doi:10.3772/j.issn.1006-6748.2022.02.002

Directional nearest neighbor query method for specified geographical direction space based on Voronoi diagram①

LI Song(李　松)②, SONG Shuang, HAO Xiaohong, ZHANG Liping
(College of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, P.R.China)

Abstract

The existing nearest neighbor query methods cannot directly perform the nearest neighbor query of specified geographical direction space. In order to compensate the shortcomings of the existing methods, a directional nearest neighbor query method in specific direction space based on Voronoi diagram is put forward. This work studies two cases, i.e. the query point is static and the query point moves with a constant velocity. Under the static condition, the corresponding pruning method and the pruning algorithm of the specified direction nearest neighbor (pruning_SDNN algorithm) are proposed by combining the plane right-angle coordinate system with the north-west direction, and then according to the smallest external rectangle of Voronoi polygon, the specific query is made and the direction nearest neighbor query based on Voronoi rectangle (VR-DNN) algorithm is given. In the case of moving with a constant velocity, first of all, the combination of plane right angle coordinate system, geographical direction and circle are used, the query range is determined and pruning methods and the pruning algorithm of the direction nearest neighbor based on decision circle (pruning_DDNN algorithm) are put forward. Then, according to the different position of motion trajectory and Voronoi diagram, a specific query through the nature of Voronoi diagram is given. At last, the direction nearest neighbor query based on Voronoi diagram and motion trajectory (VM-DNN) algorithm is put forward. The theoretical research and experiments show that the proposed algorithm can effectively deal with the problem of the nearest neighbor query for a specified geographical direction space.

Key words: nearest neighbor query, direction, Voronoi diagram, rectangular plane coordinate system

0 Introduction

The spatial nearest neighbor query has important application value. Some important research results have been achieved in the field of nearest neighbor query, such as nearest neighbor (NN) query[1-2], continuous nearest neighbor (CNN) query[3-4] and other basic query algorithms. Nearest neighbor query is widely used in real life, such as finding the nearest convenience store, recommending the best tour route for a scenic spot, and finding the best temperature for a species. Further, according to the different query requirements and different query objects, some researchers propose the nearest neighbor query between points and points, the nearest neighbor query between line segments and line segments, and the nearest neighbor query between points and line segments. Then according to different query environments, researchers propose the nearest neighbor query in the obstacle environment, the nearest neighbor query in the high-dimensional environment, and the nearest neighbor query in the road network environments. The existing researches mainly focus on reverse nearest neighbor (RNN) query[5-6], group nearest neighbor (GNN) query[7-8], K-nearest neighbor (KNN) query[9-10], moving view field nearest neighbor (MVFNN) query[11], reverse K-nearest neighbor (RKNN) query[12-13], dependent nearest neighbor (dNN)[14-15], K-aggregate nearest neighbor (K-ANN) query[16] and so on.

In practical application, the direction nearest neighbor query has the vital significance. For example,

① Supported by the National Natural Science Foundation of China (No.61872105, 62072136), the Natural Science Foundation of Heilongjiang Province (No.LH2020F047), the Scientific Research Foundation for Returned Scholars Abroad of Heilongjiang Province of China (No.LC2018030) and the National Key R&D Program of China (No.2020YFB1710200).

② To whom correspondence should be addressed. E-mail: lisongbeifen@163.com.

Received on July 1, 2021

when a vehicle is driving on a highway and needs to find a nearest gas station, if the traditional nearest neighbor inquiry is used, it maybe show the nearest gas station on the other side of the driving direction. If the vehicle turns around or goes retrograde on the highway at this time, there will be great danger. According to the actual demand, the nearest neighbor query problem with direction in the spatial database is an important issue. Therefore, the nearest gas station can be quickly judged in the direction of vehicles by adding geographic direction to the query of the nearest neighbor. The existing methods of the directional nearest neighbor queries have some limitations, and cannot be better applied in a given geographic direction. At the same time, continuous nearest neighbor queries cannot be conducted in a designated geographic direction when the query point is moving.

The method proposed in this paper is able to effectively reduce the redundant data, reduce the period for complicated calculation and receive more accurate searching results. The main contributions of this paper are as follows.

(1) In view of the data redundancy, when the query point is static, a method of combining geographical direction with the plane right-angle coordinate system is proposed. According to the positive and negative nature of the four quadrant symbols in the plane right-angle coordinate system, the pruning method can be proposed, which can effectively remove the data set points other than the query direction before the query is carried out.

(2) For the data redundancy when moving with constant velocity, on the basis of combining with the plane right angle coordinate system, a judgment circle is generated. And according to the different position of the determination circle and the plane right angle coordinate system, two pruning methods are proposed. The judgment circle can narrow the scope of the query and reduce the searching time complexity.

(3) By utilizing the nature of the Voronoi diagram and the nature of the Voronoi polygon minimum circumscribed rectangle, a specific search is made in the data set after the pruning, and the corresponding algorithm is put forward, which can make efficient query in a specific direction.

1 Related work

Researchers have proposed many query techniques for spatial data processing by using different spatial index structures. The nearest neighbor query method and pruning rules based on R^{BRT} tree were proposed in Ref. [17]. Ref. [17] proposed a NN _ FRT query algorithm, which filters out a large number of non-candidate set data points according to the characteristics of the R^{BRT} tree, and then calculates the final query results according to the trapezoidal distribution of data points. Ref. [18] proposed K-nearest neighbor query algorithm based on density grid index. During the processing of the algorithm, a series of candidate search radii are obtained by using the geometric features of the rectangle. Then, according to the density distribution of moving objects, the appropriate candidate search radius is selected to filter out the data points which are not in the search circle.

Ref. [19] introduced a novel type of spatial query called the reverse view field nearest neighbor (RVFNN) query. To process the query, Ref. [19] proposed two query processing methods on an R^*-tree, i. e. RVFNN query processing on a sector-based R^*-tree and RVFNN query processing on an origin-based R^*-tree. In addition, Ref. [19] proposed a new type of spatial data index structure called the view field R-tree (VFR-tree) and a search method for RVFNN queries on the VFR-tree. Based on the definition of influence, Ref. [20] proposed reverse approximate nearest neighbors (RANN) queries. Ref. [21] presented a progressive algorithm for approximate nearest neighbor indexing and querying, which is a novel algorithm for progressive approximate K-nearest neighbors, enabling fast KNN queries while continuously indexing new batches of data. In Ref. [22], a shared execution-based approach called standard clustered loop (SCL) was proposed that allows efficient processing of all nearest neighbor (ANN) queries on a dynamic road network. The key concept behind the shared execution technique is to exploit the coherence property of road networks by clustering objects that share common paths and processing the cluster as a single path.

