Application of The KNN Algorithm Based on KD Tree in Intelligent Transportation System

Guangyi Zhang and Fangzhen Li
Computer Science and Technology College
Shandong University of Finance and Economics
Jinan, Shandong Province, China
zhguangyi42@sina.com

Abstract—**The intelligent transportation system has demonstrated its strong advantages in solving the urban transport problem. One of its important roles is able to reflect the traffic conditions timely through the floating car. The key problem is to find out the candidate road sections from the vast road network quickly. Then we make the floating car match to the corresponding road by the map-matching algorithm. So we can get the real location of the floating car on the map. Every floating car needs to select candidate road sections from the whole road network, so the computing time is an important factor in affecting the real-time performance of the whole system. The commonly used method is to build an ellipse according to the probability criterion. It needs to determine the size of the ellipse, which is based on the statistic theory. It also needs to find these road sections which are in the ellipse from the whole road network. The whole process is complicated and time-consuming. Therefore, this paper proposes the k-nearest neighbors algorithm based on KD tree to get the candidate road sections.**

Keywords-KD tree; map-matching; k-nearest neighbors algorithm; floating car

I. INTRODUCTION

Along with the development of the Chinese economy, the number of motor vehicles is increasing year by year, which makes the city become more and more crowded. The traditional traffic management is unable to deal with this condition. So the intelligent transportation system is put forward to solve this condition. So if the intelligent transportation system needs to reflect the traffic condition[1] of the city timely and accurately, especially when the scale of the road network is very large; we need a good algorithm to find out the candidate road sections from the whole road network. Therefore, in order to find the candidate road sections quickly, this paper proposes the k-nearest neighbors algorithm which is based on KD tree and then we use the map-matching algorithm to get the road section which has the best matching degree. As a result, the intelligent transportation system can reflect the traffic condition of a city timely and accurately.

II. THE PRELIMINARY KNOWLEDGE

A. Floating Car

The floating car is a way to get traffic information and is widely used in the intelligent transportation system. It mainly refers to the vehicle equipped with the global positioning system, which is driving on the road in the city. In order to associate the data of floating car with the urban road, we use the map-matching algorithm to get its real location on the map. We can obtain the traffic condition of the road, which the floating car is driving on. If there are enough floating cars in the city and these data of floating cars are transmitted to the information center via wireless communication system timely, the information center processes the data and then obtains the traffic condition of the whole city.

B. The Concept of Map-matching

Map-matching is a technique combining electronic map with location information to obtain the real location[2] of vehicles on the map. If the location information of floating car and the data of spatial road network are accurate enough, we will just make the floating car match to the corresponding road through the data received from the GPS. However, even with the help of the fairly good sensor, there is still existing the problem that the precision[3] is not good. The electronic map can't achieve the satisfactory degree of accuracy. Therefore, we need integrate the location information of floating car and the data of electronic map. After revising the inaccurate location information by the map-matching algorithm, we can obtain the real location of floating car on the map.

C. The Commonly Used Map-matching Algorithm

The commonly used map-matching algorithm is the map-matching of single point. With the moving of the floating car, the floating car will output the GPS data every few seconds. With the help of these continuous data, we use the map-matching to get the running track of floating car. The map-matching of single point algorithm has many different ways to achieve. Each of these methods has advantages and disadvantages[4][5][6].

The map-matching algorithm based on the point-to-point means that the road is divided into several sections. It takes the endpoints of road sections as the matching object. The GPS positioning point will be matched to the endpoint which is the nearest to it. Without considering the history information of matching, the method only uses the geometry technology and is sensitive to the digital road network.

The map-matching algorithm based on the point-to-line means that we calculate the vertical distance between the GPS positioning point and the road sections .We choose the nearest road to match and take the projection point as the best matching point. The method only uses the geometry technology too. But the real-time performance is very good.

