International Journal of Innovative
Computing, Information and Control
Volume **15**, Number **4**, August **2019**

ICIC International ©2019 ISSN 1349-4198
pp. **1287–1307**

QUERY METHOD FOR NEAREST REGION OF SPATIAL LINE SEGMENT BASED ON HILBERT CURVE GRID

Liping Zhang, Lingling Ren, Xiaohong Hao and Song Li

College of Computer Science and Technology
Harbin University of Science and Technology
No. 52, Xuefu Road, Nangang District, Harbin 150080, P. R. China
zhanglptg@163.com

Received November 2018; revised March 2019

Abstract. *In order to solve the problem that the existing research results cannot effectively deal with the problem of the nearest region query of spatial line segment in the spatial database, the query method for the nearest region of spatial line segment based on Hilbert curve grids (HLNR) is proposed. The query method is mainly used to find the region which is the nearest neighbor of the line segment. We determine whether there are obstacles between the query line segment and the region set, and two HLNR query methods are proposed. One is the nearest region query method of spatial line segment based on Hilbert curve grid (HLNR query) in the barrier free environment, and the other is the HLNR query method in the obstacle environment (OHLNR query). The query method for HLNR is based on the Hilbert curve and grids characteristics. The query method for OHLNR can filter out a large number of non-candidates and non-effect obstacles in the pruning process. Theoretical research and experimental results show that the query method for nearest region of spatial line segment based on Hilbert curve grids has higher performance in searching.*

Keywords: Spatial database, Hilbert curve, The nearest neighbor query method, Spatial line segment

1. **Introduction.** The nearest neighbor query [1] is one of the most important operations in spatial database, and the nearest neighbor query in spatial database is widely used. Nearest neighbor queries are classified into many types: k nearest neighbor query [2], reverse k nearest neighbor query [3], line nearest neighbor query [4], group nearest neighbor query [5], continuous visual nearest neighbor query [6], the nearest neighbor query of mobile object [7], probability moving nearest neighbor query [8] and fuzzy data nearest neighbor query [9], etc.

In recent years, many researchers have studied the nearest neighbor query methods. In [10], the aggregate nearest neighbor (ANN) query method in the network environment is proposed, which is mainly divided into two stages: searching and pruning. Firstly, the NNs of each query point are calculated in a particular order, until all the query points find the common POI, then it can get the candidate POIs; secondly, according to the given aggregation function, the method can select the unqualified POIs based on the pruning strategy. These two processes are repeated until the remaining candidate points are used as final results. In [11], the method of Voronoi graph based on space grid is studied in order to efficiently query and update the nearest neighbor, and the related algorithms are proposed. In [12], a new query type maintenance k diversity nearest neighbor (MkDNN) algorithm is proposed which considers the availability of query results, also the incremental maintenance k diversity nearest neighbor (IMkDNN) algorithm is proposed, it can reduce

DOI: 10.24507/ijicic.15.04.1287

the query cost. In [13], the whole data space is divided into the grids of equal size, and there is no overlap between grids, points are arranged linearly in grids. Points and adjacent points in grids are accessed by the query point, and then we can get the final nearest neighbor result. In [14], a continuous nearest neighbor algorithm based on proxy is proposed. The algorithm uses distributed index to query the windows on Hilbert curve. In [15], the kNN queries based on Voronoi graph are proposed and it can enhance search efficiency. The above methods are mainly based on the spatial data points; however, when solving the query problem, many spatial objects which are abstracted into points will affect the accuracy of the query results in real life. For example, rivers, roads and so on are not suitable for abstracting into points in the process of research. In order to improve the accuracy of the query results, it can be abstracted into spatial line segments. For example, in [16], the line segment reverse k nearest neighbor in the network environment is proposed. It is mainly used to evaluate the influence scope of query object. In [17], the segment nearest neighbor query based on R*S tree is proposed, according to the creation of R*S tree to sort the distance between target line segment and the boundary rectangle, and then the line segment nearest neighbor query results are obtained. In [18], the nearest neighbor query of the plane segment and line segment is proposed. The method can solve the problem that some spatial objects cannot be abstracted into points.

However, not all objects can be abstracted as points or line segments in reality. For example, mountains and lakes are not suitable for abstracting into points or line segments; it can be abstracted into irregular shapes to ensure the accuracy of query results. In this paper, the query method for line segment nearest neighbor based on Hilbert curve grid is proposed for the first time. For example, tourists go on vacation (mostly for the seaside or mountain, etc.) and choose the nearest place of playing according to the location of the tourists (motorway or rail). Again, the earthquakes generally occur in mountains, and the mountains are not suitable for abstracting into lines, so it can be abstracted into irregular shapes; moreover, the supply station can be abstracted into lines. As a result, the location of nearest disaster can be located according to the supply station site that the survivors can get the rapid rescue and supply.

The existing methods cannot deal with the nearest neighbor problem that the query object is abstracted as a spatial line segment, and the target object is abstracted as the spatial region. The query for the nearest region of the spatial line segment is a new problem. According to whether there are obstacles between the query line segments and the region set, two cases of query method for HLNR are proposed. One is the query method for the nearest region of the spatial line segment based on the Hilbert curve grid (HLNR query) in the barrier free environment, and the other is the query method for HLNR in the obstacle environment (OHLNR query). Among them, it mainly includes three parts in the query method for HLNR: pre calculation, filtering and refining. The query method for HLNR is based on the Hilbert curve and grid characteristics, the pre calculation process can store the set of line segments and regions, and it can reduce the query time and optimize the scanning process. Then, the filtering process can cut out a large number of non-candidate regions, and it can optimize query process and decrease the running time. The refining process can refine the set of candidates in the filtering process, and the set of candidates is obtained. The query method for OHLNR can filter out a large number of non-candidates and non-effect obstacles in the pruning process. Then, the set of candidates is refined to obtain the more accurate candidates. Finally, the final result is obtained through filtering out the candidates.

Due to the poor effect of existing results on the nearest neighbor query in high-dimensional space, this paper proposes a query method of line segment nearest region based on Hilbert curve meshing. This method has high accuracy in processing line-based

nearest region query in high-dimensional space. The main contributions of this paper are as follows.

(1) The Hilbert curve and meshing method are introduced. The concepts and properties of adjacent regions are proposed.

(2) In the barrier-free environment, H value and region set R_s are cached in the pre-calculation process, and the algorithm HLNR_Pre_Cal is proposed. In the filtering process, the query line segments are filtered, and the relevant theorems and HLNR_Filter algorithm are proposed. Finally, the results of the filtering process are refined in the refining process, and the algorithm HLNR_Fine_Filter is proposed.

(3) Finally, this paper proposes a method of line segment nearest region query in obstacle environment. This method first determines whether the obstacle object has an impact on the query results. If it has an impact, it directly calls HLNR algorithm, and vice versa, it calls OHLNR algorithm.

2. Basic Definition and Properties.

Definition 2.1. [19,20] *Let H: $Rd \to I$ be a one-to-one mapping between D dimensional space and one dimensional space.*

Let line segment $L \in R^d$, $\{H_1(L), H_2(L), \ldots, H_i(L)\} \in I$, and the $\{H_1(L), H_2(L), \ldots, H_i(L)\}$ is the set of H value by line segment L.

Let region $R \in R^d$, $\{H_1(R), H_2(R), \ldots, H_i(R)\} \in I$, and the $\{H_1(R), H_2(R), \ldots, H_i(R)\}$ is the set of H value by region R.

According to the definition of Hilbert curve, we show the first and second order of Hilbert curves examples in Figure 1.