In the existing research on directional-related nearest neighbor queries, the directional-aware index structure was proposed in Ref. [23]. It first groups the points of interest (POI) according to the distance from the query point to the lower left corner of the smallest bounding rectangle containing all nearest POIs. Then, POI is sorted in each group according to the direction of the lower left corner, and the effective region-based pruning techniques and direction-based pruning techniques are further proposed. Meanwhile, an effective algorithm is put forward to solve the direction-aware spatial keyword query problem. Ref. [24] proposed a new spatial data query, that is, the direction-constrained continuous nearest neighbor (DCNN) query. DCNN query finds the nearest POI that satisfies the direc-

tion constraint. The direction constraint is an angular range that depends on the direction of the user's speed, and the range of angles is determined by the user.

During the past years, direction nearest neighbor query has gained further research. The direction-aware nearest neighbor (DNN) query was proposed in Ref. [25]. Given the query point q and the angle threshold θ, DNN query can search the nearest neighbor in all directions around the query point q, which can be applied to search the nearest object, and also can be used to search for photos of geographical objects according to the location of the photos taken. The proposed algorithms can answer point-DNN queries and range-DNN queries effectively and efficiently. Ref. [26] proposed a new type of query and devised a new index structure for efficiently indexing spatio directional objects. The index structure utilizes multiple R-trees with different directional ranges to minimize the distance search space. Two efficient query processing algorithms including single matching algorithm and adaptive matching algorithm are also proposed. Ref. [27] presented a novel algorithm of direction-aware continuous moving K-nearest neighbor (DACKNN) queries in road networks. In this method, the azimuth information of objects is adopted to determine the moving direction, ensuring the moving objects in the result set towards the query object.

2 Definitions and property

Definition 1 (Voronoi diagram[28]) Given a set of data point sets $P = \{p_1, p_2, \cdots, p_n\} \subset R^2$, of which $2 < n < \infty$, $p_i \neq p_j$ when $i \neq j$. The Voronoi region is defined as $VH(p_i) = \{q \mid D(q, p_i) \leqslant D(q, p_j)\}$, $D(q, p_i)$ is the minimum distance between q and p_i, and p_i is called Voronoi generation point. The Voronoi region $VH(p_i)$ determined by p_i is called a Voronoi polygon, the rib of the Voronoi polygon is $VL(p_i)$, and the Voronoi polygon that shares edges with $VH(p_i)$ is called $VH(p_i)$ adjacent polygon, and its corresponding generating point is called the adjacency generation point of p_i. The graph defined by $V(H) = \{VH(p_1), VH(p_2), \cdots, VH(p_n)\}$ is called a Voronoi diagram.

Property 1 The distance from any point $q(q \notin Q)$ to p in Voronoi polygon $VH(p)$ must be less than the distance from q to other Voronoi generating points[28].

Definition 2 (the minimum external rectangle of Voronoi polygon) The rectangle with the smallest area around Voronoi polygon $VH(P)$ constructed by one or more vertices of Voronoi polygon is called the smallest circumscribed rectangle of Voronoi polygon, whose edges are parallel to the coordinate axis, and is denoted as $VR(P)$.

Definition 3 (nearest neighbor query[1]) Given a set of data objects P and a query point q, the nearest neighbor query finds a subset of data objects in the set P, satisfying the following conditions:

$$\mathrm{NN}(q) = \{p \in P \mid o \in P, \mathrm{dist}(q, p) \leqslant \mathrm{dist}(q, o)\} \quad (1)$$

where $\mathrm{dist}(q, p)$ represents the shortest distance from q to p.

Definition 4 (the directional nearest neighbor query) Let the query point be q, the data object set be P, and the nearest neighbor query direction be DR, DR = {East-North-West-South}. There is a data point set $K \subset P$ in the DR direction, and the nearest neighbor query aims to find the nearest neighbor of the query point q in the set K, that is, $\mathrm{DNN}(q) = \{p_d \in K \mid \forall p_i \in P, \mathrm{dist}(q, p_d) \leqslant \mathrm{dist}(q, p_i), K \subset P\}$, p_d represents the data object in the query direction.

3 Directional nearest neighbor query under static query points

In the process of the directional nearest neighbor query when the query point is static, since the nearest neighbor in a specified direction is queried, it is unnecessary to consider the data set points in the other directions. Therefore, the directional nearest neighbor query under the static query point proposed in this section includes two steps, the data preprocessing and the directional nearest neighbor query processing.

3.1 Data preprocessing

Due to the large number of data points and the long time consumed in the query, it is necessary to preprocess the data before query, that is, to prune the data points in other irrelevant directions.

First, the direction of geographical location is set to east, west, south and north. The east, west, south, and north directions are four directions, and the plane rectangular coordinate system with the same structure also has four quadrants. Therefore, in order to determine the nearest neighbor of the concrete direction, the plane rectangular coordinate system is combined with the east, west, north and south directions, and the nature of the plane rectangular coordinate system is used to reduce the scope of the query.

Second, a plane rectangular coordinate system is established.

(1) When the query direction is in any direction of northeast, southeast, northwest and southwest, a plane rectangular coordinate system is established with

the query point q as the origin, the east-west direction as the horizontal axis, and the north-south direction as the vertical axis. Let the horizontal axis be x-axis and the vertical axis be y-axis. The northeast direction is located in the first quadrant, and its abscissa and ordinate are positive; the northwest direction is in the second quadrant, and the abscissa is negative and the ordinate is positive; the southwest direction is in the third quadrant, the abscissa and ordinate are negative; the southeast direction is in the fourth quadrant, and the abscissa is positive and the ordinate is negative.

(2) When the query direction is in a single direction, such as east, west, south, and north, a plane rectangular coordinate system is established with the query point q as the origin, the east-south direction as the horizontal axis, and the west-north direction as the vertical axis. Let the horizontal axis be x-axis and the vertical axis be y-axis. The east direction is located in the first quadrant, and its abscissa and ordinate are positive; the north direction is in the second quadrant, and the abscissa is negative and the ordinate is positive; the west direction is in the third quadrant, the abscissa and ordinate are negative; the south direction is in the fourth quadrant, and the abscissa is positive and the ordinate is negative.

Finally, because the points in the data set P are scattered in each quadrant and the nearest neighbor in a concrete direction is needed, the data points outside the required direction area are pruned by using the difference of positive and negative coordinates in each quadrant. To reduce the amount of calculation, Theorem 1 is proposed.

Theorem 1 Let p_a, $p_b \in P$, the coordinates are expressed as $p_a(x_{pa}, y_{pa})$, $p_b(x_{pb}, y_{pb})$, $CDT\_P(q)$ be the nearest candidate set of the query point q after pruning. If p_a is in the region of the desired direction, then $p_a \in CDT\_P(q)$; conversely, if p_b is not in the region of the desired direction, then $p_b \notin CDT\_P(q)$.

