An Introduction to Map Matching for Personal Navigation Assistants

David Bernstein and Alain Kornhauser
Princeton University

August 1996

New Jersey TIDE Center
Directed by Prof. Louis J. Pignataro
New Jersey Institute of Technology
Newark, NJ
pignataro@admin.njit.edu
www.njtide.org

INTRODUCTION

There are three different types of Personal Navigation Assistants (PNAs). First-generation PNAs simply provide the user with a map and the ability to search the map in a variety of ways (e.g., search for an address, search for a landmark, scroll and pan). Second generation PNAs provide both a map and the user's current location/position. Third generation PNAs provide a map, the user's location, and directions of some kind.

It should be clear why we distinguish first-generation PNAs systems from second- and third-generation systems. Clearly, a system that provides the user's current location is much more complicated than one that doesn't, and generally requires both additional hardware and software. What may not be clear is why we distinguish between second- and third-generation PNAs.

The rationale for doing so is actually quite simple. In second-generation systems the location that is provided to the user need not coincide with the street system (or subway system, etc. . .) system). However, in order to provide directions, the user's location must coincide with a street (or subway line, etc. . .) when appropriate.

There are, in essence, three different ways to determine the user's location. The first is to use some form of *dead reckoning* (DR) in which the user's speed of movement, direction of movement, etc. . . is continuously used to update her/his location (1). The second is to use some form of ground-based *beacon* that broadcasts its location to nearby users (2). The third is to use some form of *radio/satellite positioning system* that transmits information that the PNA can use to determine the user's location. This last approach is by far the most popular; a great many PNAs use the *Global Positioning System* (GPS) to determine the user's location (3).

Given a GPS receiver, it is almost trivial to convert a first-generation PNA into a second generation PNA (i.e., one that provides both a map and the user's location), and many people have done so. However, reconciling the user's location with the underlying map (or network) can be much more complicated. In other words, converting a second-generation PNA into a third-generation PNA can be quite difficult.

When both the user's location and the underlying network are very accurate, the reconciliation problem is thought to be straightforward – simply "snap" the location obtained from the GPS receiver to the nearest node or arc in the network. Hence, it is not surprising that a number of people are working on improving the accuracy of both the underlying network and the positioning system.

In order to develop more accurate maps/networks, enormous "surveying" efforts are underway (4, 5, and 6). Some of these efforts are being undertaken by government agencies and others are being undertaken by private companies.

In order to develop more accurate positioning systems, a great deal of attention is being given to combining data from multiple sources. Some systems combine GPS with dead reckoning systems (7, 8 and 9), others use differential GPS (10), and others user multiple sources of data (sometimes including maps) and then filter or fuse the data in some way (11, 12, 13, 14, 15, 16, and 17).

We are interested in situations in which it is not possible or desirable to improve the accuracy of the map/network and the user's location enough to make a simple "snapping" algorithm feasible. Such situations arise for many reasons. First, not all PNAs are vehicle-based [e.g., hand-held Personal Travel Assistants of the kind discussed in (18)]. Hence, it may not be possible to use dead reckoning or other data sources. Second, even if it is possible to develop a network/map that is accurate enough, such a network may not always be available. For example, the PNA may not have sufficient capacity to store the complete, accurate network at all times and hence, may need to either store inaccurate/incomplete networks or download less-detailed networks from either a local or central server. Third, many facilities will probably never be available from map/network vendors and will need to be obtained on-the-fly from the facility, probably with limited accuracy. For example, vendors may not provide detailed networks/maps of airports, campuses (both corporate and university), large parking facilities, and shopping centers.

Hence, the purpose of this paper is to explore *map-matching algorithms* that can be used to reconcile inaccurate locational data with an inaccurate map/network. We begin in the next section with a formal definition of the problem. We then discuss point-to-point, point-to-curve and curve-to-curve matching. In all three cases we consider algorithms that only use geometric information and algorithms that also use topological information. Finally, we conclude with a discussion of possible future research directions.

