Research on Map Matching Algorithm Based on Nine-rectangle Grid

Li Cai[1,a], Bingyu Zhu[2,b]

[1]School of Software, Yunnan University, Kunming 650091, China

[2]School of Information Science and Engineering, Yunnan University, Kunming 650091, China

[a]caili@ynu.edu.cn, [b]byunix@163.com

Keywords：Map Matching, Nine-rectangle Grid, Floating Car, Nearest Point Estimation

Abstract. Map matching is the key problem of floating car technology. On Shenzhen taxi GPS data pre-processing of the premise, using “Divide and Conquer”methodology for reference, the map matching algorithm based on nine palaces has been proposed, which can overcome the problem of low computing efficiency in common map-matching algorithm, and has better matching accuracy and efficiency. In addition, the algorithm has been verified and evaluated. Experiment results show that the map matching algorithm has a high accuracy for complex road conditions and has a good adaptability, meeting the requirements of floating car map matching.

Introduction

Floating car system is a new traffic information collection technology accompanied by intelligent transportation systems (ITS) application development in recent years. Based on floating car data (FCD), traffic congestion can be identified, travel times can be calculated, and traffic reports can be rapidly generated [1-2]. The key step of processing these data is map matching. Any global satellite positioning system (GNSS) like GPS, GALILEO or GLONASS, which is used for real-time positioning includes inevitable errors. Additionally, there's no (and will be no) absolutely accurate digital street map representing the real world. Therefore, in order to show a vehicle's position on a digital street map you have to match ("snap") its coordinates (received from a global positioning system) on top of a vector based street map. This challenging process is called "Map Matching".

Because FCD are very massive, map-matching algorithm requires high accuracy and quick processing speed. In this paper, by the premise of preprocessing Shenzhen taxi GPS trajectory, using “Divide and Conquer" methodology for reference, we addresses a map matching algorithm based on Nine-rectangle Grid, which can effectively improve low computing efficiency of the traditional algorithms.

Road Grid Generation

- Principles of Road Grid Generation

Since collection objects of floating car are the ten thousand vehicles, capture interval from 5 seconds to 30 seconds, the floating data collected from the vehicles are very massive, so the establishment of an efficient road topology is very necessary.

Road grid generation means that we will divide all nodes and road sections of map into equal grids, whose essence is “*Divide and Conquer*" methodology, the whole city map is split into a plenty of small maps for processing [5]. In the following process, map matching will greatly reduce the search range and area. Grid shapes include square, rectangular and hexagonal, their

* This work is supported by Yunnan Province Education Department Scientific Research Fund Grant No. 2012Y498

classification is based on increment of latitude and longitude and distance.

[4] proposed a divided road grid solution: according to in direction of latitude and longitude using a certain interval L, the road network was divided into many uniform grid blocks from top to bottom, from left to right. Assumed the road network is generated into M × N grid, denoted (M × N), where M and N respectively represented the row and column block numbers. The grid line number is from 0 to M-1and the column number is from 0 to N-1; its map matching algorithm used linked list to store grid information. In our paper, we improved this method of the paper [4] and proposed a new classification scheme. Detailed method is that: using 0.0005 latitude and longitude increments, dividing the whole Shenzhen digital map into a length of 51.28 m - 51.42 m, a width of 55.65 m rectangular grid, which will provide the basic data for the following algorithm. Meanwhile, using the characteristics of HashMap key values being unique, computing grid key values through the relevant algorithm and quickly generating key values of the corresponding nodes and road lines, then the information of node ID and road segment ID are stored into the HashMap for indexing, which will effectively reduce grid query time.

