Map-matching Algorithms for GPS-data - Methodology and Test on Data from The AKTA Roadpricing Experiment in Copenhagen

Otto Anker Nielsen René Munk Jørgensen

Centre for Traffic and Transport, Technical University of Denmark
Building 115, st. tv, Bygningstorvet
2800 Kgs. Lyngby, Denmark
{oan,rmj}@ctt.dtu.dk

1 Introduction

The literature has only to a limited extent considered the problem of tracking entire routes by GPS and linking this to a graph of links and nodes. Most GPS-units uses intuitive algorithms for map matching. However, this may not lead to coherent and correctly matched routes. The paper presents problems and solutions for the use of GPS to track cars routes in a road network. These are tested on data from the AKTA project which collected data on 500,000 trips, with 2,5x10^8 GPS-points map-matched related to a 300,000 link network. It appeared that intuitive algorithms were insufficient when using standard GPS equipment over the 2x12 weeks time-periods. It was accordingly necessary to develop new algorithms that can be used to process GPS-data. The results of this can be used for behavioural analyses, transport model estimation, speed measurement and congestion monitoring. The paper presents and compares different algorithms with respect to calculation time and precision.

To map-match a serie of GPS-points appears intuitively to be a trivial problem, i.e. the closest link must be the most likely link the car is using. However, in practice several problems occur with such an approach;

1) When a trip starts, it takes some time before the GPS-signal is valid. The most likely starting point must accordingly be estimated based on the last trip, co-ordinates and

time/speed.

2) When coordinates are logged, they are sometime fairly uncertain – typically when the unit is turned on and calibrates the signal, and in curves with high speed. The closest link may not be the link the car is actually driving on.

3) Due to the level of detail of the digital map and parallel or side roads, it is often not unambiguously which link, node or even route a car is following. This is especially the case for cars driving along motorways, where the centrelines of ramps, parallel and crossing roads may be nearer the car than the centreline of the motorway itself.

4) Signals are sometimes not recorded for some periods along a certain route, e.g. due to high buildings along the road, atmospheric disruptions, etc. It is in these situations necessary to carry out a qualified estimate on the routes.

5) In some situations, a trip might include a short errand: The route is apparently illogical, since the errand had not been detected. For traffic analyses and estimation of transport model, this can be a serious problem.

Although determining the location of trips starting points (ad 1) and the errands (ad 5) themselves are necessary problems to solve, the present paper concentrates on the map-matching problem itself, i.e. it is assumed that the trip ends have been determined by a prior algorithm.

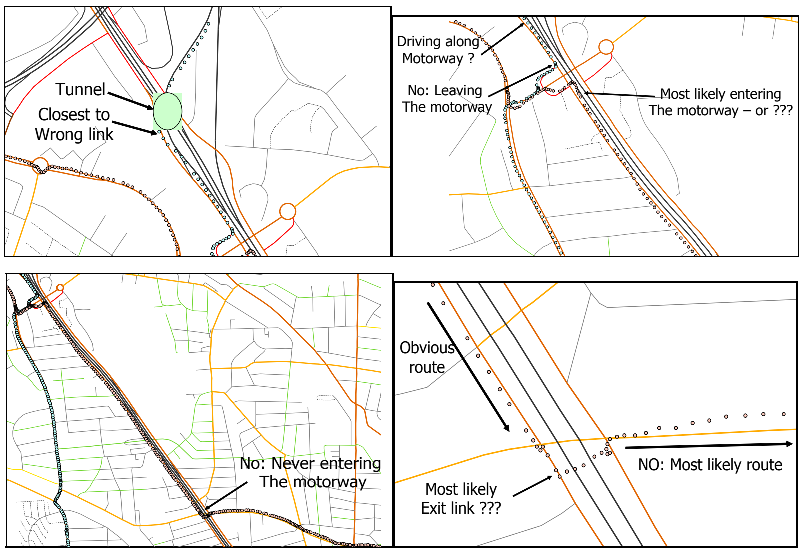

Figure 1. Different examples on map-matching problems.

Figure 1 illustrates some examples of typical map-matching problems. Upper left part illustrates a lost signal in a tunnel, followed by a sequence of points nearest the parallel road, where the car is not driving, and then along the motorway after the car in fact had exited it. This can be realised later on the route, where the car uses a local road perpendicular to the motorway. Upper right shows the same car when it re-enters the motorway. Along a long section of this, it is most likely that the car had entered the motorway. However, at the lower left and right it can be seen that the car had been driven at the parallel road along the motorway, and it is exiting this at a location where there have been no exit ramp from the motorway. In the example it was first possible to determine the route after a long sequence of links and nodes. However, in the specific case this could be done unambiguously by looking at the figure.