Proof If p_a is in the region where the desired direction is located, which means the symbol of $p_a(x_{pa}, y_{pa})$ coordinate is consistent with the symbol of the horizontal and vertical coordinates of the quadrant where the query direction is located, therefore $p_a \in CDT\_P(q)$; if P_b is not in the region where the desired direction is located, which means the symbol of $p_b(x_{pb}, y_{pb})$ coordinate is inconsistent with the symbol of the horizontal and vertical coordinates of the quadrant of the query direction, p_b will be pruned. So $p_b \notin CDT\_P(q)$. The certificate is completed.

Pruning method 1 can be further obtained from Theorem 1.

Pruning method 1 Arbitrarily take the coordinates (denoted as $p_i(x_{pi}, y_{pi})$) in the desired direction area, and respectively make vertical lines from all the points in the data set P to the x-axis and the y-axis. Then the symbols of the horizontal and vertical coordinates will be obtained. The abscissa or ordinate of the obtained point in the data set P is pruned inconsistent with the symbol of the coordinates $p_i(x_{pi}, y_{pi})$.

The query range is reduced by judging the coordinates, and the pruning algorithm of the specified direction nearest neighbor (Pruning_SDNN algorithm) is further given, as shown in Algorithm 1.

Algorithm 1 Pruning_SDNN

Input: dataset P, query direction.

Output: dataset $CDT\_P(q)$.

```
Begin:
1. According to the direction of the query, establish the plane right angle coordinate system;
2. Set any point in the area where the query direction is located with a coordinate of p_i(x_pi, y_pi);
3. Let the coordinate of the point in the data set P be denoted as p_j(x_pj, y_pj);
4. for j = 1 to P. length do
5.    if x_pi > 0 and y_pi > 0
        //The query direction is northeast
6.    if x_pj > 0 and y_pj > 0
7.      CDT_P(q) ← p_j;      //Pruning method 1
8.    else pruning;
9. if x_pi < 0 and y_pi > 0
     //The query direction is northwest
10.   if x_pj < 0 and y_pj > 0
11.         CDT_P(q) ← p_j;   //Pruning method 1
12.   else pruning;
13. if x_pi < 0 and y_pi < 0
      //The query direction is southwest
14.   if x_pj < 0 and y_pj < 0
15.     CDT_P(q) ← p_j;   //Pruning method 1
16.   else pruning;
17. if x_pi > 0 and y_pi < 0
      //The query direction is southeast
18.   if x_pj > 0 and y_pj < 0
19.     CDT_P(q) ← p_j;      //Pruning method 1
20.   else pruning;
21. return CDT_P(q);
End
```

Algorithm 1 firstly establishes the plane rectangular coordinate system, and determines the quadrant positions of northeast direction, northwest direction, southwest direction and southeast direction according to the properties of the plane rectangular coordinate sys-

tem. Then, by using the positive and negative quadrants, the data points which are not consistent with the sign of the quadrant in the required direction are pruned to reduce the data set.

3.2 Identifying neighbors

In subsection 3.1, the plane rectangular coordinate system is combined with the east, west, north and south directions, and the redundant data points are pruned by using its properties. This section will conduct the specific queries. Since the redundant data set points have been pruned, it only needs to construct Voronoi diagram in the specific query direction. In order to facilitate searching, a minimum circumscribed rectangle is constructed based on the Voronoi polygon, and the properties of the Voronoi polygon minimum circumscribed rectangles are used to determine the nearest neighbor.

A Voronoi diagram is constructed in the desired direction, and a rectangle of the smallest area is constructed surrounding $VH(P)$ by one or more fixed points of Voronoi polygon $VH(P)$, that is, the minimum circumscribed rectangles of Voronoi polygon is constructed, as shown in Fig. 1.

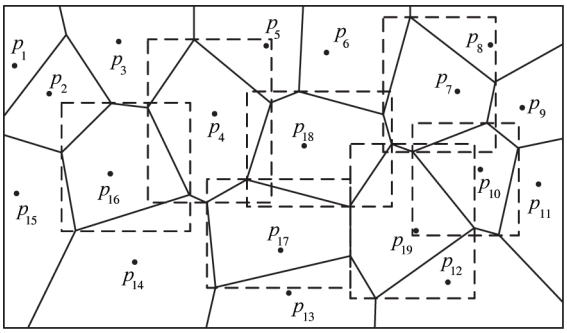


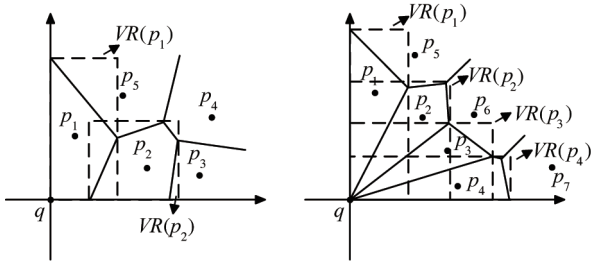
Fig. 1 Minimum circumscribed rectangle of Voronoi polygon

The positional relationship between the query point q and the minimum circumscribed rectangle generated by Voronoi polygon has the following two cases.

(1) When the number of the minimum circumscribed rectangle containing the query point q is 1, the data point p_i corresponding to the minimum circumscribed rectangle is the nearest neighbor of the query point q in the direction.

(2) When the number of the minimum circumscribed rectangle containing the query point q is $m(m>1)$, the query point q is on the edge line of Voronoi polygons. According to the definition of Voronoi diagram, the distance from the query point q to $p_1 \cdots p_m$ is equal. At this time, all of $p_1 \cdots p_m$ are the nearest neighbor of the query point q in this direction.

For example, let the query direction be northeast. As shown in Fig. 2(a), the minimum circumscribed rectangle containing the query point q is $VR(p_1)$, then the data point p_1 corresponding to $VR(p_1)$ is the nearest neighbor of the query point q. As shown in Fig. 2(b), the minimum circumscribed rectangle containing the query point q is $VR(p_1)$, $VR(p_2)$, $VR(p_3)$, $VR(p_4)$, then all of the data points p_1, p_2, p_3, p_4 corresponding to $VR(p_1)$, $VR(p_2)$, $VR(p_3)$, $VR(p_4)$ are the nearest neighbor of the query point q.



(a) A minimum circumscribed rectangle containing query point q

(b) Multiple minimum circumscribed rectangles containing query point q

Fig. 2 The relationship between query point q and $VR(P)$

In the query process, first, the Voronoi diagram of the data set P_1 is generated. Second, the minimum circumscribed rectangle of each Voronoi polygon is generated, and then the number of minimum circumscribed rectangles of Voronoi polygon determines where the query point q is located. Finally, the Voronoi generation point is determined corresponding to the minimum circumscribed rectangle and the nearest neighbor of q is found.

