J Inf Process Syst, Vol.16, No.2, pp.406~420, April 2020
https://doi.org/10.3745/JIPS.04.0167

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

A Method for *k* Nearest Neighbor Query of Line Segment in Obstructed Spaces

Liping Zhang[*], Song Li[*], Yingying Guo[*], and Xiaohong Hao[*]

Abstract
In order to make up the deficiencies of the existing research results which cannot effectively deal with the nearest neighbor query based on the line segments in obstacle space, the k nearest neighbor query method of line segment in obstacle space is proposed and the STA_OLkNN algorithm under the circumstance of static obstacle data set is put forward. The query process is divided into two stages, including the filtering process and refining process. In the filtration process, according to the properties of the line segment Voronoi diagram, the corresponding pruning rules are proposed and the filtering algorithm is presented. In the refining process, according to the relationship of the position between the line segments, the corresponding distance expression method is put forward and the final result is obtained by comparing the distance. Theoretical research and experimental results show that the proposed algorithm can effectively deal with the problem of *k* nearest neighbor query of the line segment in the obstacle environment.

Keywords
Line Segment k Nearest Neighbor, Line Segment Obstacle Distance, Line Segment Voronoi Diagram, Nearest Neighbor Query, Spatial Database

1. Introduction

At present, with the rapid development of application fields such as geographic information system, computer-aided design, intelligent identification system, Internet of Things, decision support system, the spatial database query technology has more important significance. The nearest neighbor query [1-3] is one of the foundations of the spatial database query. A new ST-R tree index is designed by [4] and the nearest neighbor query algorithm in accordance with spatio-temporal information is proposed. The authors [5] proposed the nearest neighbor query algorithm in the road network environment by using the characteristics of network Voronoi diagram. The nearest neighbor query method based on random partition tree is proposed in [6]. In recent years, many variants of the nearest neighbor query were derived based on the research of nearest neighbor query, for example, the reverse nearest neighbor query [7,8], the group nearest neighbor query [9,10], the continuous nearest neighbor query [11,12], the nearest neighbor query based on uncertain data [13], the aggregate nearest neighbor query [14], the strong nearest neighbor query [15], and so on.

※ This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

Manuscript received November 19, 2018; first revision February 8, 2019; accepted February 27, 2019.

Corresponding Author: Liping Zhang (zhanglptg@163.com)
* College of Computer Science and Technology, Harbin University of Science and Technology, Harbin, China (zhanglptg@163.com, lisongbeifen @163.com, chchsong@163.com, jiaxiyu888@163.com)

Copyright© 2020 KIPS

In reality, some restrictions on geographical are inevitable, the factors of obstacle must be taken into consideration by the shortest distance between two objects. In [16], the nearest neighbor query method is proposed to preserve the location privacy in obstacle space. In [17], the reverse *k* nearest neighbor query method is put forward, and the search of influence scope of the query space in the obstacle space is realized. The authors [18] proposed a new index structure O-tree, which is used to store the obstacle data, and solved the nearest neighbor query problem in obstacle space. According to the different position relationship of minimum outsourcing distance between the point in the data set and the query set, the authors [19] proposed a new variant nearest neighbor query in an obstacle environment. In [20], the method of the group reverse *k* nearest neighbor query in obstructed space was proposed. The algorithms in static obstacle environment and dynamic obstacle environment were given respectively.

The above research work mainly focuses on spatial data points, however, in the practical exploration of query problems, if the spatial objects are abstracted into points, the accuracy of the query results will be impacted, such as mountains, rivers, roadblock, and so on. They are not suitable to be abstracted into points in the process. In order to improve the efficiency of the query, some spatial objects can be abstracted as line segments. [21] solved the nearest neighbor problem of line segments for the first time and proposed the concept and the corresponding nearest neighbor query algorithm based on the line segments. [22] required that all the nodes in the SI-tree are ordered by the relation of its geometric position, which makes that the line segment nearest neighbor query can be quickly located the intermediate node. [23] queried the nearest neighbor properties pair by using the nearest neighbor and local dynamic properties of two Voronoi diagrams. By judging the position relationship between the query line segment and the security region, the problem of the moving *k* nearest neighbor query over the line segments is solved in [24].

For the nearest neighbor query based on the points in obstacle space, if there is an obstacle between the query point and the data point, then the query will surely be affected and it is necessary to calculate the obstacle distance. However, if the query object is abstracted as a line segment, then the query line segment and the data line segment may be completely visible, partly visible and completely invisible. Therefore, the *k* nearest neighbor query of the line segment in obstacle space is a new problem to be solved in this paper. The method proposed in this paper is widely used in real life. For example, if some people want to buy a house which is nearest to a school, then the effects of other buildings (obstacles) should be taken into consideration when calculating the shortest distance between the school and the buildings. At this point, the *k* nearest neighbor query of line segment algorithm in obstacle space which is proposed in this paper can be used. Also, during the railway construction, it will inevitably encounter obstacles such as mountains, rivers, and other factors, and in this case, the *k* nearest neighbor query of line segment algorithm in obstacle space can be used in order to reduce the query error. The main contributions of this paper are as follows:

(1) The new problem of the obstacle *k* nearest neighbor of line segment query is presented and formalized defined. The role of the query is discussed in detail.
(2) The study of this paper is divided into two stages: the pruning process and the refining process. First, according to the properties of the line segment Voronoi diagram, this paper proposes the pruning rules for data line segments and obstacles, and the corresponding pruning algorithms are given. In the refining process, the methods of expressing obstacle distance are given. Finally, the STA_OLkNN algorithm is proposed. The proposed algorithm can effectively deal with the problem of *k* nearest neighbor query of the line segment in the obstacle environment.

