Search of Continuous Nearest Target Objects along Route on Large Hierarchical Road Network

Jun FENG† and Toyohide WATANABE†

† Department of Information Engineering, Graduate School of Engineering, Nagoya University.
Furo-cho, Chikusa-ku, Nagoya, Aichi 464-8603, JAPAN

Abstract The query of continuous nearest target objects for a specific route on road network is of primary interest in geographical information systems. Existing methods for computing continuous nearest neighbor (CNN) are based on straight-line distance between points on the assumption that the whole graph can be stored in main memory. We have proposed a fast method for searching continuous nearest target objects along a route on road network. CNN search based on road network is composed of two steps: one is locating a computation point on the route; and another is searching the nearest object for this computation point based on the shortest path in the road network. We proposed heuristics for generating computation points and the region for searching the shortest path based on the intermediate results. By using our method, all the search processes can be limited to the corresponding search region, and the CNN search process can be greatly accelerated. However, when the whole graph cannot be stored in the main memory, the previous method cannot handle the problem. In this paper, we present a fast method for searching CNN on a large road network. The objective is not to artificially introduce hierarchy into a network model, but rather to investigate a search method based on the networks which bear a natural hierarchy and natural partition.

Key words distributed spatial database, Geographical Information System, path searching, hierarchical road network

1. Introduction

The capability of computing path queries is an essential feature in new database systems for many advanced applications such as navigation systems and Geographical Information Systems (GIS) [1]. For example, one of the primary functionalities in Intelligent Transportation Systems (ITS) [2] is to find routes from the current location of a vehicle to a desired destination with a minimum cost, the cost could represent shortest distance, travel time, etc. In ITS systems, many path requests can be submitted over large transportation network: the whole network is too large to be stored in the main memory at once. This problem referrs to the disk-based search. To solve it, many methods have been proposed [1] [3] [4] [5] from a pre-computation viewpoint. Proper partitions are done to the road network, and some parts of the path length are pre-computed. The cost of the following path search process is related to the number of:

1) nodes and road segments on road network in every partition;
2) boundary nodes in every partition;
3) partitions used in search process.

Comparing to the cost of processing in the main memory, the cost of accessing partitions on disks is expensive. Therefore, this paper proposes a method to decrease the times of disk access by minimizing the search region. In particular this paper investigates a typical query problem over large road network: searching continuous nearest target objects along a predefined route on large road network.

This search retrieves the nearest neighbor (NN) for every point on the route and the result is a set of triples $< point, interval, path >$, such that *interval* is a sub-route, *point* is the nearest target object of all points on the sub-route, and *path* is the shortest path from the sub-route to the *point*.

We propose heuristics to accelerate this search by

1) using the search region, so-called *r-region*, for searching NN for every computation node;
2) using the path search region, so-called *p-region*, for computing shortest path from the computation point to its NN candidates;
3) decreasing *r-region* and *p-region* in the process of computing one pair of nodes: the computation point and one of its NN candidates;
4) sorting boundary nodes inside *p-region*.

The rest of this paper is organized as follows. Section 2 discusses work related to CNN queries, our previous works and shortest path search on large hierarchical road network. Section 3 describes our approaches. Section 4 introduces the algorithms. The conclusion is drawn in Section 5.

2. Related and Previous Work

The issue of this paper referrs to the continuous nearest neighbor search and the path search on large road network.

2.1 Continuous nearest neighbor search

The existing work for CNN search is almost presented from the computational geometry perspective [6] [7] [8]. To the best of our

knowledge, the latest work dealing with CNN queries is given in [6]. CNN search for line segments was effective, based on the straight-line distance between objects. The same effect can be found in [8], which only finds the single NN for the whole line segments.

We have proposed the method [9] to solve the problem based on common situations in GIS: the distance from a point on the route to a target place should be decided by the path length or travel cost of them; and the target objects and the road network are managed in GIS datasets, respectively. Consider the example given in Figure 1, where the specific route is $[S, E]$, and the target object set is $\{t_a, t_b, t_c, t_e, t_f, t_g\}$. The output of the query is $\{< t_e, [S, Q_1], P_1 >, < t_d, [Q_1, Q_2], P_2 >, < t_d, [Q_2, C_2], P_3 >, < t_c, [C_2, Q_3], P_4 >\}$, $< t_c, [Q_3, E], P_5 >\}$: the target object t_e is NN for the interval (subroute) $[S, Q_1]$, and the shortest path from the subroute to t_e is P_1; t_d is NN for the subroute $[Q_1, Q_2]$ with the shortest path P_2; t_d is also NN for the subroute $[Q_2, C_2]$ with the shortest path P_3; t_c is that for the subroutes $[C_2, Q_3]$ and $[Q_3, E]$ with the shortest paths P_4 and P_5, respectively.

