Applied Mechanics and Materials Vols. 284-287 (2013) pp 2965-2969

doi:10.4028/www.scientific.net/AMM.284-287.2965

A Forward Moving Method For Continuous Nearest Neighbor Queries

Jun-Hong Shen[1, a], Ye-In Chang[2], Chen-Chang Wu[2] and Ta-Wei Liu[2]

[1]Dept. of Information Communication, Asia University, Taichung, Taiwan

[2] Dept. of Computer Science and Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan

[a]shenjh@asia.edu.tw

Keywords: Continuous nearest neighbor, spatial databases, space filling curves.

Abstract. A continuous nearest neighbor (CNN) query retrieves the nearest neighbor of every point on a line segment and indicates its valid segments. Zheng *et al*. have proposed a Hilbert-curve index for the CNN query. This method contains two phases, searching candidates in the approximate search range, and filtering the candidates to get the final answer. However, it may determine a wide search range in the first phase based on this method, resulting in the decrease of the accuracy and the increase of the processing time. Therefore, in this paper, to avoid this disadvantage, we propose a forward moving method to efficiently support the CNN queries. The proposed method locally expands the search range along the query line segment to find the neighbors. Experimental results show that our method outperforms Zheng *et al*.'s method in terms of the accuracy and the processing time.

Introduction

A spatial database is designed to optimize the ability of manipulating the spatial property for performing queries on spatial data. These spatial objects are made up of points, lines, regions, rectangles, surfaces, volumes, and even data of higher dimension. This paper considers the spatial objects as points. The physical organization of files can be supplemented with indices. A spatial indexing method organizes space and objects in the database to deal with proximity queries [4]. Queries on spatial data commonly concern a certain range or area, for example queries related to intersections, containment and nearest neighbors. With the proliferation of wireless communications and the rapid advances in geographic information systems, a very common spatial query is the nearest neighbor (NN) query which retrieves an object whose representative position is closest to the query position [1].

In real-life applications, people may want to know where those gas stations are along the highway from the start position to the destination. Such a query is called a continuous nearest neighbor (CNN) query. Due to the mobility of the user, the result may be changed immediately as the user is moving. Therefore, CNN queries retrieve the nearest neighbor of every position from the beginning to the destination [3]. While a route may not be a line segment, it can be decomposed into multiple line segments [7]. For answering a CNN query, issuing a nearest neighbor query at every point along the query line segment will incur significant overhead. Therefore, it is feasible that the answer results of a CNN query can contain the objects and their corresponding valid segments.

In [6], Tao *et al*. proposed an R-tree method to process CNN queries. This method traverses the nodes in the R-tree to find nearest neighbors based on a few heuristics. In [7], Zheng *et al*. adapted Tao *et al*.'s method for processing CNN queries in mobile environments by changing the traversal order of branches in the R-tree. In [7], the authors also proposed a Hilbert-curve based method for processing CNN queries. Their experimental results have shown that the Hilbert-curve based method has a better performance on tuning time than the R-tree based method on the uniformly distributed data. Note that the tuning time is the amount of time spent by a mobile client listening to the channel, which roughly corresponds to the processing time in the disk-based systems [5]. In addition, for the disk-based systems, Chen and Chang [2] pointed out that R-tree based methods access unnecessary nodes for processing NN queries, resulting in the increase of the processing time. Moreover, they have shown that Hilbert-curve based methods have a better performance for processing NN queries than R-tree based methods.

From the above observations, in this paper, we consider the Hilbert-curve based structure for processing CNN queries in the disk-based system. Zheng *et al*.'s Hilbert-curve based method contains two phases. In the first phase, it finds the nearest neighbors of the start point and the end point, respectively, and then these two nearest neighbor objects are used to determine the search range to find all candidate objects once. In the second phase, it uses a few heuristics to filter the candidate objects for the final answer. Although this method is proposed for the mobile environment, it can be applied to the disk-based system directly. However, Zheng *et al*.'s method may determine a wide search range to find the candidate objects in the first phase; that means it may check some unnecessary data blocks, resulting in the increase of the processing time.