Based on the above discussion, the direction nearest neighbor query based on Voronoi rectangle (VR_DNN) algorithm for the nearest neighbor query based on the minimum circumscribed rectangle of the Voronoi diagram is given as Algorithm 2.

Algorithm 2 VR_DNN

Input: dataset $CDT\_P(q)$, query point q.

Output: directional nearest neighbor of q.

Begin:

1. Create_Voronoi($CDT\_P(q)$);
2. Create Min_rectangle(p_i, $VH(p_i)$);
   //Create minimum circumscribed rectangles of the Voronoi polygon
3. Judge the relationship between query point q and minimum circumscribed rectangular MR(P);
4. Determine the number m of minimum circumscribed rectangles MR(P) containing q;
5. if $m=1$ then //q is only in one MR(P)
6. DNN $\leftarrow p_i$;
   //Add p_i to directional nearest neighbor set

7. else if $m>1$ then
   //q is in the overlapping region of multiple MR(P)
8. Determine the minimum distance $d_1 \cdots d_m$ corresponding to the data point $p_1 \cdots p_m$;
9. DNN←$p_1 \cdots p_m$;
   //Add $p_1 \cdots p_m$ to directional nearest neighbor set
10. return DNN;

End

In Algorithm 2, the minimum number of outer rectangles containing query point q is determined by adding the minimum outer rectangle in Voronoi diagram, and then the data set p_i or $p_1 \cdots p_m$ is got corresponding to the minimum outer rectangle of Voronoi polygon. The data point p_i or $p_1 \cdots p_m$ is the nearest neighbor of query point q in this direction.

4 Directional nearest neighbor query under uniform motion of query points

When the query points move, the directional nearest neighbor of the query points will change accordingly. So, this section will mainly focus on the nearest neighbor query problem when dynamic query points are moving at a uniform speed. Prerequisites are that the motion trajectory of the query point q is known, and the end point of the motion trajectory is recorded as z.

When the query point Q moves uniformly, it mainly moves in the following situations: (1) uniform straight line; (2) uniform fold line; (3) uniform curve; (4) combination of uniform fold line and curve.

4.1 Data preprocessing

Firstly, the minimum circumscribed rectangle of the trajectory of the query point q as $R(t)$ is constructed, and then the plane rectangular coordinate system is constructed. The horizontal axis is the east-west direction, which is the x-axis; the vertical axis is the north-south direction, which is the y-axis; and the intersection of x-axis and y-axis is the origin o, which is a vertex in $R(t)$. When the query direction is northeast, the extension line of the lower boundary of the minimum circumscribed rectangle is the horizontal axis, and the extension line of the left boundary is the vertical axis; when the query direction is northwest, the extension line of the lower boundary of the minimum circumscribed rectangle is the horizontal axis, and the extension line of the right boundary is the vertical axis; when the query direction is the southwest direction, the extension line of the upper boundary of the minimum circumscribed rectangle is the horizontal axis, the extension line of the right boundary is the vertical axis; when the query direction is the southeast direction, the extension line of the upper edge of the minimum circumscribed rectangle is the horizontal axis, and the extension line of the left boundary is the vertical axis.

Secondly, the scope of the query is determined. Many data points which are not in the desired direction in the data set points do not need to be considered, so how to determine the query range is the key to prune the data set. The query range is determined by generating a decision circle and prune it.

Finally, since the query points are moving at a uniform speed in a certain direction, the points in other irrelevant directions are pruned to reduce the amount of calculation. So Pruning method 2 and Pruning method 3 are proposed.

Pruning method 2 In the minimal circumscribed rectangle of the trajectory of the query point q, let the other end of the diagonal line with one end point as the origin o be marked as o', and a decision circle $C_{oo'}$ with the o' as the center can be generated. The straight line distance between origin o and o' is denoted as the radius of the center, marked as d_{oo}. Then the data set points in the area other than the circle are pruned, and the data set after pruning is recorded as $CDT\_PO(q)$.

Pruning method 3 In data set $CDT\_PO(q)$, the data points which are not in the quadrant of the direction of motion in the judgment circle area are pruned, and the data set after pruning is recorded as $CDT\_PI(q)$.

Theorem 2 There are two points p_c, $p_d \in CDT\_PO(q)$, their coordinates are denoted as $p_c(x_{pc}, y_{pc})$, $p_d(x_{pd}, y_{pd})$, and points p_c, p_d are both in the generated decision circle. If p_c is in the region where the desired direction is located, then the $p_c \in CDT\_PI(q)$. Otherwise, if p_d is not in the region where the desired direction is located, the $p_d \notin CDT\_PI(q)$.

Proof If the p_c is in the region where the desired direction is located, the symbol of $p_c(x_{pc}, y_{pc})$ coordinate is consistent with the symbol of the horizontal and vertical coordinates of the quadrant where the query direction is located, so $p_c \in CDT\_PI(q)$; if p_d is not in the region where the desired direction is located, according to Pruning method 1, for the symbol of the horizontal and vertical coordinates of $p_d(x_{pd}, y_{pd})$ is inconsistent with the symbol of the horizontal and vertical coordinates of the quadrant where the query direction is located, p_d will be pruned. So $p_d \notin CDT\_PI(q)$. The

certificate is completed.

For example, to query the nearest neighbor in the northeast direction, its pruning diagram is shown in Fig. 3. First, according to Pruning method 2, the data points in the region other than the decision circle are pruned, that is, the data points p_1, p_2, p_3 are pruned. According to Pruning method 3, the points in the decision circle that are not in the specific query direction are pruned, that is, p_8, p_9, p_{10} are pruned. Then the last direction nearest neighbor is in p_4, p_5, p_6, and p_7.

The query scope is determined by making a decision circle. The pruning algorithm of the direction nearest neighbor based on decision circle (Pruning _ DDNN algorithm) is further given below, as shown in Algorithm 3.

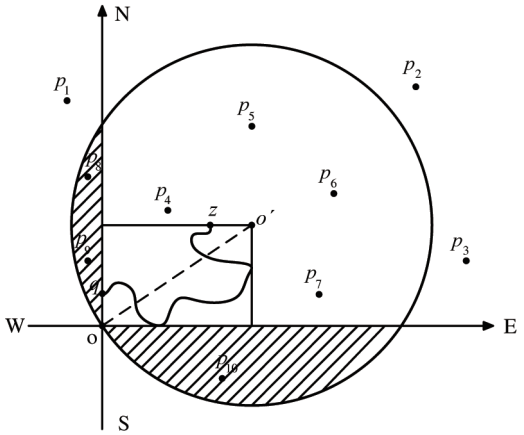


Fig. 3 Pruning diagram when the query direction is northeast

Algorithm 3 Pruning _ DDNN

Input: dataset P, query direction.

Output: dataset $CDT\_PI(q)$.