By proposing heuristics for selecting computation points (e.g., $\{S, C_1, C_2, C_3, C_4\}$ in the previous example) and initializing NN search region for these computation points, our CNN search finds the target objects with the shortest path length from all the points on the route effectively.

However, when the road network or the sub-network inside the search region is too large to be stored in the main memory, the shortest path search becomes complex and the traditional path search method (e.g., Dijkstra's algorithm) is not applicable.

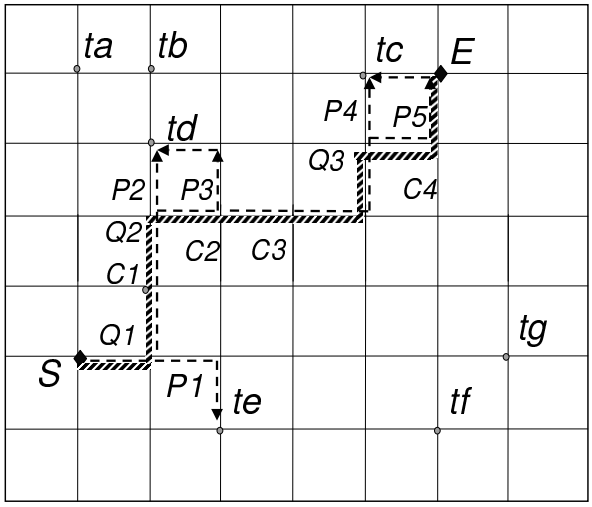

Figure 1 Road network, specific route [*S*, *E*], and target objects.

2.2 Path query on large road network

The shortest path problem on general graphs has already been well-studied: e.g., Dijkstra's algorithm is widely used and actually very fast when the whole road network can be stored in the main memory. Recently, new algorithms have been proposed to address the problem of large dataset: the whole road network cannot be stored in the main memory at once. HEPV approach proposed by [1] is typical. In this method, a graph is first divided into partitions, and then boundary nodes are pushed to the second level to form a super graph. All-pair shortest paths among the boundary nodes in the same partition are also computed, and the corresponding edges are added to the super graph. If the supper graph is still too large, it is divided further into partitions and a third level graph is generated. This process continues until the top-level super graph is small enough to be resident in the main memory. This whole set of super graphs and the ground level graph is called a hierarchical graph. For a path query from the source *s* to the destination *d*, the system first looks for the partitions containing two nodes: partitions *S* and *D*, respectively. The shortest path from *s* to *d* is the concatenation of a shortest path from *s* to a boundary node *u* in *S*, a shortest path from *u* to a boundary node *v* in *D*, and a shortest path from *v* to *d*, i.e.,

$$SP(s,d) = min_{u,v}\{SP(s,u) + SP(u,v) + SP(v,d)\}.$$

The shortest paths within *S* and *D* are easily acquired. To find the shortest distance from *u* to *v*, one must recursively find the shortest path of them in the hierarchical graph. Furthermore, to find out the shortest path from *s* to *d* should compute three parts for all pairs of (*u*,*v*). The search process would result in times of disk access. So, a method to accelerate the process and decrease the number of disks to be accessed is proposed in this paper.

3. Approach

The problem of CNN search which we address in this paper is to find NN for any point along a specific route on the large hierarchical road network. NN is the target object with the shortest path from the point on the route. In the ordinary GIS, all the target objects and road network are managed in index structures, respectively. For example, the target objects are indexed by R-tree [10].

There are two main issues in solving this problem: one is the selection of computation point on the route; and another is the computation of NN for the computation point, including the computation of the shortest path and the selection of NN.

The first problem can be solved by using heuristics, proposed in our previous work [9]. Here, the heuristics for the selection of NN and shortest path search are given as follows:

1) Although the straight-line distance is not completely consistent to the path length on road network, the object with shorter distance from the source has higher possibility to get shorter path length. So, NN search for a computation point begins from selecting a candidate based on the straight-line distance between the computation point and the candidate. With the premise that the whole dataset cannot be stored inside the main memory, the heuristic for decreasing the NN search region (*r-region*) is proposed;

2) The cost of path search is deeply dependent on the search region: the larger the region is, the more road segments and nodes of road would be. The method for initializing the path search region

(*p-region*) for the NN candidates of the computation point and the method for minimizing *p-region* during the computation of shortest path are proposed;

3) There are relations between *p-region* and *r-region*: the intermediate results during the shortest path computation can be used to decrease *r-region* and the number of NN candidates.

