Routing-based Map Matching for Extracting Routes from GPS Trajectories

Terry Griffin
Midwestern State University
3410 Taft Blvd
Wichita Falls, Texas
terry.griffin@mwsu.edu

Yan Huang
University of North Texas
1155 Union Circle #311277
Denton, Tx 76203-5017
huangyan@unt.edu

Shawn Seals
Midwestern State University
3410 Taft Blvd
Wichita Falls, Texas
shawn.seals@mwsu.edu

ABSTRACT

This paper introduces a novel offline map matching approach. We develop a routing-based map matching approach for standardizing identified routes in a collected GPS trajectories. Our approach first identifies key waypoints in a user's GPS trajectory using a modified Peucker curve reduction algorithm. Subsequently, it sends the key waypoints to a blackbox driving directions service which returns a route utilizing each of the key waypoints. The returned route is a standardized representation of the original GPS trajectory constructed using the minimum necessary set of points. A filter-and-refine approach is used to identify the incorrect portion of the returned route and a refine step is carried out by eleminating the waypoints which lead to the incorrect matching. Experiments results showed that the proposed approach works well for a dataset of 10 volunteers each collecting data an average of 34.3 days.

Keywords

Map-Matching, GPS

1. INTRODUCTION

GPS receivers are imprecise. A data point of a typical GPS receiver has an error that ranges anywhere from ten feet to hundreds of feet. A typical application of collected GPS data is to extract a route from a given GPS trajectory. The routes will then be used to train algorithms such as trip destination predication. However, the errors that are intrinsic to GPS data collection cause problems for accurate extraction of routes.

Route extraction can be done through map matching. Map Matching is the process of matching data points of a GPS trajectory (usually collected by a GPS receiver) to a known road network. In general there are two types of map matching: *online* and *offline*. Online map matching is typically performed in real time for use in vehicle navigation systems. Offline Map Matching is performed *a posteriori* with regard to the collected GPS data. This method is primarily used for the analysis of historic travel data and can also be used as a means of data cleansing, by snapping small errors within the data set to the underlying road network. When trying to identify a repeated activity by a user, variations in the driver's trajectory or additions of GPS error, can make a commonly traveled route seem different on each occasion. Individually, each variation can seem trivial, but the synergism of all the variations together can be problematic when comparing like routes. An example of some small variations in a driver's route can be seen in Figure 1, and many of the variations can be attributed to GPS inaccuracies. A driver traveling the same route on different occasions produces slightly different trajectories (portions of which actually seem to miss the road). By using Map Matching techniques, each route (a,b,c) was snapped to the road network resulting in an identical route (d).

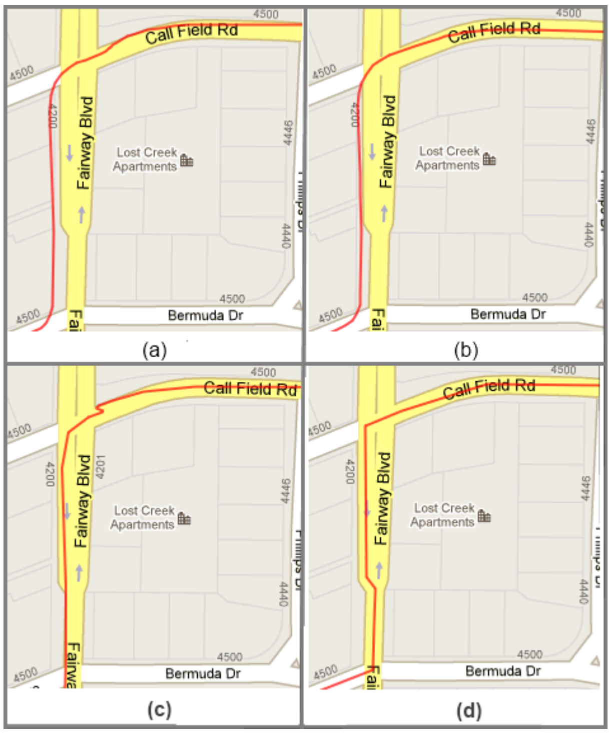

Figure 1: The original routes (a), (b), and (c) were traveled on separate occasions. All three produce the same matched result (d).