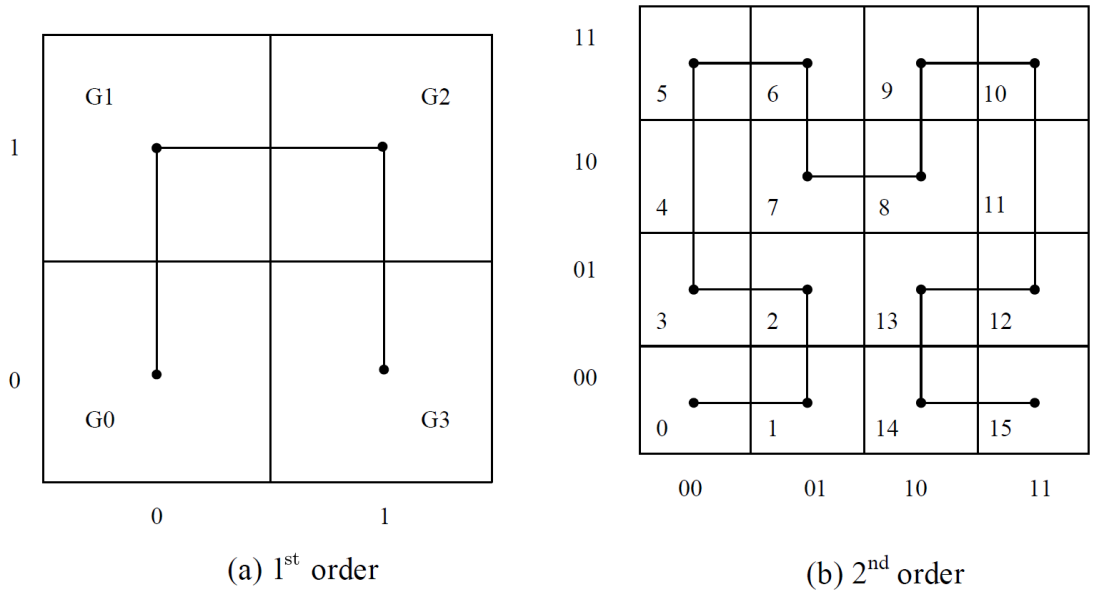

FIGURE 1. The 1st and 2nd order of Hilbert curves

Definition 2.2. [21] *The data space is divided into equal size grids, and there is no overlap between grids, the space filling curve passes through each grid, and then these grids are numbered in sequence. Each grid is accessed once and uncrossed. Finally, the spatial data points in the space filling curve are mapped into one-dimensional.*

In this paper, the concept of adjacent region is proposed, and the related properties and proofs are given.

Definition 2.3. *Given a Hilbert curves grid which is generated by a group of lines L and regions R. And the first layer element is constructed with the adjacent elements of the grid where the line segment l_i is located, that is the adjacent region of the line segment l_i.*

Lemma 2.1. *Given any line segment l_i in line L and region r_i in adjacent region, let $dist(l_i, r_i)$ be the distance between l_i and r_i, then, $dist(l_i, r_i) \leq r$. r is the farthest distance from l_i to adjacent region called a range threshold parameter.*

Proof: Assume there is a region r_i in the adjacent region of the line l_i. r is distance threshold. Then, $dist(l_i, r_i) \leq r$. Proof finished.

Definition 2.4. *The region away from the k^{th} $(k \geq 1)$ unit in the grid of the line segment l_i in any direction is the k levels adjacent region of the line segment l_i.*

Lemma 2.2. *Given any line segment l_i of L and region r_i of R, let $dist(l_i, r_i) > r$, and then the region r_i is in the k levels adjacent region.*

Proof: By Lemma 2.1, r_i exits in the adjacent region of l_i, so $dist(l_i, r_i) \leq r$. If $dist(l_i, r_i) > r$, then r_i exits in the k $(k > 1)$ levels adjacent region of l_i by Definition 2.4. Proof finished.

Definition 2.5. *Given the line $l \in L$, the region $r \in R_s$ and obstacle set O. L and R are visible only if the irregular graph formed by the L and R connection which do not intersect with any of the obstacles in the O, that is any $o \in O$, $[l, r] \cap O = \emptyset$.*

Definition 2.6. [17] *Given the line K and the point p, the points $k, k_i, k_j \in K$, $dist(p, k)$ indicates the distance from p to k, let the nearest distance between the point p and the line segment K be $dist(p, K)$, then $dist(p, K) = \{dist(p, k) | dist(p, k_i) \leq dist(p, k_j)\}$.*

Definition 2.7. *Given the line L and the region R, the points $l, l_i, l_j \in L$, the points $r, r_i, r_j \in R$, let $dist(l, r)$ be the distance from point l to point r. Let the nearest distance between the line L and the region R be $dist(L, R)$, which amount to compute the smallest distance between a point on the line L and a point on the region R, then $dist(L, R) = \{dist(l_i, r_i) | dist(l_i, r_i) \leq dist(l_j, r_j)\}$. If the line segment intersects with the region, their nearest distance is 0.*

Definition 2.8. *Given the obstacle set O, the line l and the region r, let $dist_o(l, r)$ be the obstacle distance between l and r. The $dist_o(l, r)$ is the length of the shortest obstacle path between l and r. When l and r are visible, the obstacle distance between l and r is their Euclidean distance, which is expressed as $dist_v(l, r)$.*

The query for the nearest region of spatial line segment is defined in Definition 2.9.

Definition 2.9. *Given a set of disjoint regions R, the region r_i is found in R, the distance from r_i to l_i is less than any other line segment in L, and thus, r_i is the nearest neighbor of the line segment of the l_i. The specific definition is as follows: $HLNR(R, L) = \{\exists r_i \in R, \exists l_i \in L | q \in HLNR(r_i, l_i)\}$.*

Figure 2 shows an example of the HLNR query. Inside, $l_1 \in L$, $R = \{r_1, r_2, r_3\}$, the solid line segments indicate the Euclidean distance from the line segment to the region r_i, and the dotted line segments indicate the parallel segment l_{1i} of the l_1. Through translation or extension line l_1, and l_1 intersects with the regions r_1, r_2, r_3 respectively at points p_1, p_2, p_3, there are $h_1 < h_2 < h_3$. Obviously, the query line l_1 has the nearest regions. And r_1 is the nearest region of the query line l_1.

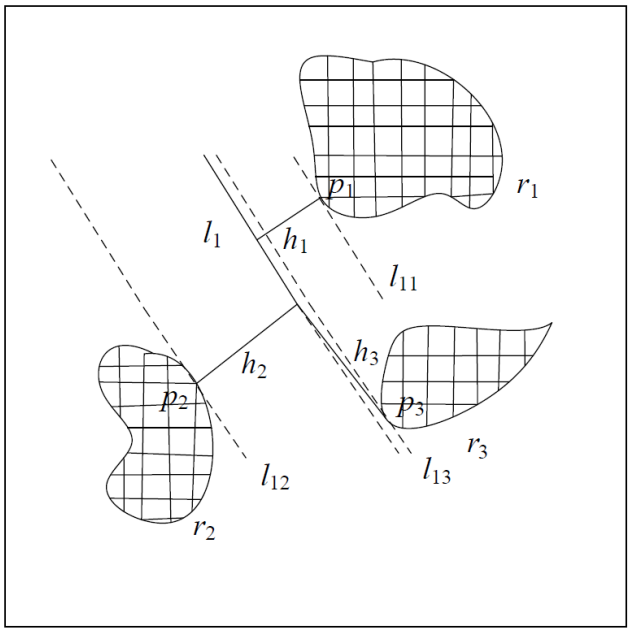

FIGURE 2. Example of HLNR query

3. **Nearest Region of Spatial Line Segment Based on the Hilbert Curve.** Depending on whether there are obstacles between the set of query line segments and regions, we divided the query for nearest region of spatial line segment based on Hilbert curve grid into the query for nearest region of spatial line segment in barrier free environment (namely HLNR query) and in the obstacle environment (namely OHLNR query). In this section, we filter the region set in filtering process. Discussing on whether the intersection of query lines and regions, we propose the algorithm HLNR_Filter and the theorems. Then according to the location between lines and regions, the refining process algorithm HLNR_Fine_Filter is proposed. Finally, HLNR Query in Obstacle Environment algorithm is proposed which includes pruning process and refining process.

3.1. **HLNR query in barrier free environment.** In the HLNR query process, a lot of data is useless, so we propose three steps, namely pre calculation, filtering and refining. According to the Hilbert curve grid characteristics, it can store the value of H of query line segments and the set of regions R_s in the pre calculation process. The process can reduce computation and decrease the running time, and then the HLNR_Pre_Cal algorithm of the pre calculation is proposed. In the filtering process, the set of regions is selected by the query line segment, and the related theorems are given. And the filter algorithm namely HLNR_Filter is proposed. In the refining process, the set of query results in filtering process is refined. And the refining algorithm namely HLNR_Fine_Filter is proposed. Finally, the set of accurate results is obtained.

3.1.1. *Pre-calculation process.* Firstly, the set of H values by the query line l_i is calculated, and then the set of H values and the adjacent region of the query line l_i are stored. Then, the set of the grid by the regions R_i is calculated. The process of the pre-calculation can greatly reduce the computation when there is a large amount of data.

Based on the above discussion, the algorithm of pre-calculation is given.

