Map Matching and Uncertainty: an Algorithm and Real-World Experiments

Kristof Ghys, Bart Kuijpers, Bart Moelans, Walied Othman and Dries Vangoidsenhoven
Hasselt University, Belgium
{firstname.lastname}@uhasselt.be

Alejandro Vaisman
Universidad de Buenos Aires, Argentina
avaisman@dc.uba.ar

ABSTRACT

A common problem in moving object databases (MOD) is the reconstruction of a trajectory from a trajectory sample (i.e., a finite sequence of time-space points). A typical solution to this problem is linear interpolation. A more realistic model is based on the notion of *uncertainty* modelled by space-time prisms, which capture the positions where the object *could* have been, when it moved from a to b. Often, object positions measured by location-aware devices are not on a road network. Thus, matching the user's position to a location on the digital map is required. This problem is called *map matching*. In this paper we study the relation between map matching and uncertainty, and propose an algorithm that combines weighted k-shortest paths with space-time prisms. We apply this algorithm to two real-world case studies and we show that accounting for uncertainty leads to obtaining more positive matchings.

Categories and Subject Descriptors

F.2 [**Analysis of algorithms and problem complexity**]: Miscellaneous; H.2.8 [**Database applications**]: Spatial databases and GIS

General Terms

Algorithms

Keywords

Map Matching, GIS, uncertainty

1. INTRODUCTION

The most popular location-aware devices nowadays are GPS-based. Even though most people use GPS as a navigational tool, it can also be used to record the position of a moving object (e.g., a car, a pedestrian) for data analysis. We can analyze the trajectories of a person and study why that person preferred a particular route over another. The main disadvantage with GPS coordinates is that they are not exact, i.e., they do not always map to a road. Therefor, every GPS device accounts for these errors by mapping the exact street driven on instead of just modeling the GPS coordinates received from a satellite. Many algorithms were devised for this task, as we discuss in Section 1.3. The process of matching GPS positions to a road network is called *map matching*.

One particular model for the management of the uncertainty of the moving object's position in between sample points is provided by the *space-time prism* model. In this model, it is assumed that aside from the time-stamped locations also some background knowledge, in particular a (e.g., physically or law imposed) speed limitation at certain locations is known. The space-time prism between two consecutive sample points is defined as the collection of space-time points where the moving objects could have passed, given the speed limitation. The chain of space-time prisms connecting consecutive trajectory sample points is called a *lifeline necklace* [1] (see Figure 1). Whereas space-time prisms were already conceptually known in the time geography of Hägerstrand in the 1970s [4], they were introduced in the area of GIS by Pfoser [8] and later studied by Egenhofer and Hornsby [1, 5], and Miller [6].

In this paper we present an algorithm that uses a combination of weighted k-shortest path algorithms and space-time prisms to solve the map matching problem and apply this algorithm to two real-world case studies.

1.1 Problem statement and case study

An emergency service in a small European city ($\pm$ 15k street segments) discovered that new employees did not know the city very well and it took them longer than experienced employees to arrive at the place of intervention. The experienced drivers are able to transfer knowledge to new drivers such as, avoid streets with schools or to take advantage of tram lanes, but there is also subconscious knowledge. The emergency services wants to discover this knowledge using the most recent knowledge discovery techniques [2].

In the scenario above, a typical problem that arises is that about ninety-five percent of the points fall off the roads actually taken and that sometimes it is ambiguous to which road segment they should be matched. Thus, there is a need to map points to the road network. This problem is called *map matching* and will be the focus of this paper.

ACM GIS 2009 Seattle, Washington, USA

1.2 Contribution and paper organization

The main contribution of this paper is a map matching algorithm that uses a combination of a techniques that handle uncertainty in trajectory data (more precisely, via *space-time prisms*), and a weighted k-shortest paths algorithm.