1.1 Problem Definition

Given a GPS trajectory of n data points represented by $G = \{< p_1, t_1 >, < p_2, t_2 >, \ldots, \{p_n, t_n >\}$, where p_i is the position represented by longitude and lattitude at time t_i, a road network, the goal is to identify a line segement represented by $R = \{r_1, r_2, \ldots, r_m\}$ where each r_i is a point on the road network and G is map matched to R for all data points of G.

This will allow one to accurately adjust and portray individual routes driven by a traveler between the same two destinations on different occasions as being the same series of logged points. The benefits of extracting a user's route can be found when comparing two routes traveling between the same destination on separate occasions. When routes are extracted, the distance metric used for comparing routes will typically have a tighter upper bound, allowing a threshold to be set with more confidence and reduce the number of false positives and negatives.

The primary focus of this paper is to develop an offline map matching for the purpose of route extraction. Without map matching, routes that travel the same path on a road network will differ from one another due to the variation and error in the GPS produced data along with differences in a driver's trajectory. Map Matching provides a way to compare, categorize, and analyze similar and dissimilar routes by providing a consistent representation of a route.

1.2 Our Contributions

In our approach to the problem, we make the following contributions. First, we propose a novel routing-based map matching algorithm for offline map matching. A new Douglas-Peucker [4] curve-reduction-based waypoint selection algorithm is proposed. The algorithm reduces the number of points needed to be sent to a black box routing algorithm while maintaining the accuracy of the returned the routes. Second, we use a classifier-based filter-and-refine approach to identify and remove sensitive waypoints that lead to incorrect map matching and refine the results. Third, we tested our approach on a large real GPS datasets and the results comfirm that our approach works well.

2. OVERVIEW OF OUR APPROACH

Our approach starts with identifying a subset of points called waypoints from a given GPS trajectory. It sends these waypoints to a blackbox routing algorithm to obtain a route. It then inspects the route and identifies the incorrect portion of the route using a classifier. It then removes the waypoints that lead to the incorrect portion of the route and re-obtains a route from the routing algorithm. This process may need to be repeated if the route obtained is not satisfying.

The blackbox routing algorithm can be obtained through the *Driving Directions* (DD) services provided by many popular web service providers including: MapQuest®, Yahoo®, and Google®. By providing the web service with a starting point A and stopping point B, the web service will interpolate all necessary points to create an accurate route connecting A and B.

All four web services were provided slightly different lat/lon pairs[1] that represented A and B. Even though each web service returned the same normalized interpolated route (as can be seen in figure 3), this is not normally the case.

[1]The differences fell within the same parameters that typical routes connecting the same starting and stopping destinations would fall into.

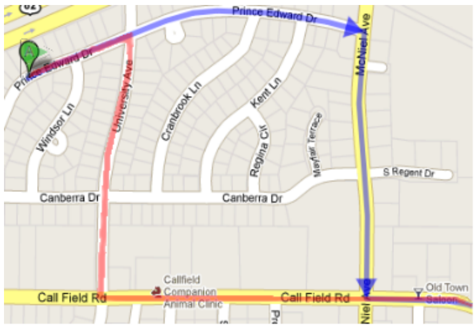

Figure 2: Two DD services can easily return different routes connecting the same starting and stopping points.

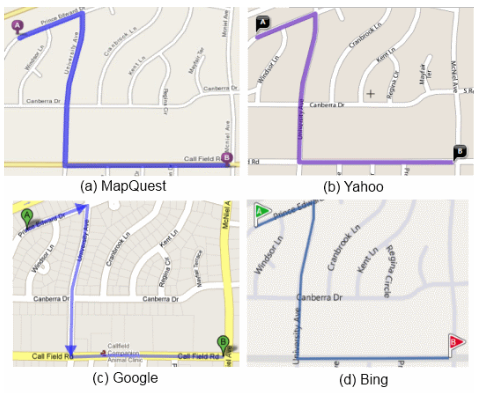

Figure 3: DD services occasionally return the same results when not given very much latitude.