After working with the problem heavily on the large-scale AKTA data-set it is clear that the map-matching problem possess two major challenges:

1) Accuracy of the matched route. The more simple approaches have a very high percentage of errors, i.e. that most routes are not matched 100% perfect. Unfortunately this can only be inspected manually. For the worse algorithms this is course estimate, e.g. about 80% of the routes on motorways are not matched correctly, while the better approaches have been manually examined on a sub-set of 2,983 routes that have been selected to secure that all major roads and commuting patterns in the Copenhagen region are cowered.

2) Calculation time. Different algorithms and software solutions differ dramatically in calculation complexity and calculation time, i.e. from 1 second per trip up to 11 hours. To process the complete AKTA dataset this equals am interval between 8 days and an estimated calculation time of 600 Years! For obvious reasons only algorithms with up to 2 weeks calculation time have been tested on the entire dataset. The others' calculation times have been estimated on a sub-set of trips.

The paper briefly describes the experiment used for the tests, followed by the formulation of basic components of map-matching algorithms and use of these.

2 Basic components of the mapmathing problem

Following our work with the map-matching problem, it appears suitable to classify the map-matching problem into different sub-problems after the pre-processing. The GPS co-ordinates were logged each second and then pre-processed. A filter criteria in the GPS-unit skipped points which 1) Indicated an unreasonable large change in speed or location (>

|10m/sek/sek|), 2) Indicated an unreasonable high speed (>200 km/h). or 3) Was fluctuating due to no or very little movement at all (< 5 km/h).

3 Nearest link problem

Intuitively map-matching is to assume that the GPS-point is located along the nearest link. Although this is not a valid assumption, as mentioned above and shown in figure 2, the nearest link problem is none-the-less an important part of the map-matching problem.

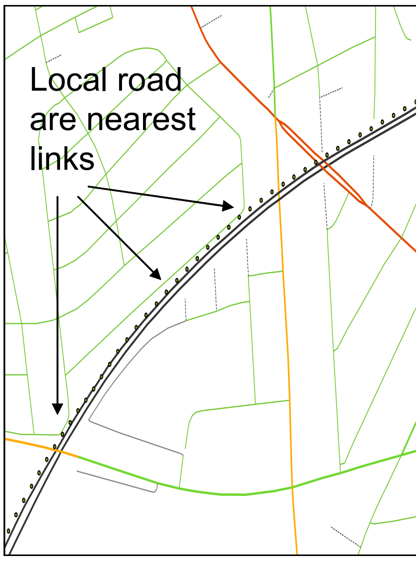

Figure 2. Example where the nearest links are clearly not on the most likely route.

The calculation time for the problem on the Copenhagen network is about half real driving time. This is feasible in on-car units for car driver guidance. But not for post-processing of the entire data-set in an experiment, it would take more than 100 years processor time for the AKTA data. Some experiments on use of GPS-data for traffic studies[1] prepossess data in the unit, i.e. store a sequence of nearest links only. Errors are hereby inherited in the raw data, before these are post-processed.

The nearest link problem can be solved more efficient by referring the network to a structure of grids and sub-grids. Each link is then assigned to a master-grid and sub-grid. The relevant sub-grid is then found for the GPS-point, the corresponding links by a binary search, whereby only distances links within the links in the sub-grid has to be calculated, instead of to all links in the network.

4 Nearest node problem

The nearest node problem is in principle the same as the nearest link, except that the calculation is simple, especially if the link consists of a series of shape-points and lines.

[1] I.e. the Gothenburg Road Pricing Experiment, the Borlänge data and speed control experiments in Malmö (all in Sweden) and Aalborg (Denmark). These experiments have not been reported in papers on map-matching.

5 Score procedure (several links or nodes)

As illustrated by figure 1 and 2 the nearest link or node in the network is in some cases not the one that the car used. It is therefore relevant to examine a set of links and nodes within a certain buffer instead of just the nearest one. At least this must have the size of the 95% confidence intervals for the accuracy of the map-matching procedure + the same for the digital map + the half width of the road, e.g. ½ x number of lanes x width per lane.

The distance between each GPS-point increases with speed. Thereby the buffer distance for the node score must be higher for high-level roads (e.g. motorways) than local roads. At 120 km/h (the speed limit at Danish motorways), the distance between each logged point is 33 meter.