PROBLEM STATEMENT

Our concern is with a person (or vehicle) moving along a finite system (or set) of streets, $\overline{\mathcal{N}}$. At a finite number of points in time, denoted by $\{0, 1, \ldots, T\}$, we are provided with an estimate of this person's location. The person's actual location at time t is denoted by $\overline{P}^t$ and the estimate is denoted by P^t. Our goal is to determine the street in $\overline{\mathcal{N}}$ that contains $\overline{P}^t$. That is, we want to determine the street that the person is on at time, t.

Of course, we do not know the street system, $\overline{\mathcal{N}}$, exactly. Instead, as illustrated in Figure 1, we have a *network representation*, $\mathcal{N}$, consisting of a set of curves in $\mathbb{R}^2$, each of which is called an *arc*. Each arc is assumed to be piecewise linear. Hence, arc $A \in \mathcal{N}$ can be completely characterized by a finite sequence of points $(A^0, A^1, \ldots, A^{n_a})$ (i.e., the endpoints of the individual line segments that comprise A), each of which is in $\mathbb{R}^2$. The points A^0 and A^{n_a} are referred to as *nodes* while $(A^1, A^2, \ldots, A^{n_a-1})$ are referred to as *shape points*. A nodes is a point at which an arc terminates/begins (e.g., corresponding to a dead-end in the street system) or a point at which it is possible to to move from one arc to another (e.g., corresponding to an intersection in the street system).

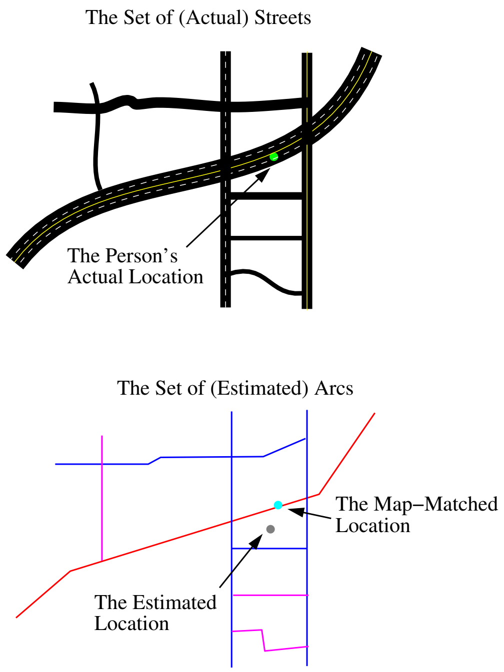


Figure 1: The Map-Matching Problem

This problem is called a *map matching problem* because the goal is to match the estimated location, P^t, with an arc, A in the "map", $\mathcal{N}$, and then determine the street, $\overline{A} \in \overline{\mathcal{N}}$, that corresponds to the person's actual location, $\overline{P}^t$. A secondary goal is to determine the position on A that best corresponds to $\overline{P}^t$.

In order to simplify the exposition, we assume that there is a one-to-one correspondence between the arcs in $\mathcal{N}$ and the streets in $\overline{\mathcal{N}}$. This assumption can easily be relaxed, however.

Not surprisingly, the problem considered here is similar to the map-matching problem in mobile robotics. There, the problem is to establish a correspondence between a current local map and a stored global map. For a review, see (19).

USING ONLY GEOMETRIC INFORMATION

We begin by considering methods of solving the map-matching problem that make use only of *geometric* information. That is, methods which make use only of the "shape" of the arcs and not the way in which they are "connected".

Geometric Point-to-Point Matching

One natural way to proceed is to match P^t to the "closest" node or shape point in the network. Of course, the question then arises of how to define "close" and the most natural way to proceed is to use the Euclidean metric (though other metrics can also be used).