Therefore, in this paper, we propose a forward moving method, *FM*, based on the local expansion for the CNN queries to avoid those disadvantages in the first phase of Zheng *et al*.'s method. To preserve the spatial locality, the spatial objects are organized based on the Hilbert curve, in which the surrounding blocks to the query point can be efficiently found by using the method in [2]. Due to the local expansion of the search range, our proposed method avoids checking unnecessary data blocks and provides a higher accuracy to get the final answer than Zheng *et al*.'s method. Experimental results show that the proposed *FM* outperforms Zheng *et al*.'s method.

Background

This section describes the underlying data structure, the Hilbert curve, to store spatial objects in our proposed method. There is no total ordering of spatial proximity among spatial data in the multi-dimensional space. Space-filling curves are used to preserve spatial proximity, such as the RGB curve, the Peano curve, and the Hilbert curve. A space-filling curve is a continuous path which passes through every point in the two-dimensional space once to form a one-one correspondence between the coordinates of the points and the one-dimensional sequence numbers of the points on the curve. Since the goal of the space-filling curves is to preserve spatial proximity, they can handle nearest neighbor queries [1]. Among the existing space-filling curves, the Hilbert curve has the best clustering property. The clustering means that after ordering, spatial objects which are close to each other in the two-dimensional space are still close to each other in the one-dimensional space [1]. Therefore, we store spatial objects in the disk according to their sequence of the Hilbert curve, and construct an index to record which data blocks contain objects. Moreover, we use the neighbor finding method in [2] to find the surrounding neighbors. Given a two-dimensional space of size , where $N = 2^n$, $n > 0$, the Hilbert curve of order n recursively divides the space into four equal-sized blocks and gives each block a sequence number from 0 to $(N^2 - 1)$ [1]. Fig. 1 shows the Hilbert curve of order $n = 3$.

The Proposed Method

Our proposed forward moving method (*FM*) based on the local expansion is designed to improve the first phase in Zheng *et al*.'s method. In the first phase, we determine a search range to obtain the blocks on the Hilbert curve. In the second phase, we use heuristics mentioned in Zheng *et al*.'s method to filter the candidate objects for the final answer.

In our proposed method, the first phase for processing a CNN query is processed by procedure *FindRange* shown in Fig. 2 by taking start point *s* and end point *e* as its parameters. In procedure *FindRange*, procedure *FindNNBlocks* uses Chen and Chang's method [2] to find the nearest neighbor blocks having data objects inside. From lines 2-3, the nearest neighbors of start point *s* and end point *e* are found out, respectively. Next, procedure *FindRoute* finds out the blocks that are passed through the query line by using the coordinates of start point *s* and end point *e*. Moreover, these blocks are put into queue *RQueue*, except the start block, to be further processed to expand the search range.

Finally, procedure *LocalExpansion* incrementally moves forward one block along the query line segment, which is stored in *RQueue*, and locally spreads the search range to find the candidate objects. In procedure *LocalExpansion*, at first, start block *s* and one block dequeued from *RQueue* are taken as the local start point and the local end point, respectively, to spread the search range. Then, this local end point is taken as the local start point for the next step, and one block dequeued from *RQueue* is taken as the local end point. These local expansions are repeatedly processed until *RQueue* is empty.