2. Basic Definitions and Properties

Definition 1 (Line segment Voronoi diagram [25]). Given a set of disjoint adjacent line segments $L=\{l_1,\ldots,l_n\}$and end points of each line segment. The line segment Voronoi diagram divided the plane into several connected regions, which is called Voronoi region. Every region is correspond to each line segment. Voronoi diagram region of each generated line segment is given by the equation: $VL(l_i)=\{p \mid d(p, l_i)\le d(p, l_j)\}$, among which $i\neq j$, $j\in I_n$, and $d(p, l_i)$ are the shortest distance between p and line segment l_i, (linear distance between point p and line segment l_i in Euclidean space). The region decided by l_i is called Voronoi polygon. The figure defined by $VL(L)=\{VL(l_1),\ldots,VL(l_n)\}$ is called line segment Voronoi diagram. Each edge of a line segment Voronoi diagram is composed of a straight line segment or a parabola segment. Voronoi polygons which share the same edge are called adjacent polygons, their generated segments are called adjacent generation line segments.

Property 1. The nearest neighbor of generated line segment l_i is in the adjacent generated line segment of l_i.

Definition 2 (k level adjacent generated line segment [25]). Given a set of generation line segments $L=\{l_1, l_2,\ldots,l_n\}$ in the Voronoi diagram, among which $2<n<\infty$, and when $i\neq j$ then $l_i\neq l_j$. Among them, i, $j\in I_n=\{1,\ldots, n\}$. The k level adjacent generation line segment of l_i is defined as $AG_k(l_i)=\{l_j \mid VL(l)$ and $VL(l_j)$ has common edge, $l\in AG_{k\text{-}1}(l_i)\}$.

Definition 3 (Line segment visualization). There are two line segments l_i and l_j in obstacle environment, we connect the left and right ends of the two line segments to form a polygon P. If P does not intersect with any obstacles, then the line segment l_i and l_j are completely visible; If there is an obstacle which has intersection with any side of P, then the line segment l_i and l_j are partly visible; If there is an obstacle which has more than one intersection with P, then the line segment l_i and l_j are completely invisible.

Definition 4 (Line segment shortest distance [21]). Given two line segment L and K, the point l, l_i, $l_j\in L$, the point k, k_i, $k_j\in K$, $dist(l, k)$ represents the distance from point l to point k, suppose the shortest distance between line L and K is $dist(L, K)$. Then $dist(L, K)=\{dist(l_i, l_k) \mid dist(l_i, k_i)\le dist(l_j, k_j)\}$.

Definition 5 (Nearest distance of line segment with obstacles). There is a query line segment l_q(suppose its left and right end points are respectively m and n) and data line segment l_i. If two line segments are visible, the Euclidean distance of two line segments is noted as $Vdist(l_i, l_q)$, then the obstacle distance of two line segments is equal to the Euclidean distance, that is $Odist(l_i, l_j)=Vdist(l_i, l_j)$. If two line segments are completely invisible and there is an effective obstacle O_k between two line segments, then the obstacle distance is the shortest distance of the two line segments bypass the obstacles, which is $Odist=min\{dist(m, O_k)+dist(O_k, l_i), dist(n, O_k)+dist(O_k, l_i)\}$.

Definition 6 (Obstacle k nearest neighbor of line segment query). Given data line segment set $L=\{l_1, l_2,\ldots, l_n\}$ and obstacle set $O=\{O_1, O_2,\ldots, O_m\}$. In the obstacle space, the k nearest neighbor of line segment query returns k data line segments which are nearest to the query line segment l_q, that is OLkNN$(l_q)=\{l\in L \mid Odist(l, l_q)\le Odist(l_i, l_q)\}$

3. The Method of *k* Nearest Neighbor Query in the Obstacle Space

In the obstacle space, the k nearest neighbor query of line segments can be divided into two parts. It includes filtration process and refining process. The main purpose of filtration process is to improve the query efficiency. In order to get the accurate result, the STA_OLkNN algorithm is proposed in refining process.

3.1 Filtration Process

In the actual query process, not all data are valid for the query, and some data are redundant. Therefore, first we need to exclude the irrelevant data. The data filtering process includes two parts: filtering of data line segments and filtering of obstacles data.

First of all, we filter the data line segments. The main work of this process is to eliminate a large number of non-candidates, and get more accurate LkNN candidate set in the obstacle space. Because the polygons can be triangulated, each obstacle is abstracted as a triangular representation, and the obstacle set $O=\{O_1,O_2,\ldots,O_m\}$ is formed. The obstacle which may have an effect on the query is called the effective obstacle. We select a set of disjoint line segments as the set of baseline segments, that is $L=\{l_1,l_2,\ldots,l_m\}$. According to the properties of line segment Voronoi diagram and the definition of k nearest neighbor query of line segments in obstacle space, Theorem 1 and Theorem 2 are given.

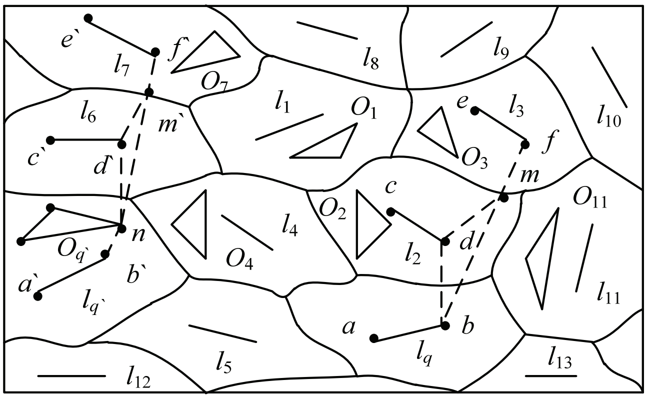

Fig. 1. Proof of Theorem 1 and Theorem 2.

THEOREM 1. Given query line segment l_q, data line segment l_i and l_j. From l_q to l_j, it need to go through the Voronoi polygon which l_i is in. If l_q and l_i are completely visible, then $Vdist(l_q, l_i)\leq Vdist(l_q, l_j)$.