- Calculation of Grid Key

Fig. 1 shows generation method of city road grid and node key value. Divided grids form a semi-enclosed package standard rectangular. If the node is located in one grid's solid line, then it belongs to this grid; otherwise, if it is located in dashed edge, then it belongs to adjacent grid. For example, the common edge of the grid 1 and grid 3 belongs to the grid 3, and the common edge of grid 1 and grid 2 belongs to the grid 2. The key values of ID 1, 2, 3 grids in data structure of HashMap respectively represents 0.0000-0.0000, 0.0005-0.0000 and 0.0000-0.0005, that means the grid key value is latitude and longitude coordinates of this grid's left corner. The reason for selecting 0.0005 increments lies in controlling the 18 place after the decimal point does not generate key drift. In Fig. 1, suppose a node's latitude and longitude is (0.0001, 0.0004), longitude 0.0001 belongs to 0.0000-0.0005 and latitude 0.0004 belongs to 0.0000-0.0005, so we can infer the node's key is 0.0000:0.0000.

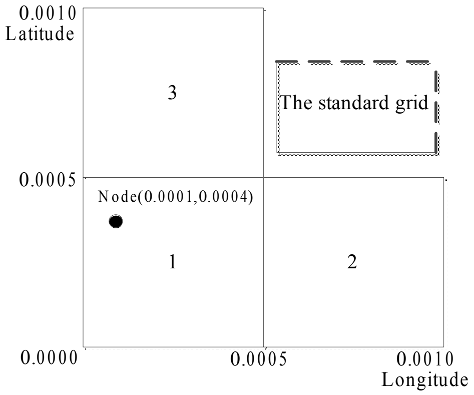


Fig. 1 Mesh Generation and Node Key Identification

The following is the calculation formula of node grid based on its latitude (abbreviated lat), the calculation of longitude is similar to these formula.

$$\mathbf{latTemp} = (\mathbf{lat} * \mathbf{100000})/\mathbf{10}. \tag{1}$$

$$\mathbf{remainder} = \mathbf{latTemp} \, \% \mathbf{latPlusInt} \,. \tag{2}$$

$$\mathbf{result} = (\mathbf{latTemp} - \mathbf{remainder})/\mathbf{10000} \,. \tag{3}$$

The first step, we compute absolute value of the node's latitude; the second step, using Eq. 1, 2 (the remainder latTemp number of 5), and 3 is calculated in order. The purpose of the Eq. 1 does not directly multiply 10,000 is to smooth carry after complementary operations for the data's mantissa with multiple 9, when its processing results need to compulsorily conversion to integer type. LatPlusInt in Eq. 2 represents 5. The key of entire grid key calculation method is complementary operations and acquire the left section of the GPS point by the Eq. 3 that means GPS grid key.

- Calculation Algorithm of Road Section in Grid

After road network grids generation, we need to calculate the road section belongs to which grid and handle the issue of the section span grids. In Fig. 2, almost the whole road section is curved, respectively crossing the grid 1, 2, 4. Inside road section may be divided into multiple lines. In order to simplify the processing, in this paper, we use infinite points to replace internal line, based on these points, and calculate grid key of the straight line, furthermore calculate grid key of the road section. Detailed mathematical formulas are shown below.

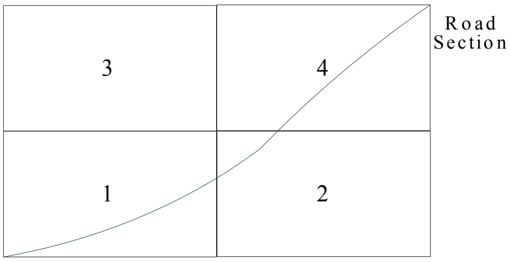


Fig. 2 Road Section Computing in Grid

$$\Delta\mathbf{lon} = \mathbf{lon2} - \mathbf{lon1}; \Delta\mathbf{lat} = \mathbf{lat2} - \mathbf{lat1}. \quad (4)$$

$$\mathbf{divisor} = \mathbf{max}\left(\frac{|\Delta\mathbf{lon}|}{\mathbf{mylonPlus}}, \frac{|\Delta\mathbf{lat}|}{\mathbf{mylatPlus}}\right). \quad (5)$$

$$\mathbf{for\ i = 1\ to\ divisor - 1}\{\mathbf{lon3} = \mathbf{lon1} + \mathbf{i} * \left(\frac{\Delta\mathbf{lon}}{\mathbf{divisor}}\right); \mathbf{lat3} = \mathbf{lat1} + \mathbf{i} * \left(\frac{\Delta\mathbf{lat}}{\mathbf{divisor}}\right)\} \quad (6)$$

