An Efficient Technique for Distance Computation in Road Networks

Xu Jianqiu[1], Victor Almeida[2], Qin Xiaolin[1]

[1]*Nanjing University of Aeronautics and Astronautics, China*
{xjq_cs, qinxcs}@nuaa.edu.cn
[2]*MaaSai Mobile Technologies, Brazil*
victor.almeida@maasai.com.br

Abstract

With recent advances in wireless communication and position technologies, it became possible to collect and record trajectories of moving objects. Thus, many services in road networks such as nearest neighbor querying and the analysis of moving objects trajectories come into sight which present challenges to the database community. The essential part of the queries behind these services is related to the distance computation in road networks. In this paper, we focus on this problem and propose a method for distance computation in road networks including the access of network connectivity information. We present an algorithm called circle which returns all moving objects within a given distance (radius) to a given network position during some time intervals in the past. We choose an existed index structure, the MON-Tree, to record the framework presented in this paper which is to store and query network connectivity information. The circle operator is used to experimentally evaluate our approach. The results show that the performance of the technique presented in this paper outperforms the only existing index structure in the literature capable to support this kind of query.

1. INTRODUCTION

With the rapid development of wireless communication and positioning technologies, more services can be provided for mobile users, e.g. querying nearest neighbors, analyzing the history information of moving objects and searching the shortest path. At the same time, more and more location data of moving objects can be collected ([8]). Especially, the trajectory can keep record of the positions of moving objects over time which are used for querying analysis and prediction.

As an example, let us analyze the queries "where was the nearest ambulance to the place of the accident at that time" and "find all taxis whose distances are within 2 kilometers to the gas-station between 9:00am and 12:00am last Monday".

In order to answer these queries, one has to analyze the details of the moving objects trajectories such as position and time, as well as the underlying network information to compute network distances, get the results and give some feedback to the query user.

The distance computation is the most important step in processing these kinds of query. The efficiency of the distance function has direct impact over these queries. Recently, there has been some research work about distance computation on Euclidean space ([5], [14], and [17]). However, the distance value there is different from the one in road networks. The former is conventional and straightforward while the latter is much more complex, especially the process of calculating and searching. Specifically, the distance in a network depends on the shortest path connecting the location points and cannot be computed accurately by Euclidean distance metric.

Consider the following example: there are three points in the network shown in the Figure 1: p1, p2, and p3. The Euclidean distance value between p2 and p3 is the length of the line directly connecting them. However, the network distance should be the length of the shortest path along the route from p2 to p3 (we assume the network is bi-directional which means that there is no single route). So, the network distance and the Euclidean distance between p2 and p3 are different.

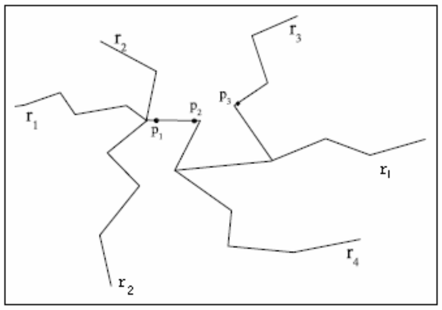


Figure 1 Sample road network

The network model of the most recent research on distance calculating is mainly graph-oriented, e.g. [2, 7, 15], which means the network is divided into nodes and edges. Some of these works only consider static objects or spatial objects without moving objects, e.g. [14, 10, 12]. The graph-oriented modeling method is straightforward and simple. However, as the real road network is quite complex, its representation ability is not well enough. It is not the best one to represent transportation networks.

In this paper, we use the route-oriented network model first presented in [4], where some discussion about the advantage of the route-oriented model over the graph-oriented one can be found. Additionally, an index structure has been presented which is capable to support range queries over moving objects data sets, called MON-Tree ([1]). However, the usage of the MON-Tree is limited to range queries and the network connectivity information is not kept in the index currently, which means the MON-Tree is not able to support the query based on the network distance, the so-called connectivity based query ([16]), such as the examples presented above.

Motivated by these observations, we present an operator called circle with its corresponding algorithm using the MON-Tree index structure that can efficiently answer connectivity-based query for moving objects in road networks. The operator receives a network position p (query point), a radius r, and a time interval [t1, t2] as arguments and returns all the moving objects that have had distance to p less than r during the time interval [t1, t2]. To the best of our knowledge, the problem discussed in this paper has not been well studied by the database community.

We show in an experimental evaluation using the circle operator that our approach is efficient and it outperforms the only existed index structure capable to handle connectivity-based query currently in the literature: the TMIS ([16]).

