ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/246773761

Matching GPS Observations to Locations on a Digital Map

Article · January 2002

CITATIONS
424

READS
10,264

1 author:

Josh Greenfeld
New Jersey Institute of Technology
26 PUBLICATIONS **591** CITATIONS

SEE PROFILE

All content following this page was uploaded by Josh Greenfeld on 18 May 2017.

The user has requested enhancement of the downloaded file.

*Matching GPS Observations to Locations on a Digital Map.

Joshua S. Greenfeld

Department of Civil and Environmental Engineering

New Jersey Institute of Technology

Newark, NJ 07102

Greenfeld@adm.njit.edu

ABSTRACT

GPS based navigation and route guidance systems are becoming increasingly popular among bus operators, fleet managers and travelers. To provide this functionality, one has to have a GPS receiver, a digital map of the traveled network and software that can associate (match) the user's position with a location on the digital map. Matching the user's location has to be done even when the GPS location and the underlying digital map have inaccuracies and errors.

There are several approaches for solving this map matching task. Some only match the user's location to the nearest street node while others are able to locate the user at specific location on the traveled street segment. In this paper a topologically based matching procedure is presented. The procedure was tested with low quality GPS data to assess its robustness. The performance of the algorithms was found to produce outstanding results.

Keywords: Global Positioning System (GPS), Map Matching, Route Guidance, and Navigation.

INTRODUCTION

The proliferation of handheld and dashboard mounted traveler guidance systems with navigation software and map databases are changing the way people travel to and arrive at their desired destinations. Contemporary navigation systems provide not only a visual map and a current location of the travelers, but also directions for the quickest route to user selected destinations. These navigation systems, sometimes referred to as Personal Navigation Assistants (PNAs), are installed in rental cars and delivery fleets, and are now available as an optional accessory in several new car models. Other PNAs are portable systems that are based on either a notebook computer or on a handheld Personal Digital Assistants (PDA) computer. PNAs are also available for off-the-road activities such as hiking, treasure hunting and other recreational endeavors. Location aspects of this type of application will not be discussed in this paper.

To provide route guidance it is necessary to have two essential components: a digital map and a device to locate the user. In essence, there are three different ways to determine the user's location (1). The first is to use some form of *dead reckoning* (DR) system in which the user's speed and direction of movement are continuously used to update her/his location (2). Many DR systems use a compass and/or a gyro system to augment the determination of the absolute heading. The biggest problem with dead reckoning is that it is a relative positioning scheme. This means that the absolute positioning error grows proportionally with the distance traveled. Therefore, in lengthy trips DR will produce poor positioning.

The second method to determine the user's location is to use some form of ground-based (*Terrestrial*) *Radio Frequency* or *beacon* that broadcasts its location to nearby users (3). The received signals are used to triangulate the position of the user. This system is an absolute scheme, thus the accuracy does not degrade with time. However, the usage of such a system is limited to areas that are within visibility of the radio towers emitting the broadcast signals. Usage of such a system is not commonly available to the general public.

*Proceedings of the 81th Annual Meeting of the Transportation Research Board, January 2002, Washington, DC.

The third method to determine the user's location is to use some form of *radio/satellite positioning system* such as the *Global Positioning System* (GPS) (4). GPS signals and other transmitted information are received by the PNA and are used to compute the user's location. GPS based location devices are by far the most popular and least expensive system for determining the user's location. Therefore, henceforth, GPS will be referred to as the default locating device. GPS can provide not only a position in terms of coordinates but also speed and direction. In this paper only GPS derived coordinates will be considered to make the algorithm applicable to other locating devices that do not provide speed and heading information.

The second element necessary to provide route guidance is a digital map. The map offers the users a spatial reference for their location. It allows users to associate their observed position with a physical location in the real world. For a PNA to be effective, the location indicated on the digital map should correspond to the actual position of the user. If both the digital map and the GPS location were perfectly accurate this would have been a straightforward task. All one has to do is to snap the location obtained from the GPS to the street network. However, GPS can have locational errors and the digital map can have not only positional error but suffers from incomplete information such as missing street segments, etc. To make the PNA function effectively one should improve the accuracies of the map (see (5) and (6)) and develop algorithms that are capable of determining the correct user location even with inaccurate map/network data.

Map-matching algorithms are used to reconcile inaccurate locational data with an inaccurate map/network data. The necessary sophistication of the map-matching algorithms depends on the nature of the application and the available data. There are three complexity levels that a map-matching algorithm has to resolve. The most straightforward algorithm is needed when the user travels on a fixed network. For example, a bus traveling on known street segments, thus, all that is needed is to locate the bus on one of the street segments that makes up the bus route. That is, the search domain for a street segment as a match candidate is very limited.

