PERGAMON

Transportation Research Part C 8 (2000) 91–108

TRANSPORTATION RESEARCH PART C

www.elsevier.com/locate/trc

Some map matching algorithms for personal navigation assistants

Christopher E. White [a], David Bernstein [b,*], Alain L. Kornhauser [a]

[a] *Department of Operations Research and Financial Engineering, Princeton University, Princetown, NJ 08540, USA*
[b] *Department of Computer Science, James Madison University, MSC 143 Harrisonburg, VA 22807, USA*

Abstract

Third-generation personal navigation assistants (PNAs) (i.e., those that provide a map, the user's current location, and directions) must be able to reconcile the user's location with the underlying map. This process is known as *map matching*. Most existing research has focused on map matching when both the user's location and the map are known with a high degree of accuracy. However, there are many situations in which this is unlikely to be the case. Hence, this paper considers map matching algorithms that can be used to reconcile inaccurate locational data with an inaccurate map/network. © 2000 Published by Elsevier Science Ltd.

1. 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 from second- and third-generation systems. Clearly, a system that provides the user's current location is much more complicated than one that does not, 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).

* Corresponding author. Tel.: +1-540-568-1671; fax: +1-540-568-2745.
E-mail address: bernstdh@jmu.edu (D. Bernstein).

0968-090X/00/$ - see front matter © 2000 Published by Elsevier Science Ltd.
PII: S0968-090X(00)00026-7

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 (Collier, 1990). The second is to use some form of ground-based *beacon* that broadcasts its location to nearby users (Iwaki et al., 1989). 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 (Hofmann-Wellenhoff et al., 1994).

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 (Deretsky and Rodny, 1993; Schiff, 1993; Shibata, 1994). 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 (Degawa, 1992; Mattos, 1994; Kim, 1996), others use differential GPS (Blackwell, 1986), and others use multiple sources of data (sometimes including maps) and then filter or fuse the data in some way (Krakiwsky et al., 1988; Tanaka, 1990; Abousalem and Krakiwsky, 1993; Scott and Drane, 1994; Watanabe et al., 1994; Jo et al., 1996).

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. Firstly, not all PNAs are vehicle-based. Hence, it may not be possible to use DR or other data sources. Secondly, 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. Thirdly, 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 discuss some simple *map matching algorithms* that can be used to reconcile inaccurate locational data with an inaccurate map/network. We begin in the following 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. Then, we consider the performance of each of the algorithms in practice (in an admittedly limited number of tests). Finally, we conclude with a discussion of possible future research directions.

Our objective in this paper is not to provide a definitive evaluation of different map matching algorithms. Rather, our objective is to describe some simple algorithms and to consider, both theoretically and in a small number of tests, why they might or might not work well in practice.

2. 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, T, 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 Fig. 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 piece-wise 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 node 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 move from one arc to another (e.g., corresponding to an intersection in the street system).

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 $\overline{P}^t$.[1]

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 (and often does not hold in practice).

3. Map matching algorithms

There are a number of different ways to approach the map matching problem, each of which has advantages and disadvantages. We will briefly discuss several of them before moving on to a discussion of the specific algorithms that we considered.

[1] 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.

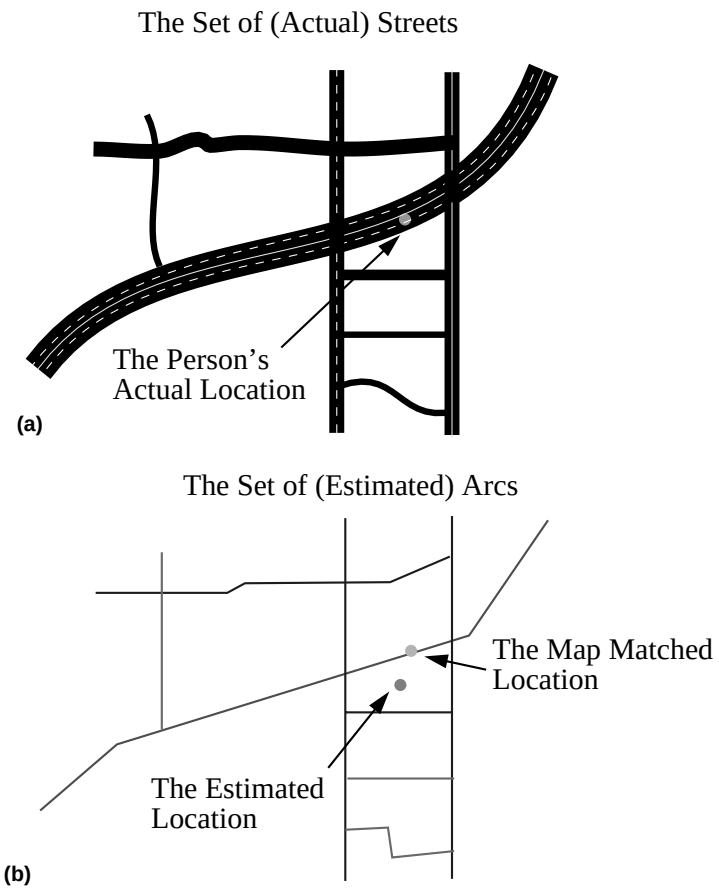

Fig. 1. The map matching problem.