Firstly, the algorithm HLNR_Pre_Cal creates a line region Hilbert curve grid for the generated data of the region set R_s and query line l_i. And the endpoint coordinates of line l_i are stored in the Hilbert curve grid. At the same time, three arrays A_1, A_2, A_3 are initialized. According to the Hilbert curve grid characteristics, the H value is calculated

Algorithm 3.1. HLNR_Pre_Cal(L, R_s, X, Y)

Input: query line set L, Region set R_s, the coordinate of query line segment (X, Y).
Output: A_1, A_2, A_3.
Begin:

```
Create LRHC; /* Creating Hilbert curves grids of line segments. */
Let ((X_i, Y_i), (X_j, Y_j)) be the coordinate of query line segment l_i;
Let A_1, A_2, A_3 be three arrays;
for i = 0 to L.length do
    if (X_i == X_j && Y_i == Y_j) then
        (X_i, Y_i) is the coordinate of l_i;
        Calculate the H value of l_i.
        A_1 ← H(l_i);
            Calculate the adjacent region of l_i.
        A_2 ← N(l_i);
    end if
    else
        ((X_i, Y_i), (X_j, Y_j)) is the coordinate of l_i;
        (X_m, Y_m) = ((X_i + X_j)/2, (Y_i + Y_j)/2) is the midpoint coordinate of l_i;
        if (X_i == X_m && Y_i == Y_m || X_j == X_m && Y_j == Y_m) then
            H(l_i) = {H_i, H_j};
        end if
        else
            H(l_i) = {H_i, H_m, H_j};
    A_1 ← H(l_i);
end for
for j = 0 to R_s.length do
    Calculate the set of grids by the region r_i namely H(r_i);
    A_3 ← H(r_i);
end for
return A_1, A_2, A_3;
End
```

by the coordinate of the line segment l_i, and the results are stored in the A_1. Then, the adjacent region of l_i is calculated according to the H value, and the results are stored in A_2. Finally, the set of the H values by region R_s is calculated, and the results are stored in A_3. Thus, A_1, A_2, A_3 are obtained. The algorithm creates the Hilbert curve grid which time complexity is $O(nd)$, and the total time complexity of the both *for* loops are $O(2n)$, so the algorithm time complexity is $O(n)$.

3.1.2. *Filter process.* The process can filter out a large number of non-candidates, and the more accurate set of nearest regions of segments is obtained. Firstly, the location of the query line l_i is determined. Then, the adjacent regions of the query line l_i are determined. Finally, we determine whether there are regions r_i in the adjacent regions. According to the characteristics of Hilbert curves and the query for nearest region of line segment algorithm, we can discuss it in two cases.

(1) Intersection of Query Line Segment and Regions r_i.

Theorem 3.1. *Given the set of query line segments L and the regions R_s, the line segment $l_i \in L$, the region $r_i \in R_s$. If the line segment l_i and the region r_i are intersected, the nearest neighbor region of the query line l_i is r_i.*

Proof: Because the line segment l_i and the region r_i are intersected, the shortest distance between the line segment l_i and the region r_i is 0, that is, the region r_i is the nearest region of line l_i. As shown in Figure 3, the query line l_1 and the region r_1 are intersected at point m, the shortest distance between the line segment l_1 and the region r_1 is 0, and the nearest region of the query line l_1 is r_1. Proof finished.

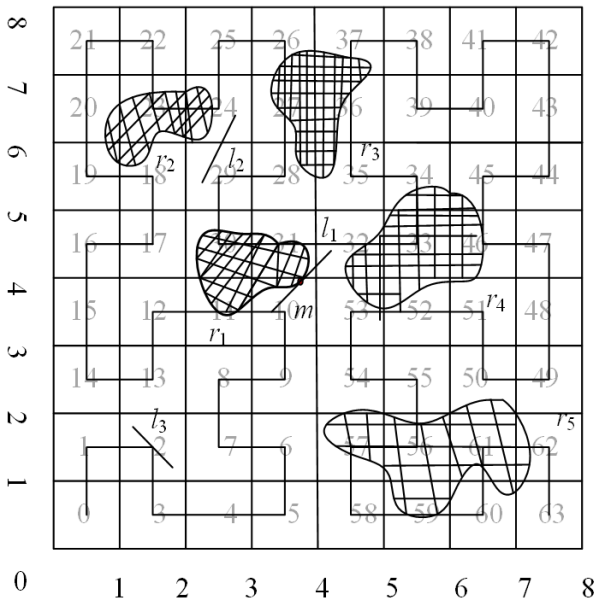

FIGURE 3. Example diagram based on Theorem 3.1

(2) Disjoint of Query Line Segment and Regions r_i.

Theorem 3.2. *In the Hilbert curve grid, if any query line l_i has the nearest region, the region will be in k ($k \geq 1$) levels adjacent regions of l_i.*

Proof: Suppose the query line l_i has the nearest neighbor region. By Lemma 2.1, there must be a region r_i in the k levels adjacent regions of the line l_i and the distance between l_i and r_i is r. Thus, if the query line l_i has the nearest region, the region will be in k ($k \geq 1$) levels adjacent regions of line l_i. Proof finished.

If the query line l_i and the region r_i are not intersected, it is also necessary to determine the location of the region r_i, so it is discussed in two cases.

1) r_i in Adjacent Region. By Algorithm 3.1, the H value of the query line l_i and the set of adjacent regions are obtained, and the set of regions r_i is also obtained. By the location of the query line segment l_i, the set of nearest neighbor region can be determined, and many non-candidates are removed and the running time is shortened. As shown in Figure 3, the set of H of query line segment l_2 is $\{24, 29\}$, then the adjacent regions of the line segment l_2 are $\{17, 18, 22, 23, 25, 26, 27, 28, 30, 31\}$, and the regions that intersect with the adjacent regions are $r_1\{10, 11, 30, 31\}$, $r_2\{18, 19, 20, 23, 24\}$, $r_3\{26, 27, 28, 35, 36, 37\}$. The region r_4 does not intersect with the adjacent regions of l_2, so r_4 is removed. Finally, the candidate results set of the nearest neighbor regions of line l_2 is $\{r_1, r_2, r_3\}$.

2) r_i not in Adjacent Region. From Theorem 3.2, we can know that if there is no region r_i in the adjacent region of the line segment l_i, then r_i must exist in the k ($k > 1$) levels adjacent regions of the line l_i. The nearest neighbor regions of the line segment l_i are obtained by looking for the k levels adjacent regions.

The core idea of the filtering process algorithm: Firstly, by the result of Algorithm 3.1, the location of the query line l_i is determined. If the region r_i is in the adjacent regions

of the line segment l_i, then r_i is one of the candidate results of nearest neighbor regions of the line segment l_i. If the region r_i is not in the adjacent regions of the line segment l_i, then search the k ($k > 1$) levels adjacent regions of the line segment l_i, by Theorem 3.2, we can cut all regions in greater than k levels adjacent regions, and thus, a large number of non-candidates can be filtered out. If there is no such step, it is necessary to calculate the distances between the query line segment and all the regions in the data set, which not only increases the amount of query computation, but also reduces the query efficiency. In short, we can get the more precise candidate results, and the efficiency of query algorithm can be improved by Theorem 3.2.

Based on the above discussion, the filtering algorithm is given.

Algorithm 3.2. HLNR_Filter(l_i, R_s)

Input: query line segment l_i, Region set R_s.
Output: the set of the nearest neighbor region R_l.
Begin:
1: $R_l \leftarrow \emptyset$;
2: Call HLNR_Pre_Cal(); /* To import all the relevant data of line segments l_i and region r_i */
3: for $m = 1$ to R_s.length do
4: if l_i intersects with R_m then
5: $R_l \leftarrow R_m$; /* R_m is the nearest neighbor region of the query line l_i */
6: end if
7: else if l_i does not intersect R_m then
8: if R_m is in the adjacent region of l_i then
9: $R_l \leftarrow R_m$;
10: end if
11: else if R_m is in the k levels adjacent region of l_i then
12: for $j = 2$ to $k + 1$ do
13: $R_l \leftarrow R_j$;
14: end if
15: end if
16: end for
17: if the query results have only one then
18: return R; /* R is the final query result */
19: end if
20: return R_l;
End

Firstly, by Theorem 3.1, the set of regions R_s is filtered. Then, the candidate results are filtered again from Theorem 3.2. The process is discussed in two cases. Firstly, r_i is in adjacent area, and then the set of regions R_l is updated. Secondly, the r_i is not in the adjacent region, then the k levels adjacent regions of l_i are detected until the region r_i is found, and finally, r_i is added into the R_l. The R_l is a set of candidate results that excludes non-candidates. Moreover, the time complexity of the algorithm is $O(n)$.

3.1.3. *Refine process.* The refining algorithm is proposed for dealing with the set of candidates R_l. Firstly, we calculate the shortest distances between the query line segment and R_l, then the shortest distances are compared, and finally the nearest neighbor query result of the line segment is obtained.