The map-matching algorithm based on the line-to-line means comparing the trace of floating car with the shape of the road. The accuracy of matching depends on the method of

978-1-4799-3279-5 /14/$31.00 ©2014 IEEE

calculating distance between the trace of floating car and the road. One wrong matching may cause more mistakes.

Besides, there are more advanced map-matching algorithm, such as Kalman Filter, The Extended Calman Filter[7], Dempster-Shafer Evidence Theory[8], Particle Filter[9], Fuzzy Logic[10][11][12], Bayesian Inference[13] and so on. Due to the use of advanced and new technology, it makes the matching precision greatly improved. However, these new technologies often need to build a model or study for a long time to obtain the corresponding parameters. The complexity of these algorithms is high. They are not suitable for the intelligent transportation system which needs the better real-time performance.

III. THE KNN ALGORITHM BASED ON KD TREE

A. The Introduction of KNN Algorithm

First, let us introduce the KNN algorithm to you. The full name of KNN is k-nearest neighbors. For the given point, we can simply think that the number of its nearest neighbors is k. In order to figure out whether these points are the nearest neighbors, we usually take the Euclidean distance as the criterion.

Whether the matching of feature point or the image retrieval is the same problem essentially, they can be classified as the problem which is to find out several nearest neighbors quickly in the high-dimensional space. In order to find several nearest neighbors, the easiest way is to linearly scan. We calculate the distance between the given point and the points in the sample set in turn, and then find out which points are the nearest to the given point. This method is simple. But when the scale of sample set is large, it will take lots of time to calculate. Another way is to build the data index. Generally speaking, data will be presented to be the cluster shape. Therefore, we expect to build the data index tree. The basic idea is to divide the searching space into some small space. The KD tree is this kind of algorithm.

B. The Definition of KD Tree

The KD tree is the multi-dimensional data structure, which is presented by the Bentley[14]. It is widely used to search the key data in the high-dimensional space. The K in the KD is different from the K in the KNN. The K in the KD tree stands for the number of dimension. In fact, the KD tree is a kind of balanced binary tree. The KD tree divides the whole space into specific sections, and then we start to search in the specific space.

C. Illustrate the Building and Searching Of KD Tree

Let us give you a simple example to demonstrate the KD tree algorithm. Assume that there are six two-dimensional points. They are {（2,3）,（5,4）,（9,6）,（4,7）,（8,1）,（7,2）}. The whole space is divided into several small parts. The space is divided into two parts by the bold black line. Then in the two subspaces, the entire space is divided into four parts by the fine lines. Finally, the black dashed lines will further divide the four parts. Fig.1 shows the whole process. After dividing the whole space, we can get the KD tree. Fig.2 shows the KD tree.

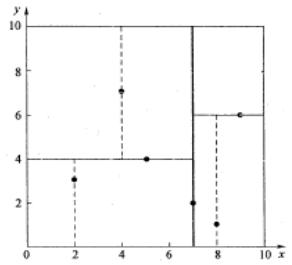

Figure 1. How to divide the data space

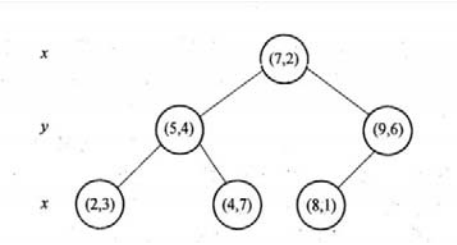

Figure 2. The KD tree

In the KD tree, the core issue is to find out the point which is the nearest to the given point. We give you a simple example to describe the whole process. Please look at Fig.3. The asterisk represents the given point. Its coordinate is (2.1, 3.1). By the binary searching, we can quickly find the nearest point. It is the point (2, 3). But this point may not be the nearest one. In order to find the real nearest point, we need to backtrack. In this example, the order of searching is {(2, 3), (5, 4), (7, 2)}. First, we take the point (2, 3) as the nearest point and calculate the distance between the point (2, 3) and the given point. The distance is 0.1414. Then we back to its father point (5, 4) and figure out whether its other child nodes are nearer to the given point. Last we back to the point (7, 2) and do the same operation as above. Finally we end the whole searching and get the nearest point (2, 3).