In particular, recall that the *Euclidean distance* between two points x and y in R^2 is given by:

$$||x - y||_2 = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}. \tag{1}$$

In a point-to-point matching algorithm, one need only determine the distance between P^t and each point in the network sequentially, storing the closest point found along the way.

Of course, in practice, it is not necessary to determine the distance between P^t and every node and shape point in the network. Instead, one can first identify those node and shape points that are within a "reasonable" distance of P^t, and then calculate the distance only to "reasonable" points (e.g., those points within some multiple of the known accuracy of the GPS receiver being used). A number of data structures and algorithms exist for identifying all of the points "near" a given point (often called a *range query*). See, for example, (20 and 21).

While this approach is both reasonably easy to implement and fast, it has many problems in practice. Perhaps most importantly, it depends critically on the way in which shape points are used in the network. To see this, consider the example shown in Figure 2. Here, P^t is much closer to B^1 then it is to either A^0 or A^1, hence it will be matched to arc B even though it is intuitively clear that it should be matched to arc A. Hence, this kind of algorithm is very sensitive to the way in which the network was digitized. That is, other things being equal, arcs with more shape points are more likely to be matched to.

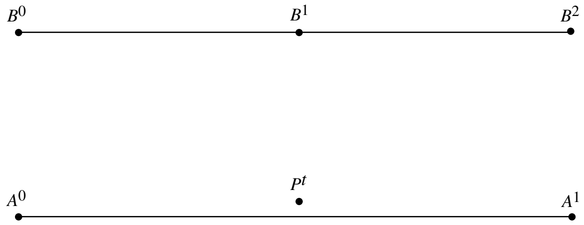


Figure 2: One Problem with Point-to-Point Matching

One might argue that this problem could be overcome simply by including more shape points for every arc. Unfortunately, this dramatically increases the size of the network and is not guaranteed to correct the problem.

Geometric Point-to-Curve Matching

Perhaps the next most natural way to proceed is to attempt to identify the arc in $\mathcal{N}$ that is "closest" to P^t, rather than the point that is closes to P^t. Again, we must ask how to define "close" and the most common approach is to use the minimum distance from the point to the curve. Since we are dealing with piecewise linear curves, to find the minimum distance from a point x to a curve A we must find the minimum distance from x to each of the line segments that comprise A and select the smallest.

Of course, it is fairly simple to find the minimum distance between a point and a line. In particular, suppose that we let $\{\lambda a + (1-\lambda)b, \lambda \in \mathbb{R}\}$ denote the line, A, through a and b. Then, the minimum distance between some point c and this line is given by:

$$d(c, A) = \sqrt{\frac{[(a_2 - b_2)c_1 + (b_1 - a_1)c_2 + (a_1 b_1 - b_1 a_2)]^2]}{(a_2 - b_2)^2 + (b_1 - a_1)^2}} \tag{2}$$

which is the distance along the "perpendicular" from c to the line.

As shown in Figure 3, calculating the minimum distance between a point and a line segment is slightly more complicated than calculating the minimum distance between a point and a line in some cases. Calculating the minimum distance between p and the line segment between A^0 and A^1 is straightforward since it is the same as the minimum distance between p and the line through A^0 and A^1. However, when we calculate the distance between q and the line through A^0 and A^1 we see that the "perpendicular" intersects the line outside of the line segment. Hence, we must also calculate the distance between q and both A^0 and A^1 and choose the smallest. Thus, calculating the minimum distance between a point, P^t, and an arc A, involves finding the minimum distance between P^t and the line segments $\{\lambda A^0 + (1 - \lambda A^1), \lambda \in [0,1]\}, \{\lambda A^1 + (1 - \lambda A^2), \lambda \in [0,1]\}, \ldots, \{\lambda A^{n_{A-1}} + (1 - \lambda A^{n_A}), \lambda \in [0,1]\}$ and choosing the smallest.