Theorem 3.3. *Give a query line l_i and the adjacent regions r_i of l_i. Translating l_i and intersecting with adjacent region at point p_i, then $dist(p_i, l_i) \leq dist(p_j, l_i)$, $p_j \in r_i$.*

Proof: Let the line segment l_3 be in the Hilbert curve grid in Figure 4, so the nearest region of line l_3 is r_4. Let the nearest region of line l_3 be not r_4, but be r_5. Translating l_3 to l_3' and l_3' intersects with region r_4 at point p_4, the line segment l_3 is extended to r_5 at point p_5. Let the endpoints of line segment l_3 be a and b. Then, $dist(r_4, l_3) = dist(p_4, l_3)$; $dist(r_5, l_3) = dist(b, p_5)$. We can connect p_4 to p_5, connect p_4 to the point b, and $dist(p_4, p_5) + dist(p_4, b) > dist(b, p_5)$ is obtained. By the Hilbert curve grids characteristics, $dist(p_4, p_5) > dist(p_4, r_4)$ is obtained. The reason is that the nearest region of line l_3 is r_5, then $dist(b, p_5) < dist(p_4, r_4)$. By the nature of a triangle, $\angle\alpha > 90°$, the $dist(p_5, b)$ is the longest side of a triangle. Contradicted with the above conclusions, the r_4 is closer to the line l_3 than the r_5. On the whole, r_4 is the nearest region of line l_3, which is inconsistent with the hypothesis. Proof finished.

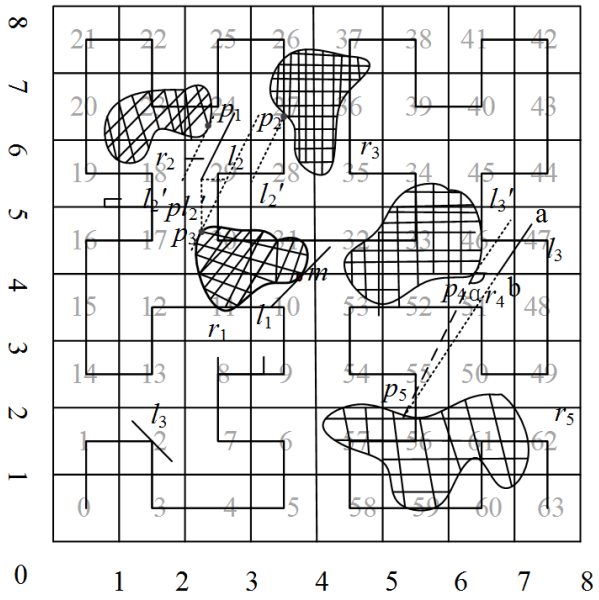

FIGURE 4. Schematic diagram of region r_i in different directions of line segment

Theorem 3.4. *In the Hilbert curve grids, the distances from the query line l_i to its k levels adjacent regions are less than the distances from the $k+1$ levels adjacent regions to the l_i.*

Proof: By the Hilbert curve grids characteristics, the distances from the query line l_i to its $k+1$ levels adjacent regions must be passed through the k levels adjacent regions. Thus, the distances from the l_i to its k levels adjacent regions are less than the distances from the $k+1$ levels adjacent regions to the l_i.

The line segments have four directions which are up, down, right and left. When region r_i is in different directions of line segments l_i, the translation method is different. The above discussion is divided into two cases.

(1) When the region r_i is on the left (right) side of the query line. When the region r_i is on the left (right) side of the query line, by the nature and structure of line segments, we translate left (right) line segment, and the line after translating that intersects with the region at point p, and then passes the point p to do the vertical line of the query line, and lastly the distance between the region and the query line segment is calculated.

(2) When the region r_i is on the upper (lower) side of the query line. When the region r_i is on the upper (lower) side of the query line, then, the line segment is extended up (down), and translating left (right) and the line after translating that intersects with region at the point p, and the upper (lower) endpoint of the query line is connected with p, which is the nearest distance between the region and the query line segment.

According to Theorems 3.3, 3.4, the core idea of refining algorithm is proposed. Firstly, the set of query results is initialized, and it calls the filtering algorithm. Then, the set of candidates R_l is obtained. Thirdly, the shortest distance is calculated by Definition 2.7. Finally, the set of candidates R_l is refined. Moreover, the final result is obtained.

Based on the above discussion, the refining algorithm is given.

Algorithm 3.3. HLNR_Fine_Filter(l_i, R_s, R_l)

Input: query line segment l_i, region set R_s, the nearest regions candidate R_l.
Output: the nearest region R.
Begin:
$R_l \leftarrow \emptyset$;
$dist(r, l) \leftarrow \emptyset$;
$R_l \leftarrow$HLNR_Filter(l_i, R_s);
while (R_l.length > 0) do
    Let the end points of the l_i be a, b;
        if the region r_i is between a and b then
            Translating l_i and intersects with the region r_i at point p_i;
            $dist(r_i, l_i) = dist(p_i, l_i)$;
        end if
        else if the region r_i is on top of a then
            Reverses extension l_i;
            if reverses extension of l_i to r_i at point p_i then
                $dist(l_i, r_i) = dist(a, p_i)$;
            end if
        end if
    else if the region r_i is below b then
            extends l_i;
            if l_i and r_i are intersected at point p_i then
                $dist(l_i, r_i) = dist(b, p_i)$;
            end if
            else if l_i does not intersect with r_i then
                Translating l_i and intersects with r_i at point p_i;
                $dist(l_i, r_i) = dist(b, p_i)$;
            end if
    end if
        $dist(l, r) \leftarrow dist(l_i, r_i)$;
    Compare distance between query line segment and each region r and then compute the minimum distance
        $R \leftarrow r$;
        R_l.length $\leftarrow R_l$.length $- 1$;
end while
return R;
End

Firstly, the HLNR_Fine_Filter algorithm initializes the results set and calls Algorithm 3.2 to obtain the set of candidate regions R_l. By the nature of the line segment, the shortest distance can be calculated in two cases. The first is the region between the ends of the line; the second is the region beyond the ends of the line. Suppose that the two endpoints of the line segment are a, b, and the line l_i and the region r_i are intersected at point p_i. If the region exists in between a and b, then $dist(r_i, l_i) = dist(p_i, l_i)$. If the region does not exist in between a and b, then $dist(r_i, l_i) = \min(dist(a, p_i), dist(b, p_i))$. In that case, the shortest distance from the nearest region and the query line l_i is obtained. Finally, the accurate query result is obtained. And the time complexity is $O(n)$.

3.2. **HLNR query in obstacle environment.** In Section 3.1, it studies the method for HLNR query in the barrier free environment, but there are many obstacles between two objects in real life. The query methods in obstacles environments are given in this section, which is divided into two steps: the pruning process and the refining process. Firstly, the pruning process can filter out a large number of non-candidates and the obstacles which not affect the query results. Then, we refine the set of candidates. Finally, the set of the results is obtained.

3.2.1. *Pruning process.* The main work of the pruning process is to optimize the query line segments, and then prune a large number of non-candidates and no affected obstacles, and the set of candidates is obtained. The obstacles are abstracted into line segments. Firstly, Algorithm 3.2 which can filter out a large number of non-candidates is called. By the Hilbert curve grids characteristics, an efficient pruning strategy is obtained. Then, the set of regions is determined by the pruning strategy. If the condition is not satisfied, then it is pruned. The remaining regions are constituting as the set of candidates. The pruning process can filter out a large number of non-candidates which greatly reduces the scope of query and improves the query speed.

In this section, the pruning strategy theorems and relevant proofs are given firstly.

Theorem 3.5. *The nearest region $NR(l_i)$ of the query line l_i exists in the k levels of adjacent regions, so the adjacent regions of $NR(l_i)$ which are larger than k levels and obstacles are pruned.*

Proof: By Theorem 3.3, any line segments l_i is in the Hilbert curve grids. Suppose the nearest region of l_i is r, the k levels adjacent regions r_k of l_i that $r_k \in NR_1(l_i) \ldots \cup NR_k(l_i)$ (k is an integer, $k \geq 1$), there is an r_k in the k level adjacent of l_i. Therefore, $k+1$ and higher than $k+1$ levels adjacent regions of l_i may not exist in the nearest region of l_i. $r_j \in NR_n(l_i)$ $(n > k)$, that is, r_j exists in the k levels adjacent regions of l_i. When there are obstacles, the distance between r_j and l_i must be greater than the Euclidean distance of them, so the nearest region of l_i is not r_j. In conclusion, in obstacle environments, the nearest region of l_i exists in the k levels adjacent regions, so we prune regions and obstacles beyond the k levels. Proof finished.