Although all links and nodes within the buffer may have been used, it is still likely that the closest are most likely used. A score for each link can then be calculated, e.g. by the following formula:

$$Score = 1 \Big/ \min\{\alpha_n \cdot dist(point, node)^{\beta_n}, \max score_{(n)}\} +$$
$$\sum_{points_a} 1 \Big/ \min\{\alpha_a \cdot dist(point, link)^{\beta_a}, \max score_a\}$$

The score of the from-node *n* at the beginning of the link is a function of the distance between the GPS-point and node. This can be weighted by a node-type dependent coefficient α_n and powered by $\beta_n < 0$. The smaller β the more importance of nearness between the two. However as the function could be infinite with very small distances it is reasonable to assume a maximum score. The link score is the sum of the score of each GPS-point along the link. Practical tests have shown that cars are most likely following major roads, why a link-type-dependent weight can improve the performance of the algorithm in cases where there are some disturbances of the GPS-signal[2].

6 Connectivity

Another – not so often used – component of map-matching procedures is to check the network

[2] The coefficients and link types must be calibrated according to the link-classification in the specific digital map, the accuracy of the map, and the accuracy of the specific GPS-units being used. A guideline though is $\beta<-1.5$ and α twice as high for the motorways and major roads compared to residential streets. Node score should be weighted 2-5 times higher than link-score, since there are much more GPS-points along links than nodes.

connectivity of the present link or node with prior links and nodes. The problem after the tunnel in figure 1 could e.g. have been avoided by checking one link backward. In real time GPS this can improve performance significantly. If post-processing backward as well as forward, connectivity check can improve performance even further.

When implementing this rule it was realised that the entire graph should be built in memory, and that each node must have pointers to both forward and backward links, while each link as well should have pointers to from and to nodes.

7 Path building

When a sequence of map-matched links and nodes has been found, there might be intervals of signal fall-outs, tunnels, etc. without GPS-signals. The most likely route between then can be estimated as shortest path that minimise a function of e.g. travel time, congestion time, travel length and costs. AKTA had enough information per person (200-1000 routes) to estimate individual functions. The path building has a Dijkstra like complexity, which turned out to be far less than the map-matching itself, i.e. about ½ second for a long path, while the map matching itself had calculation times between 1 second and 11 hours depended on approach. A short path, e.g. in case of a short fall out, can be calculated much faster, since the search is stopped when the end node is removed from the heap.

8 Loop detection

After paths have been built, there may be illogical routes with loops (a node or link is used twice on the same route) or infeasible travel times (e.g. more than 100% faster than allowed). This is especially the case when using too simple map-matching algorithms. If for example a closest link approach has been used on figure 2, the car has to exit the motorway to find a path to the (wrongly) nearest local road, and then drive back and re-enter the motorway using the same ramp as before to continue to the next section of the motorway which has been map-matched.

An improvement of the loop detection test is to check whether the second last node in the prior sequence of matched nodes can be connected to the first in the second sequence (or the last with the second). If this avoids the loop, this is a more reasonable assumption on the route, than just removing the loop.

9 different map matching algorithms

In the following, experiences with different map matching algorithms are presented.

9.1 Nearest link approach

Although being the most intuitive this approach didn't work in practice. Even with a post loop detection approach most trips along motorways had some errors, which were also the case for some roads with median strip in urban areas. This approach also had an infeasible high calculation time if being used on the entire AKTA-dataset.

9.2 Nearest node approach

The nearest node approach posses the same methodological problems as nearest link, although to a slightly less extent. In the case in figure 2 no nodes where e.g. too close to the motorway, why errors still occurred in the case in figure 1. However, some nodes along motorways may not be matched due to the high speed of cars. The alternative is to increase the buffer size. This however adds a lot of nodes at local roads along the motorways, which makes things worse. The results of the nearest node approach are in general worse than by the nearest link approach, e.g. almost no routes along motorways are matched correctly. The calculation time is feasible, since it is much faster to find the nearest node than link, i.e. about 1 second calculation time per route in AKTA.

9.3 Semi topological relationship, nodes

The idea with this approach is only to accept a matched node, if the prior and next node in a candidate list of nodes has been map matched[3]. Due to the high calculation time of the nearest link approach, topology has only been tested for nodes. Each route is here made up by joining a series of nodes that have been selected in the following way.

1) For every GPS point it is examined whether it is close to a node in the network and if it is

[3] Some methods in GPS-equipment examine the prior link and/or node, but not the next. Loop detection is not part of these approached as well. However, the tested method can still be considered as a modified improved version of a well-known approach.

closer than any of the other GPS points. If this is true then the node is selected and a timestamp is imported from the GPS point.

2) In this way a series of nodes are collected and then joined by the links connecting them which hereby represents a trip with a time stamp on each node. The process is actually a little more complex, as it also uses information from one way roads and if any of the directly connected nodes have been selected.