A second level of map-matching algorithms is needed when the user inputs a destination and the PNA determines a suggested traveling route. In this application, the algorithm assumes that the user follows the suggested route and matching is performed to that route. If the user deviates from the suggested route, the system detects a large discrepancy between the GPS location and the matched location. Normally in such cases, a new route and subsequently a new match are computed. The main drawback of using "known route" information is that it can result in an incorrect match if the user deviates only slightly from his/her "known route". For example, if the user elects to travel on a nearby parallel street, most of these algorithms will determine that the user is traveling on the "known route". Detecting this type of a mistake could be very difficult.

The third and most general map-matching algorithm does not assume any knowledge or any other information regarding the expected location of the user. It uses only coordinates such as x,y or Latitude, Longitude, and the relevant street network to locate the user. In this paper only the general map-matching algorithm will be discussed.

In the next section, a formal definition of the map-matching problem is given. Next, some prior approaches for solving the map-matching problem are reviewed and discussed. Some of these approaches are based only on geometric information (e.g. nearest node) while others are based on statistical (e.g. (8)) and topological analysis (e.g. consistency with prior matches). In the given review only algorithms that use geometric and topological are discussed. Our approach/algorithm for solving the problem is subsequently described. Finally, conclusions and possible future research directions are outlined.

PROBLEM STATEMENT (FOLLOWING (1))

A vehicle (or a person) is moving along a finite system (or set) of streets, $\overline{N}$. A location device such as GPS provides an estimate for the vehicle's location at a finite number of points in time, denoted by $\{0,1,\ldots,t\}$. The vehicle's actual location at time t is denoted by $\overline{P}^t$ and the estimate is denoted by P^t. The

goal is to determine the street in $\overline{N}$ that contains $\overline{P}^t$. That is, to determine the street that the vehicle is on at time, *t*. In addition, the location of the vehicle relative to the end points of the street is also to be determined

The street system, $\overline{N}$, is virtually never known exactly. Instead, a *network representatio*n, *N*, of $\overline{N}$ is constructed using various mapping techniques. As illustrated in Figure 1, *N*, consists of a set of curves in $\boldsymbol{R}^2$, each of which is called an *ar*c. Each arc is assumed to be piecewise linear. Hence, arc $A \in N$ can be completely characterized by a finite sequence of points (A^0, A^1,…, A^n). A^0 and A^n, the endpoints the of the line segment *A*, are referred to as *nodes* while A^1, A^2,…, A^{n-1} are referred to as *vertices* or *shape point*s. A node is an end point at which an arc terminates or begins while a shape point is used to show the geometry of the arc. A node can be a transition point from one arc to another (e.g., corresponding to an intersection in the street system) or a terminal (e.g., corresponding to a dead-end in the street system) point in the network.

Figure 1: The Map-Matching Problem

The goal of the *map matching problem* is to match the estimated location, P^t, with an arc, *A* in the "map", *N*, and then to determine the street, $A^i \in \overline{N}$, that corresponds to the vehicle's actual location. In some applications such as matching the location of the bus to a bus stop, it is necessary to determine the position on A^i that best corresponds to $\overline{P}^t$. An assumption that we have to make here is that there is a one-to-one correspondence between the actual street network $\overline{N}$ and the map network *N*, otherwise $\overline{P}^t$ cannot be correctly mapped into *N*.

MAP MATCHING ALGORITHMS

It is worthy to distinguish between map-matching methods that use some known facts about the expected route of the user and methods that use no such information. Knowing the user's route can make the map-matching task easier since the search for the arc A^i is very much constrained by the known route. For example, matching the location of a bus along its route is a relatively easy task since the bus is expected to follow a fixed set of arcs in a predetermined sequence. Similarly, PNDs that compute the users route to his/her destination can use this information to narrow the matchable search space to the suggested route. However, confining the search space to only "expected to be traveled" arcs is not always a good idea, especially in the latter case. The bus or the PND user could intentionally or unintentionally deviate from their routes. Various circumstances such as bad traffic conditions or inaccessibility of a given street segment could force them to travel on an alternative route. Therefore, a more general map-matching algorithm has to be developed to augment such circumstances.

It is also customary to distinguish between map-matching methods that use only geometric information (7) and those using topological information as well (1). When using only geometric information, one makes use only of the "shape" of the arcs and not of the way in which they are "connected". When using

topological information one makes use of the geometry of the arcs as well as of the connectivity, proximity and contiguity of the arcs. Thus, the match is done in context and in relationship to the previously established matches. That makes the topological solution much more likely to be correct.

