Efficient map-matching of large GPS data sets - Tests on a speed monitoring experiment in Zurich

F. Marchal* J. Hackney K.W. Axhausen†

26th July 2004

Abstract

The logging of GPS data is quickly becoming an important source of data for travel behavior researchers. Post-processing these data requires to match the GPS data stream to the coded map of the transportation network. The output of the map-matching process is the identification of the routes that were actually taken. This paper presents an innovative map-matching algorithm that relies only on the GPS coordinates and on the network topology. Examples are provided on a large data set for the Zurich area. The paper demonstrates the efficiency of the algorithm in term of accuracy and computational speed.

Preferred citation: F. Marchal, J. Hackney and K.W. Axhausen (2004). Efficient map-matching of large GPS data sets - Tests on a speed monitoring experiment in Zurich. *Arbeitsbericht Verkehrs- und Raumplanung*, Institut für Verkehrsplanung und Transportsysteme, ETH Zürich, Zürich

*marchal@inf.ethz.ch, CoLab, ETH Zurich

†{hackney,axhausen}@ivt.baug.ethz.ch, IVT, ETH Zurich

1 Introduction

The usage of Global Positioning Systems (GPS) is getting increasingly popular among travel behavior researchers to collect reliable data [1, 3, 5, 9]. More specifically, GPS data loggers allow to record accurately the spatial displacements of travelers and to improve activity-based analysis by enhancing classical travel diaries. Off-the shelf devices available nowadays can record the position of a traveler every second and for extended periods that span several hours or even days. The computing power of GPS loggers only allows for a limited amount of real-time processing. The raw data have to be downloaded and post-processed off-line at the end of the measurement. Given their relatively low cost of set-up, large volumes of data can be generated for a single study. In a companion paper [2], we describe how 2.5 millions of points were collected in a three week experiment to monitor driving speeds in the Zurich area. The purpose was to measure network performances with floating cars. Larger experiments led in Sweden and in Denmark [10, 4] recently collected respectively about $250,000$ trips and $500,000$ trips during record periods from six months to two years. The size of the data available in these studies amounts to a billion of GPS points. An important step of the post-processing of the data consists in matching these points to the coded map of the transportation network to identify the routes which were taken. Therefore, efficient map-matching algorithms are required to handle such large data volumes in reasonable computation times. Intuitive methods such as nearest-node search or nearest-link search are quite fast but do not insure the consistency of the whole route since they ignore the correlation between subsequent points. This paper describes first an algorithm that achieves high performances. The algorithm takes into account the network topology to overcome the route consistency problem using the multiple hypothesis technique. Second, tests are presented on the experiment in Zurich.

2 Problem statement

The goal is to identity the route taken by a traveler equipped with a GPS data logger. Assuming that the traveler is moving along a transportation network, we can restrict ourselves to a network representation of the spatial environment. The network is coded as a set of nodes connected by directional links. A path is a set of connected links. The *map-matching problem* consists in finding the path that is the best estimation of the route that was actually taken by the user. There are two sources of errors. First, the GPS data loggers have their own limitations depending on the environment (e.g. canyon streets, tree canopies, etc.) and their accuracy. Second, the coded network is but an approximation of the physical world: it can be sparse because roads are missing and the description of the roads themselves (i.e. curvature and location) has a limited resolution.

Most of the existing map-matching algorithms and their limitations have been reviewed and described in recent papers [8, 7]. It has been recurrently noted in these papers that efficient map-matching should integrate the information about the network topology. The multiple hypothesis technique (MHT) allows to keep track of several positions or paths at once and to select eventually which candidate is the best. MHT has been proposed by [6] for on-line map-matching with embarked GPS logger and Dead Reckoning (DR) device. However, these previous works were more focused on the accuracy than on the computational speed of the algorithms. The aforementioned paper [4] recognizes that the algorithms developed so far are not tractable for map-matching large data sets. Therefore, our concern is slightly different from the existing literature. First, we assume in this paper that the only source

of data concerning the traveler is a stream of 2D coordinates collected by a GPS logger embarked in a car. In particular, we do not use a DR device as in [7, 6] and we do not use information about the heading nor the speed of the vehicle. Second, the measurement of the overall error as the distance between the GPS points and the coded network is not transferable from one study to another, since it highly depends on the resolution of the coded network. Therefore, the comparison of the algorithms can only be performed on the same data sets. Ideally, the performance of the algorithms should be measured as the ratio of routes that were correctly matched, which in turn would require a tedious manual checking (e.g. using street names). For these reasons, we focus only on the *operational* performance of the algorithms, that is how much faster that real-time they can process large volumes of data with reasonable matching errors (i.e. the routes are checked visually on a map). Note that most embarked GPS navigation systems have map-matching abilities. The algorithms are often proprietary and have different purposes and constraints than we do. The processing power of these devices is limited but they do not have to perform faster than real-time. As is, they are inadequate to our problem.