Assuming a linear with start node and end node, their latitude and longitude coordinates of GPS points are respectively lon1, lat1, lon2 and lat2, the increment of latitude (latPlus) and longitude increment (lonPlus) are equal to 0.0005, and dividing frequency (divider) is 10000. According to the Eq. 4 and 5 to calculate the actual cycle number (divisor), then using the loop statement to compute the latitude and longitude of numerous GPS points; finally make these GPS points replace these line sections, and get their grid values. Because calculated multiple points maybe exist duplicate grid values, we need to execute duplicate removal processing. However, even with a high equal division number, we cannot avoid the problem of grid omission. Such as in Fig. 2, if equal division number being too big will omit grid 2, the corresponding solution is described in the Nine-rectangle Grid.

- Experimental results of road grid algorithm

After corresponding processing, we obtained Shenzhen city road grid table shown in Table 1. The first column of table 1 indicates HashMap key value, the second column represents the node number, and the third column indicates the node belongs to road section number and the corresponding linear number. If we compute road grid with nodes, the computation time of a ten million GPS points is 62 seconds, and 100 million GPS point calculation takes 593 seconds. If we compute grid with sections, according to Shenzhen 2012 electronic map with a total of 95,392 road sections and 363,274 line segments, using 10000 equal divisions, the time consuming of this algorithm is 173.06 seconds.

Table 1 Road Grids Table of Shenzhen

Column	Hash MapKey	NodeIds	RoadSection&StraightLineIds
Sample	22.5845:114.0530	39509:39510:	989-7:992-0:993-0:993-1:994-2:994-3:

The Algorithms of Nine-rectangle Grid

- Nine-rectangle Grid

Nine-rectangle Grid is an extended grid for common grid shown in Fig. 3. From top left to bottom right of the grid is divided into a total of nine grids, each grid's HashMap key has been identified in the figure. The total length of the Nine-rectangle Grid is 3 × 51.28 m to 3 × 351.42 m, a width of 3 × 55.65 m, which is approximated as a square with 150 m side length. If the range of Nine-rectangle Grid does not satisfy the requirements of the following algorithm, whose range can also be extended to 25 grids (side length of 250 meters), 49 grids (side length of 350 meters), 81 grids (side length of 450 m) and so on.

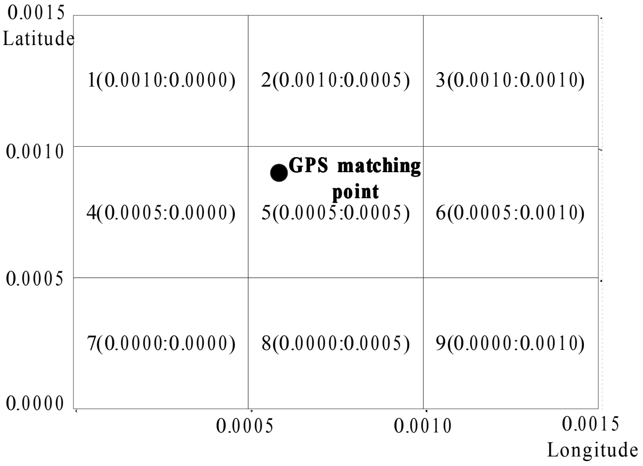


Fig. 3 Nine-rectangle Grid

During the stage of map matching, after using the Nine-rectangle Grid, depending on the input GPS matching point, we can calculate its grid key value. For example, the matching point in Fig. 3, which belongs to the grid 5, only executing 8 times addition or subtraction operation for 0.0005, we can quickly get eight adjacent grids(1,2,3,4,6,7,8,9) ' key values.