Begin:

1. Establish a plane rectangular coordinates according to the query direction;
2. Creat _ Circle (z, $d_{oo'}$);
   //Generate a decision circle $C_{oo'}$ with the o' as the center, the straight line distance between origin o and o' denoted as the radius of the center, marked as d_{oo}.
3. Let the coordinates of any point in the area where the query direction is located as $p_i(x_{pi}, y_{pi})$;
4. Let the coordinate of data set P as $p_j(x_{pj}, y_{pj})$;
5. for $j=1$ to $P.\ length$ do
6. if p_j is in C_{qz}
7. $CDT\_PO(q) \leftarrow p_j$; //Pruning method 2
8. else pruning;
9. Let the coordinate of data set $CDT\_PO(q)$ as $p_k(x_{pk}, y_{pk})$;
10. for $k=1$ to $CDT\_PO(q).\ length$ do
    //$CDT\_PO(q).\ length$ represents the number of data points in the data set $CDT\_PO(q)$
11. if $x_{pi} > 0$ and $y_{pi} > 0$
    //The query direction is northeast
12. if $x_{pk} > 0$ and $y_{pk} > 0$
13. $CDT\_PI(q) \leftarrow p_k$; //Pruning method 3
14. else pruning;
15. if $x_{pi} < 0$ and $y_{pi} > 0$
    //The query direction is northwest
16. if $x_{pk} < 0$ and $y_{pk} > 0$
17. $CDT\_PI(q) \leftarrow p_k$; //Pruning method 3
18. else pruning;
19. if $x_{pi} < 0$ and $y_{pi} < 0$
    //The query direction is southwest
20. if $x_{pk} < 0$ and $y_{pk} < 0$
21. $CDT\_PI(q) \leftarrow p_k$; //Pruning method 3
22. else pruning;
23. if $x_{pi} > 0$ and $y_{pi} < 0$
    //The query direction is southeast
24. if $x_{pk} > 0$ and $y_{pk} < 0$
25. $CDT\_PI(q) \leftarrow p_k$; //Pruning method 3
26. else pruning;
27. return $CDT\_PI(q)$;

End

Algorithm 3 establishes a plane rectangular coordinate system, and determines the quadrant position of northeast, northwest, southwest, and southeast directions according to the properties of the plane rectangular coordinate system. A decision circle $C_{oo'}$ will be generated with the o' as center and the straight line distance $d_{oo'}$ between origin o and o' as radius. It prunes the data set points outside the decision circle, and prunes the area where the decision circle outside the quadrant of the motion direction.

4.2 Dircetional nearest neighbor query

In order to ensure the accuracy of the query results, this section will perform real-time search. A Voronoi diagram in the decision circle after pruning is established, and the nearest neighbor of the query point q is found by using the properties of the Voronoi diagram. Theorem 3, Theorem 4 and Theorem 5 are given.

Theorem 3 There is $p_e \in CDT\_PI(q)$, and at a certain time T_i during the motion, if the query point q and data set point p_e are in the same location, then the data set point p_e is the nearest neighbor of the query point q at the time of T_i.

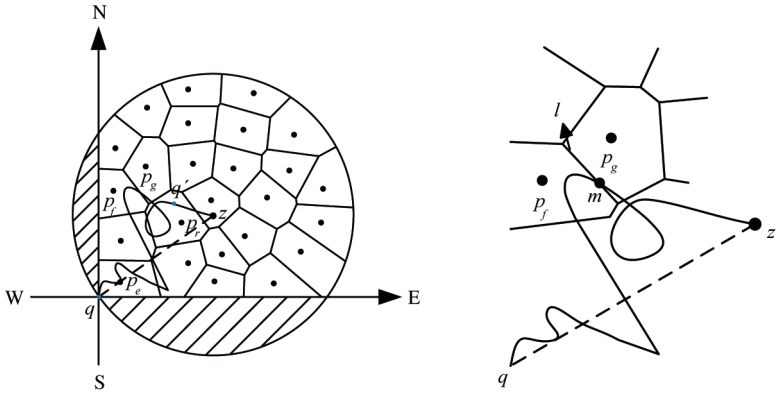
Proof As shown in Fig. 4(a), there is $p_e \in CDT$

$\_PI(q)$. At a certain time T_i in the process of uniform movement of the query point q, the Euclidean formula between query point q and data set point p_e is 0, that is

$$D_{qpe} = \sqrt{|x_q - x_{pe}|^2 + |y_q - y_{pe}|^2} = 0 \quad (2)$$

so at this time, the data set point p_e is the nearest neighbor of the query point q. The certificate is completed.

Theorem 4 There is p_f, $p_g \in CDT\_PI(q)$. The data points p_f and p_g share a edge l of Voronoi polygon, and the edge l intersects with the trajectory of the query point q. At a certain time T_j during the movement, the query point q moves to the intersection point m. At this time the Euclidean distance D_{zpf} and D_{zpg} from p_f, p_g to the end point z are calculated. The corresponding data points of D_{zpf} and D_{zpg} with smaller values are the nearest neighbors of the query point q at the time of T_j.



(a) Nearest neighbor of q is p_e (b) Nearest neighbor of q is p_g

Fig. 4 When the query direction is northeast

Proof As shown in Fig. 4(b), p_f, $p_g \in CDT\_PI(q)$, the distance from point p_f to z is represented as D_{zpf}, the distance from point p_g to z is represented as D_{zpg}, and the points p_f, p_g share the edge l of the Voronoi polygon. The query point q intersects the edge l at point m. At this time, the distances from the query point q to p_f, p_g are equal. In order to reduce the movement time of query point q, the point closer to the end point z is selected as the directional nearest neighbor. As shown in Fig. 4, $D_{zpf} > D_{zpg}$, so p_g is the direction nearest neighbor of query point q during the time of T_j. The certificate is completed.

Theorem 5 Let $p_j \in CDT\_PI(q)$, and at a certain time T_y during the movement, if the data set point p_j is in the Voronoi polygon region where the query point q is located, then p_j is the nearest neighbor of the query point q at the time of T_y.

Proof It is evident that this theorem can be proved by Property 1. For example, as shown in Fig. 4, when the query point q moves to q', q' is in the Voronoi polygon generated by the data point p_r, where the data point p_r is the nearest neighbor in the direction of the query point q.

Based on the above discussion, the direction nearest neighbor query based on Voronoi diagram and motion trajectory (VM_DNN) algorithm for the nearest neighbor query based on the minimum circumscribed rectangle of the Voronoi diagram is further shown as Algorithm 4.

Algorithm 4 VM_DNN

Input: dataset $CDT\_PI(q)$, query point q.

Output: directional nearest neighbor of q.