USING ONLY GEOMETRIC INFORMATION

The most intuitive approach for solving the map-matching problem is to match P^t to the nearest node or shape point A^i. While this is a relatively simple solution to the problem, it yields in many cases undesirable results. Figure 2 illustrates this approach and the map matching results. The user's observed GPS positions are indicated by P^i, *i=0…7* and the network is composed of arcs *A*, *B* and *C*. One can see that the user is traveling along arc *B* and then turns left to arc *C*. However, the matching algorithm associates the GPS points with arc *B* then *A* and finally with arc *C*.

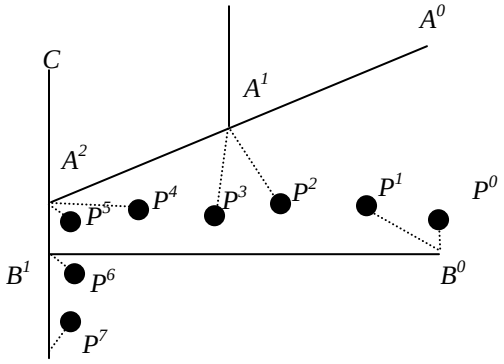

Figure 2. Matching GPS points to the nearest node (or shape point)

Another intuitive map matching solution is to find the closest arc, in *N*, to the user's observed position P^t. The closest arc can be found by projecting P^t onto the network *N* and find the arc with the smallest distance from P^t. The smallest distance between a point and a line is the perpendicular distance or the normal to a line. With reference to figure 3, the perpendicular distance can be computed from the following equations:

$$X = \frac{m(y_3 - y_1 + m \cdot x_1) + x_3}{m^2 + 1} \quad (1)$$

$$Y = y_1 + m(X - x_1) \quad (2)$$

$$D = \sqrt{(X - x_3)^2 + (Y - y_3)^2} \quad (3)$$

Where :
(x_1, y_1) , (x_2, y_2) , (x_3, y_3) and (X,Y) are node coordinates as shown in Figure 3

$$m = \frac{y_2 - y_1}{x_2 - x_1} \quad (4)$$

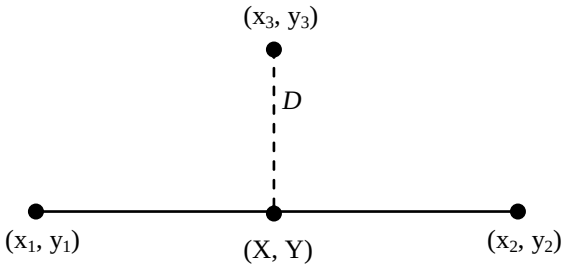

Figure 3. The geometry of a perpendicular distance.

As seen in figure 4, this algorithm also produces incorrect results by matching the user's location first to arc *B* then to arcs *A, C, B* and finally again to arc *C*. Other difficulties that may arise from this approach is that when P^t is projected onto the nearest arc it may fall on the extension of the arc rather than on the arc itself. Therefore, one must be careful to consider only arcs that the projected point of P^t indeed lays on the arc, not on its extension.

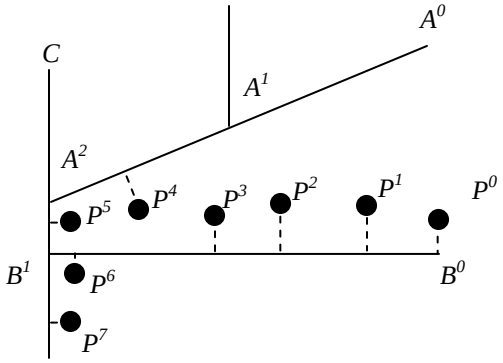

Figure 4. Matching GPS points to the nearest arc

The main shortcoming of the geometric information only approach is that it assumes that every new point in P^t is matched without considering the matched location of point P^{t-1}. Using prior information on the arc that was matched to point P^{t-1} could provide vital information on which arc should point P^t be matched to. It makes sense that if point P^t is near point P^{t-1}, then point P^t should match either with the same arc or with another arc that is connected to the current arc in the general traveling direction.

Map matching methods that consider the connectivity, adjacency and proximity characteristics of the network and the location sequence of the user, are called topologically based map matching methods. Considering the topological characteristics of the network and the progression of the user along the network could potentially prevent the algorithm from jumping from one arc to another as seen in figures 2 and 4.

A PROPOSED MAP MATCHING PROCEDURE

A map matching procedure was developed with the following characteristics: It is based on topological analysis and it uses only coordinate information on the observed position of the user. It assumes no

knowledge of the expected traveling route and it does not use any GPS determined heading and/or speed information.