Proof. We prove the theorem by using the method of reduction. The query line segment l_q, data line segment l_2 and l_3 are given, which as shown in Fig. 1. Supposing that the distance from the query line segment l_q to the data line segment l_3 is closer than that to l_2, that is assuming $Vdist(l_q, l_3)<Vdist(l_q, l_2)$. According to the line segment distance definition, the distance between l_q and l_2 can be expressed as $Vdist(l_q, l_2)=dist(b, d)$, the distance between l_q and l_3 can be expressed as $Vdist(l_q, l_3)= dist(b, f)=dist(b, m)+dist(m, f)$. The equation $dist(m, d)=dist(m, f)$ will be obtained in accordance with the property of line segment Voronoi diagram. As in the Euclidean space, the triangle inequality theorem is satisfied, thus

dist(b, m)+*dist*(m, d)>*dist*(b, d), *dist*(b, f)>dist(b, d), that is *dist*(l_q, l_3)>*dist*(l_q, l_2). The original theorem is established because of the contradiction with the hypothesis. Prove finished.

Theorem 1 is applicable to the case that there is no effective obstacle between the query line segment and the data line segment. In other words, the distance between the query line segment and its adjacent data line segment is shorter than that between other lines. For example, if l_{12} is a query line segment, then *dist*(l_{12}， l_5)<*dist*(l_{12},l_4), as shown in Fig. 1.

THEOREM 2. Given query line segment l_q, data line segment l_i and l_j. From l_q to l_j , it needs to go through the Voronoi polygon which l_i is in. If the effective obstacle exists between l_q and l_i, then *Odist*(l_q, l_i)≤*Odist*(l_q, l_j).

Proof. We prove the theorem by using the method of reduction. The query line segment l_q, data line segment l_6 and l_7, and obstacle $O_{q'}$ are given, as is shown in Fig. 1. Supposing that the obstacle distance from the query line segment l_q to the data line segment l_7 is closer than that to l_6, that is assuming *Odist*($l_{q'}$, l_7)<*Odist*($l_{q'}$, l_6). According to the line segment distance definition, the obstructed distance between l_q and l_6 can be expressed as *Odist*($l_{q'}$, l_6)=*dist*(b', n)+*dist*(n, d'), the obstructed distance between l_q and l_7 can be expressed as *Odist*($l_{q'}$, l_7)=*dist*(b', n)+*dist*(n, f'). In order to prove the connection point d' and m', *dist*(d', m')=*dist*(m', f') will be obtained in accordance with the property of line segment Voronoi, therefore, *Odist*($l_{q'}$, l_7)=*dist*(b', n)+*dist*(d', m')+*dist*(m', n). According to the triangle inequality theorem, *dist*(n, m')+*dist*(m', d)>*dist*(n, d') can be obtained, thus *dist*(n, b')+*dist*(n, d')<*dist*(n, b')+*dist*(n, f'), *Odist*($l_{q'}$, l_6)<*Odist*($l_{q'}$, l_7). The original theorem is established because of the contradiction with the hypothesis. Prove finished.

Theorem 2 is applicable to the case that there is an effective obstacle between the query line segment and the data line segment. The effective obstacle between the two lines will certainly affect the calculation of the distance between the two lines. The existence of obstacles will increase the distance between the two lines, but compared with other data lines, the distance between the data lines adjacent to the query lines is relatively smaller. For example, suppose l_1 is a query line segment, then *Odist*(l_1,l_3)<*Odist*(l_1,l_{10}), as is shown in Fig. 1.

According to Theorem 1 and Theorem 2, pruning rule 1, 2, 3, 4 are given as follows:

Pruning rule 1. We take the visible shortest distance from query line segment to data line segment as the radius, and generate two circles with two endpoints of l_q. The region which they cover is marked as E. Then the k nearest neighbor of l_q may be in the Voronoi polygon which intersects with E.

Pruning rule 2. If the data line segment in the Voronoi polygon which is adjacent to the query line segment, then it may be the k nearest neighbor of the line segment.

Pruning rule 3. If the Voronoi polygon *VL*(l_i) of the data line segment and the Voronoi polygon *VL*(l_q) of the query line segment have no common edge, then the data line segment l_i is pruned.

Pruning rule 4. If the minimum number of Voronoi polygons passing from the data line segment to the query line segment exceeds k, then the data line segment will be pruned.

Based on the discussion above, the filtering algorithm is shown in Algorithm 1.

Algorithm 1. OL*k*NN_Line_Filter(L, O, l_q, k)

```
Input: Data line segment set $L$, Obstacle set $O$, Query line segment $l_q$, $k$.
Output: The data line segment candidate $S_C$.
Begin
1. Generate the line Voronoi diagram in obstacle space;
2. $S_C \leftarrow \varnothing$;
3. $count$=0 ;
4. $sumcount$=0 ;
5. if $l_q$ and $l_j$ are visualization then
6.    Calculate_$MinVdist(l_q, l_j)$ ;
7.    Create_$E$ ; // The region which the circles cover
8.    if $E \cap VL(l_j)$ != NULL then
9.        $S_C \leftarrow S_C + l_j$;  //Pruning rule 1
10.       Judge_$Position(l_j)$ ;
11.       if $VL(l_j)$ is the adjacent polygon to $VL(l_q)$ then
12.          $S_C \leftarrow S_C + l_j$;  // Pruning rule 2
13.       if there is no common edge between $VL(l_j)$ and $VL(l_q)$then
14.          $S_C \leftarrow S_C - l_j$ ; // Pruning rule 3
15. else
16.    if $count \le n$ then
17.       Traversal all of the $VL(l_j)$ from $l_q$ to $l_j$ ;
18.       $sumcount$=$count$+1 ;
19.       if $sumcount < k$ then
20.          $S_C \leftarrow S_C + l_j$;
21.       else
22.          $S_C \leftarrow S_C - l_j$;  // Pruning rule 4
23. return $S_C$ ;
End
```