3.1 Heuristics for selecting NN for a computation point

In our approach the selection of computation point uses the following heuristics:

> [Heuristic 1] The start point on the route is the first computation point;
>
> [Heuristic 2] The next computation point is an intersection on the road network, which is the next intersection of the divergence between the route and the shortest path from the previous computation point to its NN.

Here, the divergence is a point on the route, where the shortest path branches off the route. If there is no overlap between the path and the route, the computation point can be regarded as the divergence. Heuristic 1 gives the start point of the search; Heuristic 2 helps to select the following computation point. In this paper, we only use this result and the detail description is given in [9].

> [Heuristic 3] The initial NN search region for a computation point *c* is a circle area, whose center is *c* and radius is *r*:
>
> $$r = Path_{cq} + Path_{qt}.$$

Here, *t* is NN for the previous computation point and *q* is a divergence. As the value of $Path_{qt}$ has computed in the previous NN search step, and the value of $Path_{cq}$ is the curve length between *c* and *q*, the computation of *r* is not expensive.

We can observe the example given in Figure 2, where *c* is the computation point, *S* is the previous one, and *t* is NN for *S* with the shortest path P_1. All NN candidates for *c* are located inside the initial search region *r-region*. The region is the circle area centering on *c* with radius *r*.

3.2 Heuristics for decreasing search region of path search

To find the shortest path from the computation point *c* to a candidate $t^{'}$ is based on an even smaller region: *p-region* shown in Figure 2.

In Figure 2, the path length from the current computation point *c* to NN of the previous computation point is *r*, and the straight-line distance between *c* and the candidate node $t^{'}$ is $d^{'}$. For an easy description we define a coordinate for them in Figure 3: the origin *O* of the coordinate is on the center of line $\overline{ct'}$, the x-axis passes along line $\overline{ct^{'}}$, and the y-axis is perpendicular to the x-axis on the origin *O*. To find the shortest path from *c* to $t^{'}$ is based on the road segments inside the region (as the dotted line in Figure 3):

$$p - region =$$

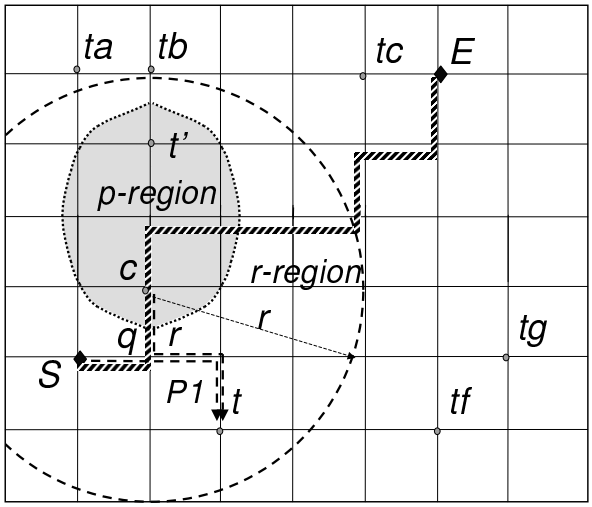

Figure 2 Search region generated for a new computation point: NN search region *r-region* and shortest path search region *p-region*

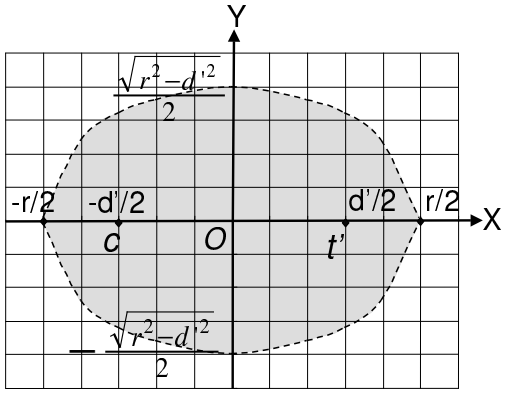

Figure 3 Shortest path search region

$$\{(x,y)|\sqrt{(x+d'/2)^2+y^2}+\sqrt{(x-d'/2)^2+y^2}\leqq r\}.$$

This means that any path from *c* to $t^{'}$ via a point on the boundary of (or inside) this region is equal to (or shorter than) *r*. All road segments on the shortest path could only be found inside this region, if there is any path shorter than *r* from *c* to $t^{'}$. Here, we prove this region is legal and is also the smallest one.