In Section 1.3 we review related work on map matching. Section 2 presents the theoretical basis of our work, mainly studying how uncertainty in road networks is addressed using the notion of space-time prisms. In Section 3 we discuss different techniques and technical problems of map matching, and introduce our algorithm. In Section 4 we report on experimental results.

1.3 Related work

Map matching algorithms can be classified into three categories: (a) *Geometric* [10, 11]: geometric information of the original road network and not topological information is considered; (b) *Topological* [3, 9]: the geometry of the network as well as the connectivity and contiguity of the links are considered; (c) *Probabilistic* [7]: these algorithms define an elliptical or rectangular confidence region. The error region is superimposed on the road network to identify a road segment. If the error region contains more than one street, probabilistic algorithms perform a weighted search on the candidate streets.

The algorithm we present in this paper can be considered a variant of a probabilistic techniques (Section 3.2) and topological techniques (Section 3.3).

2. A MODEL FOR MOVING OBJECT DATA WITH UNCERTAINTY

In this paper, we consider moving objects in the two-dimensional (x, y)-space $\mathbf{R}^2$ and describe their movement in the (t, x, y)-space $\mathbf{R} \times \mathbf{R}^2$, where t is time (we denote the set of the real numbers by $\mathbf{R}$).

Moving objects, which we assume to be points, produce a special kind of curves, which are parameterized by time and which we call *trajectories*.

In practice, trajectories are only known at discrete moments in time. This partial knowledge of trajectories is formalized in the following definition. To stress that some t, x, y-values (or other values) are constants, we will use sans serif characters.

DEFINITION 1. *A* trajectory sample *is a finite set of time-space points* $\{(\mathsf{t}_0, \mathsf{x}_0, \mathsf{y}_0), (\mathsf{t}_1, \mathsf{x}_1, \mathsf{y}_1), \ldots, (\mathsf{t}_N, \mathsf{x}_N, \mathsf{y}_N)\}$*, on which the order on time,* $\mathsf{t}_0 < \mathsf{t}_1 < \cdots < \mathsf{t}_N$*, induces a natural order.* □

Often, in practical applications, more is known about trajectories than merely some sample points. For instance, background knowledge like a physically or law imposed speed limitation at location might be available. The speed limits that hold between two consecutive sample points can be used to model the uncertainty of a moving object's location between sample points. For modeling uncertainty, time geography [4] introduced space-time prisms (Pfoser et al. [8] introduced this in moving object database literature as *beads*). We now formalize the concepts above.

The *space-time prism* between two sample points $p = (t_p, x_p, y_p)$ and $q = (t_q, x_q, y_q)$, with $t_p \leq t_q$, and maximal speed $v_{\max} \geq 0$ is the set of all points $(t, x, y) \in \mathbf{R} \times \mathbf{R}^2$ that satisfy the following constraints

$$\begin{cases} (x - x_p)^2 + (y - y_p)^2 \leq (t - t_p)^2 v_{\max}^2 \\ (x - x_q)^2 + (y - y_q)^2 \leq (t_q - t)^2 v_{\max}^2 \\ t_p \leq t \leq t_q. \end{cases}$$

It describes the set of space-time locations where the moving object could have been between sample points p and q. We denote this set by $Pr(p, q, v_{\max})$ or $Pr(t_p, x_p, y_p, t_q, x_q, y_q, v_{\max})$. The projection of a space-time prism on the (x, y)-plane is an ellipse.

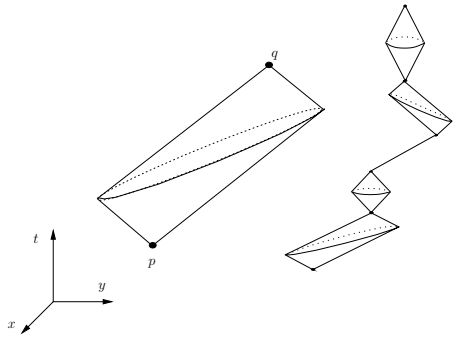

Figure 1: A space-time prism and a lifeline necklace.