To summarize, we have 1) a standard network representation of the transportation system with nodes, directed links and "polylines" describing their curvature and 2) a stream of GPS coordinates recorded for every second. The problem is to find the path in the network that is the closest to the GPS points and in a minimum amount of computations.

3 Proposed algorithm

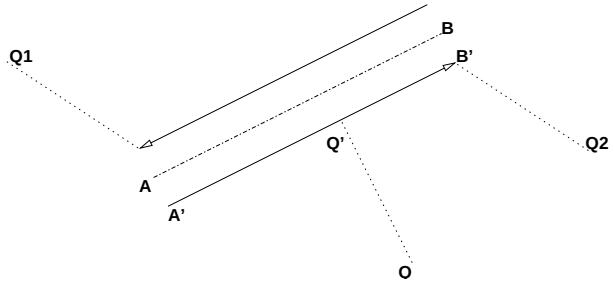
3.1 Distance between a point and a road segment

Let $G(V, E)$ be the directed graph describing the road network. V is the set of vertexes or nodes and E is the set of edges or links. Let Q_i be the set of points given by the GPS data stream $i = 1..T$. Each GPS point consists of a pair of coordinates (x_i, y_i) and a time-stamp t_i. To measure the distance between a GPS point Q and an oriented link AB, we use the distance as introduced in [8]. Let Q' be the projection of Q on line AB. The distance is defined as follows:

$$d(Q, AB) = \begin{cases} d_e(Q, Q') & \text{if } Q' \in [AB] \\ \min\{d_e(Q, A), d_e(Q, B)\} & \text{elsewhere} \end{cases},$$

where d_e denotes the euclidean distance. With this definition, $d_{Q,AB} = d_{Q,BA}$ and a point is equally distant from the two opposite directed links of a given road segment. For this reason, a small shift perpendicular to the link direction is introduced, according to the driving side of the studied area. Therefore, in the computation of the distance, an oriented link AB is replaced by $A'B'$ where $A' = A + \lambda 1_\perp$ and $B' = B + \lambda 1_\perp$ and $1_\perp \perp AB$, λ is chosen to reflect the average physical distance between the middle of the road and the middle of the driving lanes (in our application, $\lambda = 3$m). The computation of the distance between a point and a road segment is illustrated on Figure 1 where Q, Q_1 and Q_2 are equidistant to AB.

Figure 1: Distance between a point and a road segment



3.2 Score of a path

Let $P\{E_1, E_2, \ldots, E_p\}$ denote the path composed of the p subsequent links $E_1, E_2, \ldots, E_p$. The absolute score F of path P in matching the sequence Q_i is defined as

$$F = \sum_{j=1}^{p} \sum_{i=1}^{T} d\left(Q_i, E_j\right) \delta_{ij} \,, \tag{1}$$

where $\delta_{ij} = 1$ if Q_i was matched by link E_j and $\delta_{ij} = 0$ otherwise. The problem is to find the path P^*that minimizes F. The relative score F_r is defined as $F_r = \frac{F}{T}$. F_r will be used as a surrogate for the estimation of the error of the algorithm. Note that F_r is highly dependent on the resolution of the coded network. Each point is matched by only one link of the path, so that

$$\sum_{j=1}^{p} \sum_{i=1}^{T} \delta_{ij} = T \,.$$

3.3 Initialization

The algorithm is initialized by finding the set S of the N nearest links from the first GPS point Q_1. Note that we could use in practice a few more GPS points so that the initialization is more robust and less dependent on potential initial error in the measurement. For each link in the set, we create a one-link path with the initial link. The path $P_1, P_2, \ldots, P_N$ are ranked according to their scores: $F_1 \leq F_2 \leq \ldots \leq F_n$. Technically, they are stored in a sorted set. Alphabetical rank is used to discriminate paths that are different but that have the same score. Searching for the nearest link can be accelerated using a quad-tree storage for the links. Experiments done on a network with $400,000$ links showed performances of about 1 million of points per minute on a PC clocked at 2Ghz.