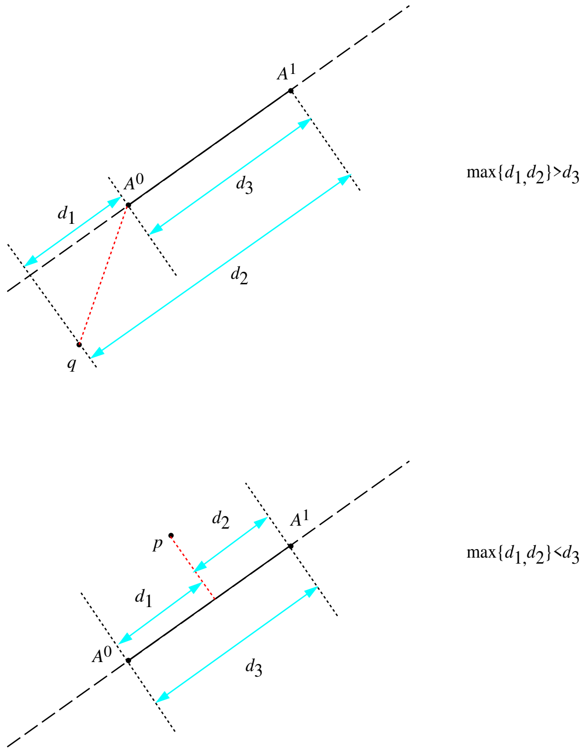


Figure 3: The Distance Between a Point and a Segment

To match P^t to an arc, one must calculate the minimum distance between P^t and all "reasonable" arcs in $\mathcal{N}$ and choose the one that is closest. While this approach may seem perfectly appropriate at first glance, it does have several shortcomings that make it inappropriate in practice.

First, point-to-curve matching does not make use of "historical" information. One problem that arises as a result is illustrated in Figure 4. The estimated position P^2 is equally close to arcs A and B. However, given P^0 and P^1 it seems clear that P^2 should be matched to arc A.

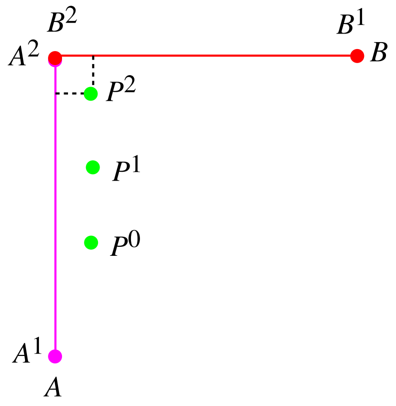


Figure 4: One Problem with Point-to-Curve Matching

Another problem with point-to-curve matching is that it can be quite "unstable". This is illustrated in Figure 5. The points P^0, P^1, and P^2 are all equidistant from arcs A and B. But, it turns out that P^0 and P^2 are slightly closer to A and P^1 is slightly closer to B. Hence, the matching oscillates back and forth between the two.

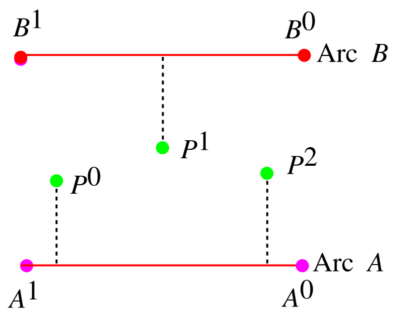


Figure 5: Another Problem with Point-to-Curve Matching

Geometric Curve-to-Curve Matching

A better way to proceed is to consider m positions simultaneously by matching to the arc that is "closest" to the piecewise linear curve, P, defined by the points $P^0, P^1, \ldots, P^m$. Of course, this requires that we have some measure of the distance between curves and there are many ways to define the distance between two curves in $\mathbb{R}^2$.

One definition of the distance between two curves A and B is:

$$||A - B||_{\text{min}} \min_{a \in A, b \in B} ||a - b||. \tag{3}$$

While this definition is quite appropriate in some circumstances, it does not work well for this type of map-matching as it is quite sensitive to outliers. This is illustrated in Figure 6 where, using this definition, the curve P is much "closer" to arc A than it is to arc B. (Obviously, $||A - B||_{\text{max}}$ has the same kinds of shortcomings.)

