Intelligent Traffic with Automatic Violation System [ITWAVs]

Location Determination Algorithm

Dr.Murad Abusbaih, Khalid I. Baradia, Abdullah Z. Ja'abary
Dept. of electrical engineering
Palestine Polytechnic University
Hebron-Palestine, May 2013
Murads@ppu.edu

***Abstract*—In order to determine the speed limit of the street where the car located, the system must match the vehicle location i.e. the current GPS reading with the pre-stored value in database. In this system, we will develop a geometrical algorithm, which will be used to approximate the current GPS data to the closest street.**

***Index Terms*—Speed limit, GPS, database.**

I. INTRODUCTION

The goal of this algorithm is to identify 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 is to find the best estimated of 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 (database) which approximates the physical world.

Most of the existing map-matching algorithms [1], [2], [3], are more focusing on the accuracy than the computational speed of the algorithms. Therefore, our concern is slightly different from the existing literature.

First, we assume in this algorithm 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 and we will 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 will 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.

To summarize, we have 1) a standard network representation of the transportation system, directed links and "polygons" describing their curvature as will be explained later, and 2) a stream of GPS coordinates recorded for every time. The problem is to find the path in the network that is the closest to the GPS points and in minimum computations.

II. Proposed Algorithm

Let G (V, E) be the directed graph describing the road network. V is the set of vertices or nodes and E is the set of edges or links. Let Q_i, Q_i' be the set of points given by the GPS data stream, set of corrected GPS points respectively, I = 1...T, Each GPS point consists of a pair of coordinates (x_i, y_i).

In order to minimize the GPS reading error, a small shift perpendicular to the link direction will be introduced, according to the end 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 +Δ, and B' = B +Δ, Δ is chosen to reflect the average physical distance from each edge of the road. In this system, Δ = 5m in each direction, since the average GPS error is about 10m.

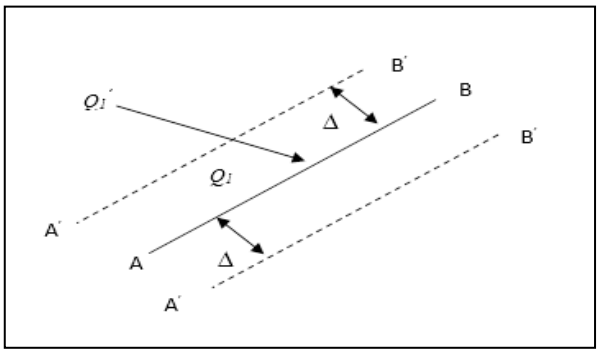

Figure 1:*Distance between a point and a road segment*

The most important thing that must be searched is how to satisfy the algorithm objectives with the least computational complexity, and minimum time. So, the Algorithm initialized by distributing the working region into N_i polygons, each polygon is restricted by the lowest and highest GPS points.

Inside each polygon, distribute the streets into S_j internal polygons, each one is restricted by the lowest first and highest end GPS points, whereas a diagonal link can be connected between this two points.

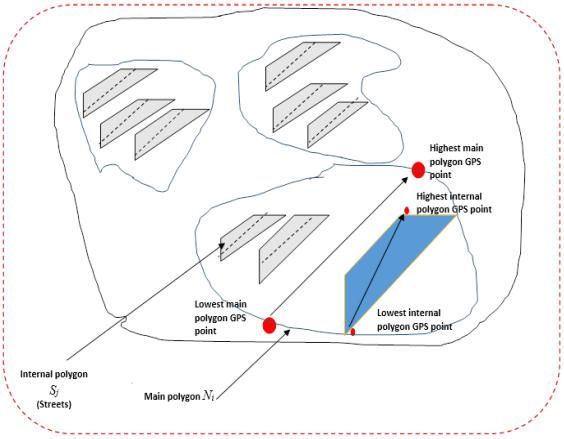

Figure 2: *Technical algorithm description*

Technically, this algorithm sorts each group of streets into one polygon, which help in decreasing the searching process inside the database. In other words, the searching process will be divided into two stages: first, inside the N polygons to determine in what region the vehicle site (N_i), and the second is the searching process inside the N^{th} polygon, to search about the S_j street.

III. Building Mobile Database

As mentioned before, this algorithm depends on determining the vehicle location with minimum possible time, so lowest computation process must apply. Therefore, this concept must apply when building streets database, by applying the following steps:

1. Distribute the intended region into polygons as mentioned before, each with the lowest and highest GPS points. For example, in the model applied for this system, we consider that "West-Bank" is the working region, and divided it into polygons, one of these polygons is "Hebron" polygon, with GPS points: [31°27'56.10"N , 35° 4'52.45"E] ~ [31°36'34.10"N , 35° 7'11.61"E], as shown below.