3.1. Map matching as a search problem

One can view the map matching problem as a simple search problem. Then the problem is to match P^t to the "closest" node or shape point in the network.

A number of data structures and algorithms exist (see e.g., Bentley and Maurer, 1980; Fuchs et al., 1980) for identifying all of the points "near" a given point (often called a *range query*). It is then a simple matter to find the distance between P^t and every node and shape point that is within a "reasonable" distance of it (regardless of the metric used), and select the closest.

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 Fig. 2. Here, P^t is much closer to B^1 than 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.

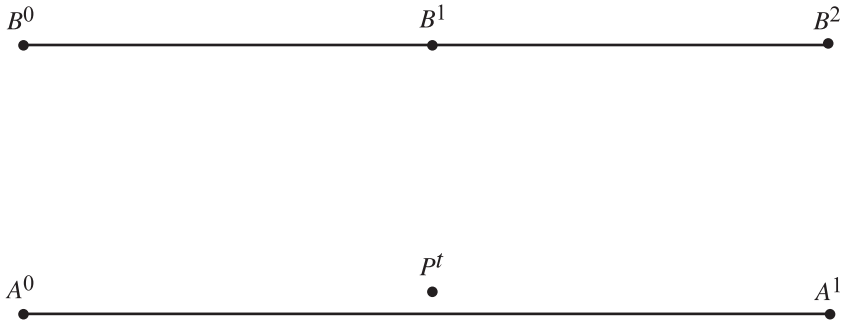

Fig. 2. One problem with point-to-point matching.

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.

3.2. Map matching as statistical estimation

One can also view map matching as a problem of statistical estimation. In this approach, one considers a sequence of points $(P^s, \ldots, P^t)$ and attempts to fit a curve to them. This curve is constrained to lie on the network.

This kind of approach has been explored in numerous papers (see e.g., Krakiwsky et al., 1988; Scott and Drane, 1994; Jo et al., 1996) and is quite appealing. It is particularly elegant when the model describing the "physics of motion" is simple (e.g., movement is only possible along a straight line). Unfortunately, in most practical applications, the physics of motion is dictated by (or constrained by) the network. This makes it quite difficult to model.

To understand why this is important, consider the network shown in Fig. 3. In this example, the positions $P^1 \ldots P^7$ have been recorded. Our objective is to fit a curve to these points, but the curve is constrained to lie on the network. In this case, there are two candidate curves, A and B (we ignore the rest of the network for simplicity).

In general (i.e., regardless of the metric), the curve P is closer to the curve B than it is to the curve A. Thus, if one uses a simple model of motion one will be led to match P to B rather than to A.

4. Algorithms used in this study

Our objective in this study was to combine the simplicity of the simple search approach with some of the ideas in the statistical approach. In the end, we implemented and tested four different algorithms.

Algorithm 1 is very simple. It finds nodes that are close to the GPS "tick" and finds the set of arcs that are incident to these nodes. It then finds the closest of these arcs and projects the point onto

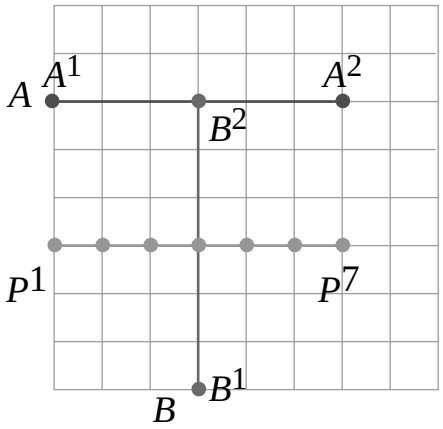

Fig. 3. Curve fitting.

that arc (using a minimum norm projection). As shown in Fig. 4, 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. 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 the line segment. Hence, we must also calculate the distance between q and both A^0 and A^1 and choose the smallest. Finally, since each arc is a piece-wise linear curve, we must find the minimum distance from the point of interest to each of the line segments that comprise A and select 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.

Obviously, this algorithm has many shortcomings. Firstly, it does not make use of "historical" information and this can cause problems of the kind illustrated in Fig. 5. 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.

Another problem with this algorithm is that it can be quite "unstable". This is illustrated in Fig. 6. 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.

Hence, Algorithm 1 will play the role of a "straw man". It is fast, easy to implement, and should be easy to beat.

Algorithm 2 is identical to Algorithm 1 except that it makes use of "heading" information.[2] If the heading of the PNA is not comparable to the heading of the arc, then the arc is discarded. So,

[2] As discussed below, this calculation is actually performed by the GPS receiver.