Figure 1 illustrates the notion of a space-time prism and *a chain of space-time prisms*, a *lifeline necklace* [1], in time-space.

3. MAPPING TO A ROAD NETWORK

A part from being discrete, the data from GPS devices can also contain gaps, e.g., because of tunnels. Measurements of locations can be (far) off-road or just between two roads.

In this section we describe a map-matching algorithm that uses space-time prisms, for handling problems of real-world data. This algorithm is summarized in pseudodcode in Listing below. In Section 3.1 we will explain Lines 6-11, how to select the region where the car could have been. Section 3.2 will describe Line 13, making a weighted road network. Finally in Section 3.3 we will describe the k-shortest path algorithm on Line 16.

```
1  Input:
2  T := array of sample points;
3  G := road network;
4
5  G_part := empty road network;
6  For i := 1 tot (|T| - 1){
7    P := calculate part of G where the
8         route between T[i] and T[i+1]
9          could have been;
10   G_part := G_part ∪ P;
11 }
12
13 G_score := Calculate a weighed
14           road network from T and G_part;
15
16 K := set of candidate routes
17                   based on G_score;
18
19 Return the route in K
20      with highest score;
```

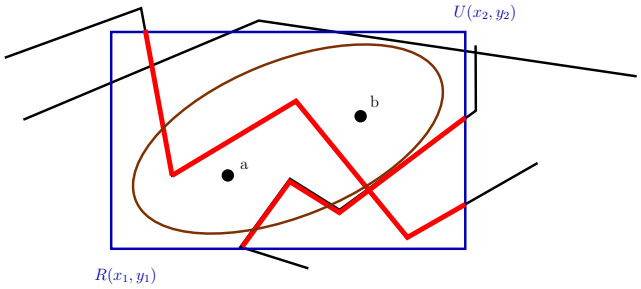

Figure 2: Projection of a space-time prism over the road network, for two points a and b.

3.1 Using space-time prisms to limit the scope

For example, in our case study, given the time between two consecutive recorded points, and a maximal speed, a car could have been in many possible locations, given by the projection of the space-time prisms over the plane. Even though, in fact, this projection is an ellipse, for simplicity we compute a bounding box given by two points $R(x_1, y_1)$ and $U(x_2, y_2)$ (see Figure 2). The point R is computed as follows: x_1 is the farthest point on the X-axis that can be reached moving away from b driving at maximum speed v_{max}. Analogously, y_1 is the is the farthest point on the Y-axis that can be reached moving away from b driving at maximum speed v_{max}. The point U is computed as follows: x_2 is the farthest point on the X-axis that can be reached moving away from a driving at maximum speed v_{max}. Analogously, y_2 is the is the farthest point on the Y-axis that can be reached moving away from a driving at maximum speed v_{max}.

It is known that GPS data can also contain outliers. These are handled with these space-time prisms, because if the distance between two consecutive points is so that one can not reach the next point in time with the given maximum speed, the space-time prism would be empty and thus no roads will be added.

We select the largest maximum speed allowed on the road network.

3.2 Assigning a score to a road network

In previous section, we limited the total road network G to a road network G_{part} (the part of G where the car could have been). We will now create a weighted road network G_{score}, using G_{part}. Initially G_{score} is G_{part} where every road segment gets a score of 0. For every sample point p we look at the n closest road segments in G_{part}. The score of the closest road segment to p will be increased with n, the score of the second closest road segment to p will be increased with $n-1$, and so on.

3.3 Weighted shortest path computation

We adapt Yen's algorithm [12] to rank the shortest paths between a pair of nodes, using a scoring algorithm discussed in Section 3.2, to give weight to the edges in a road network. In our adaptation of the algorithm, a score is calculated for each shortest path (as described in Section 3.2); if this score is greater or equal to that of the shortest path calculated at the beginning, it is added to the result list. We return the path with the highest score. The complexity of the algorithm is $O(n^3)$. Calculating the spur paths from each node is $O(n)$ and using a shortest-path algorithm (Dijkstra) $O(|V|^2+|E|)$ with $|V|$ number of vertices (intersections) and $|E|$ the number of streets (edges).