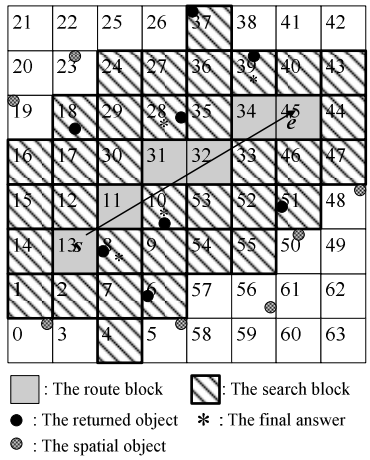
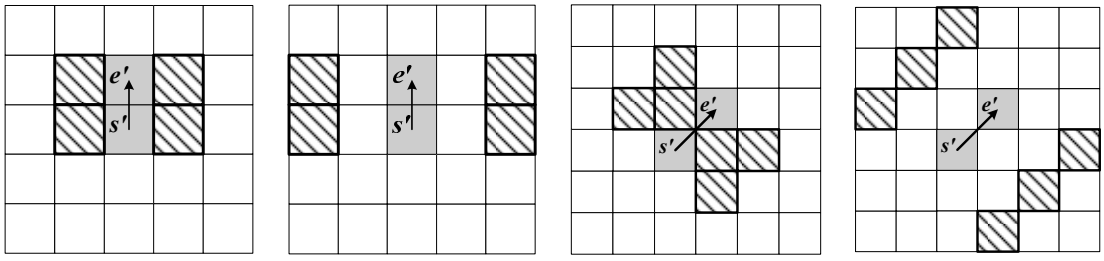


Fig. 1. The Hilbert curve of order 3

```
input: start point s and end point e
output: the candidate nearest neighbors
1: procedure FindRange(s, e)
2:   Call procedure FindNNBlocks(s)
3:   Call procedure FindNNBlocks(e)
4:   Call procedure FindRoute(RQueue, s, e)
5:   Call procedure LocalExpansion(RQueue, s, e)
6: end procedure
```

Fig. 2. Procedure *FindRange*

There are eight kinds of moving directions for the local expansion: north (N), south(S), east (E), west (W), northeast (NE), northwest (NW), southeast (SE), and southwest (SW). We classify them into two classes: (a) N, S, E, and W; (b) NE, NW, SE, and SW. Each class has the similar local expansion. For the first class, Fig. 3-(a) and Fig. 3-(b) show the local expansions of the N moving direction for the local start point *s*' and local end point *e*' with *SRadius*=1 and *SRadius*=2, respectively. *SRadius* indicates the distance away from the route in terms of blocks. In these figures, the search range are the east and west blocks of local start point *s*' and local end point *e*' and the block containing *e*'. The other three moving directions in the first class have the similar processing behavior with the different spreading search range. The local expansion is repeatedly processed with increasing the value of *SRadius* by one at a time, until the blocks of the current search range have data or the spread with *SRadius* is out of boundary of the search rectangle.



(a) (b) (c) (d)

Fig. 3. The local expansions: (a) the N direction with *SRadius*=1; (b) the N direction with *SRadius*=2; (c) the NE moving direction with *SRadius*=1; (d) the NE direction with *SRadius*=2.

For the second class, Fig. 3-(c) and Fig. 3-(d) show the local expansions of the NE moving direction with *SRadius*=1 and *SRadius*=2, respectively. In these figures, the search range are the northwest and southeast blocks of *s*', the west, northwest, south, southeast blocks of *e*' and the block containing *e*'. The other three moving directions in the second class have the similar processing behavior with the different spreading search range. In the second class, to avoid the missing candidate objects, after the blocks containing objects are found, the local expansion is spread one more time with the increase of the value of *SRadius* by one.

Take an example shown in Fig. 1 to illustrate the proposed method. A continuous nearest neighbor query is issued from the query line segment *se*. That is, the start block and the end block are blocks 13 and 45, respectively. First, the nearest neighbors of start block *s* and end block *e* are found out by procedure *FindNNBlocks*. Second, the route containing blocks {13, 11, 31, 32, 34, 45} shown in Fig. 1 is found out by procedure *FindRoute*. Third, the local expansions of the search range are processed by procedure *LocalExpansion*. The entire search range for this query is shown in Fig. 1. After the entire search range is examined in the first phase, we apply the heuristics mentioned in Zheng *et al*.'s method [7] to filter out the final answers from the candidate objects. The final answers are the blocks marked with "*" in Fig. 1.