3) If matching fails for a while, the program estimates the missing links as the optimal path. References to links and nodes in such a sequence are stored with a flag stating *calculated*, opposite step 2 that are *measured.*

4) Finally, if some points could no be logically connected (e.g. in case of a loop in the route), these are cut off, and the route is re-estimated.

After having prepared the data for the program it is able to process 1 driver's data in about 10 minutes (on a Pentium 2,400 MHz), which on average is around 1 trip per second. All 500 drivers therefore take around 4 days to map match (about 250 million GPS-points are processed).

The accuracy of this approach was far better than in 4.1 and 4.2. A manual inspection of the 2,983 trips mentioned in the introduction revealed 84 different errors. In addition 12 were repeated errors for cars driving the same route – primarily along motorways since these are used most intensively. About 70 errors were along motorways, and 10 along major roads with median strips. After inspecting the errors along motorways, it appeared that about 50 of them happened near ramps exiting or entering the motorways. These errors can be removed by manually splitting motorway links into two by adding a pseudo node perpendicular to the links at the node at the ramp. The problem is then avoided for all other cars along the road, e.g. removing repeated errors[4]. Checking connectivity more than one link back and forth do not improve the performance, since this may posses problems in cases of signal fall outs and along motorways if a single node is not matched (motorways have per definition few nodes).

10 Pre-processed score

The above methods have the weakness that only the nearest nodes or links are examined. A potentially better approach is to assign scores to all links and nodes in the network. This is done

[4] Note that the trips were stratified to cover the whole Copenhagen region, why emphases were laid to have as few as possible repeated routes among the 2,983.

by examining the sequence of GPS-points one by one and assigning a score to all links and node within a certain buffer size as in section 3.3. The optimal path is then found by minimising a function of time (take care of signal fall out) and score (should be inversed, i.e. the higher score the better).

Each calculation is a one to many relationship, which only has a marginal higher calculation complexity than in section 4.1. and 4.2. The link approach is heavier than the node approach.

The method was implemented both in ArcGIS 8.3 and MapX. In both cases it was possible to solve the map-matching problem on the routes being tested. However, the calculation time per route was in average 14 hours using ArcObjects and 11 hours with MapX, i.e. a completely infeasible calculation time for the AKTA dataset with 0.5 mio. routes.

11 Dynamic path building

A much more complex but potentially efficient algorithm is to build the route dynamically. A pre-processing step determines the most likely beginning node. From here all exit links are examined, and marked as used. Each of the to nodes of these are put into a heap, if the link got a certain minimum score. Then a new candidate node is drawn from the heap as the node first in time (it is essential that it is first in time, not lowest utility, since the order of the GPS-points must not be messed up). The links leaving the node are then examined, if not already marked as used.

When each link and node is examined the score is calculated as in section 3.3 and added to the score along the path to the link or node. If a node has already been reached and the score is lower than the new path, then the new path is defined as the optimal, regardless of the GPS-time stamps[5].

The core benefit with this approach is that the score is calculated for a sequence of GPS-points along a link, where the beginning of the sequence is defined by the time of the last GPS-point before entering the link. This is accordingly a one-to-one relationship and not a one to many as in pre-processing methods such as in section 4.1, 4.2 and 4.4. The calculation complexity is accordingly far lower than 4.4, and even lower than 4.2 and 4.3 which have a feasible calculation time. At the same time, the map match has a potentially better quality than 4.3 and 4.4.

When the heap is empty, the optimal route can be found simply by taking the end-node with the

[5] This appears to be the most efficient, since the accuracy of the point in space may lead to a not optimal choice of path in time. This will be elaborated further in the final paper.

maximum score and track the path backward. This does not finalise the algorithm, since there could be more GPS-points in time (in case of signal fall out). These are processed in the same way, and the path pieces are then connected by the path building procedure in section 3.5.

Unfortunately, the algorithm is more complicated to programme, since the path building and map-matching are done dynamically together. Thereby existing libraries in MapX (map info) or MapObject (ArcGIS) can only be used in a loop, whereby the graph is not stored in memory but on hard disk which is infeasible in terms of calculation time). Therefore, the entire algorithm has to be programmed from scratch which is being done at the moment.

12 Conclusions

Map-matching is being used in many cars' onboard GPS-units as well as to process GPS-data for traffic planning and analyses. The paper shows that the most commonly used approaches may lead to errors in many cases – especially along motorways and roads with median strips. Different improved or new algorithms are present, which have either an acceptable quality or calculation time. The semi-topological approach has a feasible calculating time and not too many errors, especially if motorways are splitted by pseudo nodes, whereby "only" about 1% of the routes had errors. A potentially better algorithm – the dynamic path building were outlined. This has potential of improving both calculation time and accuracy, since it combines the best elements of the other approaches.