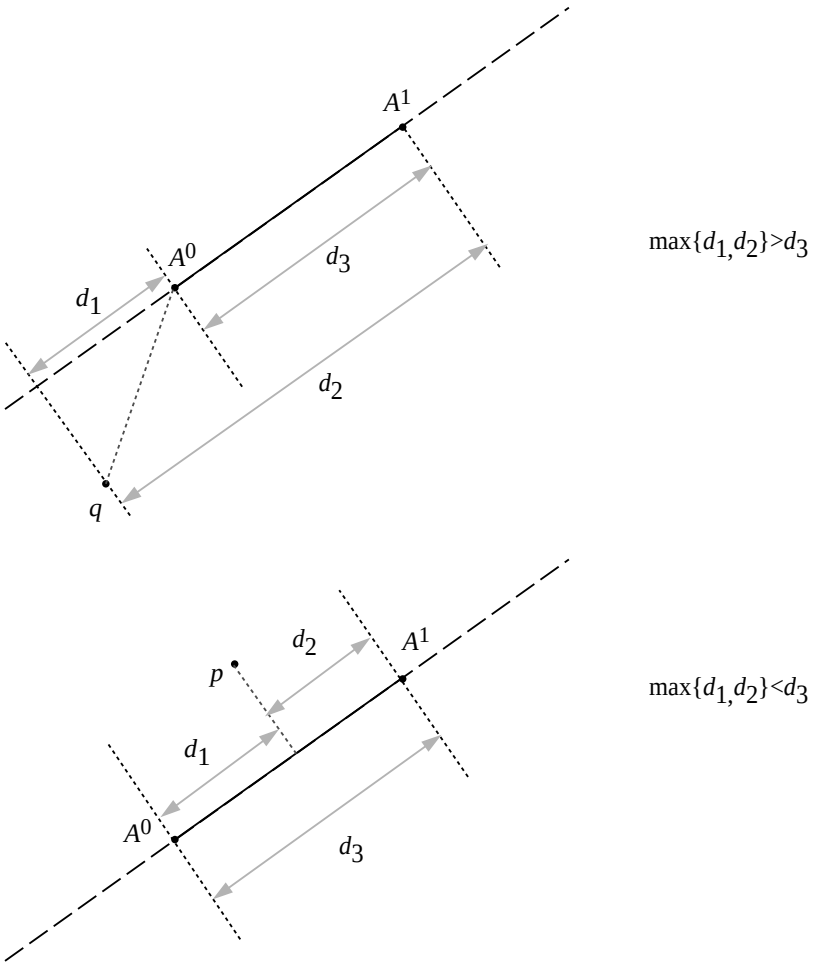

Fig. 4. The distance between a point and a segment.

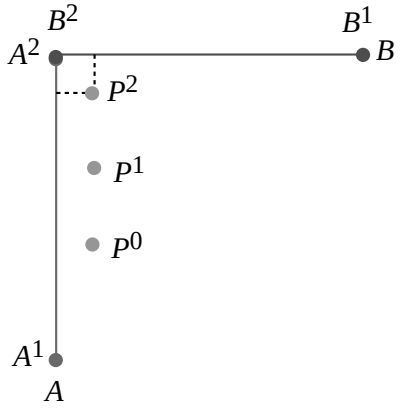

Fig. 5. One problem with point-to-curve matching.

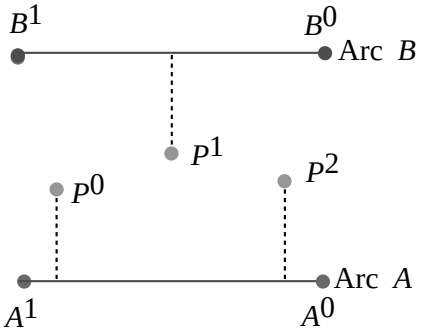

Fig. 6. Another problem with point-to-curve matching.

for example, a GPS tick will not be matched to an arc that is perpendicular to the current direction of travel (as occurred in the example in Fig. 3). Again, this algorithm is included primarily as a straw man.

Algorithm 3 is a variant of Algorithm 2 that uses topological information. In particular, whereas Algorithms 1 and 2 only use a range query to locate candidate nodes (and, hence, candidate arcs), Algorithm 3 also uses connectivity information. Specifically, if the algorithm has confidence in the previous match, it will use the topology of the network to locate candidate nodes for the next match. That is, it will only consider arcs that are reachable (in a topological sense) from the "current" arc. On the other hand, if the algorithm does not have confidence in the previous match, it will use a range query. A match is considered "good" if the error is less than the minimum of 0.15 km and twice the average error in the matches obtained thus far.

Perhaps the easiest way to understand how topological information can be used is to consider the example shown in Fig. 7. Suppose we know that the person was initially at P^0. We then know

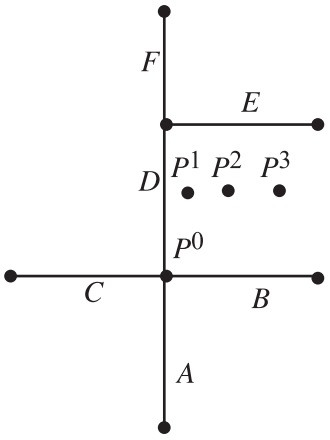

Fig. 7. Using topological information.