The map matching procedure is composed of two separate algorithms. Each algorithm has a specific function that is applied at a specific stage of the process. The first algorithm is called InitialMapping(). It finds an initial match. The initial match is needed to locate the user somewhere on the network. Based on this initial match subsequent topological analysis can then be performed. The second map matching algorithm in our procedure is called Map(). It is the main matching algorithm that uses topological reasoning and a weighting scheme to match a GPS point P^t with arc A of the network N. The Map() algorithm is applied only after an initial match was found using InitialMapping().

The InitialMapping() is applied in the following circumstances:

1. When the first GPS point P^0 is received.
2. When the distance between the new GPS point P^t and point P^{t-1} exceeds a pre-selected distance tolerance. A very large distance between two consecutive GPS points can occur if the receiver temporarily stops tracking the GPS signal due to obstructions or due to other brief mechanical failures.
3. When the main map-matching algorithm is unable to successfully map a particular GPS point. In such an instance, the procedure is reset to start a completely new matching task.

The InitialMapping() uses the following steps to determine an initial match of P^t to A.

1. Find the closest street node A^0 to point P^0. This is essentially a geometric only matching process.
2. Determine all the arcs in N that are connected to node A^0
3. When P^1 becomes available, map the point onto one of the selected arcs.

The Map() algorithm is composed of the following steps:

1. Obtain the next GPS point P^t.
2. Form a GPS line segment between points P^{t-1} and P^t.
3. Evaluate the proximity (distance) and orientation (direction or Azimuth) of the GPS line to the currently matched street segment A^i. The method for evaluating the proximity and orientation is discussed below.
4. If the new point P^t does not map onto the current segment A^i, find another street segment, A^{i+1}, which is either connected to A^i or is nearby down stream from A^i. Arc A^{i+1} is also selected based on the same proximity and orientation evaluation scheme.

Similarity Assessment

There are two measures that can be used to determine the proximity of a point P^t to an arc A^i. The first and most simple proximity measure is to compute the shortest (or the perpendicular) distance from P^t to A^i. If in fact point P^t is related to arc A^i one would expect that point P^t is close to A^i. However, as discussed earlier, the suggestion that a correct match was found only because of the geometric fact that an arc is the closest one to the point, is flawed. The fact that the arc is the closest to the point provides evidence that candidacy for a match should be considered but not that the correct match was definitely found.

The second proximity measure is to determine whether the line between points P^t and P^{t-1}, and the arc A^i, intersect. If an intersection occurs between two lines that, in general, follow the same direction, it is likely that a correct match was found. Conversely, if the two lines are close to being perpendicular to each other, it is very unlikely that a correct match was found. In both cases, the lines have to intersect on the arc segment and in between points P^t and P^{t-1}, not on the extensions of either of these lines. If the intersection occurs on the extension of either of these lines, the intersection criterion is invalid. Figure 5 illustrates the various intersection cases. Intersections *a* and *b* show correct matches. Intersection *c* is incorrect because of the large angle between the intersecting lines P^2, P^3 and A^2. Intersection *d* falls on the extension of the lines P^3, P^4 and A^1 and therefore, the decision whether point P^4 is on arc A^1 or not, should not be based on the fact that these lines intersect somewhere in space. Any two lines will inevitably intersect somewhere in space.

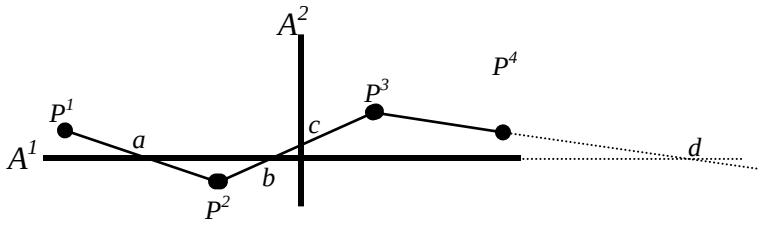

Figure 5. Intersections of GPS lines and street segments.

Another measure of likelihood for a correct match is the orientation or the direction of the arcs and the GPS lines. The direction of the line between consecutive GPS points and the direction between the two nodes of the network arc should be similar. Several orientation metrics such as angle, bearing or azimuth can be used to assess similarity. We elected to use the azimuth similarity criterion to determine whether or not an arc can be a candidate for a match. The benefit of computing the azimuth instead of a simple angle between the lines is that an azimuth has a ranges of 0 to 2π and therefore, it distinguishes between 2 parallel lines that are pointing to opposite directions.