The rest of the paper is organized as follows: In Section 2, we discuss in detail the difference between the Euclidean distance and the network distance; besides we also survey the existed research work for connectivity-based query in road network. Section 3 proposes our method including the framework for storing network connectivity information using the MON-Tree, the description of the circle operator and its algorithm. Section 4 contains the experimental evaluation of our approach. Finally, Section 5 finishes the paper with some conclusions.

2. RELATED WORK

In this section we present some discussion about the recent work related to ours in order to position our paper in the current state of the literature.

2.1 Distances in Road Network

There has been much research work about network distance computation in spatial databases, but most of them do not refer to the distance value for moving objects in road networks. Papadias et al. in [12] present the most popular queries in spatial network database, namely nearest neighbors, closest pairs and e-distance joins, but it only considers static objects in road networks. Shahabi and Kolahdouzan in [13] propose an embedding technique called RNE for distance computation in road networks which converts all the points to a higher dimensional space and uses the Chessboard metric for distance measurements in the new space. Their approach is based on transforming the road networks into a higher dimensional space and it requires a pre-computation of the distances from all or a group of points to the other points without exploring the index structure. The computation result is not precisely enough after transforming and it is just an approximation of the actual distance which in the worst case is distorted by a factor of O (log *n*). The group of Jensen in [6] concerns active, ordered k-nearest neighbor queries and also presents a data model (2D and graph representation) that serves for the queries. However, their network model is graph-oriented which is different from our network model. Besides, if the number of nodes in network is large, the time and space cost of frequently model transformation may be much high as they choose the method of pre-computing. Mouratidis's work in ([11]) is similar to the problem discussed here, but the network model they have chosen is different from ours and our method using index structure can be directly added into a database system.

2.2 MON-Tree

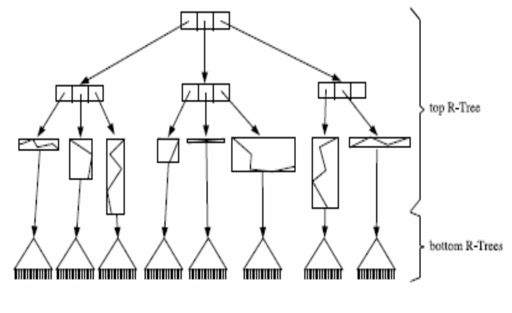


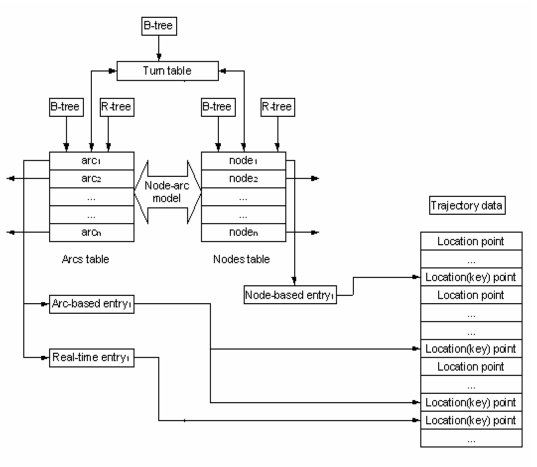
Figure 2 the index structure of the MON-Tree

For distance computation, using an index structure can improve its efficiency because the calculation can be directly processed in the index which leads to low frequency of disk accesses for retrieving the underlying information of the road network.

Since our approach presented in this work uses the MON-Tree as an underlying index structure ([1]), we present a short overview of it. The MON-Tree is an index structure capable to store the trajectories of moving objects in road networks. It is composed by two levels as can be seen in Figure 2 which consists of a 2D R-Tree (the top R-Tree) and a set of 2D R-Trees (the bottom R-Trees). The top level indices the polylines of the network and the bottom level indices the objects movements. More details can be seen in [1]. Currently, the main disadvantage of the MON-Tree is that it does not support network connectivity, not being able to answer connectivity-based query such as distance query.

2.3 TMIS

TMIS which is shown in Figure 3 presented by Li and Lin in [16] is considerably close to our work. To the best of our knowledge, this is the only index structure in the literature capable to handle connectivity-based query for moving objects in road networks. It is not a novel index structure but a group of simple and classical index structures which are linked together through the topology of a network, i.e. B-tree and R-tree. The leaf node in the tree is an entry which consists of the search value and a pointer to the trajectory data of moving objects. There are three kinds of entry: Real-time entry, Arc-based entry, and Node-based entry. They are used to locate the arc and node in the original trajectory data. Node-based entry and Real-time entry are indexed by B-tree and Arc-based entry is indexed by R-tree. A turn table ([8]) is employed to record the connectivity information during the process of building the index structure, which saves all connections between two arcs via the node.