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.

This is, in our opinion, the easiest way to incorporate the physics of motion into a search-based algorithm. While it is not nearly as sophisticated as the filtering algorithms discussed earlier, it does have the potential to preclude many kinds of errors. Of course, it can be quite sensitive to the threshold that is used. That is, one bad match that you have confidence in can lead to a sequence of bad matches. On the other hand, if you make it difficult to call a match good, then it will behave much like Algorithm 2. The threshold we chose (i.e., 0.15 km) was based on an "expected" GPS error of 0.10 m.

Algorithm 4 uses curve-to-curve matching. Firstly, it locates candidate nodes using the same techniques as in Algorithm 3. Then, given a candidate node, it constructs piece-wise linear curves from the set of paths that originate from that node. Secondly, it constructs a piece-wise linear curve using the points $(P^s, \ldots, P^t)$ and calculates the distance between this curve and the curves corresponding to the network. Finally, it selects the closest curve and projects the point onto that curve.

The details of this algorithm are illustrated in Fig. 8. P^1 and P^2, represent the previous and current points, respectively. The point P^0 is a candidate node. The dotted line is the piece-wise linear curve, P, constructed from P^0, P^1, and P^2. The two segments of this piece-wise linear curve are 8 and 5 units in length, respectively.

There are many ways to calculate the distance between two curves. Of particular importance in this context is the way in which curves of different lengths are handled. As shown in Fig. 9, one can either calculate the distance between the two curves using their actual lengths, or calculate the distance between "subcurves" of equal length. We use the latter approach.

Specifically, returning to Fig. 8, to calculate the distance between the curve and A we use the two points a and b. a is the point that is 8 units along arc A starting from P^0 and b is the point that

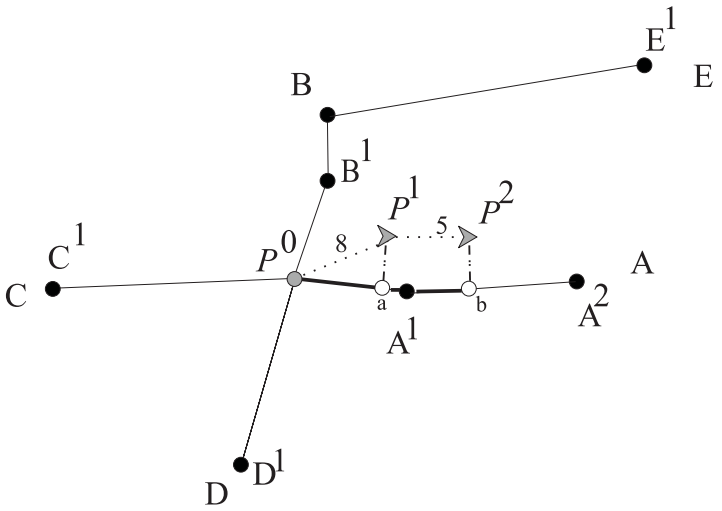

Fig. 8. Curve-to-curve distances.

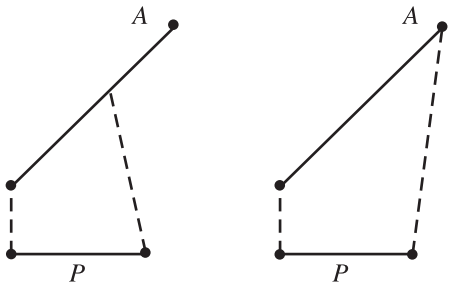

Fig. 9. The distance between curves of different length.

is of 5 units from a along arc A. The distance between P and A is then calculated as $\|P^1 - a\|_2 + \|P^2 - b\|_2$.

A similar process is used for all C and D, and for the other candidate nodes. P^2 is then projected onto the closest.

5. Performance of the algorithms

Unfortunately, there are no standard data sets that can be used to evaluate map matching algorithms. Indeed, very few evaluations have even been reported. This is in part because of the proprietary nature of much of this research and in part because of the difficulty of conducting this kind of evaluation. Hence, we constructed our own test bed of four routes.

Obviously, no conclusions can be drawn from this admittedly limited evaluation. We think the results are interesting nonetheless.

5.1. Data collection

The street network we used for this study was taken from the 1997 TIGER/Line files for Mercer County, New Jersey. We collected data while traveling on four pre-determined routes. All of the routes were "in-town". That is, they did not involve highways or arterials.

We determined our actual location by manually instructing the PNA to record the system time as we entered each link. We determined our estimated location using a GPS receiver that was manufactured by TravRoute using a chipset from Rockwell (12 channels, differential correction was turned off). We ran the receiver in NMEA-0183 mode and used the GPRMC sentences (which includes information about the time, latitude and longitude, speed, heading, and satellite count). We recorded one GPS tick per second, along with the system time when it was generated.