The main idea of the pruning algorithm is proposed: by Theorem 3.2 and Theorem 3.3, a large number of non-candidates and obstacles are pruned, and the candidate set R_l is obtained. Firstly, call Algorithm 3.2, and then the set of obstacles is put into the Hilbert curve grids, then, by Theorem 3.3, the regions in the k levels adjacent regions of the $NR(l_i)$ are putted into the candidate set R_l, and the regions in higher than the k levels adjacent regions are pruned. Finally, the set of candidates and the new obstacles are obtained, as shown in Figure 5.

Based on the above discussion, the pruning algorithm is given.

Firstly, to get the nearest region R_l of the query line l_i, the OHLNR_Filter algorithm initializes the set of candidates and calls Algorithm 3.2. Then, we must determine whether

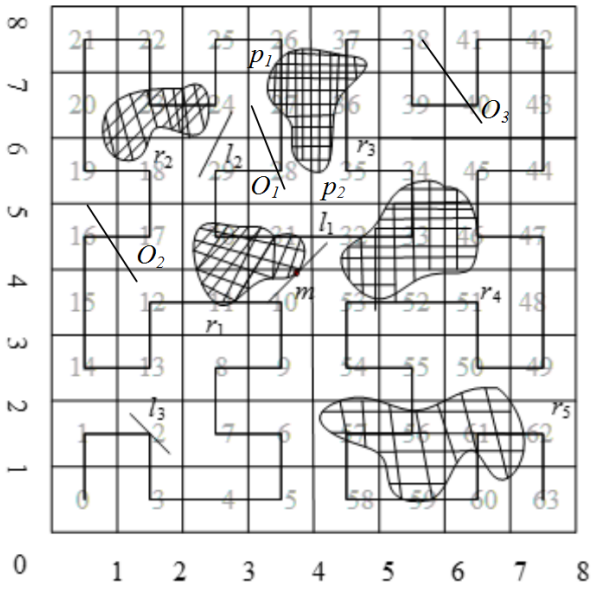

FIGURE 5. A schematic diagram based on Theorem 3.5

Algorithm 3.4. OHLNR_Filter(L, R_s, O)

Input: query line segment set L, region set R_s, obstacle set O.
Output: the nearest region set R_l, the set of obstacles after pruning O_{new}.
Begin:
1: $R_l \leftarrow \emptyset$;
2: $l_i \in L$;
3: $o \in O$;
4: Let coordinate of query line segment be (X, Y);
5: $NR(l_i) \leftarrow$ HLNR_Pre_Cal(L, R_s, X, Y); /* record the adjacent region of l_i */
6: $R_l \leftarrow$ HLNR_Filter(l_i, R_s);
7: for each $o \in O$ then
8: if $o \in NR(l_i)$ then
9: $O_{new} \leftarrow O_{new} + o$;
10: end if
11: end for
12: return R_l, O_{new};
End

there is an obstacle in the obstacle set O which exists in the adjacent region of the query line segment. If there is an obstacle between them, the algorithm updates the O_{new} as a new set of obstacles. The set of nearest regions R_l and the new obstacles O_{new} are obtained. And the time complexity is $O(n)$.

3.2.2. *Refining process.* In refining algorithm, the set of regions R_l in the nearest regions is refined. And then, the more accurate nearest neighbor region is obtained. In this section, Theorem 3.6 and Theorem 3.7 are presented.

Theorem 3.6. *Given a query line segment l_i, a region r_i, and we connect the endpoints of l_i with the vertices of the region r_i to form an irregular graph D. There is an obstacle set O, $o_i \in O$. Let $dist_v(l_i, r_i)$ be the shortest visible distance between l_i and r_i, then $dist_v(l_i, r_i) < dist_o(l_i, r_i)$.*

Proof: As shown in Figure 6, given a query line segment l_2, a region r_3, there is an obstacle O_1 between l_2 and r_3. According to Definition 2.5, the visual distance between l_2 and r_3 is expressed as $dist_v(l_2, r_3) = dist(a, d)$. By Definition 2.8, $dist_o(l_2, r_3) = dist(b, c) + dist(c, e)$. To get quadrilateral $adbe$, we connect b and e, then $dist(a, d) < dist(b, e)$. By the characteristics of the triangle, there is $dist(b, e) < dist(b, c) + dist(c, e)$. Therefore, it is concluded that $dist_v(l_2, r_3) < dist_o(l_2, r_3)$. Proof finished.

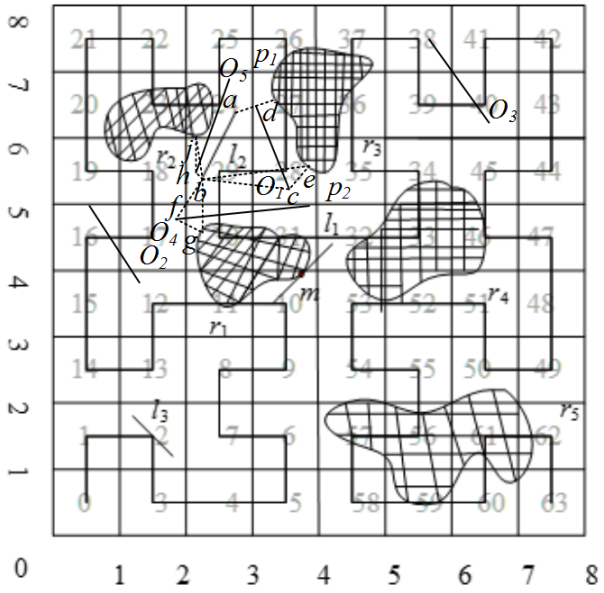

FIGURE 6. Schematic diagram of the refining process

Theorem 3.7. *In the adjacent regions NR of the query line l_i, the shortest visual distance does not exist in between the query line segment l_i and the set of nearest regions r_i, r_j. The number of grids from r_i to l_i is denoted as P, and the number of grids from r_j to l_i is denoted as Q. If $P < Q$, then $dist_o(l_i, r_i) < dist(l_i, r_j)$.*

Proof: Given a query line segment l_2, the nearest region set is $\{r_1, r_2, r_3\}$. As shown from Figure 6, it is not visible between $\{r_1, r_2\}$ and l_2. The obstacle distance from l_2 to r_2 is $dist_o(l_2, r_2) = dist(b, h) + dist(h, j)$, the number of grids that is passed from l_2 to r_2 is 2. The obstacle distance from l_2 to r_1 is $dist_o(l_2, r_1) = dist(b, f) + dist(f, g)$, and the number of grids is 3. According to Algorithm 3.1, $dist(b, j) < dist(b, g)$. By the triangle characteristic, $dist(b, j) < dist(j, h) + dist(h, b)$, $dist(b, g) < dist(b, f) + dist(f, g)$. In summary, $dist(l_2, r_2) < dist(l_2, r_1)$. Proof finished.

By Theorem 3.6 and Theorem 3.7, the refining algorithm is proposed. Firstly, the scopes of querying by the candidate regions of R_l and the candidate obstacles of O_{new} are determined. Then we determine both the region of R_l and the query line l_i which are affected by obstacles. Then the set of R_l is refined. Finally, the exact query results R is obtained.

Based on the above discussion, the refining algorithm is given, such as Algorithm 3.5.

Firstly, the query results R is initialized in the algorithm OHLNR_Refine. We call Algorithm 3.4 and the scope of searching are determined by R_l and O_{new}. The locations between each region in the nearest region and the query line l_i that are affected by obstacles are determined. If the shortest visible distance between query the line segment l_i and the region r_i, the obstacle distance from l_i to r_i is the shortest visual distance from l_i to r_i, and the obstacle distance is added into the set of final query results R. If there

Algorithm 3.5. OHLNR_Refine(R_l, O_{new}, l_i)

Input: the nearest region set R_l, new obstacle set O_{new}, query line segment l_i.
Output: the nearest region in an obstacle environment R_o.
Begin:
1: $R \leftarrow \emptyset$;
2: $P[] = 0$;
3: $count = 0$;
4: R_l, O_{new} ←OHLNR_Filter(L, R_s, O);
5: Determines the scope of searching by R_l and O_{new};
6: for each $r_i \in R_l$
7: if it is visible between l_i and r_i then
8: $dist_o(l_i, r_i) = dist_v(l_i, r_i)$;
9: $R \leftarrow R + r_i$;
10: end if
11: else if it is not visible between l_i and r_i then
12: Calculates the grids from l_i to r_i;
13: $P[i] = count + 1$;
14: end if
15: if $P[i] < P[i + 1]$ then
16: $dist_o(l_i, r_i) < dist_o(l_i, r_{i+1})$;
17: $R \leftarrow R + l_i$;
18: end if
19: end for
20: return R_o;
End

is no visible distance between the query line l_i and the region r_i, we compare the number grids from l_i to r_i. And the relatively short distances are added into the set of results R. Finally, the accurate result set R is obtained. And the time complexity is $O(n)$.