The azimuth of a constructed GPS line can be either computed or obtained as part of the GPS data string. Since our approach in solving the map-matching problem was that only coordinate information is available, the azimuth of the GPS line was computed. The azimuth of the arc *A* is not available from other sources and therefore has to be computed. The azimuth of a line from P^1 to P^2 can be computed from:

$$Az_{P^1,P^2} = \tan^{-1}\frac{x_2 - x_1}{y_2 - y_1} \qquad \text{if } (y_2\text{-}y_1)<0 \text{ add } \pi, \text{ if } <0, \text{ add } 2\pi \qquad (5)$$

Similarity Evaluation

To determine which arc A^i should be matched with point P^t, a weighting system based on similarity criteria was established. The weighting system evaluates several candidate arcs for a correct match by computing a likelihood score based on the different criteria that were discussed earlier. These criteria are: the perpendicular distance of the GPS point from the arc segment, the degree of parallelism between the GPS line and the street network arc and the intersecting angle if an intersection exists. The weighting formula is:

$$W = W_{AZ} + W_D + W_I \qquad (6)$$

Where:
W - the total score
W_{AZ} - is the weight for similarity in orientation
W_D - is the weight for proximity of the point to the line
W_I - is the weight for intersection if applicable.

To compute the various weights, we propose the following formulation:

$$W_{AZ} = C_{AZ} \cdot \cos^{n_{AZ}}(\Delta AZ) \qquad (7)$$

$$W_D = C_D - a \cdot D^{n_D} \qquad (8)$$

$$W_I = C_I \cdot \cos^{n_I}(\Delta AZ) \qquad (9)$$

Where:

ΔAZ - is the difference between the azimuths of the GPS line and the evaluated network arc

C_i - is a constant which constitutes the maximum score

n_i - is a power parameter.

By selecting different values to these parameters, one can control the amount of weight that is given to a particular matching characteristic. For example, selecting a higher value for C_{AZ} compared to C_D means that similarity in orientation is more important than proximity. Selecting a higher value for n_D than for n_{AZ} means that the matching weight will decrease more rapidly as the GPS point is further away from the arc A^i compared to the decrease in the weight as the orientation of the lines diverge away from each other. The use of *cosine* for computing azimuth related weights and applying an odd number for the power n_i not only decreases the weight as the ΔAZ increases but will also introduce a negative weight if ΔAZ is larger than 90°. Accordingly, the total weight of a match will decrease for lines that point in opposite directions and a high distance weight could be offset by the weight of incompatible orientation.

Filtering outliers

Even with selective availability turned off, the GPS signal can sometimes exhibit a jittery pattern. In other words, the user's location seems to follow a certain street segment when all of a sudden a spike occurs in one of the GPS positioning. Then, the former positioning pattern is resumed. The main cause for this behavior is the effect of Multipath on the GPS signal. An example to an irregular jump in GPS positioning can be seen in figure 6. Points P^2, P^3, P^5 and P^6 are generally parallel to arc A^0. However, point P^4 is like a spike that does not fit that pattern. A spike or an outlier such as point P^4 can cause matching errors especially if the spike was bigger and located closer to arc A^1. If point P^4 would have been evaluated based on the computed direction from P^3 - P^4, it should have been matched to arc A^1.

To avoid such matching mistakes, a safeguard is implemented. The safeguard is that if a significant change in direction of the lines P^{t-1}-P^t occurs, the mapping of point P^t is finalized only after point P^{t+1} and P^{t+2} have been observed and analyzed. In Figure 6, the lines between points P^3 and P^4 exhibit a sharp change from the previously observed directions. Therefore, point P^4 will be mapped only after the directions for the lines P^4-P^5 and P^5- P^6 are computed. The 2 second (assuming the user's position is computed every second) delay in position determination is of insignificant consequence to the user but very beneficial to the robustness of the matching procedure. This delay allows the algorithm to verify whether the computed direction for the GPS line between points P^3 - P^4 is consistent with the direction of the lines P^4 - P^5 and P^5-P^6. If the direction of line P^3 - P^4 does not fit the pattern established by the lines preceding P^3 and the lines following P^4, the position of point P^4 is considered to be an outlier.