Begin:

1. Create_Voronoi($CDT\_P(q)$);
2. Let the point p_e be in the data set $CDT\_PI(q)$;
3. if query point q coincides with data set point p_e
4. DNN←p_e;
   //Add p_e to the nearest neighbor set Theorem 3
5. Let the data set $CDT\_PI(q)$ contain points p_f, p_g, and the points p_f and p_g share a rib of a Voronoi polygon;
6. else if the trajectory of the query point q intersects with l
7. Calculate the distance between p_f and p_g to the end point z, taking the minimum distance $d_{\min}$;
8. Determine the data point p_f or p_g corresponding to the minimum distance $d_{\min}$;
9. DNN←p;
   //Add p to the nearest neighbor set, Theorem 4
10. Let there be a point p_j in the data set $CDT\_PI(q)$;
11. else if query point q falls in the Voronoi polygon area of point p_j
12. DNN <—p_j;
   //Add p_j to the nearest neighbor set, Theorem 5
13. return DNN;

End

Algorithm 4 first establishes a Voronoi diagram in the pruned data set area, and then judges the positional relationship between query point q, data point and the Voronoi polygon region. Then it uses Theorem 3, Theorem 4, and Theorem 5 respectively to determine the nearest neighbor. The algorithm constructs a Voronoi graph by using the pruned data set $CDT\_PI(q)$ (row 1). If the query point q coincides with the data point p_e, then p_e is the directional nearest neighbor at the current moment (rows 2 – 4); when the data points p_f and p_g share an edge l of Voronoi polygon, and the edge l intersects with the motion trajectory at point m, then the point closer to the end point z is the directional nearest neighbor (rows 5 – 9); when the query point q is in the Voronoi polygon region, the data points in this region are the directional nearest neighbor (rows 10 – 12), and finally the directional nearest neighbors in a specific direction can be obtained (row 13).

5 Experiments and analysis

This section analyzes the performance of the proposed algorithm through experiments. The experimental environment is 2.60 GHz CPU, 8 GB memory, 500 GB hard disk, programmed with Microsoft Visual Studio. NET2003, Windows 10 system. The experimental data are collected by using data sets generated by spatial data generation software and road network information in San Francisco and California, USA. The data have 1 965 206 nodes and 5 533 214 edges. During the experiment, the distribution of data and related information are appropriately adjusted. Some redundant information of the data is eliminated.

Two important comparison algorithms are used to compare with the method proposed in this paper. The experiment mainly tests the impact of data set size on runtime and I/O cost, the impact of dynamic query points on query time, and the impact of query time on the accuracy of the algorithm. Since the nearest neighbor query with the point as the query object cannot calculate the query in the specified geographic direction, the existing method cannot be directly compared with the method in this paper. In order to obtain the comparison algorithm, the existing algorithm is adjusted appropriately, and finally two comparison algorithms are obtained.

After adjusting the PointDNNBaseline algorithm based on direction perception proposed in Ref. [25], the comparison algorithm DNNBase + is got. Firstly, this paper mainly queries the nearest neighbors in the specific direction. Based on the results of the PointDNNBaseline algorithm given in Ref. [25], the direction constraint is added, that is, the polar angle range of the neighboring result is set according to the geographic direction of the east, west, south and north. Secondly, the nearest neighbor result in the query direction is determined according to the Euclidean distance. After the adjustment, the DNNBase + algorithm is finally formed.

The second comparison algorithm is the R_MULTI algorithm which is appropriately adjusted based on the UpperEvenLevel algorithm proposed in Ref. [26]. Firstly, since the Ref. [26] studied the K-nearest neighbor query based on the direction constraint, and this paper studies the nearest neighbor based on the Voronoi diagram, the K of Ref. [26] is adjusted to 1; secondly, all the angle of the data point defaulted leads to the query point q; finally, since the plane rectangular coordinate system established in this paper is divided into four quadrants, each of which is 90 °, the angle constraint in Ref. [26] is set to 90 °, the angle of the range is fixed to 90 °, and the R_MULTI algorithm can be formed after the adjustment.

First, the impact of the data set size on the VR_DNN algorithm is tested when query point q is static. Then the length of the query time is tested under the data sets of different scales when the query point q is static. When the data sets scale is 20 000, 40 000, 60 000, 80 000, and 100 000, the test results of time consumed to execute queries are shown in Fig. 5.

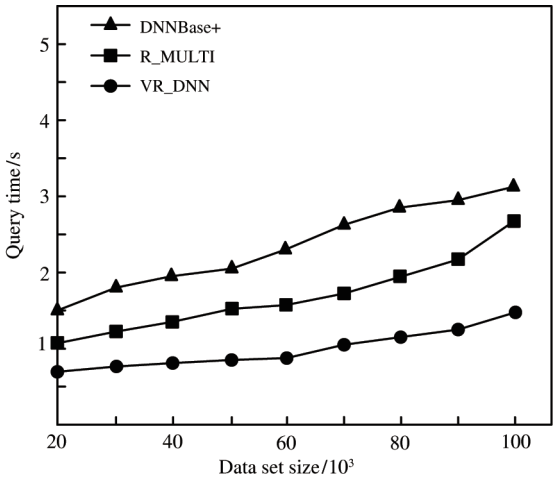


Fig. 5 Comparison of query time under different scale data sets when query point q is static

In Fig. 5, the running time of the three algorithms is gradually increasing as the number of data points increases. The VR_DNN algorithm in this paper consumes relatively less time during the query process. This is because Dist_jDNN algorithm is used to prune the data point set before the query, and some non-candidate points are removed. In addition, in the query

process, the nearest neighbor candidate set is filtered again according to the minimum circumscribed rectangle of Voronoi polygon and the query time is reduced.

The query I/O is tested under the data sets of different scales when the query point q is static. The test results of the I/O cost of executing queries are shown in Fig. 6 when the data sets scale is 20 000, 40 000, 60 000, 80 000, and 100 000. In Fig. 6, the I/O cost of the three algorithms are gradually increasing as the number of query points increases. The query I/O cost of VR _ DNN algorithm in this paper is relatively low.

Furthermore, the influence of data set size on VM _ DNN algorithm is tested when query point q moves at a uniform speed. Since Ref. [25] and Ref. [26] do not mention the movement of the query points, DNNBase + algorithm and R _ MULTI algorithm are called repeatedly.