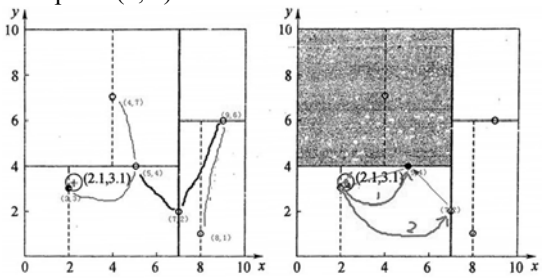

Figure 3. How to find the nearest point

This algorithm does not require the exhaustive search. It need not calculate the distance between the given point and all the sample points in turn. Especially, when the number of point set is large, it can reduce the computing time. And the algorithm can be extended to find out several nearest points easily.

IV. FIND OUT THE CANDIDATE ROAD SECTIONS AND CALCULATE THE MATCHING DEGREE

A. Use the KNN Algorithm to Find the Candidate Road Sections

For the sake of getting the candidate road sections, the commonly used method is to build an ellipse according to the probability criterion. And then we need find out these road sections which are in the ellipse from the whole road network. The whole process is complicated and time-consuming. So, this paper proposes the k-nearest neighbors algorithm which is based on KD tree to get the candidate road sections.

The entire road network would be divided into many short road sections, which is based on the road network database. The specific length of each road section should be determined by the specific circumstances of different cities. The coordinates of all the endpoints can be got by the Google Earth. So the midpoint of each road section can be easily got. The whole process only uses the information of the static road network, so it can be done offline. When the whole system starts to run, we only need to load the file. We use the coordinates of all the midpoints to build the KD tree. And then we find out that which points are the nearest to the floating car by the KD tree. Finding out which road sections the midpoints belong to means that we get the candidate road sections. Although we take the midpoint of each road section as the reference point, there might be some errors. But we can make up for its shortcomings by increasing the number of candidate road sections. The more candidate road sections we get, the better the accuracy of matching will be. But it will make the time of map-matching increased. Therefore, the specific number of the candidate road sections should be determined after many experiments.

B. Find the Best Matching Road Section for the Floating Car

We use the map-matching to find the road section that has the best matching degree. In the second section, we introduce some map-matching algorithms. In this paper, we use two variables to calculate the matching degree. One is the vertical distance between the floating car and the candidate road section. The other one is the angle between the direction of floating car and the direction of candidate road section.

$Q(t)$ stands for the matching degree. t represents the time table. The contribution of the projection distance and direction angle are (1) and (2), respectively.

$$D(p_c, p_r) = \frac{1}{1 + \frac{\|p_c - p_r\|}{\sigma^2}}. \quad (1)$$

$$A(\vec{v}_c, \vec{R}) = \frac{(\vec{v}_c, \vec{R})}{\|\vec{v}_c\| \cdot \|\vec{R}\|} = \cos\theta. \quad (2)$$

p_c: The location of the floating car

p_r: The projection location of the floating car on the candidate road

$\vec{v}_c$: The direction of the floating car

$\vec{R}$: The direction of the candidate road section

θ: The angle between the direction of floating car and the direction of candidate road section

σ: The standard deviation of the position error

We get the matching degree of each candidate road section by the following formula.

$$Q(t) = \alpha D(t) + \beta A(t) \quad . \quad (3)$$

The α and β stand for the weight of projection distance and the direction angle, respectively.

The mean value and variance of the $Q(k)$ are (4) and (5), respectively.

$$E[Q(t)] = \left(\alpha E[D(t)] + \beta E[A(t)]\right) \quad . \quad (4)$$

$$\mathrm{var}[Q(t)] = \left(\alpha^2 \mathrm{var}[D(t)] + \beta^2 \mathrm{var}[A(t)]\right) \quad . \quad (5)$$

The variance of the $Q(t)$ should be little. But the mean value of the $Q(t)$ should be large. Because different road sections have obvious difference.