(a) framework

ViaNode	FromArc	ToArc
n_1	a_1	a_2
n_1	a_2	a_1
...	...	...
n_p	a_m	a_n
...	...	...

(b) turntable

Figure 3 TMIS

The structure of TMIS is quite complex which consists of many tables each of them corresponding to an index structure for maintaining the underlying network. For each arc (in graph-oriented model) it builds two tables to store two kinds of entry representing a time instant when an object moves into the arc and a time interval during which an object stays on the arc, respectively. Thus, if the network is large, the structure becomes relatively difficult to maintain. Another drawback of TMIS is the network model used which is graph-based leading to a high number of entries in the structure and lots of update to maintain it.

3. CONNECTIVITY-BASED QUERIES

In this section, we start in Section 3.1 with an overview of our proposed framework which is to store the network connectivity information. In Section 3.2 we present the *circle* operator, in Section 3.3 an algorithm is shown which is able to efficiently calculate the network distance traversing the MON-Tree index structure and then returns the moving objects.

3.1 Framework

For supporting connectivity-based and proximity queries such as the nearest neighbors which are based on the network distance, the network connectivity information should be kept in the index structure in order to reduce the number of disk access. In this section, we present a framework to represent the connectivity information which is stored in the top level of MON-Tree index structure. The framework consists of three components which are junction, adjacency and section defined as follows.

Junction component: *junction* = <*id*, rid_1, $meas_1$, rid_2, $meas_2$, *cc*, *p*, *hv*, *cj*>. A junction description consists of an identifier *id*, two route measures $meas_1$ and $meas_2$ which represent the junction position in each route, a connectivity code *cc*, the spatial point *p*, the Hilbert value *hv*, and a list of all its connected junctions in the adjacency component. These tuples are stored in ascending order of their Hilbert values so that it can roughly guarantee that all the close junctions are physically stored close in disk.

Adjacency component: *adjacency* = <*id*, j_1, j_2, *d*, *MBR*, *s*>. Each pair of connected junctions in the junction component corresponds to a record in the adjacency component which consists of an identifier *id*, two junction identifiers j_1 and j_2 which identify them in the junction component, the distance value *d* between them, their minimum bounding rectangle *MBR* and a pointer *s* to its representation in the section component.

Section component: *section* = <*id*, *curve*>. This component stores the detailed representation (*curve*) of each route segment in the network, identified by *id*.

The three components and their relationship are presented in Figure 4. This sketch corresponds to the network in Figure 1. As shown in the figure, a tuple in the junction component is actually a spatial point where two routes intersect and it records all its connecting nodes. In addition, it is assigned a Hilbert value which tries to keep the junctions close in disk representation, hopefully in the same disk page. Records in the adjacency and section components are stored by the order of connecting junctions in the junction component, because they are always accessed via the junction component.

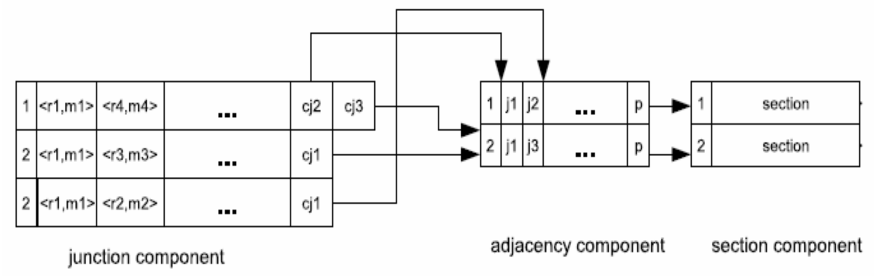


Figure 4 Framework for network connectivity

3.2 Circle Operator

In this section we propose an operator that computes the network distance, called *circle*. The operator has three arguments: a network position (center) q = (*rid, pos*), a distance (radius) r, and a time interval [t_1, t_2]. It returns all the moving objects which are at distance no more than r from q during the interval [t_1, t_2].