3.4 Breaking routes into paths

In most cases, a whole route will not be matched by a single continuous path because of interruptions in the GPS data stream. These interruptions can be due to tunnels, tree canopy, poor signal, etc. Therefore, a route will often be matched by a sequence of paths. We do not address here the problem of connecting these paths together. The outer loop of the algorithm processes the GPS points sequentially. Assume Q_i is the next data point to be processed. If the GPS device is searching for the signal,

the point is discarded. If $d_e(Q_{i-1}, Q_i)$ is bigger than a given threshold (e.g. 300 meters) or if $t_{i-1} - t_i$ is longer than a given period (e.g. 30 seconds), the matching is stopped for the current path and a fresh initialization is started with Q_i. Obviously, this procedure could be improved to be less sensitive on outliers.

3.5 Following the network topology

Given a new point Q_i, the following procedure is applied for each of the path P in the set S.

1. Assume that Q_i is matched by the last link E_p, that is $\delta_{ip} = 1$.
2. Update the score of path P using (1).
3. Insert P in a new sorted set V.
4. Check if the route has reached the next intersection (i.e. the destination of E_p).
5. If yes, create the new paths P_k^{FS} that correspond to the forward star of E_p.
6. Update the score of paths P_k^{FS} using (1).
7. Insert P_k^{FS} in the new sorted set V.

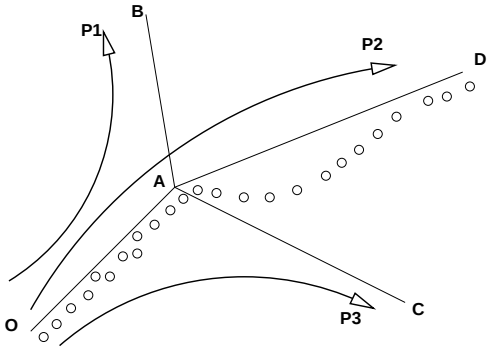
When the condition is met on step 4, new paths are created depending on the network topology. However, it is not obvious to determine if the vehicle has reached the end of the last link. We use the bird distance covered by the GPS points since the beginning of E_p to determine if the vehicle might have cruised the whole link. Let Q_{p_0} the first point matched by E_p and $L(E_p)$ the length of E_p. If the following condition holds,

$$\sum_{k=p_0}^{i-1} d_e(Q_k, Q_{k+1}) > \alpha L(E_p) \tag{2}$$

then we assume that the vehicle has reached the destination of E_p. A problem arises if a vehicle perform a u-turn in the middle of a long link. The parameter $\alpha = 0.5$ is introduced for that reason.

Let $FS(E_p)$ be the set of downstream links from E_p (forward star). If the condition (2) holds, a new path P_k^{FS} is created for each of the link in $FS(E_p)$. These children paths differ from their ancestor since they have one more link: $E_{p+1}^k \in FS(E_p)$ where E_{p+1}^k is the last link of path P_k^{FS}. The children paths are initialized with the last point so that $\delta_{ip+1} = 1$. The path creation mechanism is represented on Figure 2 where a vehicle is driving from O to D. At the intersection A, three children paths are created (P_1, P_2, P_3). Clearly the first points close to A do not allow to discriminate which direction has been taken. Indeed, the first points suggests AC was the chosen direction. Obviously, the direction was AD but the network description was not accurate enough to describe the curvature of the road section. Initially, the score of P_3 is better than that of P_2. However, as more points are added, the score of P_2 will dominate P_3 and P_1 in the end.

Figure 2: Path creation at intersections



3.6 Maintaining a set of candidate paths

Children paths are inserted into the sorted set V and ranked according to their score. Consequently, V has more elements than S. When Q_i is processed, S is emptied and only the N best elements of V are inserted into S. This avoids that the sorted set S grows indefinitely. In practice, the branching and the path creation can happen for a range of different Q_i as soon as the condition (2) is met. The range depends on α. Therefore, there are generally several instances of paths P_1, P_2, P_3 in the S, depending on the point Q_i when the branching occurred. Eventually, the less efficient paths are discarded. The sorted set S has to be large enough to keep track of unlikely trajectories that might turn to be correct ultimately as illustrated on the previous example. Intuitively, the number of candidate paths to maintain is a function of the resolution of the coded network.