After many tests, we get the best value of α, β and σ. The numeric value of them are as follows.

$$\alpha = 3, \beta = 2, \sigma = 0.5 \quad . \quad (6)$$

Look at Fig.4. Assume that the floating car has four candidate road sections. N is the position of the floating car. AB, CD, EF and AG represent the candidate road sections. We take the AB road section as the example. In triangle ANB, we use Helen formula to get the vertical distance between the floating car and the candidate road section. We also need calculate the angle between the direction of floating car and the direction of candidate road section by the (2). Then we use (3) to calculate the matching degree of the road section. Finally, we calculate the matching degree of other candidate road sections and choose the road section which has the best matching degree. So we think that the floating car is driving on the road which has the best matching degree.

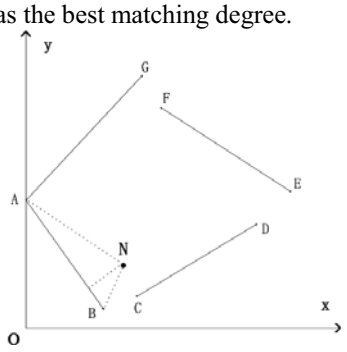

Figure 4. Calculate the matching degree for the floating car

V. THE ANALYSIS OF THE RESULTS

A. The Display of Map-matching Result

In the city where I stay, after many experiments the number of candidate road section is 9.That is, the number of nearest neighbors is 9. The length of every road section is about 1 kilometer. In this case, the accuracy of the whole system can be 90%. As is known to all, even the system is designed by big company, such as Google; the accuracy can't be 100%. In order to show the matching result easily, we only use one floating car to do the experiment. Fig.5 shows the matching result. The same car returns data every 5 seconds. There are

total 124 data as the test data. The dot represents the location of GPS and the pentagram represents the matching point.

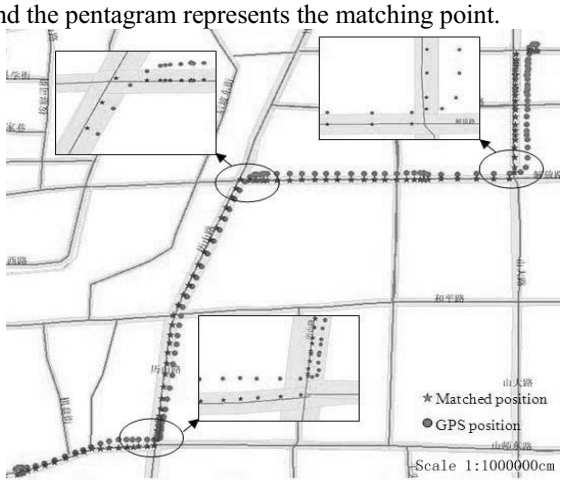

Figure 5. The matching result

B. The Analysis of Time Complexity

We don't mention the time spent on constructing the ellipse according to the probability criterion. It also will take lots of time to find out these road sections which are in the ellipse from the whole road network. If the number of all the road section is N, the time complexity of this method at least will be $O(N\log_2 N)$.

The KD tree is similar to the binary search tree. The difference is that it can provide multi-dimensional retrieval at the same time. Studies have shown that if the KD tree has N nodes, the worst time complexity is $t_{worst} = O(kN^{1-\frac{1}{k}})$.Because all the data are in the two-dimensional space, the k is equal to two. If the points are distributed evenly, the time complexity is $O(\log_2 N)$. So the KD tree algorithm is superior to other algorithms.

C. The Comparison Of Data Processing Time

Look at the table1. When the whole system runs, the time spent on processing data varies in the two kinds of algorithms. These data come from the GPS. Quite obviously, the KNN algorithm based on KD tree is superior to the original algorithm. As we all know, the faster we update the data, the better the accuracy of intelligent transportation system will be. If the intelligent transportation system updates the data every 10 seconds, the algorithm presented in this paper can better meet the real-time performance.