For the above mentioned grid omission problems, Nine-rectangle Grid can guarantee the missing grid with a minimum 50 m scope will appear in the neighboring grids. For example, in Fig. 2, the road sections omit the grid 2, but it will appear in the adjacent grids 1 and 4.

In our experiment, all grids key had been circular calculated to acquire corresponding nodes and straight lines of Nine-rectangle Grid. The results are: on average, having 473,902 / 164,894 = 2.874 nodes, having 4,012,629 / 164,894 = 24.33 line (existing duplication), and having 2,845,201 / 164,894 = 17.25 straight (removing duplication).

Improved the nearest point estimation map matching algorithm based on Nine-rectangle Grid

- Map matching principle

In the nearest point estimation map-matching algorithm [3], a GPS point p makes vertical projection operation for all line segments within the Nine-rectangle Grid in where it is, shown in

Fig. 4. Suppose there are three linear segments L1, L2, L3, P respectively does vertical projection for the three linear segments and get p1, p2 and p3 three projection points. Since the foot point of L3 is located in the extended line of L3, in this case, we put p3 be positioned to L3' vertex. Then, respectively compared the distance of P to the other three points p1, p2, p3, and choose the line segment which has the smallest distance in all candidate line segments as match segment, and get the specific point on this segment.

Nearest point estimation map-matching model is shown in Fig. 5. Points P1, P2 stand for the nodes of line segment, the coordinates of P1 are (p1x, p1y) and the ones of P2 are (p2x, p2y). P represents the matching GPS point needed; its coordinates are (px, py). P3 represents the vertical projection point of P projected on the P1P2 straight line segment; the coordinates of P3 are (p3x, p3y). Here we use mathematical formulas derived P3 coordinates.

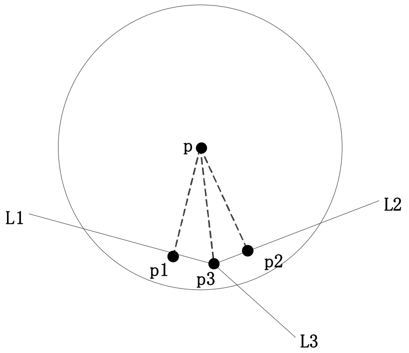


Fig. 4 Map Matching Based on Nearest Point Estimation

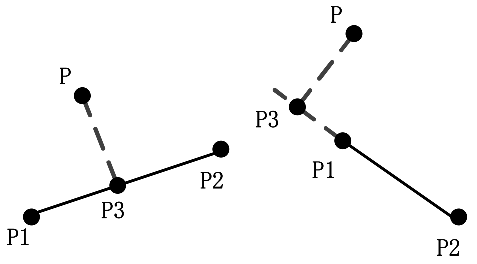


Fig. 5 Mathematical Model of Map Matching based on Nearest Point Estimation

Straight slope k generated by p1 and p2 can be obtained by Eq. (7). After acquiring slope k, p3x and p3y can be obtained by the following Eq. (8) and (9).

$$\mathbf{k} = \frac{p2y - p1y}{p2x - p1x} \tag{7}$$

$$p3x = \frac{k * (py - p1y) + (px - p1x)}{k * k + 1} + p1x \tag{8}$$

$$p3y = k * \frac{k * (py - p1y) + (px - p1x)}{k * k + 1} + p1y \tag{9}$$

We can infer detailed location of P on a straight line once we know the coordinates of p3x and p3y. Here discusses two cases:

1) If point P3 is located on the line segment P1P2, then it is the positioning point.

2) If point P3 is positioned in the extension line of the P1P2, we need to determine the specific location based on latitude range of the line segment P1P2.

In our algorithm, firstly we need to determine the smallest latitude (abbreviated minLat) and the largest latitude (abbreviated maxLat). Secondly we judge whether the latitude of P3 is within this range. If not, then P3 is positioned in the extension line of P1P2. In this case, we need to determine the minimum distance of P to point P1 and P2, the node with the shortest distance is returned as the result.

