithm detected the last link in each path for the last point. The goal is to remove the redundant links in each route, which can happen in particular configurations of GPS trace and network geometry. The most common cause of redundant links occurs when links intersect when the GPS point at the intersection is closer to the crossing link than the actual link. Normally this would not matter (for instance in the example shown in Figure 3), however, occasionally the following point is located in such a way that the link end algorithm is triggered (i.e. the point drifts to the other side of the actual link temporarily). This results in the one point connected to the crossing link having the minimum score and being included in the best path. As this happens somewhat frequently we need an algorithm to correct for this situation.

A node check is used to rule out these links, whereby and link path which has a short cycle from and to the same node or where the end node of the link does not match the start node of the next link, is eliminated. Next, a set of spurious links are frequently generated as a result of U-Turn detection step in the main algorithm. Since the U-turn step may be activated several times on one U-turn, and each time a U-turn on the middle of a link is detected, the next point will be assigned to the neighbor link, it will caused a series of links going back and forth between two links. Post-process will clean these links and

keep the first-time switching as the right one. Because central difference is used for the heading vector formation, it is quite accurate in testing. After all of the candidate paths are processed we choose the route with the lowest score as the final path for output.

5. Results

To evaluate the accuracy and computational speed of the algorithm, 20 people's driving record s were randomly picked from the two-day driving database - 10 from each day. The size of the sample was limited due to the need for manual matching of each point to a link through visual inspection, a time consuming process. The 10 records (for a total of 47 trips) from the first day were used during algorithm development to tune various parameters such as the maximum search distance, number of candidate paths, angle thresholds for the link end identification and so on. The sample was run through the algorithm many times to correct various aspects of the routine. However, in order to validate the model, the second day random sample was used as a hold back set. This sample was only run through the algorithm after it was completely developed in order to validate the routines and to ensure that the algorithm was not overfit to the initial sample. The results are shown in Table 1 below. For each trip, links are categorized into matched links, mismatched links (links that the vehicle didn't drive on but recorded by the algorithm) and missed links (links which the vehicle did drive that were not observed by the algorithm). Precision in the table is defined as: matched link in the output divided by total links identified by the algorithm, while the recall of the algorithm is matched links divided by total actual driving links. The results are shown for both the training and test sets.

Table 1. Map-matching accuracy for training and test datasets

	Training set (Day 1)		
	Actual Link	**Mismatch**	**Precision**
Matched Link	1436	7	99.5%
Missed	10	–	–
Recall	99.3%	–	–

	Test set (Day 2)		
	Actual Link	**Mismatch**	**Precision**
Matched Link	930	19	98.0%
Missed	19	–	–
Recall	98.0%	–	–

In general, the results of the algorithm have very good precision and recall, with very few errors in either data set. The training set results are slightly better than the test set results, as expected as the algorithm was updated until the training set performed to this level. However, the test set shows similar good performance, with only a 2% drop in precision and recall, demonstrating that the algorithm is not overfit to the training case.

The algorithm has been applied to the entire two-day sample of GPS traces, a total of 10,972 trips. The matching of the GPS points for each trip to the Ann Arbor network allows for interesting applications in transportation modeling. Potential uses include route choice analysis, trip table generation, network

performance monitoring, and so on. As an example, Figure 5 shows an estimate of the network performance through the use of a link speed ratio, defined as the average speed observed on the link over all trips divided by the top speed. The GPS data provides fairly good coverage with 63% of all links observed at least once, and 97% of highway links and 70% of major arterials observed more than 20 times.