However, sending a starting A and ending B often does not guarantee an accurate route. The problem arises when the chosen web service is given enough waypoints to interpolate every portion of the route indiscriminately relative to the actual route. Figure 2 shows two separate DD services that were given the same starting and ending destinations, but returned different routes. The decisions made by each DD service are based on the proprietary algorithms used by each service to calculate the shortest route between A and B.

To calculate a shortest route, DD services implement variations of Dijkstra[3] and A*[6] shortest path algorithms. These algorithms take into account the connectivity (navigability) and impedance between A and B. Impedance is the *cost* of traveling across a navigable feature. Some examples of cost are: distance, speed-limits, turn-restrictions, and one-way streets. Each DD service calculates impedance using proprietary versions of the aforementioned algorithms assigning costs in different ways [5]. This gives the possibility that given the same starting and stopping points, the resulting routes can differ. We emphasize this point because these path algorithms make decisions based on a set of cost parameters, where actual drivers tend to use personal heuristics not known to a mapping service. The results are that the "fastest" or "shortest" or "optimal" route returned by the mapping service will not always match the route chosen by a driver.

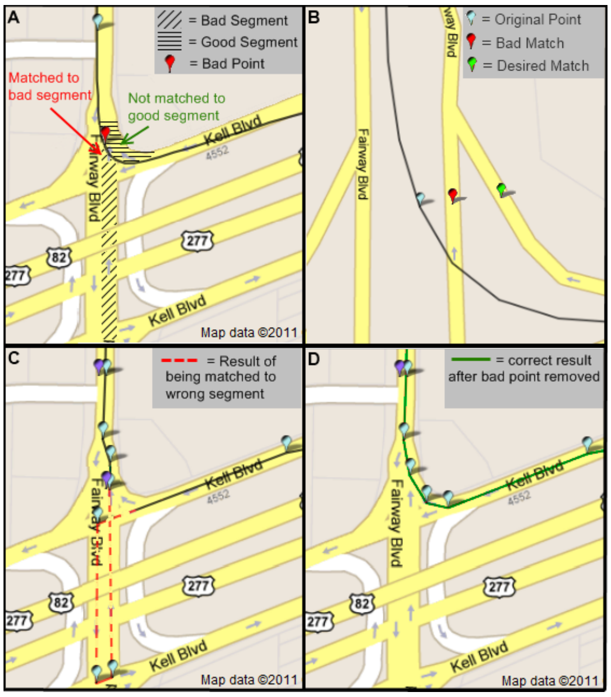

Figure 4: (A) shows a typical intersection in which the DD erroneously matches a key point. (B) is a zoomed in version of (A) depicting the actual submitted point in blue, the erroneously matched point in red, and the desired match location in green. (C) Shows the result (dashed line) of submitting the erroneously matched point to the DD service. (D) Shows the correct map-matched route after the erroneous point was removed.

Supplying only a starting point A and an ending point B to a DD service will quite possibly result in a route that does not match the actual traveled path. Therefore it is necessary to supply the DD service with a subset of points to ensure the service will produce a correct match. However, it is also not necessary or efficient to supply the DD service with the entire set of route points. By only supplying the DD service with a small subset of points, we significantly reduce the amount of time needed to match the route.

The first component (described in section 2.1) of our map matching algorithm is a point selection algorithm which addresses the obvious infeasibility of sending the entire corpus of points (670 points for the average route) to the routing algorithm for every route, and places much emphasis on identifying only the key components that need be sent which is on average about 20 points per route. This results in an average reduction of about 95 percent. By reducing the number of points submitted, the turnaround time for matching a route is held to an acceptable level.