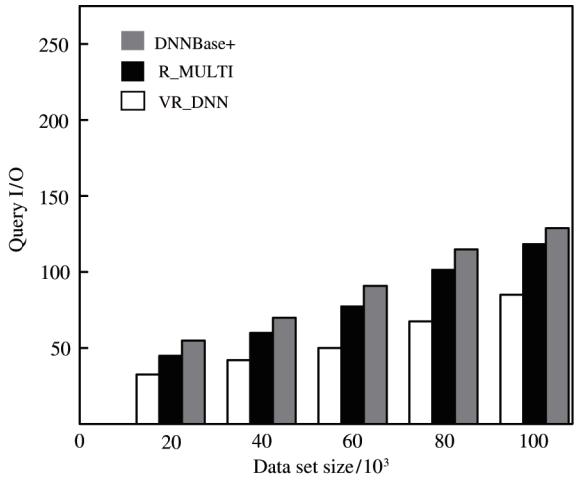


Fig. 6 Comparison of query I/O under different scale data sets when query point q is static

The query time is tested under the data sets of different scales when the query point q is moving at a uniform speed. When the data sets scale is 20 000, 40 000, 60 000, 80 000, and 100 000, the test results of time consumed are shown in Fig. 7. In Fig. 7, the running time of the three algorithms is gradually increasing as the number of data points increases. VM _ DNN algorithm in this paper consumes relatively less time in the query process. This is because the data set P is pruned before query. In the pruning process, the query range is narrowed by means of coordinate judgment and generation of the decision circle, and finally the real-time search is carried out quickly by using the properties of Voronoi diagram.

Then, the query I/O is tested under the data sets of different scales when the query point q is moving at a uniform speed. When the data sets scale is 20 000, 40 000, 60 000, 80 000, and 100 000, the test results of the I/O cost of executing queries are shown in Fig. 8.

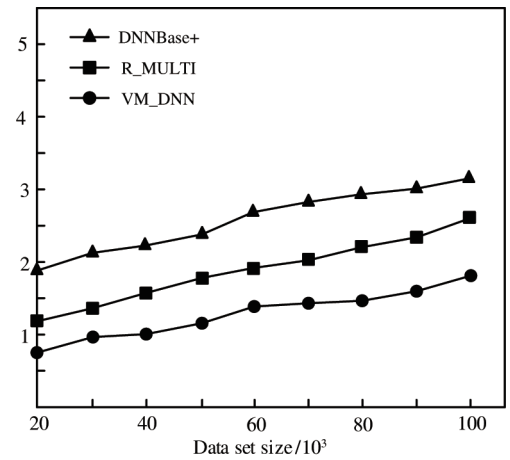


Fig. 7 Comparison of query time under different scale data sets when query point q is used for uniform motion

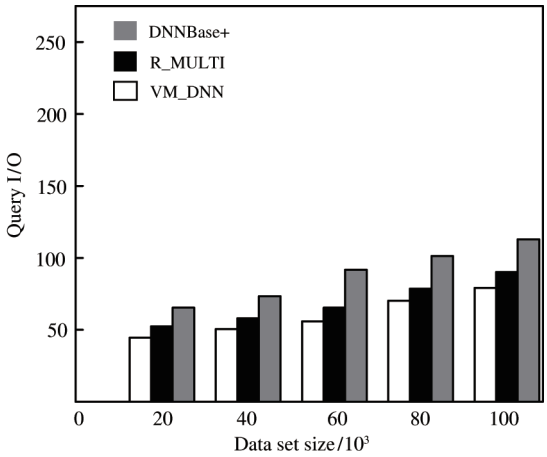


Fig. 8 Comparison of query I/O under different scale data sets when query point q is used for uniform motion

In Fig. 8, the I/O cost of the three algorithms are gradually increasing as the number of data points increases. The query I/O cost of VR _ DNN algorithm in this paper is relatively low. This is because DNNBase + algorithm and R _ MULTI algorithm can not calculate the nearest data points at a certain time from the query point in real time. After modifying and calling the original algorithm repeatedly, the amount of calculation increases.

The pruning efficiency will be tested in the data set with different density. From Fig. 9, it can be seen that the number of the candidates gradually increase as the density of the data points outside the query target grows. The pruning algorithm proposed in this paper receives relatively low affect from the density of data set.

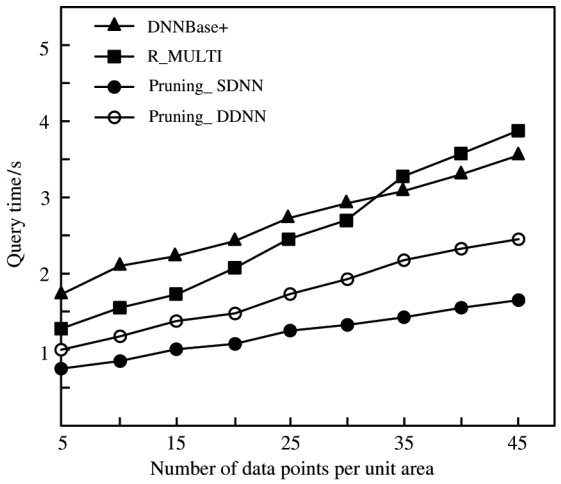


Fig. 9 The impact of data set density on query time

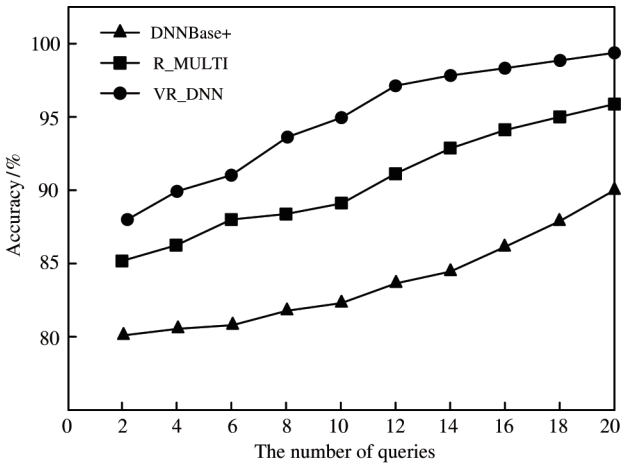


Fig. 10 Effect of queries number on accuracy

Then the performance of VR _ DNN algorithm is tested when the number of dynamic query points changes. Table 1 shows the performance comparison of VR _ DNN algorithm, DNNBase + algorithm and R _ MULTI algorithm when query points increase dynamically. In order to observe easily, five query points are added to the test. When the number of query points changes from 15 to 20, the running time of DNNBase + algorithm and R _ MULTI algorithm are longer than that of VR _ DNN.

Table 1 Performance comparison of VR _ DNN, DNNBase + and R _ MULTI

The number of query points	Algorithm	CPU time/s
15	DNNBase +	0.860
	R _ MULTI	0.727
	VR _ DNN	0.606
20	DNNBase +	1.079
	R _ MULTI	0.916
	VR _ DNN	0.718

Finally, the accuracy of the algorithm is tested.

When the data volume is 50 000 and other conditions remain unchanged, the accuracy of the three algorithms is tested in static and dynamic data sets respectively. As shown in Fig. 10, the accuracy of the three algorithms increases with the number of queries, VR _ DNN algorithm proposed in this paper can achieve a higher accuracy than DNNBase + algorithm and R _ MULTI algorithm.