The algorithm OL*k*NN_Line_Filter constructs line segment Voronoi diagram through the data line segment set, and collects the obstacle which is formed by three points except from the end points of the line segment randomly in any of $VL(l_j)$, so that the line segment Voronoi diagram in obstacle space is formed (Line 1). The algorithm initializes data line segment candidate set S_C, and counts variables (Line 2–4). Then we can judge whether there is visible distance between query line segment l_q and data line segment l_j. If it exists visible distance, then we calculate it according to Pruning rule 1. We create area E to filter the data line segment (Line 5–9). If the data line segment satisfies Pruning rule 2, it will be added into data line segment candidate set S_C, or it will be pruned (Line 10–12). According to Pruning rule 3, if there is no common side between $VL(l_q)$ and $VL(l_j)$, the data line segment l_j is pruned (Line 13–14). If there is no visible distance between query line segment l_q and data line segment l_j, according to Pruning rule 4, we can have screen on data line segment by judging the number of Voronoi polygons which are covered by the route between them. The satisfied data line segment will be added into the candidate set S_C, or it will be pruned (Line 15–22). Finally, data line segment candidate set S_C will be formed (Line 23).

We further filter the obstacle data. Given the query line segment l_q, obstacle O and data line segment l in candidate set S_C. We connect the endpoints of the data line segment and query line segment to form a polygon P. According to the different position between obstacle and polygon P, Theorem 3 and Theorem 4 are proposed.

Theorem 3. If obstacle O_x has no intersection with polygon P, then obstacle O_x is not the effective obstacle.

Proof. We provide query line segment l_q and set the left and right endpoints as point a and point b respectively. Data line segment l_1 is provided, and the left and right endpoints are set as point c and point d, as is shown in Fig. 2. We connect the endpoints of data line segment and query line segment to form quadrangle *abcd*. The obstacle O_2, O_3 and O_4 have no intersection with quadrangle *abcd*. When we compute the obstacle distance between query line segment l_q and data line segment l_1, it is unnecessary to take the effect of obstacle O_2, O_3 and O_4 into consideration, so obstacle O_2, O_3 and O_4 are not effective obstacles. Therefore, the original theorem was established. Proof finished.

Theorem 3 is used to filter the data line segments in the previous stage. For two line segments, an auxiliary line is constructed by connecting their endpoints to form a polygon. If there is no intersection between obstacles and polygons, the distance between the two lines must not be affected. So the distance between the two lines need not be considered when the distance between the lines is calculated.

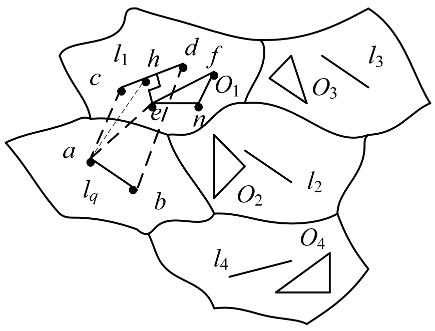

Fig. 2. Proof of Theorem 3 and Theorem 4.

Theorem 4. If obstacle O_x intersected with polygon P, and it has no intersection with the shortest visual distance from query line segment to data line segment, then obstacle O_x is not effective obstacle.

Proof. The theorem is proved by contradiction. The obstacle O_1 has three endpoints: point e, point n and point f, as shown in Fig. 2. Obstacle O_1 has intersection with quadrangle *abcd*. Assuming that obstacle O_1 has impact on the obstacle distance from l_q to l_1 that is O_1 is an effective obstacle. The shortest visual distance from l_q to l_1 is represented as $Vdist(l_q, l_1)$. According to the definition of line distance, the visual distance from l_q to l_1 can be shown as $Vdist(l_q, l_1)=dist(a, c)$. We take the effect of obstacle O_1 into consideration, the obstacle distance from l_q to l_1 can be represented as $Odist(l_q, l_1)=dist(a, e)+dist(e, h)$. From the triangle inequality theorem, we can get $dist(a, e)+dist(e, h)>dist(a, h)$. So, $dist(a, h)>dist(a, c)$, that is $dist(a, c)<dist(a, e)+dist(e, h)$. Therefore, when we compute the distance between l_q and l_1, it is unnecessary to take the effect of obstacle O_1 into consideration and obstacle O_1 is not the valid obstacle. It is inconsistent with the hypothesis. Therefore, the original theorem was established. Proof finished.

Theorem 4 is applicable to further judgment of effective obstacles. Assuming that two line segments are visible, the shortest visual distance between two line segments is constructed as an auxiliary line. For example, we calculate the distance between line segments l_2 and l_3, the obstacle O_3 is an effective obstacle in Fig. 2.

Based on Theorems 3 and 4, pruning strategy 5, 6, and 7 are given as follows.

Pruning rule 5. If the obstacle is in the Voronoi polygon which candidate set S_C of data line segments existed, then the obstacle may be an effective obstacle.

Pruning rule 6. If the shortest distance from query line segment to obstacle is longer than the distance to the diagonal of polygon *P*, then the obstacle is not an effective obstacle.

Pruning rule 7. If the shortest distance from query line segment to obstacle is shorter than the length of polygon's diagonal line, and it also has intersection with polygon *P*, then the obstacle is an effective obstacle.

Based on the discussion above, the filtering algorithm is shown in Algorithm 2.