The second component of our map matching algorithm 2.2 is a trained expert system to identify and remove problematic submitted waypoints to ensure accurate map matching of routes (note: submitted waypoints are defined in this paper as points identified by the point selection algorithm to be submitted to the DD service). This filter-and-refine step is necessary due to the possible and somewhat common occurrence of a submitted point being incorrectly matched by the DD service. Even one mismatched point will cause the route to be incorrectly map matched. Point mismatches are usually due to, but not limited to, the point being located near an intersection or underpass/overpass. For example in Figure 4, a single mismatched waypoint to a "bad" segment will cause an extra loop not existing in the actual route. Because the point selection algorithm does not have local access to a road network representation, it is unable to foresee and therefore avoid these situations. Our filter-and-refine algorithm allows us to achieve map matching at an extremely accurate level.

2.1 Peucker/Inverse Corner Point Selection

Peucker et al. [4] developed an algorithm for reducing the number of line segments required to represent a curve. The Peucker algorithm tends to function as a pseudo corner detector when used to process route data. That is, the endpoints of the line segments are typically in the location of route turns/corners. This is due to the fact that corners tend to be more orthogonally distant from the line segments formed by the algorithm. The granularity of the result depends on an ϵ value selected by the user. By changing the value of ϵ, the granularity of the final output can be adjusted to be a course or a fine representation of the original curve. In benefit to us, larger values of ϵ result in fewer points representing the route, but those few points tend to cluster around corners and turns.

Because the points selected by the Peucker algorithm are typically in very close proximity to corners or turns within the route, from here after we will referred to them as "corner points". Because turns usually occur at intersections, these points are susceptible to mismatching by the DD service. However, by simply removing the point after it has been identified as a mismatch may allow the DD service the liberty to return a route that does not accurately follow the same path that was actually traveled. Therefore, in order to "force" the DD service to return a route that accurately follows the actual path traveled, we also calculate and submit the inverse corner points of the route in addition to the originally selected corner points. An inverse corner point is a point that lies midway along the path between two corner points. By submitting both the original corner points and the inverse corner points, the neighboring inverse corner points ensure the DD service does not have enough latitude to make incorrect assumptions. Therefore we were able to reduce the probability of a removed point causing a DD service route error to near 0.

2.2 Identification/Removal of Points

After the Peucker/inverse corner point selection algorithm has been performed, the selected points are then submitted to the DD service and a map matched route is returned. However, the DD service may have mismatched some of the points submitted causing the returned route to be incorrect (discussed previously). Therefore it is necessary to identify the points that were mismatched in order to eliminate them.

In order to identify the mismatched points, the C4.5[10] algorithm was used to develop a set of rules to be applied during the comparison of the original route to the route returned by the DD service. The tree was trained using a set of attributes explained in the following sections [1].

[1]When calculating values for the attributes, any compar-

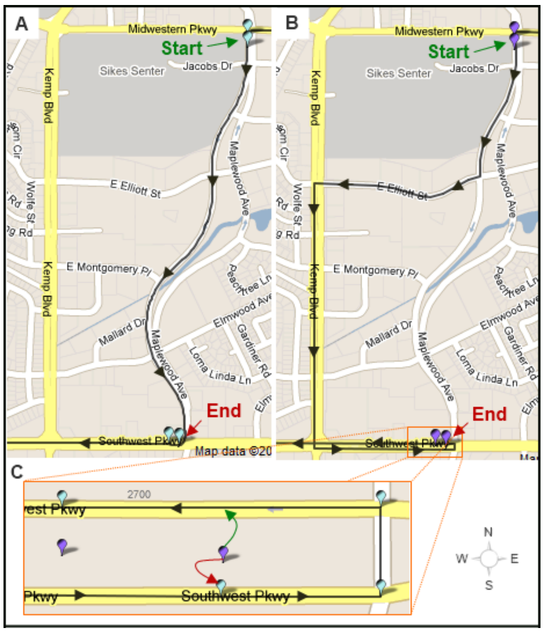

Figure 5: (A) Shows the original route with a path between "Start" and "End" and (B) shows an alternate route chosen by the DD service between the two. (C) exhibits a blown up section of the map showing why the DD service had to deviate from the original route to ensure it included the mismatched point.