Note that, given this sampling scheme, the data will be spatially biased. Unfortunately, given current technology, it is not possible to obtain a spatially unbiased sample for two reasons. Firstly, the satellites themselves broadcast messages on a pre-determined (temporal) schedule. Secondly, the GPS receiver cannot know when it has traveled a given distance, and, even if it

Table 1
General route information

Route	No. of arcs	Avg. arc length (km)
1	12	0.1706
2	12	0.2249
3	14	0.1928
4	16	0.6083

could, it would not necessarily be able to determine its location at that time. Thus, an algorithm that cannot perform well in the presence of the spatial bias is not a good algorithm for our purposes.

Note also that the GPS receiver did not provide us with raw satellite information; we let the GPS receiver do as much processing as it could in the time available to it. In fact, the GPS ticks we used were actually "smoothed" by the receiver. Of course, this filtering does introduce error that we would prefer to avoid. Unfortunately, it is not clear how to avoid this kind of error. In particular, the system is almost always under-identified (i.e., fewer than four satellites are visible) or over-identified (i.e., more than four satellites are visible). We cannot ignore all such observations because, in practice, they occur frequently (indeed, sometimes for an entire trip). Since every method of handling the identification problem introduces error, we chose to use the filtering scheme that Rockwell built into their chip set.

Table 1 provides some information about the routes that were used. The four routes had between 12 and 16 arcs, most of which had a speed limit of 25 mph. The speed limit was never exceeded while the data were being collected, but it was frequently not realized because of other vehicles on the road. Routes 1, 2, and 3 have similar mean arc lengths. Route 4 has a considerably longer mean arc length because it included a few "long" links.

5.2. The evaluation metric

To evaluate the algorithms we calculated the percentage of correct matches on a given route. That is, each estimated location was matched to an arc in the network. The match was deemed to be correct if that arc was actually being traversed at that time and it was deemed to be incorrect otherwise. While this metric is clearly not ideal, it is difficult to develop a better alternative because it is very difficult to measure the PNA's "true" location at each point in time.[3]

5.3. Results

Unfortunately, space prevents us from presenting maps of all of the results. We will, instead, try to summarize them. Given the small number and limited variability of routes in the field test, we

[3] Of course, it is possible to record the PNA's true location on a closed test track. Unfortunately, the only test tracks available to us were small and had simple topologies. Almost all algorithms work well on test tracks.

have chosen not to present a statistical analysis of the data. Indeed, we caution against drawing any strong conclusions from these results.

Table 2 displays, for each of the four pre-planned routes used in the study, the percentage of correct matches attained by each algorithm.

Overall, the best algorithm only correctly matches between 66% and 86% of the GPS ticks, however, this is not as bad as it might sound. Recall that one GPS tick was generated each second. This means that relatively more ticks are generated near intersections (because speeds are always lower near intersections and are sometimes zero) and the map matching problem is much more difficult near intersections since several arcs are very close to each intersection. In addition, the GPS receiver itself performs much more poorly near intersections because the speed of the vehicle is lower. Hence, the error in the GPS ticks is much larger at intersections.

This is relatively easy to see on a map. Fig. 10 contains a portion of the arcs that compose Route 1. In the figure, there are two sets of arrow-heads. The lighter arrow-heads represent the GPS ticks and the darker arrow-heads represent the map matched locations that were produced by Algorithm 1. As you can see, most of the problems occur at intersections.

Table 2
Matching rates for route/algorithm pairs

Algorithm	Route 1	Route 2	Route 3	Route 4
1	0.534	0.677	0.618	0.608
2	**0.663**	**0.736**	0.855	0.681
3	0.661	0.707	**0.858**	0.664
4	0.617	0.726	0.771	**0.687**

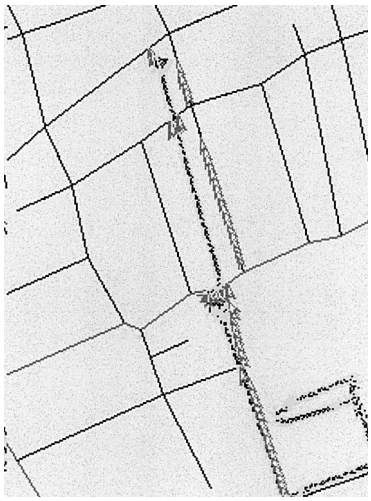

Fig. 10. Matching results of Algorithm 1 on a portion of Route 1.

As expected, Algorithm 1, has the worst performance. In contrast, Algorithm 2, performs reasonably well. Oddly, Algorithm 3 does not out-perform Algorithm 2. Referring to the figures, Algorithm 3 achieves the best performance on Route 3 (slightly outperforming Algorithm 2), the second best performance on Route 1 (slightly less than Algorithm 2), and the third best on Routes 2 and 4. The most complex algorithm, Algorithm 4, does not consistently out-perform either Algorithm 2 or Algorithm 3.