Algorithm 2. OL*k*NN_Obstacle_Filter(*L*, *O*, l_q, *k*)

```
Input: Data line segment set $L$, Obstacle set $O$, Query line segment $l_q$, $k$.
Output: The obstacle candidate $S_O$.
Begin
1. $S_O \leftarrow \varnothing$;
2. Call OLkNN_Line_Filter() algorithm  to get candidate $S_C$;
3. Partition region $VL(S_C)$ ;
4.  if $O_x \in VL(S_C)$ then
5.    $S_O \leftarrow S_O + O_x$;    //Pruning rule 5
6.    Create_$P(l_q, l)$ ;
7.    Calculate_$dist(l_q, O_x)$ ;
8.    if $dist(l_q, O_x)$>diagonal line of $P$ then
9.      $S_O \leftarrow S_O - O_x$;  //Pruning rule 6
10.   else
11.     Judge_$Position(O_x)$ ;
12.     if $O_x \cap P$ != NULL then
13.       $S_O \leftarrow S_O + O_x$;
14.     else
15.       $S_O \leftarrow S_O - O_x$;
16.       Calculate_$MinVdist(l_q, l)$ ;
17.       if $O_x \cap MinVdist(l_q, l)$ != NULL then
18.         $S_O \leftarrow S_O + O_x$;
19.       else
20.         $S_O \leftarrow S_O - O_x$;   //Pruning rule 7
21. return $S_O$;
End
```

Based on the location of the data line segment in the candidate set S_C, the algorithm OL*k*NN_Obstacle_Filter can partition the area $VL(S_C)$ (Line 1-3). Then we can make use of the Pruning rule 5 to judge whether obstacle O_x is in area $VL(S_C)$. If O_x has intersection with $VL(S_C)$, O_x should be added into the candidate set S_O (Line 4–5). We form polygon *P* and calculate the distance from query line segment to the obstacle. According to Pruning rule 6, we judge the relation between $dist(l_q, O_x)$ and length of diagonal line of P (Line 6–9). If the distance between l_q and O_x is less than the diagonal line of P, then we use Pruning rule 7 and Theorem 4 to make further judgment on obstacles, and add the satisfied obstacles into candidate set S_O (Line 10–20). Finally, the filtered valid obstacle candidate set S_O can be obtained (Line 21).

3.2 Refining Process

The refined algorithm proposed mainly has operation on the candidate set S_C and S_O which are produced during the filtration process. First, the shortest obstructed distance from query line segment to data line segment which is newly added to the candidate set is calculated. Then we compared the shortest obstructed distance and the more accurate result of the k nearest neighbor of line segment in obstacle environment can be obtained.

In order to represent the obstacle distance between the data line segment and the query line segment more accurately, the following theorem 5 is further proposed.

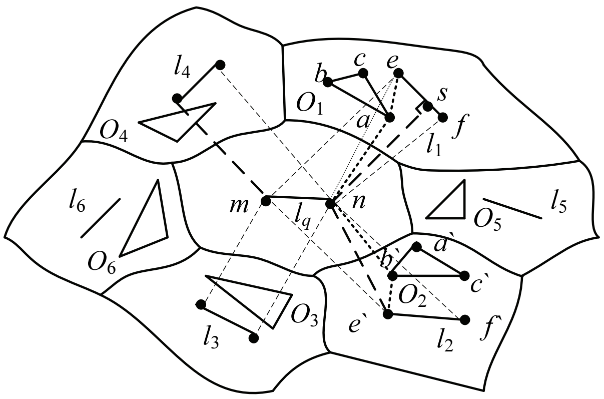

Fig. 3. Proof of Theorem 5.

THEOREM 5. Given the data line segment l, the query line segment l_q, and the obstacle O. If there is a visual distance $Vdist$ (l_q, l), then $Vdist$ $(l_q, l) < Odist$ (l_q, l).

Proof. Given the set of data line segment set $L=\{l_1, l_2, l_3, l_4, l_5, l_6\}$, query line segment l_q, and the obstacle set $O=\{O_1, O_2, O_3, O_4, O_5, O_6\}$. We create line segment Voronoi diagram as shown in Fig. 3. When there is no intersection between the obstacle and the visual distance, the visual distance can be expressed as $Vdist(l_q, l)$, such as data line segment l_1 and l_2. Because the location of the line segment is different, the distance representation method is different. When the two line segments are introverted, we structure the visual distance between l_q and l_1, that is $Vdist(l_q, l_1)=dist(n, s)$. The obstacle distance between l_q and l_1 is expressed as $Odist(l_q, l_1)=dist(n, a)+dist(a, e)$. In the Euclidean space, then $dist(n, a)+dist(a, e)>dist(n, e)$, and $dist(n, e)>dist(n, s)$, therefore $Vdist(l_q, l_1)<Odist(l_q, l_1)$. When the two line segments are extroverted, the visual distance between l_q and l_2 is constructed. As shown in Fig. 3, we can further get the visual distance $Vdist(l_q, l_2)=dist(n, e`)$, obstacle distance $Odist(l_q, l_2)=dist(n, b`)+dist(b`, e`)$, and the inequality $Vdist(l_q, l_2)<Odist(l_q, l_2)$ can be obtained . Therefore, there is an inequality $Vdist(l_q, l)<Odist(l_q, l)$ when there is a visual distance. Proof finished.

Theorem 5 is used to further judge the shortest distance between two lines when there are effective obstacles. For example, the data line segments l_3, l_5, l_6 and the query line segment l_q are completely invisible in Fig. 3, so the obstacle distance should be considered. For the partial visibility between line segment l_1, l_2, l_4 and query line segment l_q, in order to get the expression of the shortest distance between two line segments, we need to use Theorem 5 to compare the obstacles distance (*Odist*) and visible distance (*Vdist*).

Based on the discussion above, further refinement algorithm for the *k* nearest neighbor of line segment in obstacle environment is given, shown in Algorithm 3.