2.2.1 *DistPoint*

The value of this attribute is obtained by finding the distance from A to B. The idea behind this attribute is that mismatched points could be identified by having a distance over some discovered threshold. We found that GPS error rendered this useless, because the point was always matched to the closest segment (correctly or incorrectly), and therefore no discoverable threshold could be identified.

2.2.2 *DiffBearing*

This is simply the difference in bearing between A to B. Typically, a mismatched point would have a bearing completely different then the original point due to the DD service having to find an alternate path in order to keep the submitted point included in the route. In figure 6, it can be seen that the point along the original path (dotted line) was matched to the road segment to the south, when it should have been matched to the segment north of it, causing the bearing difference between A and B to be close to 180 degrees.

2.2.3 *PathRatio*

This attribute proved to be the single most important attribute when identifying mismatched points. The path ratio is obtained by calculating the sum of the distances between each point from "Start" to "End" in the original path and in the returned path. If the ratio is over some

isons made between points always refers to a point, A, originally identified by the Peucker/inverse corner point selection algorithm, and B, the closest point (based on distance) returned by the DD service.

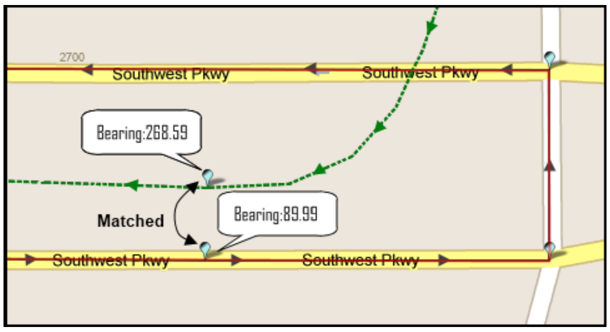

Figure 6: The original point (along the dotted path) when compared to its matched point (along the solid path) should have a similar bearing. If the bearing difference is over some threshold, it is most likely mismatched.

threshold, it then implies that the path returned from the DD service deviated from the original path in some manner. An example of this can be seen in figure 5. We calculated the path ratio difference for points before and after a submitted way point.

2.2.4 *DuplicatePoint*

A duplicate point is returned from the DD service in instances when it had to alter the route in order to keep a mismatched point within the returned route. In the context of this paper, a duplicate point can be defined as:

$$\{< p_1, t_x > \ldots < p_1, t_y >\}$$

where p is the same lat/lon, but the time difference between x and y is greater than 1. This means that the DD service returned a point that it needed to travel through twice to keep all the points submitted by our Peucker/inverse corner point selection algorithm included in the route. Any submitted point with a duplicate at $t-1$ or $t+1$ is also suspect of being mismatched.

Of the attributes listed above, the two most important were found to be 1)PathRatio and 2)DiffBearing. Either of these two values being over the discovered thresholds gave much confidence in eliminating the submitted point. Once the mismatched points have been identified, they are removed from the set of selected points. The revised set of selected points is then resubmitted to the DD service and a correctly map matched route is returned. Figure 7 shows an entire route starting with the identified by the Peucker/inverse corner points and ending with a map matched route after removing a problematic point.

3. EXPERIMENT AND RESULTS

3.1 Gps Data

Our experiments collected trace data by placing GPS loggers in the cars of 10 volunteers. The loggers are the "Genie BGT-31" handheld GPS data logger which use the SiRF Star III chip set and an optional SD card slot. Each logger was given to the subjects with a car charger and a 2GB SD card. The device was set to log raw NMEA data directly to the SD card and could hold approximately 50 days worth of