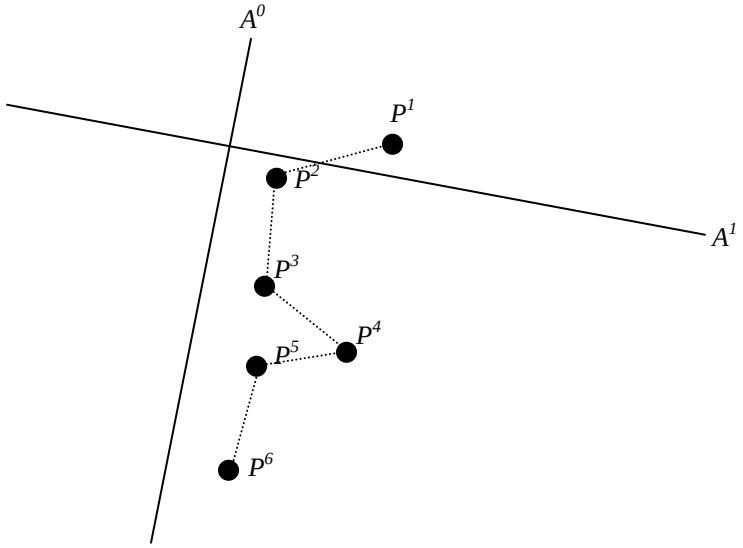

Figure 6. Spikes caused by erroneous GPS locations

Another spike related safeguard against incorrect matching is to compute the remaining distance from the last matched point to the end on the currently matched arc A^i. If that distance is larger than what can be considered as a reasonable traveling distance, the GPS point P^t will be matched to the current arc A^i.

Skipping Arcs

The topologically based matching algorithm assumes continuity between traveled arcs. This means that when a user completes traveling on arc A^i, the next "travelable" arc, A^{i+1}, must commence with the end-node of A^i. Therefore, if point P^t cannot be mapped on A^i it should be mapped on A^{i+1}. However, sometimes this assumption is not true. For example, the network may include a very short arc on which the user travels very fast and does not have the opportunity to make a location observation on that arc. The same can happen if the GPS signal is blocked for a few seconds and when position determination is resumed, the user is already on arc A^{i+2}. This situation is illustrated in figure 7.

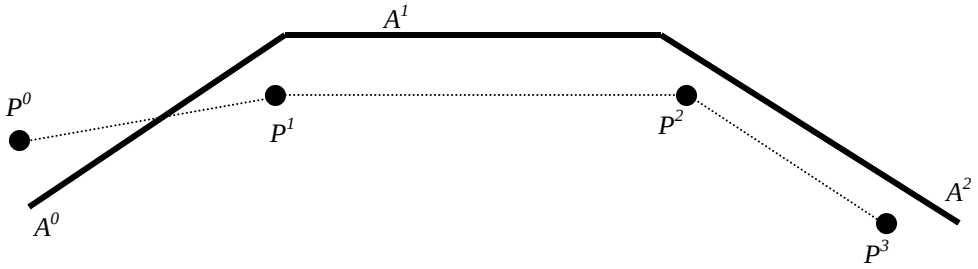

Figure 7. Skipping of an arc in the matching process.

The arc-skipping problem can be addressed in two different ways. The first and most simple solution is to restart the matching process with InitialMapping() whenever an arc-skipping situation arises. That is, if point P^t cannot be matched on the current arc A^i or on any other arc A^{i+1} that is connected to arc A^i, then restart the matching process with P^t becoming P^0. The drawback of this easy solution is that the already known information about the location of the user is lost in the initialization process and the user has to be relocated with respect to the entire network N.

Another solution to the arc-skipping problem is to attempt to map point P^t on all of the network arcs that are connected to the end-nodes of all possible A^{i+1}. That is, instead of looking for a match only at arcs that are connected to the end-point of the current arc, one can look for a match at the arcs that are connected to the end points of the arcs that are connected to the end points of the current arc. This may increase the need

for topological bookkeeping but at the same time it will confine the next matched arc to the vicinity of the current one.

IMPLEMENTATION AND EVALUATION

The map matching procedure was tested with GPS observations made on a bus route between Dunellen, New Jersey, and the Port Authority (PA) bus terminal in New York City (Manhattan). This route encompasses many different operational environments ranging from urban canyons to open suburban areas. The data was collected while Selective Availability (SA) was still turned on. This makes the data much less accurate than one should expect to have with today's GPS capabilities. The rationale for selecting this low-grade data set was to test the thoroughness of the procedure under adverse conditions.

Given the fluctuation in GPS positioning, very short lines may exhibit irregular noisy patterns that may make the computed orientation and thus the similarity evaluation irrelevant. To avoid forming very short lines between two consecutive GPS points, the user was expected to move a minimum of 10 units (feet) from his/her last position. If the next GPS point was less than 10 units away from the last mapped GPS point, it was skipped.

Figure 8 shows the results of the matching procedure on the first few arcs of the tested bus route. The weighting parameters that were used are shown in Table 1. Point 2 which is the right-most point in figure 6, was mapped using the InitialMapping() algorithm while all other points were mapped with Map(). The numbering of the GPS points in figure 6, jumped from 2 to 134 because the bus was in a bus stop and did not move more than 10 feet for about 2 minutes.