The output of the k-shortest path algorithm depends on the choice of start and end road segment. However, the choice of start and end segment is one aspect of a map matching algorithm. To increase the chance of choosing the the correct road segments as start or end segment, we altered the k-shortest path algorithm.

We can now elaborate the Line 16 in above Listing of our implementation:

```
K := is empty set of routes;
S1 := set of road segments inside space-time
      prism of the first two points of L;
S2 := set of road segments inside space-time
      prism of the last two points of L;
For every segment r1 in S1{
 For every segment r2 in S2{
  While there are shortest routes left {
    R := calculate shortest route
         between r1 and r2 based Gscore
         and in a way such that R ∉ K;
    K := K ∪ R;
 } } }
```

4. EXPERIMENTS

In this section we report the experimental results obtained by applying the algorithm. For these experiments we used a naive geometric map matching algorithm to compare with.

4.1 Setup of experiments

A simple approach to map matching consists of using a geometric algorithm that computes the road segments closest to the GPS coordinates (measuring the perpendicular distance from the point to the segment), and matches a point to that road segment. In this way, a trajectory sample that lies within the road is obtained. In spite of its simplicity, which makes it very (computationally) efficient, this method has some drawbacks, given the characteristics of real-world data discussed above. In our experiments, these drawbacks sometimes prevent from obtaining a matched trajectory (e.g., if there are large gaps in the data).

The experiments were run on a Compaq 8510p machine, Core(TM)2 Duo CPU T8100 Intel processor, at 2,1 GHz, with 4 GB DDR RAM.

We used two datasets. The first corresponds to our emergency service case study presented throughout the paper. The emergency service started recording when they got a call for an intervention, and they stopped recording when they arrived at the intervention site. Every route is a sequence of GPS coordinates (GPS points), stored in (x, y, t) form, where (x, y) represents the location and t is the time where the object was at this location. It is worth noting that a great portion of the available data had to be discarded, mainly for two reasons: (a) they contained less than 10 GPS points; (b) they contain more than 3400 GPS points, indicating that, most likely, the users did not turn off their device as requested. We can see then, that this dataset contains very precise measurements. Our second dataset corresponds to traffic in the Italian city of Milan, recorded during a week. Here, GPS coordinates are recorded at irregular intervals, i.e., data are more imprecise.

4.2 Results

We ran several experiments, aimed at: (a) revealing the possible sensitivity of the space-time prisms method to the selection of the maximal speed (i.e., the size of the space-time prisms). We investigated how this affects precision and execution time, on two datasets of different characteristics; (b) comparing the space-time prisms method against a simple geometric method, which, likely, runs faster, at the expense of obtaining less reconstructed trajectories. The results are shown in Tables 1 and 2. We report the maximum, minimum, and average execution times (we exclude the time for loading the data), for the two datasets, using the geometric algorithm and the new uncertainty-based algorithm, referred to as STP (space-time prism) in the first column. For the latter, we used the maximum speed as a parameter, ranging from 40km/h to 120km/h.

Table 1 shows that for the space-time prism method with maximum speed 120 km/h and the one with 80 km/h, the results are very similar. However, the difference is significant when the maximum speed is 60 km/h. Note that the speed limit in the city ranges from 50 km/h and 70 km/h. The results suggest then, that the trajectories were not recorded during an urgent intervention that requires driving faster than these limits. Looking at the maximum execution time in Table 1, one can see that our algorithm using a maximum speed of 80 km/h does not only give much better results then the naive algorithm, it is also faster. The explanation for this is, that our algorithm considers less roads. Then, since execution times are better for a maximum speed of 80 km/h, it would be better, in this case, to use this speed for analysis.