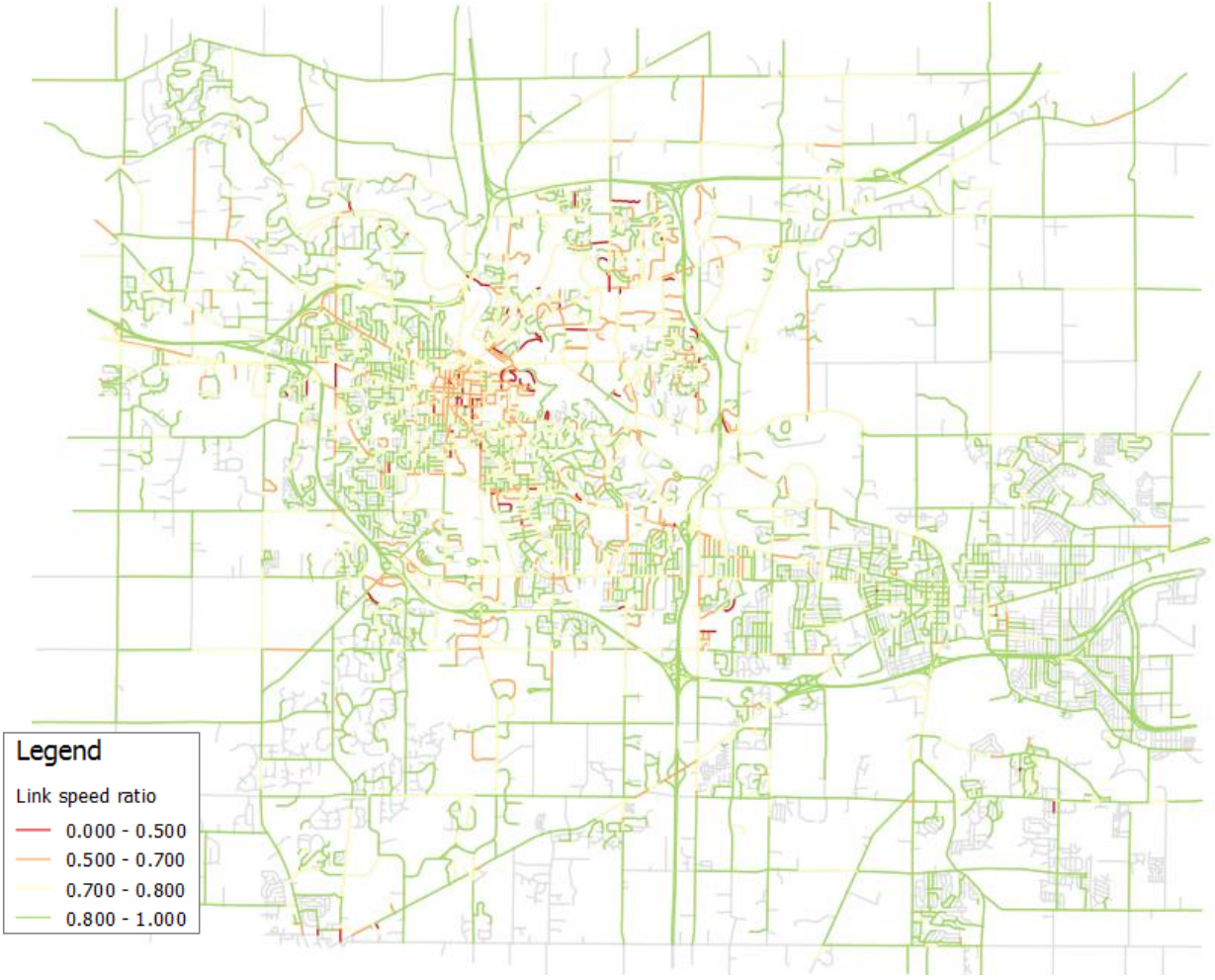

Figure 5. Link speed ratio (avg. speed / max speed) generated from GPS traces for Ann Arbor network, observed over two days

6. Discussion and Conclusions

In this paper we proposed improvements to the multiple hypothesis testing algorithm for performing map-matching using high-frequency GPS data. We showed how the proposed algorithms provides good matching results when validated against correct routes determined by visual inspection. An implementation of the algorithm using Python language and SQLite tools was developed and published. We also developed a testing data set that can be used by other researchers for evaluating and further developing other matching algorithms. The methodology can handle large scale datasets and efficiently identify road segments to be matched. The approach was demonstrated using a data set collected as

part of the safety pilot deployment conducted at Ann Arbor, where accuracies of 98% where achieved in out-of-sample testing.

There are a number of possible avenues for future improvements. First, the presence of large sampling errors is not treated in our current approach since it was not an issue for our dataset, but can be an important consideration for other GPS studies. Especially when data is used in real-time applications, when data is transmitted with low frequency and when the communication signal suffers disruptions. Another important improvement would be to automate the parameter tuning procedure. In our current implementation parameters were tuned using part of the manually matched data (training data). We selected a few candidate sets of parameters and evaluated the performance of the algorithm in the remaining manually matched data (testing data). Even though we were able to find manually a set of parameters, that lead to high level of accuracy in matching results, it is will not necessarily be the case for larger data sets. A possible approach is to use local search algorithm to minimize the error of matches when performed on out-of-sample data. A hill climbing algorithm could be a good candidate algorithm for that purpose.

Acknowledgement

We would like to acknowledge Jim Sayer and the University of Michigan Transportation Research Institute for providing the data used in this project. This project was funded by Argonne National Laboratory, Laboratory Directed Research and Development funding.

References

Argonne National Laboratory (n.d.) Transims Studio Download. Recalled on July 31, 2015 from http://sourceforge.net/projects/transimsstudio/

Auld, Joshua et al. "An automated GPS-based prompted recall survey with learning algorithms." Transportation Letters 1.1 (2009): 59-79.