1) To prove the region is legal, we give an assumption that there was a point *p1 (x1,y1)* outside *p-region* locating on a path from *c* to $t^{'}$, and the length of this path $SP(c,t^{'})$ is not longer than *r*:

$$SP(c,t^{'})\leqq r.$$

According to triangular inequality, the path length is not shorter than the straight-line distances among *c*, *p1* and $t^{'}$, which is:

$$Straight-line-distance(c,p1,t^{'})=$$

$$\sqrt{(x1+d'/2)^2+y1^2}+\sqrt{(x1-d'/2)^2+y1^2};$$

$$Straight-line-distance(c,p1,t^{'})\leqq SP(c,t^{'})\leqq r;$$

and

$$\sqrt{(x1+d'/2)^2+y1^2}+\sqrt{(x1-d'/2)^2+y1^2}\leqq r.$$

By the definition of *p-region*, *p1* is inside *p-region*. This result is contradictory to the assumption.

2) To prove the region is the smallest region for the search, we assume that there is a point *p1 (x1,y1)* on the boundary of *p-region*, and a path from *c* to $t^{'}$ crosses *p1*. There are

$$Straight-line-distance(c,p1,t^{'})=r;$$

By the definition, the length of this path $SP(c,t^{'})$ is not longer than *r* and $SP(c,t^{'})$ is not shorter than $Straight-line-distance(c,p1,t^{'})$:

$$Straight-line-distance(c,p1,t^{'})\leqq SP(c,t^{'})\leqq r;$$

then

$$SP(c,t^{'})=r.$$

This means that any region of smaller than *p-region* may be results in an answer missing and *p-region* is the smallest region for this search.

The region can also be simplified to a rectangle with length *r* and width $\sqrt{r^2-d'^2}$.

3.3 Relation between *p-region* and *r-region*

The selection of a candidate for NN search is based on the straight-line distance between the computation point and the target objects: therefore, the next candidate for the computation point is located in a ring, which is between circle *d* and circle *r* in Figure 4. After the computation of shortest path from *c* to a candidate $t^{'}$, if the shortest $r^{'}$ is shorter than *r*, $r^{'}$ is set up as a smaller *r-region*

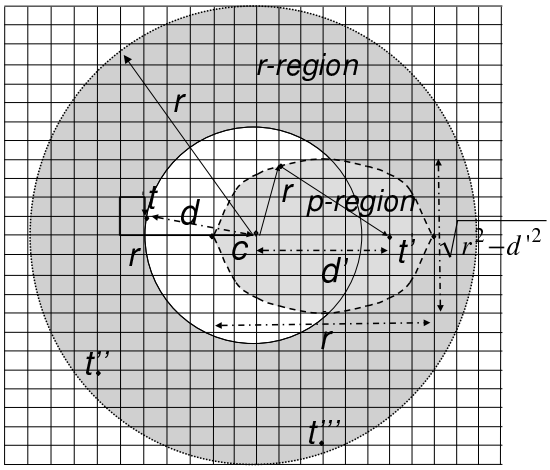

Figure 4 Relations between *p-region* and *r-region* for one computation point

for the following search. And *p-region* for the next candidate becomes smaller, too. With the search steps, *d* becomes longer and longer while *r* becomes shorter and shorter, and *p-region* becomes smaller and smaller, too. The area of *p-region* for the candidate $t^{'}$ is simplified as

$$Area(p-region^{'})\propto r\times\sqrt{r^2-d'^2}.$$

The area of *p-region* for the next candidate $t^{''}$ can be simplified as

$$Area(p-region^{''})\propto r^{'}\times\sqrt{r'^2-d''^2}.$$

As $r^{'}\leq r$ and $d^{''}\geqq d^{'}$;
so

$$Area(p-region^{''})\leqq Area(p-region^{'}).$$

The relation between *p-region* and *r-region* is very useful in the CNN computation based on the large hierarchical road network.