Algorithm 3. STA_OL*k*NN(S_C, S_O, l_q, k)

```
Input: Data line segment candidate $S_C$, Obstacle candidate $S_O$, Query line segment $l_q$, $k$.
Output: The query result set $S_R$ in static obstacle space.
Begin
1. $S_R \leftarrow \varnothing$;
2. $Odist(l_q, l)=0$;
3. if $l \in S_C$ and $O_x \in S_O$ then
4.    Create_$P(l_q, l)$ ;
5.    if $O_x$ and two sides of $P$ have intersection then
6.        No visual distance exists;
7.        Calculate_$MinOdist(l_q, l)$ ;
8.    if $Odist(l_q, l_i)<Odist(l_q, l_j)$ then
9.         $S_R \leftarrow S_C - l_j$;
10.  else if there is visual distance then
11.        Calculate_$Vdist(l_q, l)$ ;
12.        $Odist(l_q, l)=Vdist(l_q, l)$ ;  // theorem5
13.        if $Odist(l_q, l_x)<Odist(l_q, l_y)$ then
14.             $S_R \leftarrow S_C - l_y$;
15. return $S_R$;
End
```

First, the algorithm STA_OL*k*NN initializes obstacle candidate set S_R and the obstacle distance from query line segment l_q to data line segment l (Line 1–2). Polygon P can be formed by connecting the endpoints of the two lines, and then we can have a judgment on the visualization between query line segment and data line segment (Line 3–4). If the obstacle O_x is intersected with the two edges of the polygon P, the data line segments and the query line segment are completely invisible, so there is no visual distance. The obstacle distance between the two lines should be expressed and further compared to get the shortest obstacle distance (Line 5–9). If there is visual distance between the data segment and the query line segment, then the visual distance can be represented. According to Theorem 5, if the visual distance is less than the obstacle distance, then the distance between the two line segments is recorded as visual distance. The algorithm further deletes the data line segments which do not meet the query requirements from the result (Line 10–14). Finally, the more accurate query result set S_R can be obtained in the obstacle space (Line 15).

4. Experiment

In this section, the performance of the proposed algorithm is analyzed through experiments. The operating environment of the experiments is as follows: 2.0 GHZ CPU, 4 GB memory, 500 GB hard disk, Windows 7 system. The adopted data set is the road network information of the State of California [26]. In the experiment, the distribution of the target data is adjusted appropriately. Let *D* represent the size of the data line segment set, *m* represent the number of obstacles in the obstacle data set *O*, *k* represent the number of nearest neighbor of line segment in the obstacle space.

The running time of the algorithm is treated as test indicator, and each experimental result takes the average of 100 queries. Firstly, the OL*k*NN query algorithm (STA_OL*k*NN) is tested under the condition of the static obstacles. Further we test the influence of the *k* value, the number of obstacles *m*, the data set size *D*, and the I/O cost on the running time of the algorithm.

As the *k* nearest neighbor query of line segment in the obstacle environment is proposed for the first time in this paper, the existing methods of the nearest neighbor queries in obstacle space are based on points, so the proposed method cannot be directly compared with the existing research results. In order to get the comparison algorithm, this paper discussed the two nearest neighbor query algorithms which are proposed in [22] and [23]. If there is an effective obstacle between the two line segments, then we express the obstacle distance in combination of the definition of obstacle distance of line segment which is proposed in this paper. After some proper adjustments of these two algorithms, OL*k*NN_Basic1 and OL*k*NN_Basic2 are obtained.

Firstly, we test the influence of *k* value on the running time of the algorithm. The test results are shown in Fig. 4. In this paper, the running time of CPU is increased with the *k* value. But the trend of the algorithm which is proposed in this paper is more smoothly. Because the two comparison algorithms need to consider the impact of all data line segments and obstacles in the query process, they will cost much CPU running time compared with the STA_OL*k*NN algorithm. Therefore, the proposed algorithm in this paper is superior to the two comparison algorithms in terms of running time.

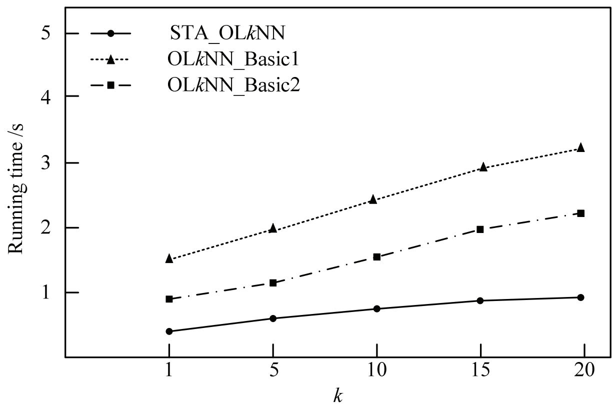

Fig. 4. The effect of *k* on query running time.

In the case of other conditions unchanged, we test the impact of *k* value on the cost of I/O. The test results are shown in Fig. 5. The I/O costs of STA_OL*k*NN algorithm and the two comparison algorithms are increased with the change of the *k* value. Compared with OL*k*NN_Basic1 and OL*k*NN_Basic2, in accordance with the filtering process of data line segments and obstacles, the algorithm proposed in this paper excluded a large amount of useless data and decreased data access amount, therefore, the cost of I/O is lower than that of the two comparison algorithms. It can be concluded that the STA_OL*k*NN algorithm proposed in this paper is better than the two comparison algorithms.

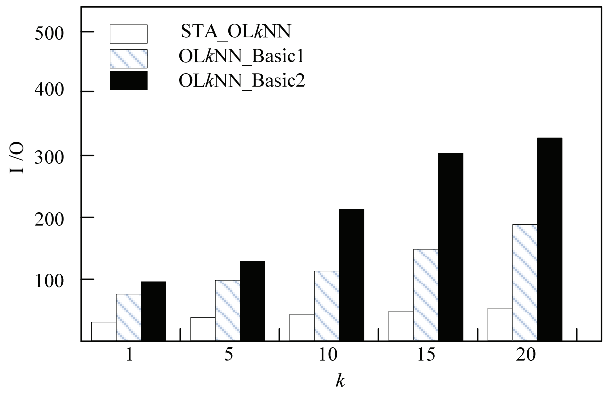

Fig. 5. The effect of *k* on I/O costs.