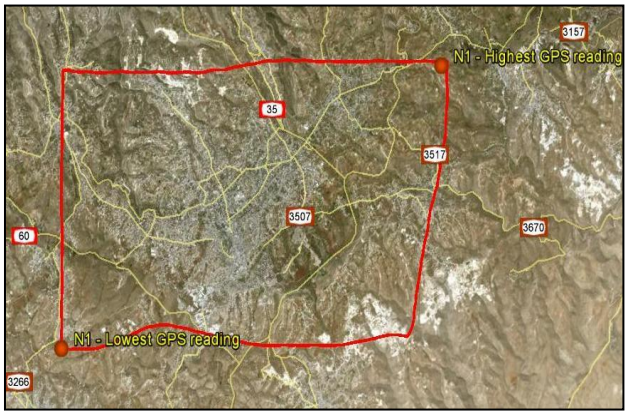

Figure 3: *Hebron Polygon*

2. Inside Each polygon, arise streets polygons, each one determining by two diagonal GPS points as shown in the following figure, and for each street Expand them by (5m) on each direction (East, West, South, North), in order to minimize the probability of error, as shown in the following figure.

Figure 4: *Street polygons samples*

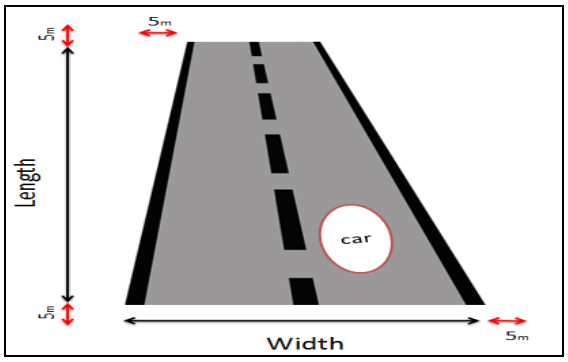

Figure 5: *Expand Street by 5m in each direction*

3. From the principle of GPS coordination, the GPS point value will increase or decrease as the direction of moving. Benefit from this principle; expand the diagonal line from the start GPS-stored data, to the end on the opposite side, as shown in figure 4.20, in order to contain the max. GPS coordinates in each sub-polygon

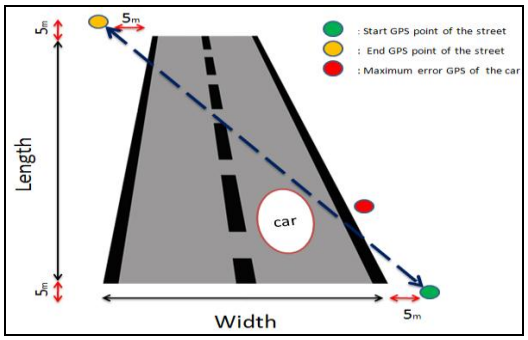

Figure 6: *Expanded street with start- end GPS points moreover, the max GPS error point of the car*

In our model, a part of sub-polygon appears in the following figure:

Figure 7: *Hebron Sub-polygon*

4. In order to make a comparison, choose latitude coordinates arranged from [Max latitude -Min latitude] for each diagonal sub-polygon, where Max. and Min. latitude represent the start and end GPS of each sub-polygon, that will be stored in the database Then the system compares the latitude coordinate of vehicle location with these latitudes range stored in database.

5. The problem here, that there will be many streets have the same latitude coordinates .To solve this problem: the system will compare the vehicle GPS location (latitude and longitude) with the street range latitude and longitude, never to get two street having the same range of latitude and longitude.

6. Built in database of each main or sub polygon will be presented by 4 points (two for latitude and the same for longitude).

Table 1: *Sample of streets database*

SPEED LIMIT	ID	END POINT (Longitude)	FISRST POINT (Longitude)	END POINT (Latitude)	FIRST POINT (Latitude)
50	S1	350602	350600	313325	313320
60	S2	350601	350553	313318	313310

7. GPS coordinates form conversion from (degrees) to (XY) form will be used here, in order to simplify the comparing process, using the conversion equations.

IV. BREAKING ROUTES INTO PIECES

When the streets are too long, or do not have a linear shape, these streets will be divided into pieces, each one has 4 points.

Query will be used to determine the car location by holding the current GPS data, and then compare it with the database.