When the last point Q_i is processed, the best path from S is kept as the result of the map-matching process. Similarly, when the route is broken and a new path has to be started, only the best path from S is kept as the best matching part for the previous GPS points. The flow-chart of the whole algorithm is presented on Figure 3. It depends on two parameters: α and the size of the set of candidates N.

4 Results

Three vehicles were equipped with GPS data loggers to monitor speed in the Zurich area. The drivers were given pre-defined routes supposed to cover most of the major roads and drove for three weeks. The routes include motorway sections as well as urban streets. About 2.5 millions of GPS data points were collected. Two coded networks were available for the map-matching process: a low resolution network ($4.5k$ nodes and $11k$ links) used by the local transport agency for road planning and a high resolution network ($166k$ nodes and $411k$ links with polylines) bought from Navtech. The latter contains about every single street of the studied area. The algorithm was coded in Java and run on a PC equipped with an Intel Pentium 4 processor clocked at 2.5Ghz.

4.1 Computational speed

A first experiment is performed on a single vehicle tracking which accounts for about 10% of the data ($231k$ GPS points). The algorithm decomposed the whole journey

Figure 3: Flow-chart of the map-matching algorithm

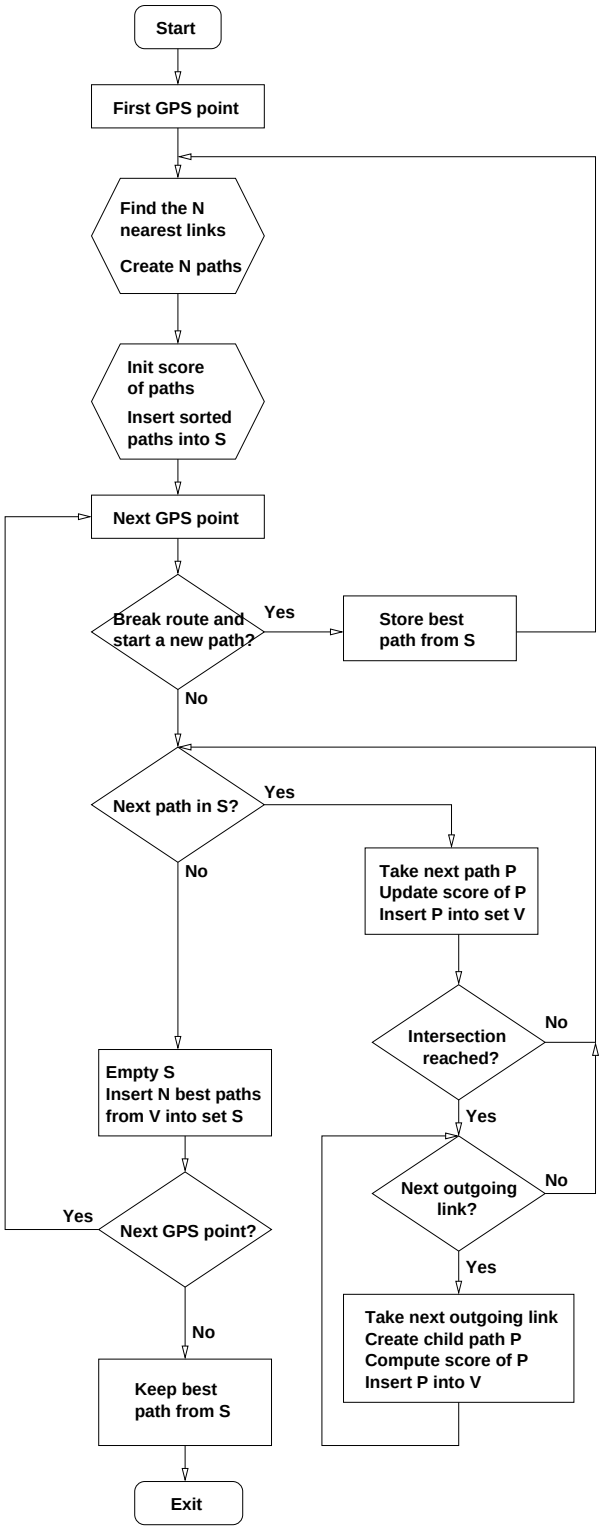


Table 1: Performance of the algorithm for the high (H) and low (L) resolution networks.. The first accuracy is the score for the overall route. The second accuracy corresponds to the number of points whose distance is bigger than twice the maximum accuracy achievable on the corresponding network (resp. 10m/pt and 50m/pt). The real-time ratio (RTR) is computed as the CPU time divided by the acquisition time (1pt/s).