4. **Experiments and Analysis.** In this section, the performances of the algorithms are analyzed by experiments. The query for nearest region of spatial line segments is a new kind of spatial query problem. To solve this problem, the query algorithm for nearest region based on Hilbert curve grids is proposed for the first time in this paper. The exiting algorithms for nearest neighbors are based on the points as the query objects. In this paper, the algorithms for nearest region query based on Hilbert curve grids are proposed for the first time which use line segments as query objects. There are a great deal of different methods for the nearest neighbors that use points as query objects; therefore, the exiting methods cannot be compared directly with the method in this paper. In order to get the comparison algorithms, we use the SI-tree LNN algorithm proposed in [4], the R*S-tree based on LNN algorithm proposed in [17] and the LNN_Search based on R-tree algorithm proposed in [18] respectively, and then the three algorithms are appropriately adjusted. The sets of regions and obstacles are added into the three algorithms. We named these three LNN algorithms as HLNR_SI_LNN, HLNR_R*S_LNN and HLNR_R_LNN respectively. Their main idea is that the region sets R_s and the obstacle sets O are added in the Hilbert curve grids, and the LNN algorithm is called the query method for nearest region of the line segment. Firstly, we determine whether there are obstacles between the line segments and the regions. If there are not obstacles between the segments and the regions, the LNN algorithm is called for a line segment in the query lines, and the nearest

region of the query line segment is obtained. Query results are added to the set of query results. If there are obstacles between the line segments and the regions, the obstacle distance is expressed by the definition of the line segment obstacle distance, and then the query for nearest region of line segments in the obstacle environment is obtained. These three algorithms are compared with the algorithms in this paper.

Experimental environment configuration is 2.5GHz, CPU, 8G memory, 500G hard disk, Windows 7 operating system. The experimental data set uses road network information from San Francisco and California [22]. During the experiment, the data distribution is adjusted appropriately, two endpoints (a, b) of the line segment are randomly selected in the grid, and the length of the line segment is length (l). Let $l_{\min}$ be the shortest allowable distance between the query line segment and its adjacent area, and $l_{\max}$ the maximum allowable distance between the query line segment and its adjacent area. Random variables are used as the distance between lines and regions called *dist* and the angle θ between line segments and regions, and the distributions of *dist* and θ are uniform. The H values of line segments and regions in the grid can be obtained by multiple sampling of random vector groups $\{dist, \theta\}$ ($l_{\min} = 0$, $l_{\max} = 1.6$ μm). The experimental targets are CPU time and I/O cost, and then the average query time of the 200 times is calculated. Firstly, the experiment tests the influence of the number of query segments and region sets on the CPU time and the cost of I/O of the HLNR query algorithm (HLNR) in the barrier free environment. Then we test the impact of the number of query lines, the number of regions and the number of obstacles on the CPU time of the HLNR query algorithm (OHLNR) in an obstacle environment. Finally, the accuracy of the four algorithms is tested in an obstacle environment.

Firstly, the effect on the number of query segments on the CPU time is analyzed. The numbers of regions are 5000, as shown in Figure 7, the CPU time of the four algorithms is increasing as the number of query segments is increasing. Among them, the rising trend of CPU time in HLNR_R_LNN algorithm and HLNR_R*S_LNN algorithm are the most significant, and the rising trends in HLNR_SI_LNN and HLNR algorithm are relatively gentle, but the CPU time of the HLNR_SI_LNN algorithm is still higher than the HLNR algorithm in general. Because the HLNR_SI_LNN algorithm does not have the pre-calculation process, the subsequent process needs to calculate the query line and the

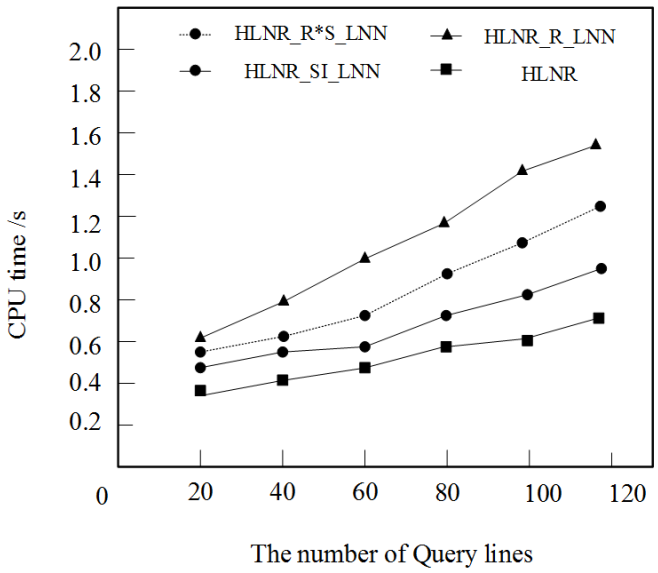

FIGURE 7. Effect of the number of query lines on CPU time

coordinates of the region, and it wastes a lot of time. The nearest regions of each query segment are calculated in the HLNR_R_LNN algorithm and the HLNR_R*S_LNN algorithm, and both of them cause overlapping search regions and slowing down the query speed. Pre-calculation on the query segments and regions set is proposed in the HLNR algorithm, it can reduce the range of searching and improve the speed, so the algorithm's time complexity is slightly lower than the three comparison algorithms. From the above discussion, we can conclude that the HLNR algorithm is superior to HLNR_R_LNN, HLNR_R*S_LNN and HLNR_SI_LNN algorithms in general.

Figure 8 shows the impact of the number of line segments on the costs of the I/O. With the increasing of the number of the line segments, the costs of I/O of four algorithms are increasing. Because the algorithm HLNR includes pre calculation, filtering and refining algorithms, it can reduce the amount of computation and a large amount of irrelevant data. Therefore, the growth trend of I/O cost is slow, and obviously lower than the three groups of comparison algorithms.

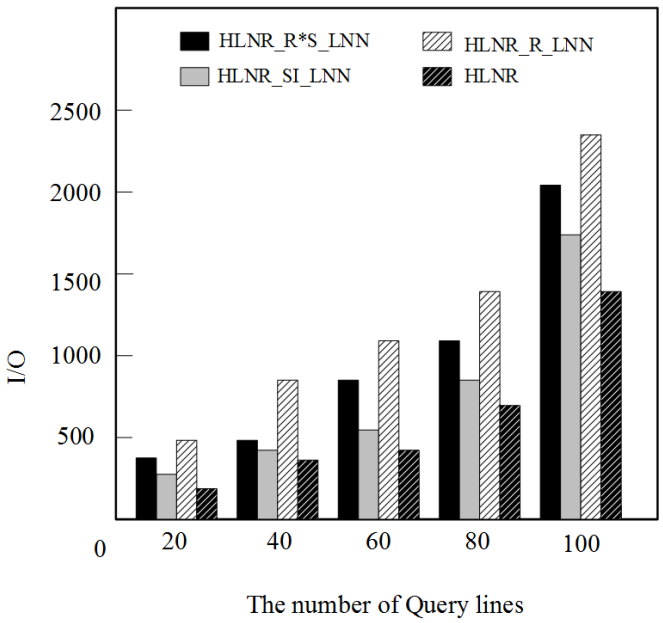

FIGURE 8. Effect of the number of query lines on I/O costs

Figure 9 shows the influence of the number of the regions on the CPU time in the four algorithms. The number of query lines is 50. The CPU time of these four algorithms is rising as the number of regions becomes larger. Among them, the CPU time of three comparison algorithms is raising obviously, but HLNR algorithm trend is gentler. The HLNR algorithm reduces the regions in the filtering process, and then filters the nearest regions again in the refining process which reduces the query time effectively. The HLNR_R_LNN algorithm and the HLNR_R*S_LNN algorithm waste a lot of time on searching regions. Although HLNR_SI_LNN algorithm has a pruning process, there is no pre-calculation for the regions. When the number of regions becomes larger, the running time is increasing obviously in the algorithm HLNR_SI_LNN. Above all, the HLNR algorithm is better.