TABLE I. THE COMPARISON OF COMPUTING TIME

The Amount Of Data	The Total Time Required When Use The Original Algorithm (second)	The Total Time Required When Use The KNN Algorithm (second)
10000	0.725	0.513
20000	1.401	0.947
30000	2.172	1.324
40000	2.815	1.723
50000	3.604	2.152

VI. CONCLUSION

This paper focus on explaining the application of the KNN algorithm based on KD tree in the intelligent transportation system. When the scale of the road network or the number of GPS data is very large, this algorithm can save some time for the whole system. In order to improve the accuracy of the matching, we often use the KNN algorithm to find out more candidate road sections from the whole road network. Therefore, it will make the time of matching increased. So in the case of guaranteeing the accuracy of the intelligent transportation system, to reduce the number of candidate road section is the future work to study.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (No.30600121, No.61004003), Shandong Doctoral Foundation, China (No.2007BS09002), and the Natural Science Foundation of Shandong Province, China (No.ZR2009GQ015).

REFERENCES

[1] Kuehne R, Schaefer R-P, Mikat J, Thiessenhusen K-U,Boettger U, Lorkowski S. New approaches for traffic management in metropolitan areas, Proceedings of the IFACCTS Symposiun. Tokyo, Japan, 2003: 42-45Marshall Breeding, Cloud Computing for Libraries, 1rd ed, 2012, pp.68–73.

[2] Christopher E. White, David Bernstein. Alain L. Kornhauser, Some map matching algorithms for personal navigation assistants Transportation Research Part C 8,2000:91-108..

[3] Lv WF, Zhu TY, Wu DD.A heuristic path-estimating algorithm for large-scale real-time traffic information calculating. Science in China Series E: Technological Sciences, 2008 Vol. 51(zkI): 165-174.

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

[5] Bernstein, D., Kornhauser, A., 1996. An introduction to map-matching for personal navigation assistants. intro.pdf Accessed June 19, 2002.

[6] Mohammed A. Quddus, Washington Y. Ochieng, Robert B. Noland. "Current map-matching algorithms for transport applications: State-of-the-art and future research directions", Transportation Research Part C 15. 2007:312–328

[7] Obradovic, D., Lenz, H., Schupfner, M., 2006. Fusion of map and sensor data in a modern car navigation system. Journal of VSLI Signal Processing 45, 112–122.

[8] El Najjar, M.E., Bonnifait, P., 2005. A Road-matching method for precise vehicle localization using Kalman filtering and belief theory. Autonomous Robots 19 (2), 173–191.

[9] Gustafsson, F., Gunnarsson, F., Bergman, N., Forssell, U., Jansson, J., Karlsson, R., Nordlund, P., 2002. Particle filters for positioning, navigation, and tracking. IEEE Transactions on Signal Processing 50, 425–435.

[10] Nassreddine, G., Abdallah, F., Denoeux, T. Map matching algorithm using belief function theory, 11th International Conference on Information Fusion,July 2008:1-8.

[11] Quddus, M.A., 2006. High integrity map-matching algorithms for advanced transport telematics applications, PhD Thesis. Centre for Transport Studies, Imperial College London, UK.

[12] Yongqiang Zhang, Yanyan Gao. A Fuzzy Logic Map Matching Algorithm, Fuzzy Systems and Knowledge Discovery, 2008. FSKD '08. Fifth International Conference on Volume 3, 18-20 Oct. 2008:132-136

[13] Pyo, J., Shin, D., Sung, T., 2001. Development of a map-matching method using the multiple hypothesis technique. IEEE Proceedings on Intelligent Transportation Systems, 23–27.

[14] Bentley JL. and Friedman JH. Data structures for range searching. ACM Computing Surveys, 1979,11(4):397－409.