Further we test the impact of *m* on CPU running time. The size of the *k* value set is 10 and the size of data line segments set is 20000. Fig. 6 shows the change of CPU running time of these three algorithms. The experimental results show that the STA_OL*k*NN algorithm proposed in this paper is superior to the two comparison algorithms. Compared with OL*k*NN_Basic1 algorithm and OL*k*NN_Basic2 algorithm, the growth trend of STA_OL*k*NN algorithm is more smoothly. The reason for this is that it filtered obstacles in the filtering process and the number of obstacles influence on CPU running time is not too much. While the two comparison algorithms need to consider the impact of all obstacles to the query, so they will spend more running time.

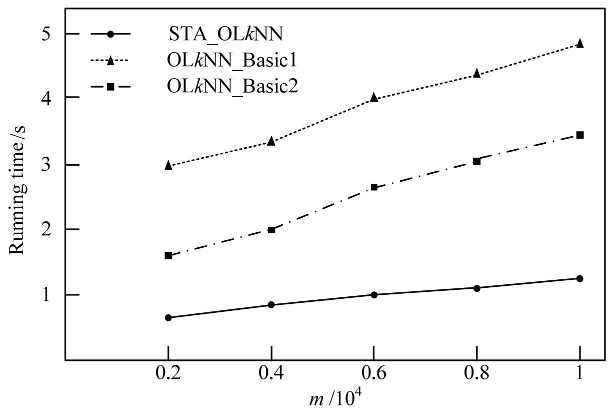

Fig. 6. The effect of m on query running time.

In the case of other conditions unchanged, we test the impact of *m* value on the cost of I/O. The experimental results in Fig. 7 show that STA_OL*k*NN algorithm is better than the two comparison algorithms. OL*k*NN_Basic1 algorithm and OL*k*NN_Basic2 algorithm appear a clear upward trend, the reason is that the comparison algorithms need to consider each obstacle in the process of query. With the increasing number of obstacles, the rate of the quantity of visiting data get larger.

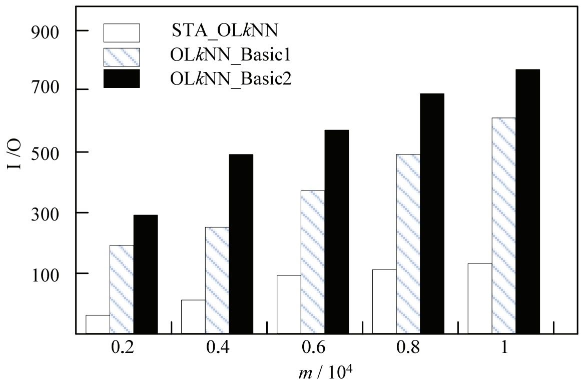

Fig. 7. The effect of m on I/O costs.

Then we test the effect of the data size *D* on CPU running time. We set the *k* value as 10 and the size of the obstacles is 2000. The experimental results are shown in Fig. 8. The STA_OL*k*NN algorithm can exclude a large number of non-candidate data in the filtering process, so even if the data size increases, the increasing of CPU running time of the query algorithm is not significant, the upward trend is more smoothly. For the two comparison algorithms, the CPU running time is affected greatly. Therefore, in terms of the impact of the data size on the algorithm running time, the STA_OL*k*NN algorithm proposed in this paper is better than the two comparison algorithms.

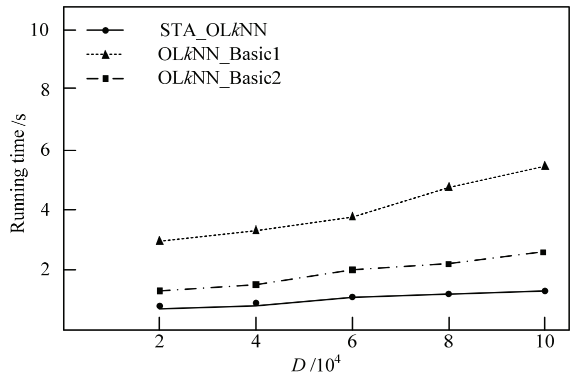

Fig. 8. The effect of *D* on query running time.

5. Conclusions

In this paper, the *k* nearest neighbor query method of line segment in obstacle space is proposed. In different situations of the line segment and the obstacle, the pruning strategies for data line segments and obstacles are proposed based on the property of the line segment Voronoi diagram. The pruning strategies can exclude a large number of non-candidate data and the efficiency of the algorithm can be improved. Then the STA_OL*k*NN algorithm in static obstacles space is finally obtained. Theoretical research and experimental results show that the proposed algorithm has high efficiency. The focus of future research is mainly on the nearest neighbor query of line segment in uncertain data and the skyline query for the line segment.

Acknowledgement

This paper is supported by the National Natural Science Foundation of China (Grant No. 61872105), the science and technology research project of Heilongjiang provincial education department (Grant No. 12531z004), and the Scientific Research Foundation for Returned Scholars abroad of Heilongjiang Province of China (No. LC2018030).

References

[1] C. Efstathiades, A. Efentakis, and D. Pfoser, "Efficient processing of relevant nearest-neighbor queries," *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, vol. 2, no. 3, article no. 9, 2016.

[2] S. Suri and V. Verbeek, "On the most likely voronoi diagram and nearest neighbor searching," *International Journal of Computational Geometry & Applications*, vol. 26, no. 3-4, pp. 151-166, 2016.

[3] O. F. Ertugrul and M. E. Tagluk, "A novel version of k nearest neighbor: dependent nearest neighbor," *Applied Soft Computing*, vol. 55, pp. 480-490. 2017.