Figure 10 shows the influence of the number of the regions on the costs of I/O in the four algorithms. With the increasing of the number of regions, the I/O costs of these four algorithms are raising. However, the trend of the HLNR algorithm is slower than the other three algorithms. Because the HLNR algorithm stores the data into the cache in the pre-calculation process which can reduce a large amount of calculation time, and then

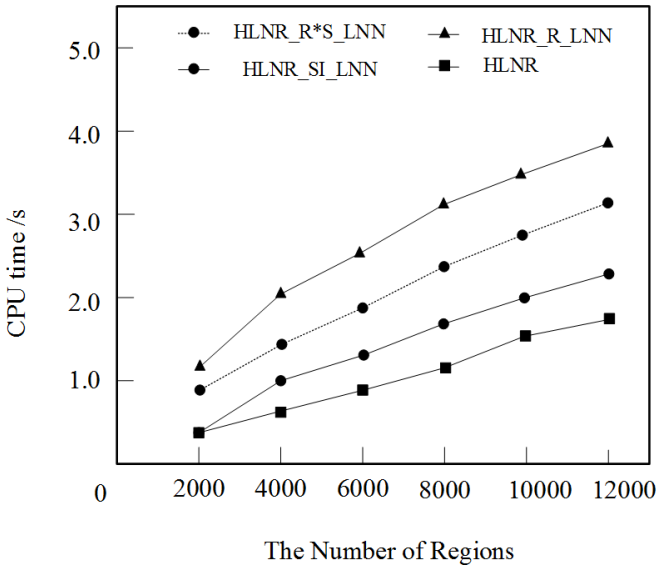

FIGURE 9. Effect of region numbers on CPU time

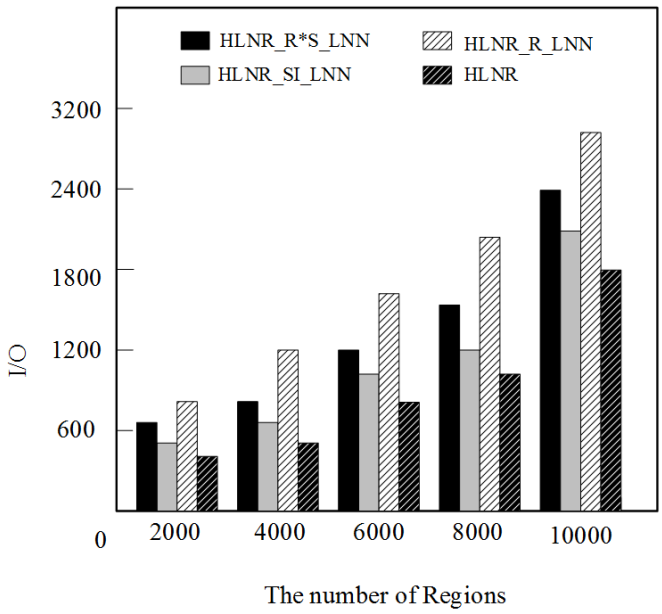

FIGURE 10. Effect of the number of regions on I/O cost

the algorithm filters out a large number of unrelated region sets in the filtering process, the running time is reduced. As a result, the HLNR algorithm is better than the other three algorithms.

Furthermore, the influence of the number of query segments on CPU time is analyzed. The number of the set of regions is 5000, and the number of obstacles is 1000. As shown in Figure 11, the four algorithms are rising. However, the CPU time of the HLNR algorithm is increasing slowly. Because the HLNR_R_LNN algorithm and the HLNR_R*S_LNN algorithm cause regions overlapped when searching the nearest regions of each query line segment, it wastes a large amount of querying time and reduces the performance of the algorithms. The HLNR_SI_LNN algorithm cannot process the data sets in advance, so that the account of calculation is too large and wastes a lot of time. The HLNR algorithm

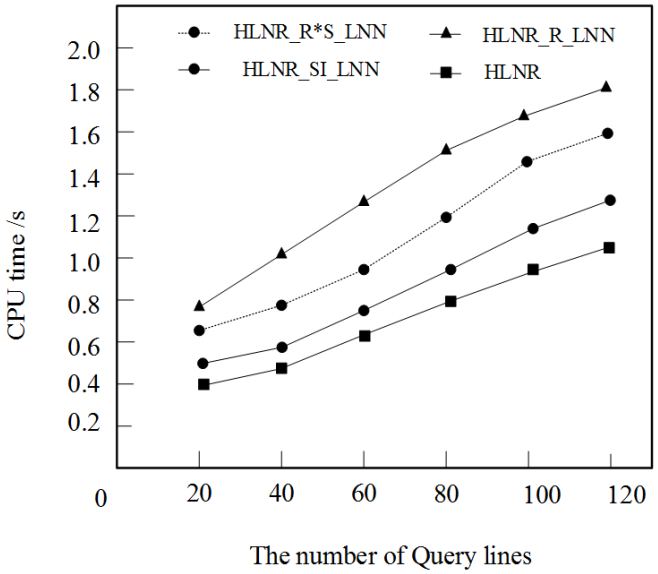

FIGURE 11. Effect of the number of query lines on CPU time in obstacle environment

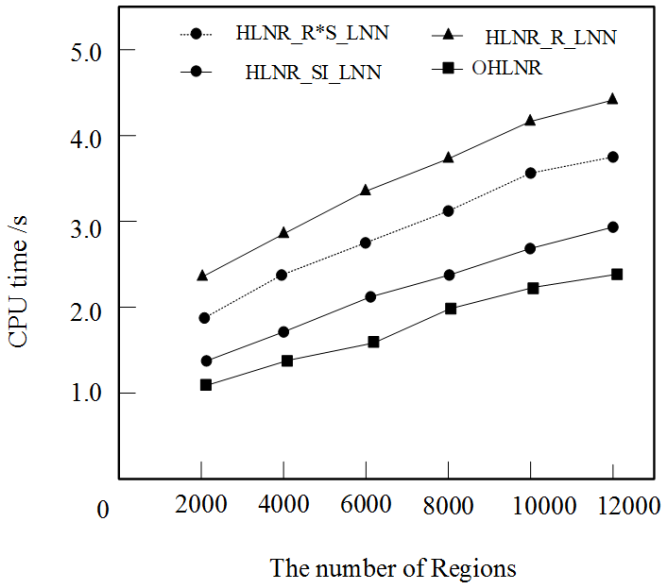

FIGURE 12. Effect of the number of regions on CPU time in obstacle environment

can reduce the amount of computation time in the pre-calculation process. By the pruning process, we remove a large number of the non-candidate regions. Then, the refining process filters the candidates set and improves the algorithm performance.

Figure 12 shows the influence of the set of regions on the CPU time in the obstacle environment. The number of line segments is 50, and the number of obstacles is 1000. As we can see from the figure, the CPU time of the three comparison algorithms is rising obviously. However, the trend of the OHLNR algorithm is gentler. This is because the HLNR_R_LNN algorithm and HLNR_R*S_LNN algorithm need to calculate each region segment. With the number of regions increasing, the regions that need to store and access also increase in the query process, so the CPU time is rising rapidly. And the HLNR_SI_LNN algorithm filters out a large amount of irrelevant data according to pruning rules. However, the algorithm does not store data in advance, so it wastes a lot of time.

For the OHLNR algorithm, the number of queries is reduced and the query performance is greatly improved because a large number of non-candidate regions are filtered out in the pruning and refining process. Therefore, the OHLNR algorithm is better.

Then, the influence of the number of the obstacle lines on CPU time is analyzed. In the obstacle environment, the number of query lines is 20, and the regions are 1000. As shown in Figure 13, with the increasing number of obstacles, CPU time is increasing faster. The HLNR_R_LNN algorithm and the HLNR_R*S_LNN algorithm need to calculate the nearest region of each query line segment, which is more expensive to calculate the distance of obstacles, and the data sets cannot be processed in advance in the HLNR_SI_LNN algorithm, so that amounts of calculation of the subsequent process is too large to reduce a lot of time. The OHLNR algorithm prunes the obstacles, and the obstacles have no effect in pruning process. Therefore, as the number of obstacles is increasing, the CPU time of the algorithm is increasing slowly. Thus, the OHLNR algorithm is slightly better than the other three algorithms in dealing with obstacles.