Part of the difference in performance is clearly due to performance at intersections. Fig. 11 displays the same "ticks" on the same portion of Route 1 as in Fig. 10 except that the darker arrow-heads in this figure were produced by Algorithm 2. As you can see, these two algorithms had comparable performance away from the intersections. However, because Algorithm 2 uses heading information, it performs much better near intersections, even though the heading information is sometimes inaccurate at lower speeds. This result holds in general.

We were, to say the least, disappointed by the relatively poor performance of Algorithms 3 and 4. In spite of our expectations, they did not out-perform Algorithm 2, which is considerably simpler.

One is next led to ask when the algorithms performed well and when they performed poorly. We considered six factors: the length of the matched arc, the speed of the PNA, the distance from

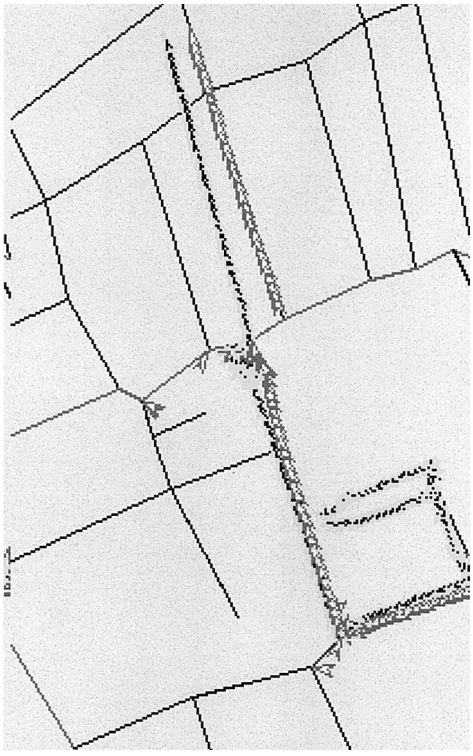

Fig. 11. Matching results of Algorithm 2 on a portion of Route 1.

Table 3
Detailed performance of Algorithm 1 (point-to-curve matching)

Algorithm 1						
Attribute	Match		Route 1	Route 2	Route 3	Route 4
Arc length (km)	Correct	Mean	0.246	0.290	0.268	1.12
		S.D.	0.074	0.089	0.072	1.12
	Incorrect	Mean	0.146	0.239	0.191	0.345
		S.D.	0.098	0.116	0.115	0.375
Speed (mph)	Correct	Mean	21.8	21.5	22.2	32.8
		S.D.	5.43	5.92	4.69	14.4
	Incorrect	Mean	15.8	17.8	15.2	32.4
		S.D.	6.16	7.12	6.16	38.1
Closest arc (km)	Correct	Mean	0.025	0.020	0.021	0.025
		S.D.	0.019	0.023	0.019	0.022
	Incorrect	Mean	0.020	0.023	0.020	0.079
		S.D.	0.013	0.022	0.015	0.096
Next closest arc (km)	Correct	Mean	0.065	0.061	0.069	0.144
		S.D.	0.027	0.032	0.032	0.087
	Incorrect	Mean	0.040	0.037	0.039	0.122
		S.D.	0.019	0.024	0.019	0.107
Serial correlation	Correct	Mean	10.3	18.6	12.8	16.7
		S.D.	2.13	2.02	1.92	4.33
	Incorrect	Mean	8.18	8.88	7.91	11.5
		S.D.	3.43	3.12	2.49	3.55
Closest intersection (km)	Correct	Mean	0.073	0.076	0.079	0.173
		S.D.	0.042	0.047	0.047	0.127
	Incorrect	Mean	0.040	0.035	0.038	0.063
		S.D.	0.022	0.025	0.024	0.060

the GPS tick to the matched arc, the distance from the GPS tick to the next best arc, the serial correlation in the GPS ticks, and the distance from the GPS tick to the nearest intersection. This information is summarized in Tables 3–6.

As you can see, all algorithms were more likely to produce an incorrect match when the PNA is traversing a relatively short road segment. This is almost certainly because the PNA is never far from an intersection.

As one might expect, all of the algorithms worked better with "better" GPS ticks (i.e., when the distance between the location approximation and the closest arc is relatively small). Note, however, that because of errors in the map database, this does not necessarily mean that a more accurate GPS receiver (e.g., a DGPS receiver) will yield better results.

Speed can also play a role and can be associated with arc length. Indeed, correct matches tend to occur at greater speeds than incorrect matches. This may simply be because the mean speed of travel is higher on longer arcs (and, hence, the GPS readings tend to be better). However, in the case of Algorithm 4, it may also be a direct result of the inclusion of topological information. Oddly, however, Algorithm 4 seems to perform badly at high speeds on Route 4.