The general idea can be seen as follows: the first parameter represents a position in the road network. The operator firstly tries to find all the segments which are within the given distance r to the query point q. It is straightforward if all the segments are in the same route. Otherwise, the processing is relatively complex because the segments in different routes have to be included. When it comes to this case, the connectivity framework presented in Section 3.1 is used to search and find the segments in other routes. So, all the segments whose network distance to the center of circle (q) is less than the circle radius's value have to be retrieved. Afterwards, with the route intervals and the given time interval it returns all the moving objects. As the bottom R-trees in MON-Tree record all the objects movements, the historical information of moving objects can be easily retrieved.

3.3 The Algorithm.

The algorithm for the *circle* operator performs in two steps: firstly, it locates the route where q lies on, and then all the pieces of routes are expected starting from the network position argument q. It finds all the route segments within the given network distance to the query point. This step is done using the infrastructure presented in Section 3.1 (see Figure 4). The query point has to be transformed to its nearest junction point which is an intersection point of routes because it can easily find the connected segments with the connectivity information. This step finishes when the distance r is reached or when there is nothing else to expand. In the end of this step, every piece of the route retrieved becomes a query to the bottom R-Tree in the second step. Secondly, every piece of route is translated into a route interval together with the query time interval [t_1, t_2] and then becomes an argument into the search in the corresponding bottom R-Tree of that route. This step traverses the bottom level R-trees in MON-tree to retrieve all the objects movements during the given time interval.

Algorithm1 Algorithm Circle

```
INPUT: MON-Tree $R$, $q$ = ($rid$, $pos$), $r$, $t_1$, $t_2$
1: $Q \leftarrow \varnothing$
2: $R_i$ = GetRoute ($R$, $q$);
3: starting from $R_i$, find all the route segments which are within
   the given distance $r$ to $q$ and put them to queue $Q$ which records
   route segments. The route segments in $Q$ are stored by the
   increasing order of their network distance to $q$.
4:  for every element $e$ in $Q$ do
5:   Traverse the second level R-trees of MON-Tree to retrieve the
        moving objects in the route interval represented in $e$ during
        the query time interval [$t_1$, $t_2$]
6: end for
```

Let us try to explain the algorithm with an example (Figure 5). The query performs starting from q in route r_1. After the first step of the algorithm, the route intervals [0.7, 0.93] of route r_1 and [0, 0.06] of route r_3 together with the query time interval [t_1, t_2] will be queried in the corresponding R-Trees of route r_1 and r_3, respectively. All moving objects which were in these intervals are returned by the algorithm.

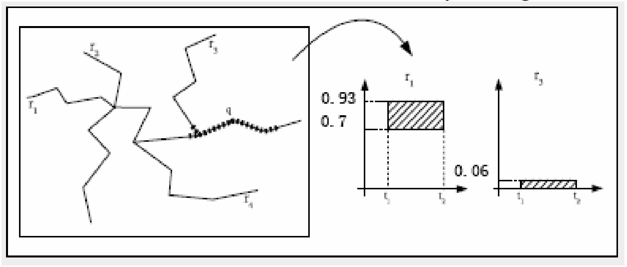


Figure 5 An example of the circle operator

4. EXPERIMENTAL EVALUATION

To examine the performance of our proposal, we performed an experimental evaluation to compare our work with TMIS.

4.1 Environment

For our experiments we used a personal computer IBM ThinkPad R50 with the following configuration: Pentium 1.5 GHz and 512 Mbytes of main memory, running Fedora

Core 2. The implementation was done with C++ and compiled using g++ version 3.3.

4.2 Data Sets and queries

The experiments are performed using a real data set of the road network of Germany downloaded from [19]. The statistics about the data sets can be seen in Table 1.

TABLE 1 STATISTICS ABOUT THE NETWORK DATA SET

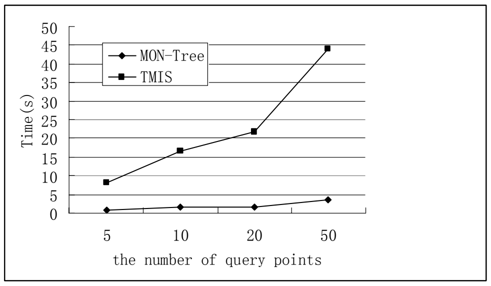
Data Set Model	Germany data
Nodes	19,016
Segment	30,649
Route	995
Arcs	61,298

We generated 13k moving objects in the road network of Germany. For queries, we randomly generated 5, 10, 20, and 50 query points with radii of 100, 200, and 500. The circle representation can be divided into two parts; one is spatial information including the location in the network and the radius value, the other is time information representing the querying time interval. As the network model is different for the two methods, the representation of location in road network is also different. In TMIS, the format is <*arcid*, *nid*, *offset*>, while in our approach the format is <*rid*, *p*>.