N	Accuracy [m/pt]		Accuracy [%]		CPU time [s]		RTR [k]	
	H	L	H	L	H	L	H	L
10	435.8	56.8	55.2	8.5	56	12	4.2	19.5
15	15.9	53.2	18.2	8.5	58	14	4.0	16.8
20	13.1	50.0	10.9	4.4	62	18	3.8	13.0
30	11.3	49.8	4.5	4.4	70	21	3.4	11.2
40	10.6	49.8	3.6	4.4	78	31	3.0	7.6
50	10.5	49.8	3.3	4.4	85	37	2.8	6.3
100	10.3	49.8	3.3	4.4	122	72	1.9	3.2

in 84 paths. Results reported in Table 1 show that $N = 30$ yields reasonable results in short computing times in both cases. Increasing N further improves the quality of the matching but marginally. The computation time is roughly proportional to $\log(N)$. This is consistent with the time required to sort the candidate paths in the sets S and V. Moreover, the algorithm scales very well with the size of the network. The computational performances are such that it takes approximately one second of CPU time to process a one hour journey even with the high resolution network. Therefore, the algorithm is suitable for processing entire databases consisting of thousands of trips in acceptable wall clock times.

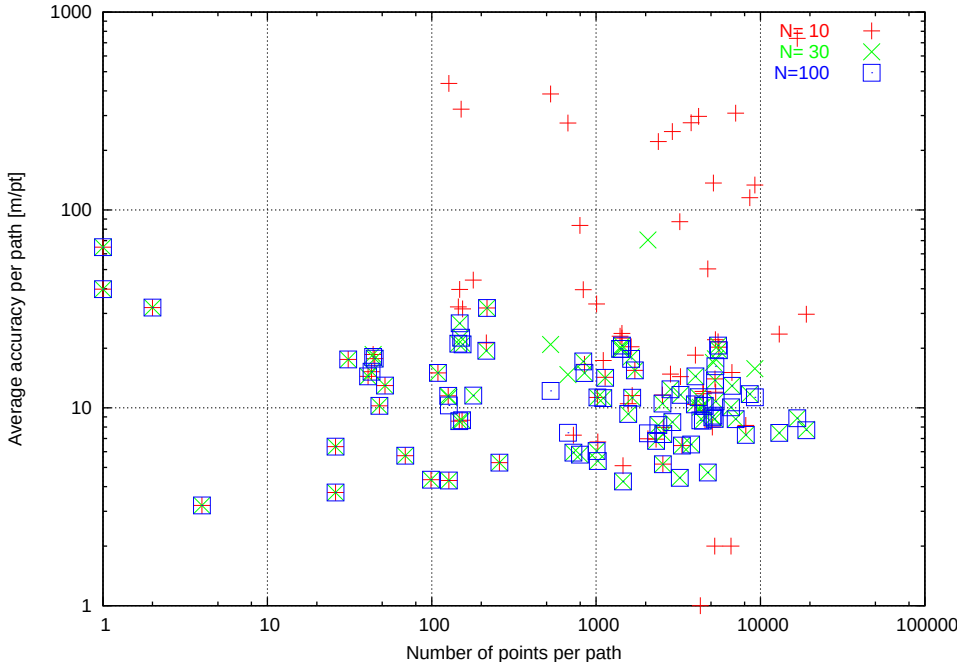
4.2 Accuracy

The maximum accuracy that can be achieved for the high resolution network is about 10 m/pt. This is consistent with the resolution of the GPS device itself. The accuracy of the 84 individual paths is plotted as a function of the number of points matched per paths on Figure 4 for different values of N for the high resolution network. First, the distributions are quite similar for $N = 30$ and $N = 100$. Second, for $N \geq 30$, outliers with poor accuracy are mostly short paths with less than two hundred points (e.g. less than three minutes). Consequently, using $N \geq 30$ seems adequate for a speed monitoring experiment or for the analysis of route choice.