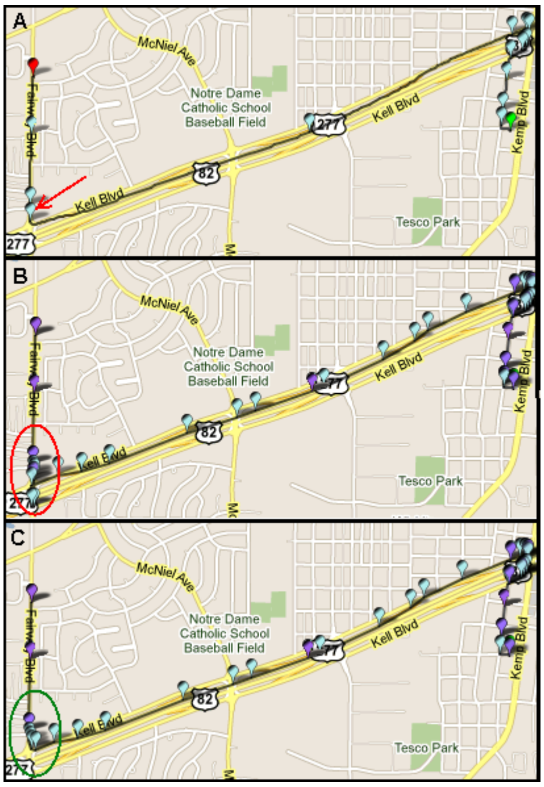

Figure 7: (A) Original route with points selected for submittal. Red arrow shows bad point needing removal. (B) First result from the DD service before bad point removal. (C)Final Map-Matched result.

GPS data. Five subjects collected the corpus of data which contained over 8.9 million individual data points with a time granularity of 1-2 second intervals. Each subject collected data an average of 34.3 days with a total number of 584 stops and 341 routes identified.

3.2 Raw GPS Segmentation

As a precursor to testing our Map Matching technique, we had to collect a significant amount of GPS traces. Working with raw GPS traces is undesirable and clarifying the data is a crucial step as mentioned in [2]. The majority of the data was collected in a 70 square mile geographical area surrounding Wichita Falls, Texas, a city with a population of just over 100,000. We segmented the raw GPS data by sorting it first chronologically and then identifying stops within the recorded points $(P_1, ..., P_n)$. Our collected GPS traces were logged using "continuous" mode, which means that data points were being logged continuously regardless of whether the subject was moving or not.

To identify stops with data logged in "continuous" mode, one can not simply look for time gaps between P_i and P_j over some threshold. First, identify when a subject is not moving (change in time without change in location), and second determine if the stop in movement is a purposeful stop. We assumed any stop over a threshold of 180 seconds as being purposeful. This threshold was chosen through no objective means, simply a heuristical choice.

After stops were identified, subsequently identifying the routes that connected each of the stops was possible. We identified 584 stops and 341 routes. To ensure that the identified features within the data were reliable, we did not use a significant portion of the data that did not fit into our expected parameters for stops or routes. With the data that was usable, the average trip time was 11.17 minutes, the average trip distance was 4.38 miles, and the number of trips per day averaged 3.8. This fits into the expected parameters as summarized in a 2001 national household travel survey[7].

3.3 Results

The experiment utilized a framework written by the authors that identified key features within GPS streams for individual users. The resulting data consisted of a total of 584 stops and 341 routes. Out of the 341 routes, 200 were selected for analysis due to there proximity within the test city and the availability of the subjects which allowed visual verification from the original drivers. Out of the 200 routes analyzed, we had 100% accuracy with the two stage submission process of our algorithm, and achieved a pristine standardized set of driven routes. Although our experimental step seems simple (visual confirmation), considerable time was spent inspecting possible problematic route features such as turn-arounds and overpasses to ensure the algorithm functioned properly on those difficult routes.

4. RELATED WORK

Many different online and offline map matching algorithms have been developed and implemented (Marchal et al. [8], Yin and Wolfson [12], Quddus et al. [9]).

White et al. [11] implemented and tested four different online Map Matching algorithms. All four algorithms attempt to match GPS data to a road network that is represented by a graph $G(V, E)$. The first algorithm simply snaps the GPS data point to the nearest edge in the graph G. The second algorithm implemented by White et al. builds upon the first algorithm by also considering the heading of the GPS data. The third algorithm takes the graph connectivity into consideration. It only snaps a new point to an edge that is reachable from the previous edge (connectivity). The fourth algorithm implemented by White et al. map matches a new GPS point by using the new point combined with the previous points to construct a curve (series of line segments). This curve and the new point is then matched to the most similar series of edges in the road network graph.