For the Milan data (Table 2), results were different, due to the fact that the GPS data was not as dense as in the first case. In consequence, the space-time prisms between two points are larger, and include more roads, which the algorithm has to process. The Milan GPS data consists of three types of trajectories. The first type are trajectories that do not enter the center of the city (e.g. people that park their car and use the public transport in the center). The second type are trajectories that come from outside the city and enter the center. The last type are the trajectories recorded within the center of the city itself. When we look at the results we notice that when the speed limit decreases, the number of trajectories that can be reconstructed also decreases. This is because the GPS points are further away from each other, compared with our first dataset.

Table 1: Results for the emergency service data

Algorithm	Min. (msec)	Avg. (msec)	Max. (msec)	Reconstructed trajectories(%)
Geometric	50,56	141,32	472,66	78,8%
STP 120km/h	103,49	246,23	825,45	96,7%
STP 100km/h	99,42	219,93	616,80	93,4%
STP 80km/h	62,06	193,47	445,88	90,1%
STP 60km/h	60,24	177,09	449,21	54,5%
STP 40km/h	58,38	167,23	455,82	15,1%

Table 2: Results for the Milan data

Algorithm	Min. (msec)	Avg. (msec)	Max. (msec)	Reconstructed trajectories(%)
Geometric	79,69	134,99	193,98	13,3%
STP 120km/h	1132,82	2680,88	13665,48	96,7%
STP 100km/h	902,73	1889,39	8303,49	83,3%
STP 80km/h	778,41	1452,79	5475,36	53,3%
STP 60km/h	674,68	1066,19	1957,36	26,73%
STP 40km/h	561,39	845,75	1244,86	10,0%

Acknowledgments

This work was partially funded by the Information and Communication Technologies Programme of the European Commission, Future and Emerging Technologies under the ICT-245410 MODAP project.

5. REFERENCES

[1] M. J. Egenhofer. Approximation of geospatial lifelines. In E. Bertino and L. D. Floriani, editors, *SpadaGIS, Workshop on Spatial Data and Geographic Information Systsems*. University of Genova, 2003.

[2] F. Giannotti and D. Pedreschi, editors. *Mobility, Data Mining and Privacy - Geographic Knowledge Discovery*. Springer, 2008.

[3] J. S. Greenfeld. Matching gps observations to locations on a digital map. In *Proceedings of 81th Annual Meeting of the Transportation Research Board*, 2002.

[4] T. Hägerstrand. What about people in regional science? In *Papers Regional Science Association*, volume 24, pages 7–21, 1970.

[5] K. Hornsby and M. J. Egenhofer. Modeling moving objects over multiple granularities. *Ann. Math. Artif. Intell.*, 36(1-2):177–194, 2002.

[6] H. J. Miller. A measurement theory for time geography. *Geographical Analysis*, 37:17–45, 2005.

[7] W. Y. Ochieng, M. A. Quddus, and R. B. Noland. Map-matching in complex urban road networks. *Brazilian Journal of Cartography*, 55(2):1–18, 2003.

[8] D. Pfoser and C. S. Jensen. Capturing the uncertainty of moving-object representations. In R. H. Güting, D. Papadias, and F. H. Lochovsky, editors, *SSD*, volume 1651 of *Lecture Notes in Computer Science*, pages 111–132. Springer, 1999.

[9] M. A. Quddus, W. Y. Ochieng, L. Zhao, and R. B. Noland. A general map matching algorithm for transport telematics applications. *GPS Solutions*, 7(3):157–167, 2003.

[10] G. Taylor, G. Blewitt, D. Steup, S. Corbett, and A. Car. Road reduction filtering for gps-gis navigation. In *Proceedings of 3rd AGILE Conference on Geographic Information Science*, pages 114–120, 2001.

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

[12] J. Y. Yen. Finding the lengths of all shortest paths in n-node nonnegative-distance complete networks using $\frac{1}{2}n^3$ additions and n^3 comparisons. *J. ACM*, 19(3):423–424, 1972.