Auld, J.A., A. Mohammadian, M.S. Oliveira, J.Wolf and W. Bachman (2015). Demographic Characterization of Anonymous Trace Travel Data. Transportation Research Record: Journal of the Transportation Research Board (in press).

Brakatsoulas, Sotiris et al. "On map-matching vehicle tracking data." Proceedings of the 31st international conference on Very large data bases 30 Aug. 2005: 853-864.

Chen, Jingmin, and Michel Bierlaire. "Probabilistic multimodal map matching with rich smartphone data." Journal of Intelligent Transportation Systems ahead-of-print (2014): 1-15.

Chen, W et al. "Integrated vehicle navigation system for urban applications." Proceedings of the 7th International Conference on Global Navigation Satellite Systems (GNSS), European Space Agency, Graz, Austria 22 Apr. 2003: 15-22.

Doherty, Sean T et al. "Moving beyond observed outcomes: integrating global positioning systems and interactive computer-based travel behavior surveys." Transportation Research Circular Mar. 2001.

Goh, Chong Yang et al. "Online map-matching based on hidden Markov model for real-time traffic sensing applications." Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on 16 Sep. 2012: 776-781.

Greenfeld, Joshua S. "Matching GPS observations to locations on a digital map." Transportation Research Board 81st Annual Meeting 2002.

Herrera, Juan C et al. "Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment." Transportation Research Part C: Emerging Technologies 18.4 (2010): 568-583.

https://github.com/anl-tracc/map-matching

Leick, Alfred, Lev Rapoport, and Dmitry Tatarnikov. GPS satellite surveying. John Wiley & Sons, 2015.

Li, Yang et al. "Large-scale joint map matching of GPS traces." Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems 5 Nov. 2013: 214-223.

Lo, Chia-Hao et al. "Carweb: A traffic data collection platform." Mobile Data Management, 2008. MDM'08. 9th International Conference on 27 Apr. 2008: 221-222.

Marchal, F, J Hackney, and K Axhausen. "Efficient map matching of large global positioning system data sets: Tests on speed-monitoring experiment in Zürich." Transportation Research Record: Journal of the Transportation Research Board 1935 (2005): 93-100.

Obradovic, Dragan, Henning Lenz, and Markus Schupfner. "Fusion of map and sensor data in a modern car navigation system." Journal of VLSI signal processing systems for signal, image and video technology 45.1-2 (2006): 111-122.

Ochieng, Washington Y, Mohammed Quddus, and Robert B Noland. "Map-matching in complex urban road networks." Revista Brasileira de Cartografia 2.55 (2003).

Pfoser, Dieter, and Christian S. Jensen. “Capturing The Uncertainty of Moving-Object Representations.” Advances In Spatial Databases Lecture Notes in Computer Science, 1999, 111–31. doi:10.1007/3-540-48482-5_9.

Pink, Oliver, and Britta Hummel. "A statistical approach to map matching using road network geometry, topology and vehicular motion constraints." Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on 12 Oct. 2008: 862-867.

Pyo, Jong-Sun, Dong-Ho Shin, and Tae-Kyung Sung. "Development of a map matching method using the multiple hypothesis technique." Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE 2001: 23-27.

Quddus, Mohammed A, Robert B Noland, and Washington Y Ochieng. "A high accuracy fuzzy logic based map matching algorithm for road transport." Journal of Intelligent Transportation Systems 10.3 (2006): 103-115.

Quddus, Mohammed A, Washington Y Ochieng, and Robert B Noland. "Current map-matching algorithms for transport applications: State-of-the art and future research directions." Transportation Research Part C: Emerging Technologies 15.5 (2007): 312-328.

Schuessler, Nadine, and Kay W Axhausen. "Map-matching of GPS traces on high-resolution navigation networks using the Multiple Hypothesis Technique (MHT)." (2009).

Stopher, Peter, Camden FitzGerald, and Min Xu. "Assessing the accuracy of the Sydney Household Travel Survey with GPS." Transportation 34.6 (2007): 723-741.

University of Michigan Transportation Research Institute (n.d.) "Safety Pilot: Model Deployment". Retrieved July 30, 2015 from http://safetypilot.umtri.umich.edu/

Wolf, Jean, Marcelo Oliveira, and Miriam Thompson. "Impact of underreporting on mileage and travel time estimates: Results from global positioning system-enhanced household travel survey." Transportation Research Record: Journal of the Transportation Research Board 1854 (2003): 189-198.

Work, Daniel B et al. "An ensemble Kalman filtering approach to highway traffic estimation using GPS enabled mobile devices." Decision and Control, 2008. CDC 2008. 47th IEEE Conference on 9 Dec. 2008: 5062-5068.

Yin, Huabei, and Ouri Wolfson. "A weight-based map matching method in moving objects databases." Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on 21 Jun. 2004: 437-438.