W_{AZ}	$C_{AZ} = 10$ $n_{AZ} = 5$
W_D	$C = 10$ $a = 0.1$ $n_D = 1.2$
W_I	$C_{AZ} = 10$ $n_{AZ} = 10$

Table 1. An example of weighting parameters.

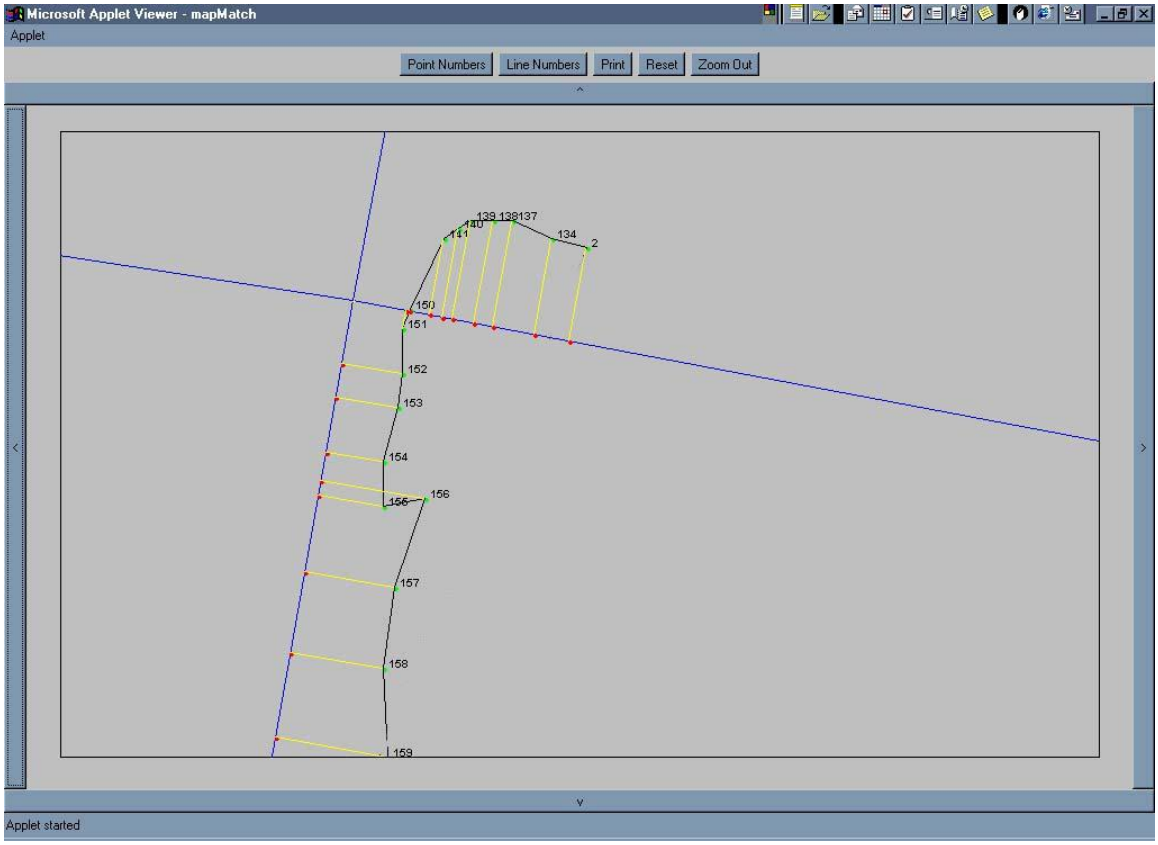

Figure 8. Results of the map matching procedure at the beginning of the bus route.

Examining the results of the entire route we found that a correct match was made virtually along the entire length of the tested route. The very few isolated mismatches that were computed, were immediately corrected by consequently computed matches.

LOCATING THE USER ON THE MATCHED ARC

In many applications it is important to know where the user is not only in terms of the arc on which he/she is traveling on, but also where the user is on the arc itself. For example, in order to know whether a bus has arrived at a bus stop or in order to determine how far the bus is from the next bus stop, we need to know the bus's position relative to the end points of the arc. This problem is a rather challenging task especially because of network mapping and GPS positioning errors. A typical approach to solve this problem is to use Kalman filtering ((9) and (10)) to predict the location of the user on the arc. The arrows shown in figure 9 illustrate the actual location of the user on the network which are different from the projected location of the points on the arc. Currently several averaging and smoothing operators are being tested to address this problem.