[4] C. Li, D. R. Shen, M. D. Zhu, Y. Kou, T. Z. Nie, and G. Yu, "kNN query processing approach for content with location and time tags," *Journal of Software*, vol. 27, no. 9, pp. 2278-2289, 2016.

[5] Y. Jing, L. Hu, W. S. Ku, and C. Shahabi, "Authentication of k nearest neighbor query on road networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 6, pp. 1494-1506, 2013.

[6] S. Dasgupta and K. Sinha, "Randomized partition trees for exact nearest neighbor search," *Algorithmica*, vol. 72, no. 1, pp. 237-263, 2015.

[7] S. Yi, C. Shim, and Y. D. Chung, "Reverse view field nearest neighbor queries," *Information Sciences*, vol. 402, pp. 35-49, 2017.

[8] A. Hidayat, S. Yang, M. A. Cheema, and D. Taniar, "Reverse approximate nearest neighbor queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 2, pp. 339-352, 2017.

[9] Z. Liu, C. Wang, and J. Wang, "Aggregate nearest neighbor queries in uncertain graphs," *World Wide Web*, vol. 17, no. 1, pp. 161-188, 2014.

[10] T. Hashem, L. Kulik, K. Ramamohanarao, R. Zhang, and S. C. Soma, "Protecting privacy for distance and rank based group nearest neighbor queries," *World Wide Web*, vol. 22, no. 1, pp. 375-416, 2019.

[11] A. P. Sistla, O. Wolfson, and B. Xu, "Continuous nearest-neighbor queries with location uncertainty," *The VLDB Journal*, vol. 24, no. 1, pp. 25-50, 2015.

[12] Y. Wang, Y. H. Dong, J. B. Qian, and H. H. Chen, "Continuous nearest-neighbor query in location privacy preserving," *Journal of Beijing University of Posts & Telecommunications*, vol. 39, no. 5, pp. 83-88, 2016.

[13] P. K. Agarwal, B. Aronov, S. Har-Peled, J. M. Phillips, K. Yi, and W. Zhang, "Nearest-neighbor searching under uncertainty II," *ACM Transactions on Algorithms*, vol. 13, no. 1, article no. 3, 2016.

[14] Z. Liu, C. Wang, and J. Wang, "Aggregate nearest neighbor queries in uncertain graphs," *World Wide Web*, vol. 17, no. 1, pp. 161-188, 2014.

[15] S. Li, L. Zhang, and Z. Hao, "Strong neighborhood pair query in dynamic dataset," *Journal of Computer Research and Development*, vol. 52, no. 3, pp. 749-759, 2015.

[16] H. Zhu, J. Wang, B. Wang, and X. Yang, "Location privacy preserving obstructed nearest neighbor queries," *Journal of Computer Research and Development*, vol. 51, no. 1, pp. 115-125, 2014.

[17] X. N. Yu, Y. Gu, T. C. Zhang, and G. Yu, "A method for reverse k-nearest-neighbor queries in obstructed spaces," *Jisuanji Xuebao (Chinese Journal of Computers)*, vol. 34, no. 10, pp. 1917-1925, 2011.
[18] H. Zhu, X. Yang, B. Wang, and W. C. Lee, "Range-based obstructed nearest neighbor queries," in *Proceedings of the 2016 International Conference on Management of Data*, San Francisco, CA, 2016, pp. 2053-2068.
[19] Z. Yang and Z. Hao, "Group obstacle nearest neighbor query in spatial database," *Journal of Computer Research and Development*, vol. 50, no. 11, pp. 2455-2462, 2013.
[20] L. Zhang, L. Liu, and X. Hao, S. Li, and Z. Hao, "Voronoi-based group reverse k nearest neighbor query in obstructed space," *Journal of Computer Research and Development*, vol. 54, no. 4, pp. 861-871, 2017.
[21] Z. Hao, Y. Wang, and Y. He, "Line segment nearest neighbor query of spatial database," *Journal of Computer Research and Development*, vol. 45, no. 9, pp. 1539-1545, 2008.
[22] R. Liu and Z. Hao, "Fast algorithm of nearest neighbor query for line segments of spatial database," *Journal of Computer Research and Development*, vol. 48, no. 12, pp. 2379-2384, 2011.
[23] Z. Yi and Z. Yang, "Research on line segment kNN query in spatial database," *Computer Engineering and Applications*, vol. 51, no. 18, pp. 131-134, 2015.
[24] Y. Gu, H. Zhang, Z. Wang, and G. Yu, "Efficient moving k nearest neighbor queries over line segment objects," *World Wide Web*, vol. 19, no. 4, pp. 653-677, 2016.
[25] M. Held, "VRONI: an engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments," *Computational Geometry*, vol. 18, no. 2, pp. 95-123, 2001.
[26] KONECT, "Road network of the State of California," 2016; http://konect.uni-koblenz.de/networks/roadNet-CA.

Liping Zhang **https://orcid.org/0000-0003-2463-1846**

She received the M.S. degree from Harbin University of Science and Technology in 2006. She is an associate professor at College of Computer Science and Technology, Harbin University of Science and Technology. Her research interests include database theory and application, data mining, data query.

Song Li **https://orcid.org/0000-0003-4456-2040**

He received the Ph.D. degree from Harbin University of Science and Technology in 2009. He is a professor at College of Computer Science and Technology, Harbin University of Science and Technology. His research interests include database theory and application, data mining, data query.

Yingying Guo **https://orcid.org/0000-0003-2262-7368**

She is a master candidate at College of Computer Science and Technology, Harbin University of Science and Technology. Her research interests include data mining, data query.

Xiaohong Hao **https://orcid.org/0000-0002-3521-3175**

She received the M.S. degree from College of Computer Science and Technology, Harbin University of Science and Technology in 2003. She is currently senior lab master. Her research interests include database theory and application, data mining, data query.