4.3 Resolution

The map-matching is compared for the two networks in Figure 5 on the same area and with the same GPS stream using $N = 30$. It can be seen that the path on the high resolution network captures the whole journey, including u-turns and stops. On the low-resolution map, the path seems to follow the most likely route. Whenever some network part is missing, the algorithm sticks to the closest roads available (see the two circles on the left). However, the algorithm can also miss some links (see the circle on the right, the GPS points are not covered by a path) when there is an intermediary stop in some missing part of the network. As one would expect, the accuracy of the output highly depends on the quality of the network

Figure 4: Path accuracy as a function of the number of matched points



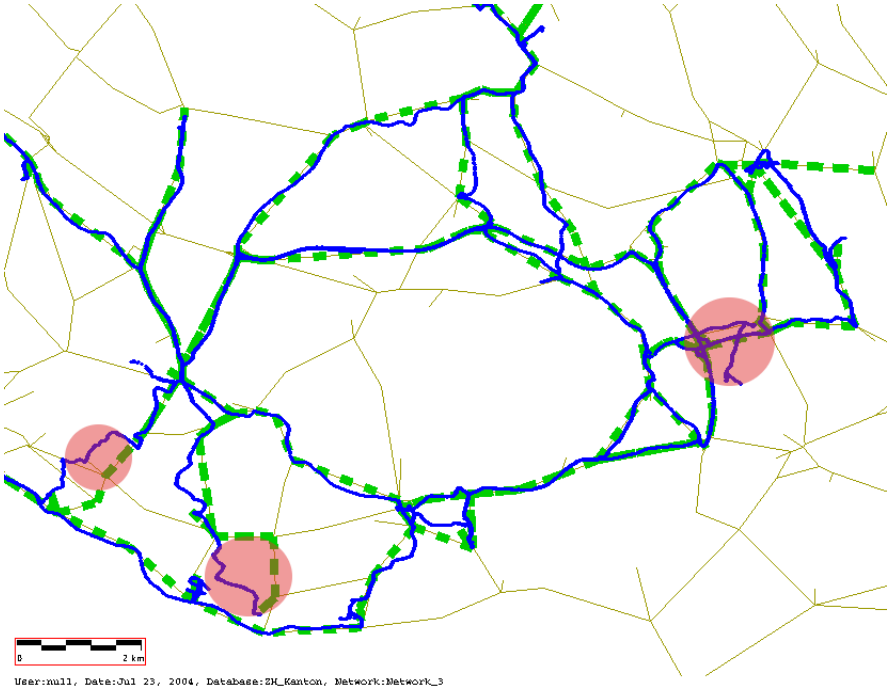
description. A practical problem encountered with the low resolution network in the speed monitoring experiment [2] is that trips initiated and ended at locations that correspond to traffic zones. As in most planning networks, the zone connectors are artificial links that do not correspond to physical road segments. The problem arises if there is not any zone connector in the vicinity of the actual trajectory of the GPS points. In that case, points are wrongly matched to the "physical" network and the information about the connection to the traffic zones might be lost. A potential solution might be to over-connect automatically the traffic zones to ensure that they can be reach from different directions.

4.4 Limitations

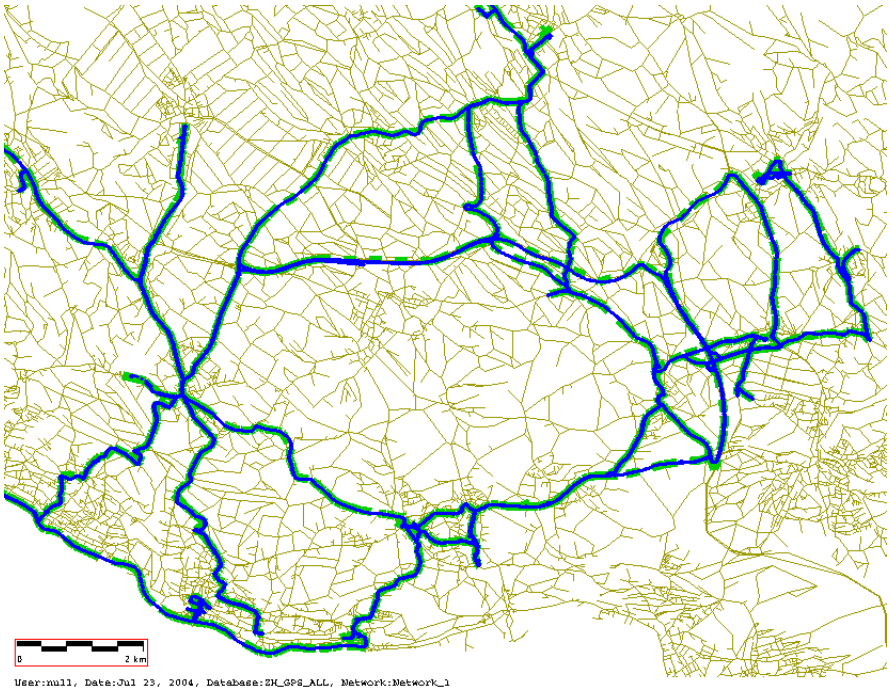
The algorithm depends on two parameters α and N. Running the map-matching on the whole data set has revealed that the accuracy of the algorithm can be quite dependent on the number of candidates N so that it is difficult to recommend a default value. Indeed, it can be seen from Table 1 that the accuracy does not increase smoothly with N. Figure 6 shows how the matching for $N = 11$ and $N = 12$ can behave quite differently. In the first case, the algorithm obviously misses its mark while in the second case the matching looks almost perfect. (Note that the discrepancy at the top of the map is due to the fact that polylines were not used in the graphical output.) Ideally, N should not be preset but adapted during the map-matching process. In practice, the algorithm is sufficiently fast and one can increase N manually after a first run if the accuracy that was achieved was not good enough. As noted above, increasing N increases the computation time only logarithmically.