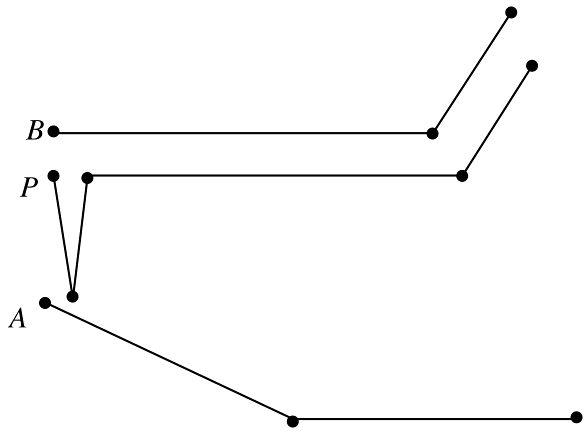


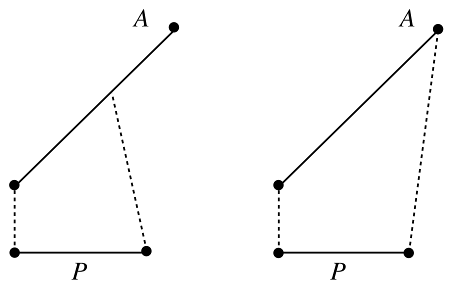
Figure 6: A Problem with One Metric for Curve-to-Curve Matching

Instead, it seems more appropriate to use some measure of the average distance between the curves. Such a measure can be defined relatively easily if we parametrize the curves that we need to compare. In particular, suppose the curve A, is parametrized by the function $a : [0, 1] \to A$. Then, one possible measure of the distance between two curves, A and B is:

$$||A - B||_2 = \int_0^1 ||a(t) - b(t)||dt. \tag{4}$$

The problem with this measure is that it is not a terribly good way to compare curves of different length.

If one has confidence in the length of P, it makes much more sense to measure the distance between P and equal length portions of the arcs under consideration. This is illustrated in Figure 7.



The Distance Between Equal Length Subsets of *P* and *A*

The Distance Between All of *P* and All of *A*

Figure 7: The Distance Between Curves of Different Length

It is also important to note that this measure of distance depends critically on the way the endpoints of the two curves are ordered. Hence, one must actually make the calculation both ways for each pair of arcs and choose the smaller of the two values.

Unfortunately, even when these issues are dealt with properly, this approach can still yield some unexpected and undesirable results, as shown in Figure 8. In this example, the positions $P^1 \ldots P^7$ have been recorded and there are two candidate arcs, A and B. The three curves can be parametrized as follows:

$$a(\alpha) = \begin{pmatrix} 6t \\ 6 \end{pmatrix}, \alpha \in [0,1] \tag{5}$$

$$b(\alpha) = \begin{pmatrix} 3 \\ 3t \end{pmatrix}, \alpha \in [0,1] \tag{6}$$

$$p(\alpha) = \begin{pmatrix} 6 \\ 3 \end{pmatrix}, \alpha \in [0,1]. \tag{7}$$

Hence, the distance between P and A is:

$$\begin{aligned}
\int_0^1 ||p(\alpha) - a(\alpha)||d\alpha &= \int_0^1 \sqrt{(p_1(\alpha) - a_1(\alpha))^2 + (p_2(\alpha) - a_2(\alpha))^2} d\alpha & (8)\\
&= \int_0^1 \sqrt{(6\alpha - 6\alpha)^2 + (3-6)^2} d\alpha & (9)\\
&= \int_0^1 3 d\alpha = 3 & (10)
\end{aligned}$$

and the distance between P and B is:

$$\begin{aligned}
\int_0^1 ||P(\alpha) - B(\alpha)||d\alpha &= \int_0^1 \sqrt{(p_1(\alpha) - b_1(\alpha))^2 + (p_2(\alpha) - b_2(\alpha))^2} d\alpha & (11)\\
&= \int_0^1 \sqrt{(6\alpha - 3)^2 + (3 - 6\alpha)^2} d\alpha & (12)\\
&= \int_0^1 \sqrt{2(36\alpha^2 - 36\alpha + 9)} d\alpha = \sqrt{4.5}. & (13)
\end{aligned}$$