In this paper, the improved nearest point estimation map-matching algorithm is as follows:

1) Use Nine-rectangle Grid to limit matching range, which means converting the large map to the small map for matching.

2) Improve the returned results of the nearest point estimation map-matching algorithms. If the projected foot point on the line, then return to the straight-line segment ID and the GPS of matching point; If the projected foot point on the extended line, then compare the distance of P to start point and end point of the line, and return the GPS point with minimum distance and the line segment ID.

3) Determine whether the matching point is the start point or end point of the line section, if yes, correct matching line segment ID as the node ID.

4) Return the results were stored in an ArrayList array sorted by ascending distance to facilitate future correction.

Detailed algorithm flow chart is shown in Fig. 6. Improved nearest point map matching algorithm is simple and effective. When we use Nine-rectangle Grid to complete scope limitation and execute this algorithm in the 0.0005 increments grid, only take 17.25 times of for loop in average, we can get the results, and the system load is very light. While this algorithm is very dependent on the accuracy of the GPS latitude and longitude, the accuracy of matching results is determined by the GPS device performance.

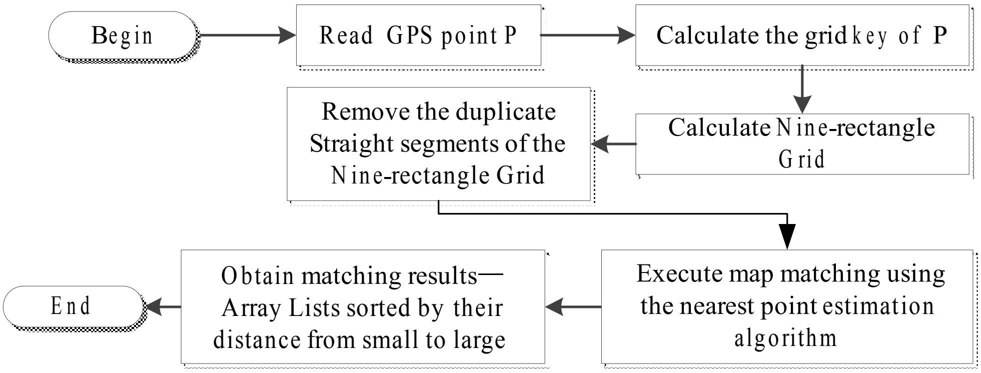

Begin
Read GPS point P
Calculate the grid key of P
Calculate Nine-rectangle Grid
Remove the duplicate Straight segments of the Nine-rectangle Grid
Execute map matching using the nearest point estimation algorithm
Obtain matching results—Array Lists sorted by their distance from small to large
End

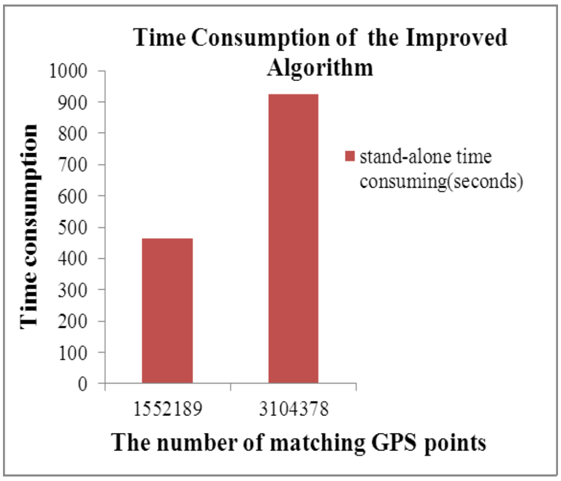
Fig. 6 Improved Map-matching Algorithm Flowchart based on Nearest Point Estimation

- Analysis of map matching results

This experimental map comes from Shenzhen city digital map (1:10000 scale) and the coordinate system is WGS84 coordinate system. This electronic map has a total of 95,392 road sections, 363,274 line segments. The algorithm runs in stand-alone mode, using 0.0005 increments as the grid standard. After a preliminary screening, processed time of reading 1,552,189 GPS points for map matching is 464.703 seconds; the one of reading 3,104,378 GPS points for matching is 924.406 seconds, the results is shown in Fig. 7.