The sensitivity on α has been only briefly tested and will not be reported here since a potential improvement of the algorithm is to replace equation (2) with a more

Figure 5: Map-matching with high and low resolution networks. GPS points are blue solid lines, the matched path is the green dashed line.



(a) Low resolution



(b) High resolution

realistic condition. In practice, $\alpha = 1$ constraints the path search too strongly and causes errors when vehicles perform u-turns in the middle of a road segment. On the opposite, $\alpha \simeq 0$ increases considerably the number of paths that are created. It also leads to unrealistic numbers of u-turns as GPS points can be alternatively closer to the left or right side of a given road segment. In practice $\alpha = 0.5$ yields reasonable results. Obviously, more elaborated -though slower- strategies can be implemented to replace equation (2).

5 Conclusion

A new map-matching algorithm has been developed that produces routes that are fully consistent with the coded transportation network. The core idea of the algorithm is to maintain a set of candidate paths as GPS data are fed in and to update constantly their matching scores. Despite its relative simplicity, the algorithm has proved to be very efficient at handling huge volumes of data. Its computational speed is in the order of magnitude of $1,000$ times faster than the collection time ($2,000$ GPS points/s). This implies that the map-matching of an average car trip can be performed every second and that entire trip databases could be processed in reasonable amount of times. Nevertheless, the efficiency of the algorithm depends on two parameters and is sensible to their variation. Further efforts will be dedicated to improve the robustness of the algorithm in that respect. Turn prohibitions at intersections could easily be taken into account as well. Finally, it remains to be tested if the algorithm is transferable to other modes, namely biking and walking modes which are less constrained by a network representation.

Acknowledgments

The authors express their thanks to T. Nideröst and to the Canton of Zurich for sponsoring this study. Computer resources were provided by CoLab, ETHZ.

References

[1] S. Doherty. Meeting the data needs of activity scheduling process modeling and analysis. *Presented at the 80th Annual Meeting of the Transportation Research Board*, 2001.

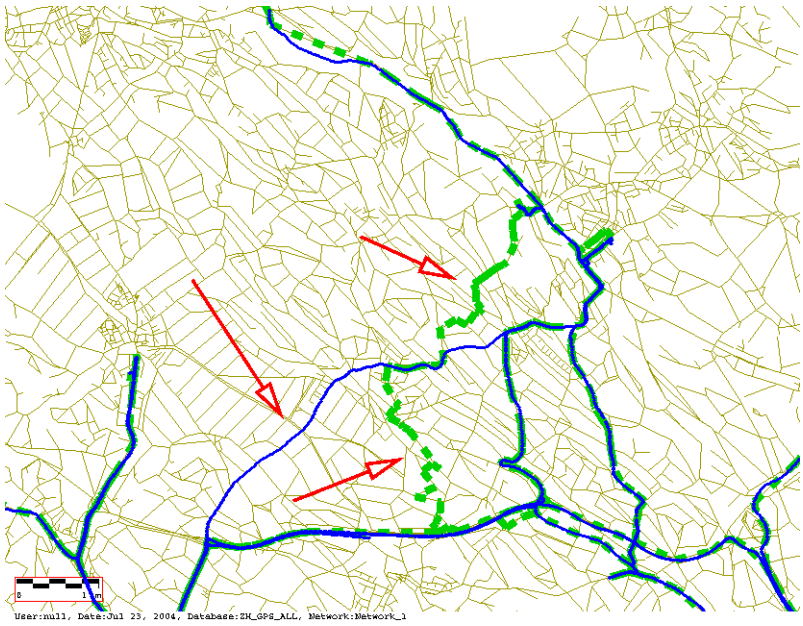
[2] J. Hackney, F. Marchal, and K.W. Axhausen. Monitoring a road system's level of service: The canton of Zurich floating car study 2003. *Submitted to the 84th Annual Meeting of the Transportation Research Board*, 2004.