Performance Evaluation

In this section, we compare the performances of *FM* and Zheng *et al*.'s method [7]. The uniform data set and the real data set are used to evaluate the performance of *FM* and Zheng *et al*.'s method. The uniform data set contains 10,000 points uniformly generated in a square Euclidean space. The real data set contains 5,922 points of cities and village of Greece (http://www.rtreeportal.org). These spatial objects are linearly ordered to the one-dimensional space from the two-dimensional space by the Hilbert curve. Two parameters are used in our simulation: *QLength* and *QAngle*. *QLength* is the ratio of the query length to the total side length of the two-dimensional space, and its value is in [0.1, 1]. *QAngle* is the angle relative to the *x*-axis of the CNN query, and its value is in [0,360). The position of each start point is randomly generated in the two-dimensional space. Simulation results are the average of 500 queries. Because the processing time is proportional to the number of fetched blocks, we use the processing time as the performance measure. In addition, we also make comparisons of the block accuracy. The block accuracy is the number of the final answers divided by the number of the search data blocks.

In the first experimental result, we evaluate the block accuracy and the processing time for the uniform data set with different values of *QAngle*: 0, 15, 30, and 45. These four degrees can represent other fifteen multiple degrees from 0 to 360. Fig. 4-(a) and Fig. 4-(b) show the block accuracy and the processing time of both methods with *QLength*=0.5, respectively. In Fig. 4-(a), we can observe that although the block accuracy of *FM* is affected by the value of *QAngle*, the block accuracy of *FM* is higher than that of Zheng *et al*.'s method. This is because the determined search range of Zheng *et al*.'s method is larger than that of *FM*. In Fig. 4-(b), we can observe that the processing time of *FM* is shorter than that of Zheng *et al*.'s method. Since the number of the candidate objects in Zheng *et al*.'s method is larger than that in *FM*, Zheng *et al*.'s method should take more time to process them in the second phase.