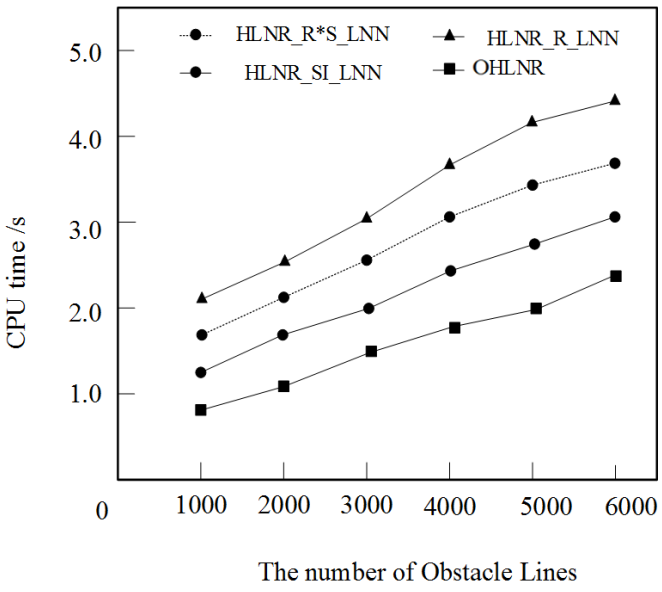

FIGURE 13. Effect of the number of obstacle lines on CPU time in obstacle environment

Finally, the accuracy of the algorithm is tested. Other conditions remain unchanged, the number of regions is 5000, the query line segments are 50, and the number of obstacles is 1000. The accuracy of the four algorithms was tested 20 times respectively in the obstacle environment, and the average of the results is obtained. As shown in Figure 14, it is the effect of query times on accuracy. From the figure, the accuracy of the four algorithms is increasing as the number of query times is increasing, but the OHLNR algorithms in this paper are faster achieving high accuracy than the other three algorithms.

Through this experiment, we can see that both of algorithms OHLNR in the obstacle environment and the HLNR in the barrier free environment are proposed which are better than the other three algorithms.

5. **Conclusions.** In this paper, to solve the problem for nearest neighbor from line segment to regions in obstacle or barrier free environment, and then the method of nearest region of spatial line segment based on Hilbert curve grid is proposed. This paper is divided into two parts: the query for nearest region of line segment in the barrier free environment and in the obstacle environment. In a barrier free environment, the process of solving the query for nearest region of a line segment includes pre-calculation, filtering

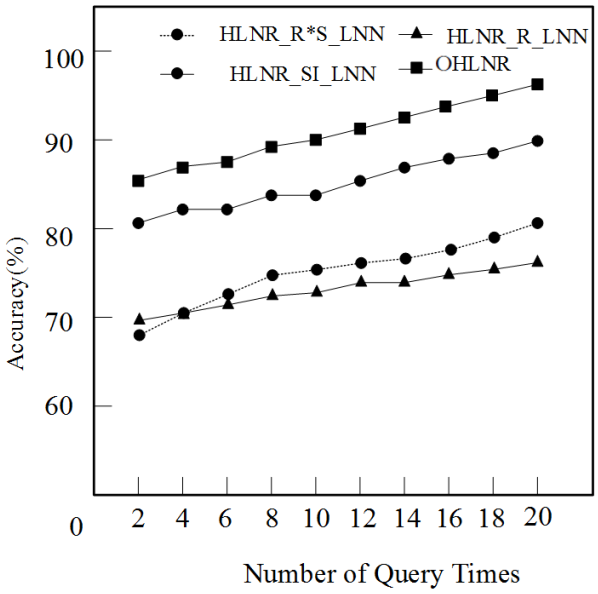

FIGURE 14. Effect of queries on accuracy

and refining. By using the Hilbert curve grids characteristics, the pre-calculation process can store the line segments and the set of the regions, and the query time is shortened. The filtering process can cut out a large number of non-candidate regions, and it can optimize query process and reduce the running time. The refining process is based on the filtering process, the set of query results in the filtering process is refined, and the nearest region results of the line segment are obtained. Next, the OHLNR query method is given, including pruning process and refining process. The pruning process can filter out a large number of non-candidates, it also prunes many obstacles that have no impact on the query process, then the candidates are refined and the more accurate candidates are obtained. Finally, the candidates are refined to get the set of accurate results. Experimental results show that the algorithms have higher efficiency in this paper. Future research focuses on the query method for reverse nearest region of spatial line segments.

Acknowledgment. This work is partially supported by the National Natural Science Foundation of China under Grant No. 61872105, the Science and Technology Research Project of Heilongjiang Provincial Education Department (1253lz004) and the Scientific Research Foundation for Returned Scholars Abroad of Heilongjiang Province of China (LC2018030). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] N. Roussopoulos, S. Kelley and F. Vincent, Nearest neighbor queries, *ACM Sigmod Record*, vol.24, no.2, pp.71-79, 1995.

[2] X. Yi, R. Paulet, E. Bertino et al., Practical approximate k nearest neighbor queries with location and query privacy, *IEEE Trans. Knowledge & Data Engineering*, vol.28, no.6, pp.1546-1559, 2016.

[3] Y. Gao, Q. Liu, X. Miao et al., Reverse k-nearest neighbor search in the presence of obstacles, *Information Sciences*, vol.330, pp.274-292, 2016.

[4] R. T. Liu and Z. Hao, Fast algorithm of nearest neighbor query for line segments of spatial database, *Journal of Computer Research & Development*, vol.48, no.12, pp.2379-2384, 2011.

[5] T. Hashem, L. Kulik and R. Zhang, Privacy preserving group nearest neighbor queries, *Proc. of the 13th International Conf. on Extending Database Technology*, Lausanne, Switzerland, pp.489-500, 2010.

[6] Y. Wang, Y. H. Dong, J. B. Qian et al., Continuous nearest-neighbor query in location privacy preserving, *Journal of Beijing University of Posts & Telecommunications*, vol.9, no.5, pp.83-88, 2016.
[7] Y. Gu, H. Zhang, Z. Wang et al., Efficient moving k, nearest neighbor queries over line segment objects, *World Wide Web – Internet & Web Information Systems*, vol.19, no.4, pp.1-25, 2016.
[8] M. E. Ali, E. Tanin, R. Zhang et al., Probabilistic voronoi diagrams for probabilistic moving nearest neighbor queries, *Data & Knowledge Engineering*, vol.75, no.2, pp.1-33, 2012.
[9] M. Pluciński, Application of the k nearest neighbors method to fuzzy data processing, *Przeglad Elektrotechniczny*, vol.1, no.1, pp.79-83, 2017.
[10] W. W. Sun, C. N. Chen, L. Zhu et al., On efficient aggregate nearest neighbor query processing in road networks, *Journal of Computer Science & Technology*, vol.30, no.4, pp.781-798, 2015.
[11] S. Li, L. Zhang, L. Peng et al., New methods for the construction of Voronoi diagram and the nearest neighbor query, *The 9th International Forum on Strategic Technology (IFOST)*, Cox's Bazar, Bangladesh, pp.255-258, 2014.
[12] W. Mulzer, P. Seiferth and Y. Stein, Approximate k-flat nearest neighbor search, *The 47th ACM Symposium on Theory of Computing*, USA, pp.783-792, 2015.
[13] H. Xu and Z. Hao, Nearest-neighbor query algorithm based on grid partition of space-filling curve, *Computer Science*, vol.37, no.1, pp.184-188, 2010.
[14] R. Rathika, G. D. Raj and T. Rajendran, A proxy based querying approach using Hilbert curve in mobile environment, *The 5th International Conf. on Advanced Computing*, Chennai, India, pp.426-431, 2014.
[15] L. Zhu, W. Sun, Y. Jing et al., Voronoi-based k-aggregate nearest neighbor query processing in road networks, *Journal of Computer Research & Development*, vol.48, no.S3, pp.155-162, 2011.
[16] L. P. Zhang, Y. Y. Guo, S. Li et al., Line reverse k nearest neighbor query based on Voronoi diagram in road network, *Computer Science & Exploration*, vol.11, no.6, pp.908-920, 2017.
[17] D. Z. Sun, Y. W. Sun, X. C. Kang et al., Line segment nearest neighbor query of spatial database based on R*S-tree, *Advanced Materials Research*, vols.201-203, pp.194-197, 2011.
[18] Z. X. Hao, Y. Wang and Y. He, Line segment nearest neighbor query of spatial database, *Journal of Computer Research & Development*, vol.45, no.9, pp.1539-1545, 2008.
[19] H. L. Chen and Y. I. Chang, All-nearest-neighbors finding based on the Hilbert curve, *Expert Systems with Applications*, vol.38, no.6, pp.7462-7475, 2011.
[20] X. Sui, X. Chen, J. Ge et al., Research on a toolpath generation method of NC milling based on space-filling curve, *High Technology Letters*, vol.23, no.4, pp.418-425, 2017.
[21] H. Xu and Z. X. Hao, An approximate k-closest pair query algorithm based on Z curve, *Journal of Computer Research & Development*, vol.45, no.2, pp.310-317, 2008.
[22] *Dataset [EB/QL]*, http://konect.uni-koblenz.de/networks/roadNet-CA, 2016.