[3] E. Murakami and D.P. Wagner. Can using global positioning system (GPS) improve trip reporting? *Transportation Research Part C*, 7:149–165, 1999.

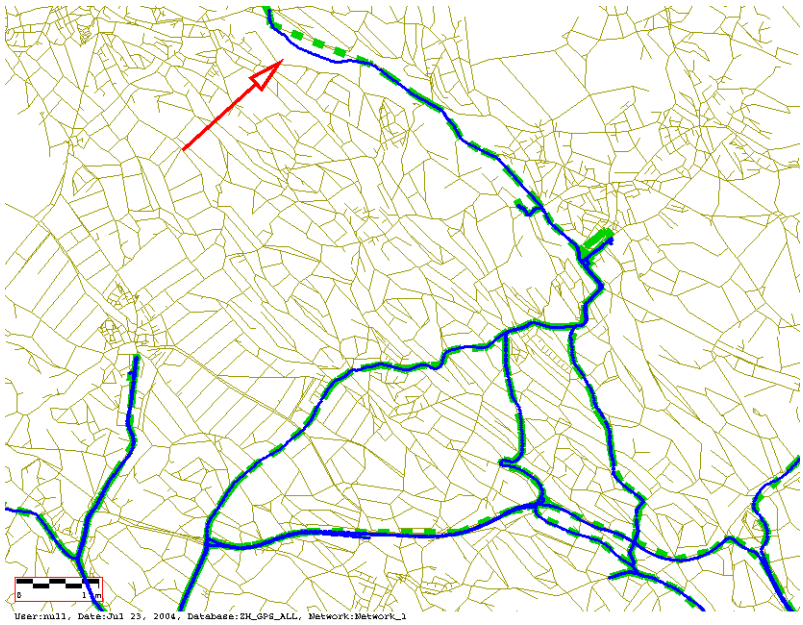
[4] O.A. Nielsen and R.M. Jørgensen. Map-matching algorithms for GPS-data - methodology and test on data from the AKTA roadpricing experiment in Copenhagen. *Presented at 5th TRIannual Symposium on Transportation ANalysis (TRISTAN V), Le Gosier*, 2004.

[5] J. Ogle, R. Guensler, W. Bachman, M. Koutsak, and J. Wolf. Accuracy of global positioning system for determining driver performance parameters. *Transportation Research Record*, 1818:12–24, 2002.

Figure 6: Sensitivity of the map-matching algorithm to the parameter N.



(a) N=11



(b) N=12

[6] J.-S. Pyo, D.-H. Shin, and T.-K. Sung. Development of a map matching method using the multiple hypothesis technique. *IEEE Intelligent Transportation Systems Conference Proceedings*, pages 23 – 27, 2001.

[7] M.A. Quddus, W.Y. Ochieng, L. Zhao, and R.B. Noland. A general map matching algorithm for transport telematics applications. *GPS Solutions*, 7:157–167, 2003.

[8] C.E. White, D. Bernstein, and A.L. Kornhauser. Some map matching algorithms for personal navigation assistants. *Transportation Research Part C*, 8:91–108, 2000.

[9] J. Wolf, S. Hallmark, M. Oliveira, R. Guensler, and W. Sarasua. Accuracy issues with route choice data collection by using global positioning system. *Transportation Research Record*, 1660:66–74, 1999.

[10] J. Wolf, S. Schönfelder, U. Samaga, and K.W. Axhausen. Eighty weeks of GPS traces: Approaches to enriching trip information. *Presented at the 83rd Annual Meeting of the Transportation Research Board, Washington, D.C.*, 2004.