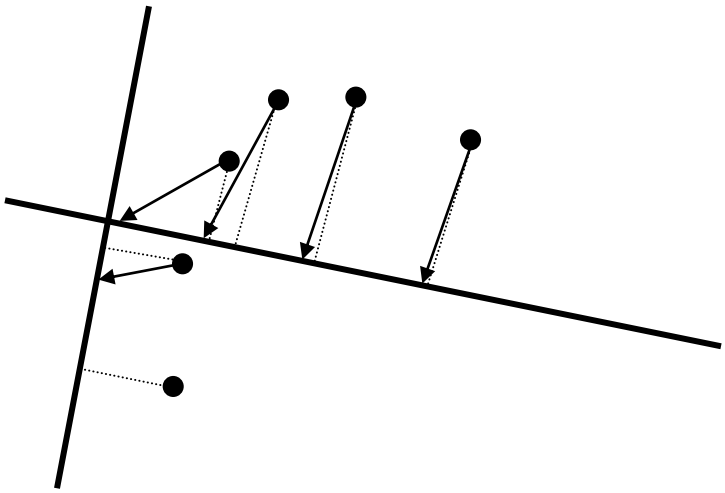

Figure 9. Actual vs. projected location of the user on the map.

CONCLUSIONS AND FUTURE RESEARCH

Associating the location of a traveler in the real world with a location on a digital map is important in many applications. While for most people this may be an easy task to resolve visually, it is a fairly difficult task to implement as an algorithm.

In this paper we reviewed several approaches for solving this problem and proposed a weighted topological algorithm. The algorithm was based on assessing the similarity between the characteristics of the street network and the positioning pattern of the user. A weighted score was computed and the match was determined by selecting the highest score or the most likely candidate for a correct match. Some measures to remove outliers due to GPS errors and to handle skipping of unmatched arcs have been implemented as well.

The algorithm seems to work well even with relatively inferior or bouncy GPS data. Tests showed that the procedure computed correct matches virtually everywhere. Though our results thus far have been good, additional research still needs to be done to verify the accurate performance of the algorithm. Additional research has to be done to make an accurate position determination along the traveling route. This is essentially locating the traveler in the correct position on the arc, not only locating him/her on the correct arc. Finally, the algorithm has to pass the ultimate test, which is comparing the matching results with the actual true location of the traveler. Such a test can be done by setting out control points along the route and correlate GPS location events with these control points.

ACKNOWLEDGEMENT

This research was sponsored by the New Jersey Center for Transportation Information and Decision Engineering (www.njtide.org) and the New Jersey Commission of Science and Technology.

REFERENCES

1. White, C.E., D. Bernstein and A.L. Kornhauser (2000) "Some Map-Matching Algorithms for Personal Navigation Assistants " , Transportation Research C , Vol. 8 , pp. 91-108.
2. Collier, W.C. “In-Vehicle Route Guidance Systems Using Map Matched Dead Reckoning”, *Proceedings of IEEE Position Location and Navigation Symposiu*m, pp. 359-363, 1990.
3. Iwaki, F., M. Kakihari, M. Sasaki. “Recognition of Vehicle’s Location for Navigation”, *Proceedings of the Vehicle Navigation and Information Sys-tems Conferenc*e, pp. 131-138, 1989.
4. Hofmann-Wellenhoff, B., H. Lichtenegger, and J. Collins. *GPS: Theory and Practic*e, 4th Edition, Springer-Verlag, New York, 1997.

5. Shibata, M. “Updating of Digital Road Map”, Proceedings of the Vehicle Navigation and Information Systems Conference, pp. 547-550, 1994.
6. Schiff, T.H. “Data Sources and Consolidation Methods for Creating, Im-proving and Maintaining Navigation Databases”, *Proceedings of the Vehicle Navigation and Information Systems Conferenc*e, pp. 3-7, 1993.
7. Kim, J.-S., “Node Based Map Matching Algorithm for Car Navigation Sys-tem”, *Proceedings of the International Symposium on Automotive Technology and Automation* pp. 121-126, 1996.
8. Scott, C.A. and C.R. Drane. “Increased Accuracy of Motor Vehicle Position Estimation by Utilizing Map Data, Vehicle Dynamics and Other Information Sources”, *Proceedings of the Vehicle Navigation and Information Systems Conferenc*e, pp. 585-590, 1994.
9. Krakiwsky, E. J., C. B. Harris, and R. V. C. Wong. “A Kalman Filter for Integrating Dead Reckoning, Map Matching and GPS Positioning”, *Proceedings of IEEE Position Location and Navigation Symposiu*m, pp. 39-46, 1988.
10. Jo, T., M. Haseyamai and H. Kitajima. “A Map Matching Method with the Innovation of the Kalman Filtering”, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science*s, Vol. E79-A, pp. 1853-1855, 1996.

View publication stats