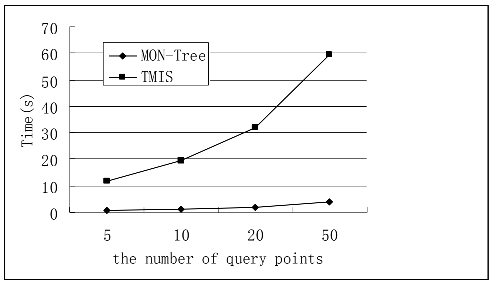
4.3 Results

Using the queries generated as explained above, we recorded the time spent and number of disk accesses by running 10 times for each kind of query. The results are shown in Figure 6 and Figure 7. The query results are moving objects within the distance value (the radius in circle) to the query point during the given time interval.

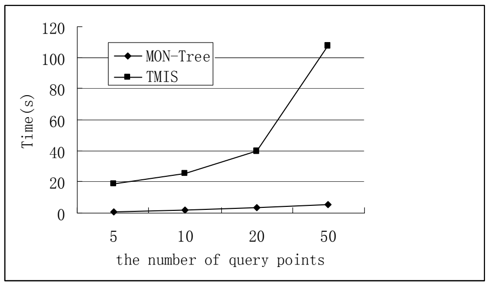
We analyze the main reason for the advantage of the MON-Tree is the access method for network connectivity. All the connectivity information is recorded in the junction component and the records are saved by Hilbert value leading to less time of traversing the structure. Besides, it is not necessary to record the connectivity in the same route according to route-oriented model. However, in TMIS it lacks an efficient data access method among nodes table, arcs table, and turn table. Although it creates B-tree indices on record identifiers for the three tables, there is no direct connection between each two of them. Moreover, as one node in nodes table may have several corresponding records in turn table, it has to traverse all the records in the turn table whose node equals to the one in nodes table. This leads to more disk accesses and consequently querying time. It is also seen from the figures that the advantage of our approach is more significant when the number of queries is larger. Moreover, the increase of the number of queries (query points) has a linear effect in our approach, which is not true for the TMIS.



(a) radius = 100



(b) radius = 200



(c) radius = 500

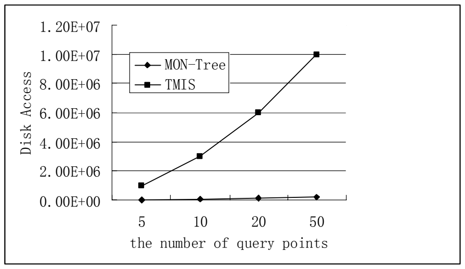
Figure 6 Query response time

5. CONCLUSION

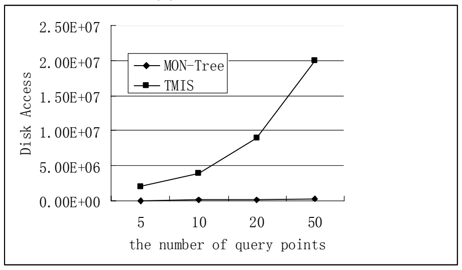
In this paper, we propose an efficient technique for distance computation in road networks which supports network connectivity information and connectivity-based query, such as nearest neighbors and the shortest path.

We have experimentally evaluated our method against the TMIS which is, to the best of our knowledge, the only index structure that supports connectivity-based query. Our tests are based on the *circle* operator, which we also provide an efficient algorithm. The experimental and some analytical results have demonstrated that our approach outperformed the TMIS, especially when the number of queries is large.

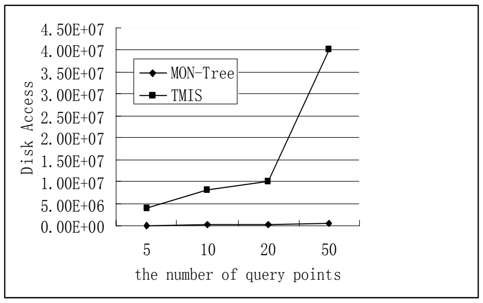
Our plans for future work are to add our work to Secondo ([18]) which is an extensible database system. Thus, the database system can efficiently support various queries which are closely related to road network distance, such as nearest neighbors, shortest path, and closest pair queries.