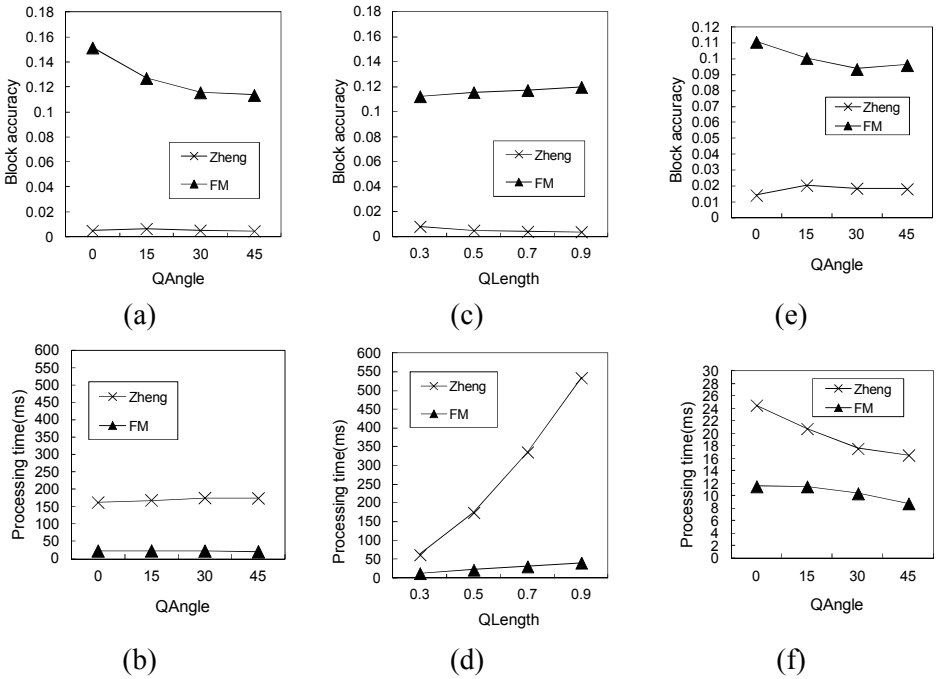


Fig. 4. Experimental results

In the second experimental result, we evaluate the block accuracy and the processing time for the uniform data set with different values of *QLength*: 0.3, 0.5, 0.7, and 0.9. Moreover, the value of *QAngle* is set to 30. Fig. 4-(c) shows that the block accuracy of *FM* is higher than that of Zheng *et al*.'s method. With the increase of the value of *QLength*, the block accuracy of *FM* is slightly increasing, and that of Zheng *et al*.'s method is slightly decreasing. Fig. 4-(d) shows that the processing time of *FM* is shorter than that of Zheng *et al*.'s method. Moreover, the processing time of Zheng *et al*.'s method is increasing dramatically with the increase of the value of *QLength*, whereas that of *FM* is increasing slightly. That means *FM* has a stable performance on the processing time.

In the third experimental result, we evaluate the block accuracy and the processing time for the real data set with different values of *QAngle*: 0, 15, 30, and 45. Moreover, the value of *QLength* is set to 0.3. Fig. 4-(e) shows that the block accuracy of *FM* is higher than that of Zheng *et al*.'s method. When the value of *QAngle* of the query is set to 0, the block accuracy is higher than that of other values of *QAngle*. For processing the query with *QAngle*=0, the size of each local search range of *FM* in the horizontal moving direction is the same. In this case, the search range is smaller as compared to the other cases. Therefore, this case achieves a higher accuracy than the others. On the other hand, when the value of *QAngle* of the query is not 0, the size of each local search range of *FM* is increased by the times of the spreading. Fig. 4-(f) shows the processing time of *FM* is shorter than that of Zheng *et al*.'s method. In the real data set, when the value of *QAngle* increases, the processing time of *FM* or Zheng *et al*.'s method decreases. The processing time of Zheng *et al*.'s method decreases quickly, when the value of *QAngle* increases. The reason is that the distribution of the data objects in the real data set is sparse, so the search range contains the smaller number of the candidate objects, which still be needed to be processed in the second phase.

Conclusions

In this paper, we have presented a forward moving method, *FM*, for continuous nearest neighbor queries. Our method uses an index, which records that which data blocks contain objects, to locally search the data blocks around the query line segment for the candidate objects. Therefore, we can determine a search range with a higher accuracy than that of Zheng *et al*.'s method. Experimental results show that *FM* can reduce the number of the fetched data blocks as compared to Zheng *et al*.'s method under two different data distributions, including the uniform data set and the real data set.

Acknowledgments

This research was supported in part by the National Science Council of Republic of China under Grant No. NSC-93-2213-E-110-027 and Grant No. NSC 99-2221-E-468-019.

References

[1] H.L. Chen and Y.I. Chang: Neighbor-finding based on space-filling curves, Information Systems 30(3) (2005) 205-226.

[2] H. L. Chen and Y. I. Chang: All-nearest-neighbors finding based on the Hilbert curve, Expert Systems with Applications 38(6) (2011) 7462-7475.

[3] Y. Gao, B. Zheng, G. Chen, Q. Li and X. Guo: Continuous visible nearest neighbor query processing in spatial databases, The VLDB Journal 20(3) (2011) 371-396.

[4] R.H. Guting: An introduction to spatial database systems, The VLDB Journal 3(4) (1994) pp. 1-32.

[5] T. Imielinski, S. Viswanathan, and B. R. Badrinath: Data on air: organization and access, IEEE Trans. on Knowledge and Data Eng., 9(3) (1997) 353-372.

[6] Y. Tao, D. Papadias and Q. Shen: Continuous nearest neighbor search, Proc. of the 28th Conf. on Very Large Data Base (2002) 287-298.

[7] B. Zheng, Lee, W.C. Lee, and D.L. Lee: Search continuous nearest neighbor on the air, Proc. of the 1st Annual Int. Conf. on Mobile and Ubiquitous Systems: Networking and Services (2004) 236-245.