Table 4
Detailed performance of Algorithm 2 (point-to-curve matching supplemented with heading information)

Algorithm 2						
Attribute	Match		Route 1	Route 2	Route 3	Route 4
Arc length (km)	Correct	Mean	0.223	0.284	0.246	0.098
		S.D.	0.084	0.094	0.097	1.043
	Incorrect	Mean	0.153	0.246	0.192	0.466
		S.D.	0.111	0.115	0.094	0.706
Speed (mph)	Correct	Mean	21.2	21.8	20.4	30.9
		S.D.	5.70	5.62	5.85	11.0
	Incorrect	Mean	14.7	16.0	14.3	36.5
		S.D.	5.79	7.08	6.21	43.6
Closest arc (km)	Correct	Mean	0.030	0.023	0.028	0.032
		S.D.	0.020	0.024	0.022	0.032
	Incorrect	Mean	0.048	0.042	0.050	0.175
		S.D.	0.073	0.061	0.072	0.197
Next closest arc (km)	Correct	Mean	0.073	0.066	0.072	0.186
		S.D.	0.031	0.035	0.028	0.143
	Incorrect	Mean	0.093	0.066	0.070	0.259
		S.D.	0.10	0.081	0.099	0.23
Serial correlation	Correct	Mean	10.7	12.5	16.3	20.0
		S.D.	1.91	2.86	3.52	3.90
	Incorrect	Mean	5.00	4.46	2.75	9.36
		S.D.	3.04	1.65	0.94	3.48
Closest intersection (km)	Correct	Mean	0.065	0.076	0.063	0.145
		S.D.	0.044	0.048	0.049	0.116
	Incorrect	Mean	0.054	0.047	0.038	0.094
		S.D.	0.076	0.064	0.072	0.147

6. 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 and attempted to evaluate four of them. Obviously, more work needs to be done.

Firstly, more 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 or test track. 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.

Secondly, a wider variety of algorithms need to be evaluated. Particular attention needs to be given to the problems that arise at intersections. While there are many things that can be tried (including turning the algorithm off when speeds are very low), it is not immediately obvious what approach will work best. In addition, one can argue that intersections are the most important portion of the network since most route changes occur at intersections. We are currently in the

Table 5
Detailed performance of Algorithm 3 (point-to-curve matching supplemented with heading information and connectivity)

Algorithm 3						
Attribute	Match		Route 1	Route 2	Route 3	Route 4
Arc length (km)	Correct	Mean	0.219	0.279	0.250	0.942
		S.D.	0.082	0.092	0.098	1.02
	Incorrect	Mean	0.163	0.267	0.177	0.569
		S.D.	0.117	0.117	0.073	0.846
Speed (mph)	Correct	Mean	21.2	21.3	20.1	31.1
		S.D.	5.78	5.84	5.97	11.1
	Incorrect	Mean	15.6	18.4	16.6	35.7
		S.D.	5.78	7.60	7.00	42.6
Closest arc (km)	Correct	Mean	0.029	0.019	0.029	0.033
		S.D.	0.020	0.021	0.023	0.033
	Incorrect	Mean	0.035	0.058	0.051	0.158
		S.D.	0.032	0.062	0.066	0.178
Next closest arc (km)	Correct	Mean	0.106	0.116	0.119	0.639
		S.D.	0.059	0.084	0.059	0.633
	Incorrect	Mean	0.079	0.105	0.098	0.258
		S.D.	0.053	0.104	0.105	0.217
Serial correlation	Correct	Mean	10.3	16.9	19.3	22.8
		S.D.	1.81	2.69	3.35	3.33
	Incorrect	Mean	4.85	7.00	3.20	12.5
		S.D.	3.01	2.75	2.46	3.45
Closest intersection (km)	Correct	Mean	0.061	0.071	0.059	0.288
		S.D.	0.04	0.046	0.045	0.385
	Incorrect	Mean	0.033	0.044	0.019	0.084
		S.D.	0.039	0.059	0.056	0.140

process of developing and testing both statistical algorithms and search-based algorithms that make use of additional information (e.g., speeds and speed limits, preferences for turns of different types, and road classification).

Thirdly, we do not yet know whether these results will hold for other filtering schemes. It may be that, in practice, different map matching algorithms will need to be used with different kinds of filtering schemes (and, hence, hardware). This requires further study.

Fourthly, the algorithms need to be evaluated on a wider array of routes. Our discussion here focused on in-town routes rather than highway routes because most algorithms work well on highways. However, more in-town routes should be considered. In addition, it is important to consider truly urban routes because of the problems that arise in ''urban canyons''. It is also important to consider the impact of weather conditions, season (i.e., the amount of foliage that blocks the view of the sky), time of day, and other factors. The limited number and variety of routes considered here obviously prevent us from drawing any real conclusions.