(a) radius = 100



(b) radius = 200



(c) radius = 500

Figure 7 Query disk accesses

ACKNOWLEDGMENTS

The authors would like to thank Prof. Man Lung Yiu in Aalborg University for providing the library of E00 file converting. This work was supported by grant China National Natural Science Funding (69973022).

6. REFERENCES

[1]Almeida, V. T. , Güting, R. H. Indexing the Trajectories of Moving Objects in Networks., GeoInformatica 9:1, 33-60, 2005.

[2]Frentzos, E., Gratsias, K., Pelekis, N. and Theodoridis, Y., Nearest Neighbor Search on Moving Object Trajectories. Proc. 9th International Symposium on Spatial and Temporal Databases (SSTD'05), Angra dos Reis, Brazil, August 2005.

[3]Guttman, A. R-trees: a Dynamic Index Structure for Spatial Searching. In SIGMOD, 1984.

[4]Güting, R.H., Almeida, V.T. de , and Ding, Z. Modeling and Querying Moving Objects in Networks. Fernuniversität Hagen, Informatik-Report 308, April 2004. VLDB Journal 2005.

[5]Hage,C., Jensen, C.S., T.B. Pedersen, Speicys, L. and Timko, I., Integrated Data Management for Mobile Services in the Real World. Proc. of the 29th Intl. Conf. on Very Large Database (VLDB, Berlin, Germany), 2003, 1019-1030.

[6]Jensen, C.S. , Kolar J., Pedersen, T.B., and Timko, I., Nearest neighbor queries in road networks. In ACM-GIS, 2003.

[7]Jensen, C.S., Pedersen, T.B., Speicys, L. and Timko, I., Data Modeling for Mobile Services in the Real World. In Proc. of the 8th Intl. Symp. on Spatial and Temporal Databases (SSTD), 2003, 1-9.

[8]Jagoe, A., Mobile Location Services: The Definitive Guide (Upper Saddle River, NJ: Prentice-Hall).

[9]Kolahdouzan M.R. and Shahabi C. Voronoi-based k nearest neighbor search for spatial network databases. In Proc. Of 30 th Intl. Conf. on Very Large Daba Bases (VLDB), pages 840-851, 2004.

[10]Miller, H.J. and Shaw, S.L., 2001, Geographic Information Systems for Transportation: Principles and Applications (New York: Oxford University Press).

[11]Mouratidis, K., Yiu, M., Papadias, D., Mamoulis, N. Continuous Nearest Neighbor Monitoring in Road Networks. Proceedings of the Very Large Data Bases Conference (VLDB), pp. 43-54, Seoul, Korea, Sept. 12 - Sept. 15, 2006.

[12]Papadias, D., Zhang, J., Mamoulis, N., Tao, Y. Query Processing in Spatial Network Databases. Proceedings of the Very Large Data Bases Conference (VLDB), pp. 802-813, Berlin, September 9-12, 2003..

[13]Shahabi, C., Kolahdouzan, M.R. and Sharifzadeh, M. A Road Network Embedding Technique for K-Nearest Neighbor Search in Moving Object Databases. In Proc. ACM GIS, pp. 94-100, 2002.

[14]Tao, Y., Papadias, D., Shen, Q. Continuous Nearest Neighbor Search. Proceedings of the Very Large Data Bases Conference (VLDB), pp. 287-298, Hong Kong, August 20-24, 2002.

[15]Vazirgiannis, M., and Wolfson, O., A Spatiotemporal Query Language for Moving Objects on Road Networks. Proc. of the 7th Intl. Symp. on Spatial and Temporal Databases (SSTD), 2001, 20-35.

[16]Xiang Li, Hui Lin: Indexing network-constrained trajectories for connectivity-based queries. International Journal of Geographical Information Science 20(3): 303-328 (2006)

[17]Yiu, M., Mamoulis, N., Papadias, D., Aggregate Nearest Neighbor Queries in Road Networks. IEEE Transactions on Knowledge and Data Engineering (TKDE), 17(6), 820-833, 2005.

[18]SECONDO-AnExtensibleDatabaseSystem. http:www.informatik.femuni-hagen.de/secondo/

[19] http://data.geocomm.com/catalog/c_index.html

Xu Jianqiu obtained his undergraduate diploma in Nanjing University of Aeronautics and Astronautics in Jun. 2005 in computer. Currently, he is a graduate student in College of Information Science and Technology in Nanjing University of Aeronautics and Astronautics. He has won special graduate scholarship in Nanjing University of Aeronautics and Astronautics for the research work he has done. He mainly research about moving objects database, spatial database and spatio-temporal database.