Marchal et al. [8] developed an offline Map Matching algorithm that uses only the coordinate data that is collected by the GPS. It does not use a DR (dead reckoning) device, nor is heading or speed data considered. The road network used by the algorithm is represented by a directed graph $G(V, E)$ where V is the set of vertices or nodes and E is the set of edges or links. Q_i is the set of coordinate points (x_i, y_i) from the GPS data where $i = 1 \ldots T$. The algorithm starts by finding the set of N nearest links from the first GPS point which is Q_1. For each GPS point in Q_i, the N nearest links are added to the previous link thus forming multiple paths. Finally, the difference (error) between the points in Q_i and the paths is calculated and scored. The path with the lowest score is chosen to be the best estimation of the actual route that was traveled.

Yin and Wolfson [12] developed a weight based offline Map Matching algorithm. In addition to the coordinate data collected by the GPS, the algorithm uses heading data. The distance between the GPS coordinate data and nearby road network links (edges in a graph) is calculated as well as the heading difference between the GPS coordinate data and nearby road network links. These calculated values are used to assign weights to define how well a coordinate point matches a specific link. The route (series of links) with the lowest total weight is considered to be the map matched route.

Alt et al. [1] implemented a Map Matching algorithm which uses Frechet distance to match a route formed by GPS data points to a sequence of edges in a road network graph $G(V, E)$. The Frechet distance is commonly described as follows: Suppose a person is walking a dog, the person is walking on one curve (route) and the dog on another. Both are allowed to control their speed but they are not allowed to go backwards. Then the Frechet distance of the curves (routes) is the minimal length of a leash that is necessary for both to walk the curves from beginning to end.

Quddus et al. [9] implemented an online Map Matching algorithm using fuzzy logic to match GPS data to a road network graph. In addition to GPS point data, the algorithm requires data provided by a gyroscope, vehicle speed sensor, back-up indicator, and a 24-channel dual-frequency geodetic receiver.

Our approach is unique in the way of selecting waypoints, using a black-box routing algorithm, and the refine-and-filter step to achieve high accurate results.

5. CONCLUSIONS AND FUTURE WORK

We conclude that our novel offline map matching algorithm succeeded in accurately creating a standardized set of routes for individual users. In the visual inspection process, there was little or no variation between like routes. The creation of a successful and accurate offline map matching algorithm is a preliminary step toward our goal of using the map matched data for the creation of prediction algorithms, whether it be route or location prediction.

6. REFERENCES

[1] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 589–598. Society for Industrial and Applied Mathematics, 2003.

[2] L. Cao and J. Krumm. From gps traces to a routable road map. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 3–12, New York, NY, USA, 2009. ACM.

[3] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[4] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[5] S. Erle, R. Gibson, and J. Walsh. *Mapping hacks: tips & tools for electronic cartography*. O'Reilly Media, Inc., 2005.

[6] P. Hart, N. Nilsson, and B. Raphael. Correction to a formal basis for the heuristic determination of minimum cost paths. *ACM SIGART Bulletin*, (37):28–29, 1972.

[7] P. Hu. Summary of Travel Trends 2001 National Household Travel Survey. Technical report, ORNL, 2005.

[8] F. Marchal, J. Hackney, and K. Axhausen. Efficient Map-Matching of Large GPS Data Sets-Tests on a Speed Monitoring Experiment in Zurich, volume 244 of Arbeitsbericht Verkehrs und Raumplanung. 2004.

[9] M. Quddus, R. Noland, and W. Ochieng. A high accuracy fuzzy logic based map matching algorithm for road transport. *Journal of Intelligent Transportation Systems*, 10(3):103–115, 2006.

[10] J. Quinlan. *C4. 5: programs for machine learning*. Morgan Kaufmann, 1993.

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

[12] H. Yin and O. Wolfson. A weight-based map matching method in moving objects databases. 2004.