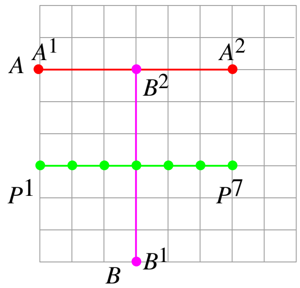


Figure 8: A Surprising Result

USING GEOMETRIC AND TOPOLOGICAL INFORMATION

In our experience, the performance of all of the algorithms discussed above can be improved if topological information is also used. In particular, given a known initial point, the topology of the network makes it possible to reduce the set of "likely" arcs dramatically.

We assume throughout the discussion that follows that the person using the PNA provides it with their origin and their destination at the beginning of their trip (and/or at other times). We further assume that the origin and destination are on the network, $\mathcal{N}$. For example, the user might provide the PNA with the addresses of their origin and destination. Obviously, the person's destination is required if the PNA is going to provide her/him with directions. Requiring the person to also enter their destination does not seem like much of an additional burden.

Improving Point-to-Curve Matching

Perhaps the easiest way to understand how topological information can be used is to consider the point to curve matching algorithm discussed above. In particular, consider the example shown in Figure 9.

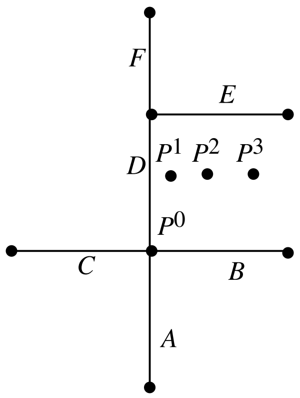


Figure 9: Using Topological Information

We know that the person was initially at P^0. Hence, we know that P^1 can only be on A, B, C, or D. In fact, given a sufficiently small amount of time between measurements, we might also know that P^3 can only be matched to A, B, C, or D. This kind of information could prevent us from mistakenly matching P^3 to, say E.

Improving Curve-to-Curve Matching

When topological information is used with curve-to-curve matching the results can be amazingly good. One such algorithm that we have tested calculates the distance between the curve P and equal length subsets of all of the arcs reachable from P^0. It then matches the position to the arc that is closes to P. The distances are calculated using a discrete approximation of equation (4). Though equation (4) can, in fact, be calculated exactly for the piecewise linear curves we are using here, the extra work does not seem warranted, especially given the frequency with which GPS positions can be obtained.

Perhaps the best way to explain the algorithm is with the simple example shown in Figure 10. In this example, the person started at P^0, the intersection of arcs A, B, C, and D (where $A^0 = B^0 = C^0 = D^0$). Then, three positions were recorded, P^1, P^2, and P^3. We want to determine the distances between the piecewise-linear curve formed by connecting $P^0 \ldots P^3$ and A, B, C, and D, respectively.

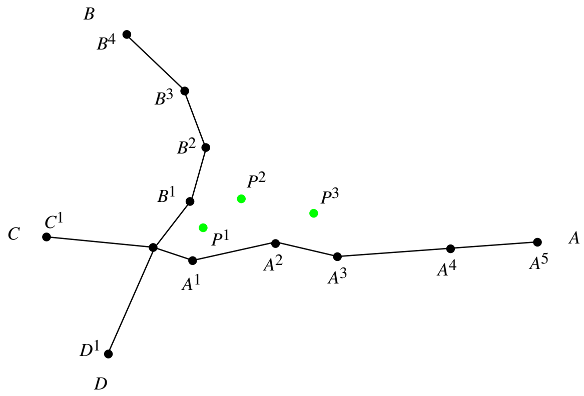


Figure 10: A Simple Example

The initial steps in the process are illustrated in Figure 11. First, a piecewise-linear curve is constructed by connecting P^0, P^1, P^2, and P^3. Then piecewise-linear curves of equal length are constructed along all of the arcs emanating from this same intersection. (For simplicity, we limit our attention only to arcs A and B.)