6 Conclusions

A directional nearest neighbor query method is proposed based on Voronoi diagram for the nearest neighbor query problem in a specific geographic direction. This work studies two case, i. e. the query point is static and it does uniform motion. When the query point is static, the plane rectangular coordinate system is first combined with the east, south, west and north of the geographical position, and the positive and negative properties of the four quadrant symbols are used in the plane rectangular coordinate system for pruning. Then the directional nearest neighbor is determined by the minimum circumscribed rectangle of Voronoi polygon. When a query point moves at a uniform speed, at first, a plane rectangular coordinate system is established, after that a decision circle for pruning is generated. Finally the positional relationship is got between the query point q, the data points, and the Voronoi polygon region, so as to determine the directional nearest neighbor. The theoretical research and experiments show that the algorithm proposed in this paper has good performances.

References

[1] ROUSSOPOULOS N, KELLEY S, VINCENT F. Nearest neighbor queries[C]//ACM Sigmod International Conference on Management of Data, New York, USA, 1995: 71-79

[2] GÍSLI R H, SAMET H. Distance browsing in spatial databases[J]. *ACM Transactions on Database Systems* (TODS), 1999, 24(2):265-318

[3] TAO Y F, PAPADIAS D, SHEN Q. Continuous nearest neighbor search[C]//Proceedings of the 28th Very Large Data Bases International Conference, Hong Kong, China, 2002:287-298

[4] MOURATIDIS K, YIU M L, PAPADIAS D. Continuous nearest neighbor monitoring in road networks[C]//Proceedings of the 32nd Very Large Data Bases International Conference, Seoul, Korea, 2006:43-54

[5] PARK J H , CHUNG C W, KANG U. Reverse nearest neighbor search with a non-spatial aspect[J]. *Information Systems*, 2015,54: 92-112

[6] DAI Q Z, XIONG Z Y, XIE J, et al. A novel clustering algorithm based on the natural reverse nearest neighbor structure[J]. *Information Systems*, 2019, 84:1-16

[7] MOUTAFIS P, GARCIA-GARCIA F, MAVROMMATIS G, et al. Algorithms for processing the group *K* nearest-neighbor query on distributed frameworks[J]. *Distributed and Parallel Databases*, 2021, 39:733-784

[8] ZHANG L P, LIU L, LI S, et al. Group reverse nearest neighbor query based on Voronoi diagram in spatial databases[J]. *Journal of Frontiers of Computer Science and Technology*, 2016, 10(10): 1365-1375

[9] YI X , PAULET R, BERTINO E, et al. Practical approximate k nearest neighbor queries with location and query privacy[J]. *IEEE Transactions on Knowledge and Data Engineering*, 2016, 28(6):1546-1559

[10] YU Z Q, LIU Y, YU X H, et al. Scalable distributed processing of *k* nearest neighbor queries over moving objects[J]. *IEEE Transactions on Knowledge and Data Engineering*, 2014, 27(5):1383-1396

[11] KIM W, SHIM C, HEO W, et al. Moving view field nearest neighbor queries[J]. *Data and Knowledge Engineering*, 2019, 119:58-70

[12] ZHANG L P, LIU L, HAO X H, et al. Voronoi-based group reverse k nearest neighbor query in obstructed space[J]. *Journal of Computer Research and Development*, 2017, 54(4):861-871

[13] GAO Y, LIU Q, MIAO X, et al. Reverse k-nearest neighbor search in the presence of obstacles[J]. *Information Sciences*, 2016, 330: 274-292

[14] LI J J, LI Y X, XIA X F, et al. Continuous *k*-neighbor query for time dependent road network[J]. *Journal of Frontiers of Computer Science and Technology*, 2019, 13(5):788-799

[15] ERTUGRUL O F, TAGLUK M E. A novel version of *k*-nearest neighbor: dependent nearest neighbor[J]. *Applied Soft Computing*, 2017, 55:480-490

[16] LI S, LI B H, YU J X, et al. Probabilistic threshold *k*-ANN query method based on uncertain voronoi diagram in internet of vehicles[J]. *IEEE Transactions on Intelligent Transportation Systems*, 2021, 22(6):3592-3602

[17] TANG Y X, GUO W L, CUI Y L, et al. Research for the nearest neighbor query based on R^{BRT} tree[C]// The 9th International Forumon Strategic Technology, Cox's Bazar, Bangladesh, 2014:251-254

[18] DENG Y Z, XIANG L I, COMPUTER S O, et al. A *k*-nearest neighbor query algorithm for density grid-based index[J]. *Acta Electronica Sinica*, 2017, 45(2):376-383

[19] YI S, SHIM C, CHUNG Y D. Reverse view field nearest neighbor queries[J]. *Information Sciences*, 2017, 402: 35-49

[20] HIDAYAT A, YANG S Y, CHEEMA M A, et al. Reverse approximate nearest neighbor queries[J]. *IEEE Transactions on Knowledge and Data Engineering*, 2018, 30(2):339-352

[21] JAEMIN J, JINWOOK S, JEAN D F. PANENE: a progressive algorithm for indexing and querying approximate k-nearest neighbors[J]. *IEEE Transactions on Visualization and Computer Graphics*, 2020, 26 (2):1347-1360

[22] BHANDARI A, HASANOV A, ATTIQUE M, et al. Efficient processing of all nearest neighbor queries in dynamic roadnetworks[J]. *Mathematics*, 2021, 9:1137-1158

[23] LI G, FENG J , XU J. DESKS: direction-aware spatial keyword search[C]// The 2012 IEEE 28th International Conference on Data Engineering, Washington, USA, 2012:474-485

[24] MIAO X, GUO X, WANG H, et al. Continuous nearest neighbor query with the direction constraint[C]// International Symposium on Web and Wireless Geographical Information Systems, Kyoto, Japan, 2019: 85-101

[25] GUO X, YANG X. Direction-aware nearest neighbor query[J]. *IEEE Access*, 2019, 7:30285-30301

[26] LEE M J, CHOI D W, KIM S Y, et al. The direction-constrained k nearest neighbor query[J]. *GeoInformatica*, 2016, 20(3):471-502

[27] DONG T Y, YUAN L L, SHANG Y H, et al. Direction-aware continuous moving K-nearest-neighbor query in road networks[J]. *ISPRS International Journal of Geo-Information*, 2019, 8(9):379-400

[28] SACL J R, URRUTIA J. Voronoi Diagrams of Handbook on Computational Geometry[M]. Ottawa: Elsevier Science, 2006:201-290

LI Song, born in 1977. He received the Ph. D degree from Harbin University of Science and Technology. He is a professor at Harbin University of Science and Technology. His research interests include database theory and application, data mining, and data query.