4. Algorithm for Large Hierarchical Road Network

In order to find CNN for a predefined route on large hierarchical road network, we adopt a partition method similar to HEPV [1]. Roughly speaking, the partitions of road network are regarded as a set of rectangles: e.g., bold-line rectangles in Figure 5. The boundary nodes are pushed to the upper level to form a super graph. All-pair shortest paths among the boundary nodes in the same partition are computed, and the corresponding edges are added to the super graph. *c* and $t^{'}$ are the computation point and NN candidate; and u_i and v_j are boundary nodes in *c-partition* and $t^{'}$*-partition*, respectively. In the super graph, there are pre-computed edges among u_i (i=1,2...) and those among v_j (j=1,2...). The gray region in Figure 5 represents current *p-region*. The shortest path from *c* to $t^{'}$ is:

$$SP(c,t^{'})=min_{u,v}\{SP(c,u)+SP(u,v)+SP(v,t^{'})\}.$$

Here, *u* and *v* are limited to be inside *p-region*.

p-region may be decreased with the computation steps: thus, the number of *u* and *v* becomes smaller and smaller. In this section, we first give a method for selecting the pair of (*u, v*), and then give the algorithm for CNN search based on a large hierarchical road network.

4.1 Heuristic for sorting boundary nodes

In order to find the shortest path from the source node to the end node, the method proposed in [1] should compare the path length for all the pairs of the source and end boundary nodes. In our method, the computation is limited to the boundary nodes inside *p-region*: furthermore, *p-region* may be decreased during the computation process; and some pairs of (*u, v*) in the previous *p-region* may be not inside the following *p-region*. *p-region* is decreased along the directions of black arrows in Figure 5. The boundary node on *c-partition*, which is further from u_0, has a higher possibility to be ignored in the following search. So, we select boundary nodes in the sequence of the distance between them and the cross-node of the corresponding boundary and line $\overline{ct'}$. In Figure 5, u_i (v_j) are sorted on the distances between u_i (v_j) and u_0 (v_0). The directions of white arrows in Figure 5 depict this situation.

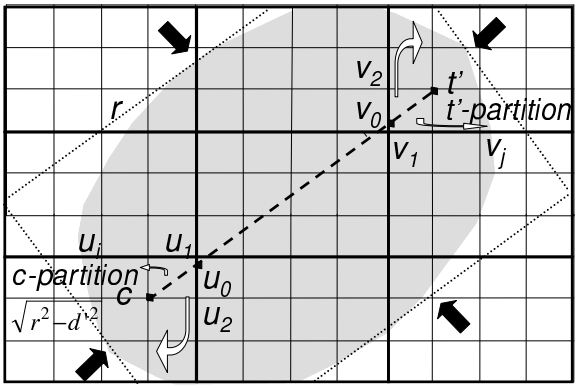

Figure 5 Situation of *p-region* and boundary nodes when computation point and NN candidate belong to different partitions.

4.2 Algorithm for CNN search on large hierarchical road network

Here, we describe the algorithms of CNN search, and NN-search with the shortest path search based (SP-search) on *p-region*. As we divide CNN search into a series of NN searches for computation points on the route, the algorithm of CNN search takes the responsibility of generating the computation point and initializing NN search region. At the beginning, the computation point is the start point of the route, and the search region for NN search is set to a Maximum value. The Maximum value could be specified by the user (e.g., less than 2km away from the route) or determined by some heuristics. The following steps run repeatedly to compute the result of triples for every computation point by calling NN search procedure, and generate the computation point and corresponding search region *r-region*. We give the algorithm as follows:

==========

Algorithm CNN-search

/* Input: route [S, E], target object set T

Output: Result set of triples

$\{< point, interval, path >,...\}$*/

1. Initialize:

set first computation point: CP = S;

set NN search region for CP: r = Max;

2. Do steps 3 to 5 until CP equals to E;

3. Call NN-search with CP, r; and get a triple of

$< t, [CP, q], Path_{CPt} >$;

4. Replace interval [CP, q] with [q_{pre}, q],

insert $< t, [q_{pre}, q], Path_{CPt} >$ into Result set;

5. Generate next computation point CP:

CP = next intersection from q along route;

set NN search region for CP:

r = $Path_{CPq} + Path_{qt}$;

==========

The NN search process bases on the R-tree index and a priority queue *Queue*. *Queue* is used to record the intermediate results: the R-tree nodes which are overlapped with the search region and the target objects. The key used to order the elements on *Queue* is straight-line distance of R-tree node and path length computed for target object. *Queue* is initialized as a node of the target object R-tree, which overlaps with the search region. When a target object turns out on the head of the priority queue, it becomes the candidate for further computation: the path length is computed for the candidate based on the search region. If the path length is smaller than the key of the head element of the queue, the candidate is the result. Otherwise, when the path length is smaller than the radius of search region, the search region is reset with the new length as radius. The value of radius is decreased by keeping step with the ongoing path length computation for the candidates, and the search region is adjusted until there is no candidate inside the search region. The NN search algorithm is given as follows:

==========

Algorithm NN-search

/* Input: route [S, E], target object set T;

source point CP and search region r;

Output: a triple $< point, interval, path >$*/

1. Initialize: priority queue *Queue*;

2. Locate first node overlapping with region r

on R-tree, compute straight-line distance d between

it and CP, and insert node into *Queue*;

3. Do steps 4 to 5 unless *Queue* is Null;

4. If head of *Queue* is leaf node of R-tree,

Then

(1) initialize *p-region* with r and d,

(2) Call SP-search to compute shortest path SP
from CP to it,

(3) insert result to *Queue*;

(4) If SP is smaller than r
Then reset r with SP;
Else

(1) compute straight-line distance between them,

(2) insert result to *Queue*;

5. If head of *Queue* is leaf node with computed
shortest path
/*object t of leaf node is NN for CP, computed path $Path_{CPt}$
is shortest path from CP to t*/
Find divergence q of $Path_{CPt}$ and route;
Return result of triple $< t, [CP, q], Path_{CPt} >$;

==========

==========

Algorithm SP-search

/* Input: source point CP, candidate point t and p-region;
Output: shortest path length*/

1. Locate CP and t to partitions on ground level;
2. Compute out u_0 and v_0 of corresponding partitions;
3. Do steps 4 to 6, until there is no new pair of (u_i, v_j);
4. Sort u_i (v_j) in order of straight-line distance
between u_i (v_j) and u_0 (v_0);
5. Select pair of (u_i, v_j), do step 6;
6. Compute SP with path length of (CP,u_i), (u_i,v_j) and (v_j,$t^{'}$),
If SP is less than r then reset p-region;
7. Return r as shortest path length;

==========

5. Conclusion

From a viewpoint of decreasing the times of disk access, we proposed a method for CNN search on the large hierarchical road network by minimizing the search region. The method does not only initialize NN search region for every computation point, but also minimizes the path region for NN search on road network. By using this method, CNN search along a route on the large hierarchical road network is greatly accelerated.

We have used our method in an urban district GIS to realize CNN search based on shortest path length over road network. Because the travel time cannot be simply regarded as proportionating the path length, this method cannot be used to compute CNN based on travel time over dynamic transportation network, i.e., provide up-to-date query results on the keep-changing transportation network. To solve the problem is in our future work.

Acknowledgments Our research is partly supported by the Hori Information Science Promotion Foundation.

References

[1] N. Jing, Y.W. Huang and E.A. Rundenstteiner: "Hierarchical Encoded Path Views for Path Query Processing:An Optimal Model and Its Performance Evaluation", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, No. 3, pp. 409–431 (1998).

[2] S. Shekhar and A. Fetterer: "Path Computation in Advanced Traveler Information Systems", *Proc. of Intelligent Transportation Systems* (1996).

[3] Y.W. Huang, N. Jing and E.A. Rundensteiner: "Effective Graph Clustering for Path Queries in Digital Map", *Proc. CIKM'96*, pp. 215–222 (1996).

[4] Y.W. Huang, N. Jing and E.A. Rundensteiner: "Path Queries for Transportation Networks: Dynamic Recordering and Sliding Window Paging Techniques", *Proc. GIS'96*, pp. 9–16 (1996).

[5] E.P.F. Chan and N. Zhang: "Finding Shortest Paths in Large Network Systems", *Proc. of GIS'01*, pp. 160–166 (2001).

[6] Y.F. Tao, D. Papadias and Q.M. Shen: "Continuous Nearest Neighbor Search", *Proc. of VLDB'02*, pp. 287–298 (2002).

[7] Z.X. Song and N. Roussopoulos: "K-Nearest Neighbor Search for Moving Query Point", *Proc. of SSTD'01*, pp. 79–96 (2001).

[8] S. Bespamyatnikh and J. Snoeyink: "Queries with Segments in Voronoi Diagrams", *SODA* (1999).

[9] J. Feng and T. Watanabe: "A Fast Method for Continuous Nearest Target Objects Query on Road Network", *Proc. of VSMM'02*, pp. 182–191 (2002).

[10] A. Guttman: "R-Trees: A Dynamic Index Structure for Spatial Searching", *Proc. of ACM SIGMOD'84*, pp. 47–57 (1984).