Next, as shown in Figure 12, each of the piecewise-linear curves is divided into s segments of equal length. (In this example, $s = 3$.) Finally, the relevant

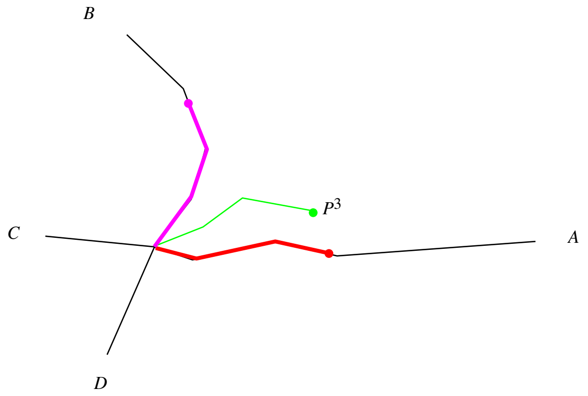


Figure 11: The Initial Steps of the Algorithm

distances are calculated. In this example, arc A is clearly the closest and P^3 is matched to it.

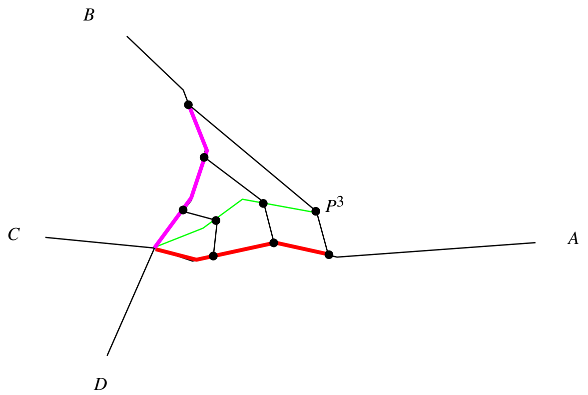


Figure 12: The Final Steps of the Algorithm

CONCLUSIONS AND FUTURE RESEARCH

In this paper we have described several algorithms (or parts of algorithms) for matching an estimated position to a network representation of the street system. We believe two things should be apparent from this discussion.

First, it should be clear that this is a fairly difficult task. Point-to-point and point-to-curve matching are unlikely to work very well, especially when there are errors in the position and/or errors in the network representation. Hence, other, more complicated algorithms must often be used.

Second, though a number of different algorithms can be used, it seems clear that it is both important to perform some kind of curve-to-curve matching and to incorporate topological information in the algorithm. Indeed, in our experience, the more attention given to the topological information, the better the algorithm performs.

Though our results thus far have been encouraging, it is clear that a considerable amount of research still needs to be done. We put this research falls into two categories.

On the one hand, attention needs to be given to how different algorithms can be compared empirically. This is a particularly thorny problem because it is quite difficult to measure the "true position" outside of a laboratory. In addition, it is not immediately clear what measures of performance are most appropriate or what scenarios should be evaluated. In an abstract sense it is clear that we would like the algorithm to perform perfectly when the errors go to zero, but it is not entirely clear what that means in practice.

On the other hand, more work needs to be done on the ways in which mistakes in map matching influence the overall performance of the PNA. In some situations, directions do not change much as a result of small errors in the map-matched location. In other cases, the directions change dramatically. Hence, work needs to be done both on more robust path-finding algorithms [e.g., providing alternative paths as in (22)] and on varying the map matching algorithm in different situations.