Finally, 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

Table 6
Detailed performance of Algorithm 4 (curve-to-curve matching)

Algorithm 4						
Attribute	Match		Route 1	Route 2	Route 3	Route 4
Arc length (km)	Correct	Mean	0.233	0.288	0.250	1.04
		S.D.	0.081	0.087	0.094	1.08
	Incorrect	Mean	0.147	0.238	0.199	0.347
		S.D.	0.106	0.124	0.104	0.416
Speed (mph)	Correct	Mean	21.4	21.2	21.0	31.5
		S.D.	5.30	6.12	5.64	10.7
	Incorrect	Mean	16.0	19.2	15.8	36.0
		S.D.	6.19	6.51	5.78	44.3
Closest arc (km)	Correct	Mean	0.03	0.021	0.028	0.031
		S.D.	0.022	0.022	0.022	0.031
	Incorrect	Mean	0.035	0.037	0.051	0.166
		S.D.	0.028	0.027	0.058	0.153
Next closest arc (km)	Correct	Mean	0.00	0.00	0.00	0.00
		S.D.	0.00	0.00	0.00	0.00
	Incorrect	Mean	0.00	0.00	0.00	0.00
		S.D.	0.00	0.00	0.00	0.00
Serial correlation	Correct	Mean	9.67	15.6	10.8	21.5
		S.D.	2.14	3.17	2.58	4.48
	Incorrect	Mean	5.54	5.9	3.19	9.77
		S.D.	2.78	2.25	1.10	4.00
Closest intersection (km)	Correct	Mean	0.067	0.076	0.065	0.289
		S.D.	0.045	0.048	0.046	0.379
	Incorrect	Mean	0.027	0.027	0.036	0.125
		S.D.	0.037	0.03	0.063	0.153

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 and on varying the map matching algorithm in different situations.

Acknowledgements

This research was sponsored by the NJ Center for Transportation Information and Decision Engineering (www.njtide.org) and the NJ Commission on Science and Technology. The authors would like to thank the anonymous referees for helpful comments on an earlier draft of this paper.

References

Abousalem, M.A., Krakiwsky, E.J., 1993. A quality control approach for GPS-based automatic vehicle location and navigation systems. In: Proceedings of the Vehicle Navigation and Information Systems Conference, pp. 466–470.

Bentley, J.L., Maurer, H.A., 1980. Efficient worst-case data structures for range searching. Acta Inf. 13, 155–168.

Blackwell, E.G., 1986. Overview of differential GPS methods. Global Positioning Sys. 3, 89–100.

Collier, W.C., 1990. In-vehicle route guidance systems using map matched dead reckoning. In: Proceedings of IEEE Position Location and Navigation Symposium, pp. 359–363.

Degawa, H., 1992. A new navigation system with multiple information sources. In: Proceedings of the Vehicle Navigation and Information Systems Conference, pp. 143–149.

Deretsky, Z., U. Rodny, 1993. Automatic conflation of digital maps: how to handle unmatched data. In: Proceedings of the Vehicle Navigation and Information Systems Conference, pp. A27–A29.

Fuchs, H., Kedem, Z.M., Naylor, B.F., 1980. On visible surface generation by a priori tree structures. Comput. Graphics 14, 124–133.

Hofmann-Wellenhoff, B., Lichtenegger, H., Collins, J., 1994. GPS: Theory and Practice. Springer, Berlin.

Iwaki, F., Kakihari, M., Sasaki, M., 1989. Recognition of Vehicle's Location for Navigation. In: Proceedings of the Vehicle Navigation and Information Systems Conference, pp. 131–138.

Jo, T., Haseyamai, M., Kitajima, H., 1996. A map matching method with the innovation of the Kalman filtering. IEICE Trans. Fund. Electron. Comm. Comput. Sci. E79-A, 1853–1855.

Kim, J.-S., 1996. Node based map matching algorithm for car navigation system. In: Proceedings of the International Symposium on Automotive Technology and Automation, pp. 121–126.

Krakiwsky, E.J., Harris, C.B., Wong, R.V.C., 1988. A Kalman filter for integrating dead reckoning, map matching and GPS positioning. In: Proceedings of IEEE Position Location and Navigation Symposium, pp. 39–46.

Mattos, P.G., 1994. Integrated GPS and dead reckoning for low-cost vehicle navigation and tracking. In: Proceedings of the Vehicle Navigation and Information Systems Conference, pp. 569–574.

Scott, C.A., Drane, C.R., 1994. Increased accuracy of motor vehicle position estimation by utilizing map data, vehicle dynamics and other information sources. In: Proceedings of the Vehicle Navigation and Information Systems Conference, pp. 585–590.

Schiff, T.H., 1993. Data sources and consolidation methods for creating, improving and maintaining navigation databases, In: Proceedings of the Vehicle Navigation and Information Systems Conference, pp. 3–7.

Shibata, M., 1994. Updating of digital road map. In: Proceedings of the Vehicle Navigation and Information Systems Conference, pp. 547–550.

Tanaka, J., 1990. Navigation system with map-matching method. In: Proceedings of the SAE International Congress and Exposition, pp. 45–50.

Watanabe, K., Kobayashi, K., Munekata, F., 1994, Multiple sensor fusion for navigation systems. In: Proceedings of the Vehicle Navigation and Information Systems Conference, pp. 575–578.