Next, we compared our improved algorithm with those algorithms in [6]. [6] used the Chongqing city bus map and processed 16,163 experimental data using the nearest point estimation map-matching algorithm(abbreviated Algorithm 1) and using map-matching algorithm based on map pretreatment (abbreviated Algorithm 2). Results of efficiency comparison between algorithm 1 and algorithm 2 showed that the former's processing time is 61.953005 s and the latter's processing time is 47.108995 s; efficiency of algorithm 2 is improved by 23.96%. In our algorithm, 16,163 experiment data were randomly selected from Shenzhen Electronic map, processing time by improved algorithm based on Nine-rectangle Grid is 5.093s, compared to algorithm (2), our algorithm gets 7-11 times performance improvement. Contrast diagram was shown in Fig. 8.

Although the data resources of our map matching algorithm and other two algorithms in [6] are not inconsistent, the electronic map of Shenzhen is more complex than Chongqing city bus electronic map, so our algorithm's performance improvement is worthy of recognition.



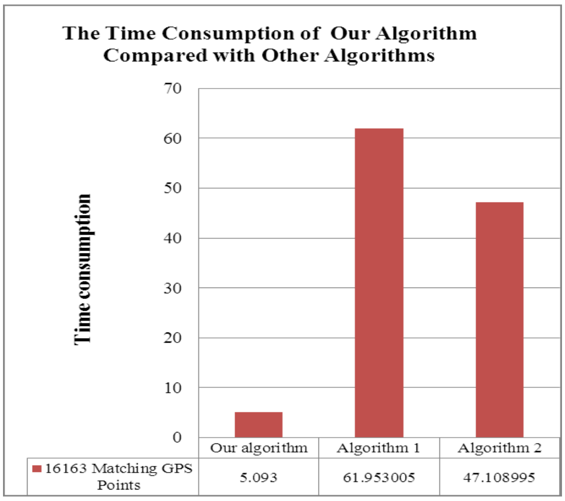


Fig. 7 Time Consumption of improved nearest point estimation map-matching algorithm

Fig. 8 our matching algorithm compared with other algorithms

Conclusion

Quickly Map matching for massive data generated from floating car system of is a key issue in this field. GPS data error and electronic map error also increase the complexity and difficulty of map matching. This paper improved traditional nearest point estimation map matching algorithm and proposed a new map-matching algorithm based on Nine-rectangle Grid. At the same time, we completed the matching experiments by the Shenzhen City taxi actual GPS data and the road network data. Results show that our algorithm has high efficient and practical features, which is better than traditional used algorithms in matching time and performance.

References

[1] Liu Pei. The Research of Map Matching Algorithm Based on Floating Car Data [M]. Bei Jing: Beijing Jiaotong University, 2007.

[2] Zhu Tongyu, Guo Shengmin. Study on The Technology of Floating Car Information Processing Technology [J]. Journal of Image and Graphics, 2009,14(7):1230-1237.

[3] Li Qingquan, Huang Lian. A Map Matching Algorithm of GPS Trajectories Data [J]. Acta Geodaetica et Cartographica Sinica, 2010,39(2):207-211.

[4] Zhang Wei, Xu Jianmin, Lin Mianfeng. Map Matching Algorithm of Large Scale Probe Vehicle Data[J]. Journal of Transportation Systems Engineering and Information Technology, 2007,7(2):40-45.

[5] Li Jie, Zhang Wendong, Zhang Xi. An Algorithm on Navigation Map Matching [J]. Journal of Computer-Aided Design & Computer Graphics, 2007,19(2):268-271.

[6] Xiao Feng. Study on the GIS and GPS Data Pretreatment Key Technology for Traffic Condition Detection [D].Chong Qing: Chongqin University,2008.