REFERENCES

1. Collier, W.C. "In-Vehicle Route Guidance Systems Using Map Matched Dead Reckoning", *Proceedings of IEEE Position Location and Navigation Symposium*, pp. 359-363, 1990.

2. Iwaki, F., M. Kakihari, M. Sasaki. "Recognition of Vehicle's Location for Navigation", *Proceedings of the Vehicle Navigation and Information Systems Conference*, pp. 131-138, 1989.

3. Hofmann-Wellenhoff, B., H. Lichtenegger, and J. Collins. *GPS: Theory and Practice*, Springer-Verlag, New York, 1994.

4. Shibata, M. "Updating of Digital Road Map", *Proceedings of the Vehicle Navigation and Information Systems Conference*, pp. 547-550, 1994.

5. Schiff, T.H. "Data Sources and Consolidation Methods for Creating, Improving and Maintaining Navigation Databases", *Proceedings of the Vehicle Navigation and Information Systems Conference*, pp. 3-7, 1993.

6. Deretsky, Z. and U. Rodny. "Automatic Conflation of Digital Maps: How to Handle Unmatched Data", *Proceedings of the Vehicle Navigation and Information Systems Conference*, pp. A27-A29, 1993.

7. Kim, J.-S., "Node Based Map Matching Algorithm for Car Navigation System", *Proceedings of the International Symposium on Automotive Technology and Automation* pp. 121-126, 1996.

8. Degawa, H. "A New Navigation System with Multiple Information Sources", *Proceedings of the Vehicle Navigation and Information Systems Conference*, pp. 143-149, 1992.

9. Mattos, P.G. "Integrated GPS and Dead Reckoning for Low-cost Vehicle Navigation and Tracking", *Proceedings of the Vehicle Navigation and Information Systems Conference*, pp. 569-574, 1994.

10. Blackwell, E.G. "Overview of Differential GPS Methods", *Global Positioning Systems*, The Institute of Navigation, Vol. 3, pp. 89-100, 1986.

11. 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 Symposium*, pp. 39-46, 1988.

12. Tanaka, J. "Navigation System with Map-Matching Method", *Proceedings of the SAE International Congress and Exposition*, pp. 45-50, 1990.

13. 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 Sciences*, Vol. E79-A, pp. 1853-1855, 1996.

14. Abousalem, M.A. and E.J. Krakiwsky. "A Quality Control Approach for GPS-based Automatic Vehicle Location and Navigation Systems" *Proceedings of the Vehicle Navigation and Information Systems Conference* pp. 466-470, 1993.

15. Watanabe, K., K. Kobayashi and F. Munekata. "Multiple Sensor Fusion for Navigation Systems", *Proceedings of the Vehicle Navigation and Information Systems Conference*, pp. 575-578, 1994.

16. 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 Conference*, pp. 585-590, 1994.

17. Tsakiri, M. "Integrated GPS and DR for Land Vehicle Navigation", *Proceedings of the Western Australia Land Information System Forum*, 1996.

18. Padmos, J. and D. Bernstein. "Personal Travel Assistants and the World Wide Web", *Transportation Research Record*, forthcoming.

19. Borenstein, J., H.R. Everett and L. Feng. *Navigating Mobile Robots: Systems and Techniques*, 1996.

20. Bentley, J.L. and H.A. Maurer. "Efficient Worst-Case Data Structures for Range Searching", *Acta Informatica*, Vol. 13, pp. 155-168. 1980.

21. Fuchs, H. Z.M. Kedem and B.F. Naylor. "On Visible Surface Generation by A Priori Tree Structures", *Computer Graphics*, Vol. 14, pp. 124-133, 1980.

22. Scott, K., G. Pabón-Jiménez, and D. Bernstein. "Finding Alternatives to the Best Path", presented at the *Annual Meeting of the Transportation Research Board*, Preprint 970682, 1997.