The following flow-charts show the Map-Matching algorithm:

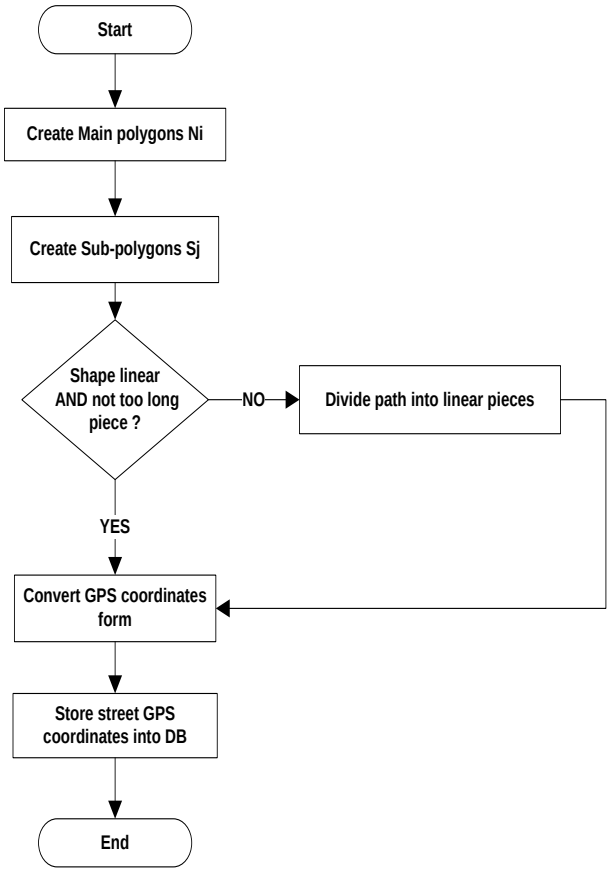

Figure 8: *Breaking Route into pieces flowchart*

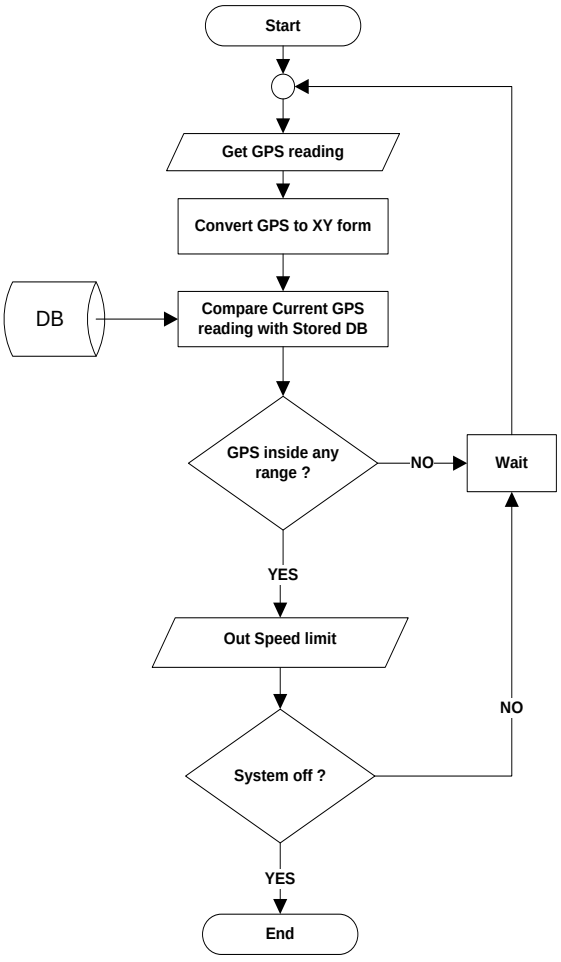

Figure 9: *Map-Matching algorithm flowchart*

RESULTS

After perform three tests using this algorithm, we have the following results:

1. Decrease the GPS accuracy from 15m error in average to lower than 5m.
2. Localize the vehicles in any street inside or outside the Palestinian cities.
3. Enhance the relation between the vehicle speed and GPS accuracy, as the following results:

Table 1: Tests results of the algorithm

Test #	Average Vehicle speed(kmph)	GPS error (m)
Test 1	20	1.5
Test 2	50	3.5
Test 3	70	5-7

CONCLUSION

After perform the location determination algorithm in the ITWAVs project, we conclude that the GPS error solved depends in the previous criteria to minimize the previous algorithm.

REFERENCES

[1] M. Ochieng, "MAP-MATCHING IN COMPLEX URBAN ROAD NETWORKS," Centre for Transport Studies, Department of Civil and Environmental Engineering, London, 2006.

[2] J. Marchal, "Efficient map-matching of large GPS data sets – Tests on a speed monitoring experiment in Zurich," Institut for Verkehrsplanung, 2004.

[3] L. Friese, "Updating the Spatial Alignment Attributes of Digital Maps Using GPS Points," Princeton University, May, 2005.

Figure 10: *Map-Matching algorithm flowchart*