International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 9 (2017) pp. 2054-2058
© Research India Publications. http://www.ripublication.com

Visible Neighbor Search Using Skyline Query

Jongwan Kim
Smith Liberal Arts College, Sahmyook University,815 Hwarang-ro, Nowon-gu, Seoul, 01795, Korea

ORCID: 0000-0003-4716-8380   Scopus Author ID: 36537411000

Abstract.
In LBS (location-based services), users search the nearest neighbors from users' location for their preferences. Conventional schemes for searching spatial objects are based on the Euclidean distance between a query point and objects. However, it is hard to access an invisible building, i.e., a building behind other buildings, and it is not preferred by users since they focus on visible objects. It is an important feature to separate the visible or invisible objects in LBS because users determine to access by the distance and the visibility.
In this paper, we propose a novel scheme, visible neighbor (VN) query that uses a skyline query in spatial data. This scheme searches the surrounding objects of the users, including the interval objects that are located between objects, using angles. Users can see and access the interval object as a surrounding object if it exists between two objects that are considerably near to each other. The accessible objects are evaluated by the angles, and processed on the skyline query to find suitable objects for the users.
Through simulation, it is observed that VN shows better performance in finding visible neighbors than the distance-based nearest neighbor (NN) search. NN has the highest computation cost of comparing all objects to find nearest object however there is no consideration about the interval objects in surroundings. Although VN needs much time to find interval objects with its angles, skyline query decreases the comparison cost searching surrounding objects.

Keywords: Nearest Neighbor, R-tree, Skyline, Visible Object, Visible Neighbor

INTRODUCTION

In recent years, rapid development in the mobile environment has led to an expansion in LBS field [1, 2]. Various services are provided such as maps that utilize user location and offer augmented reality, directions, and nearby restaurants search, etc. A location-based service is a one-dimensional information service that finds and provides nearby objects based on the user's location.

As the diversity of the services increases, queries for multiple attributes such as location and price are required. For example, users need to find "rooms with reasonable accommodation cost in nearby hotels." This is actively researched using a skyline query in the database field.

Since users generally prefer neighbors that are noticeable and accessible, it is very important to explore the user's visible objects. However, it takes a long time to compare all hidden and surrounding objects using the nearest neighbor (NN) search that is based on distance. Regardless of whether the objects are hidden or not, NN uses the proximity of the objects to the query point for searching.

Furthermore, in terms of skyline queries, a hotel hidden behind other objects is referred to as a skyline object if the hotel's accommodation cost is very low, despite it being out of the range of the user's visibility. Thus, distance-based search results cannot guarantee an optimal solution because despite being inexpensive and close to the destination, the hidden objects are inaccessible and not preferable to users.

In this study, we propose a visible neighbor search scheme that can search spatial objects around the user using a skyline query. Visibility is an important factor for the user's preference and convenience. Hence, searching for visible objects is necessary in various applications. The proposed scheme improves the search speed for visible neighbors with high preference by eliminating obscured objects in the queries, including the distance.

The main contributions of this paper are:

- We explore the accessibility of users with their preferences. The convensional schmes consider the distance from query point to object than accessibility.
- We present a novel scheme for searching the surrounding objects which is in visibility area.
- We focus on user preferences and accessibility to serve user-oriented services using VN scheme.
- We provide an extensive simulation of our VN algorithm and show the comparative benefits between NN and VN scheme.

The paper is organized as follows. In Section PRELIMINARIES, works concerning the technical knowledge and theory behind skyline queries and nearest neighbor search are reviewed. In Section VISIBLE NEIGHBOR SKYLINE SEARCH, we describe the proposed scheme, VN (visible neighbor) and its algorithm. We simulate and analyze the performance between the VN and NN schemes in Section SIMULATION. The paper is summarized in Section CONCLUSION.

PRELIMINARIES

Skyline query

Skyline query was suggested in 2001 [3] and has since become a popular scheme in the database and sensor network fields [4, 5, 6]. The skyline query searches the nearest objects from the query point in a two-dimensional space with x- and y-axes. In general, the axes are considered a two dimensional plot of an object that a hotel has the distance to the beach and the accommodation price. The goal of this multiple criteria is to find a nearest hotel to the beach and the lowest price.

The skyline query is performed as follows. The skyline query divides the quadrant-like object o9 in Fig. 1 to search for objects with more suitable conditions than the other objects. Objects in the first, second, and fourth quadrants around the object o9 are less suitable than o9 in terms of price or distance. Namely, the objects o2, o5, o6, and o8 are more expensive and farther than o9. Therefore, objects of the first quadrant are all out of the point of interest. It is believed that o9 dominates the objects of the first quadrant.

The second and fourth quadrants are re-evaluated similarly. Thus, by eliminating objects that do not satisfy the condition in each quadrant, the skyline can swiftly search spatial objects. If o9 is finally selected as shown in Fig. 1, o9 is called the dominator and the objects in the first quadrant that were removed from the skyline query results are called as dominatees.

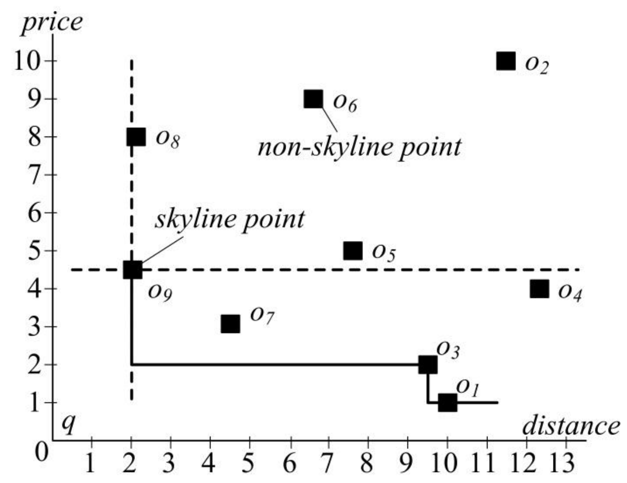

Figure 1: Skyline query and objects

Nearest Neighbor Search

The NN search [7, 8] is the most commonly used scheme for searching the nearest spatial data. It calculates the distance from the query point to the spatial objects represented by the MBR (Minimum Bounding Rectangle) as MINDIST (MINimum DISTance) based on the R-tree [6], and searches for the closest object. MINDIST is the closest minimum possible distance between the coordinates of the face that fits in the MBR and the query point, as shown in Fig. 2. It is very expensive to use NN queries for calculating the distance of all the spatial objects in the database in terms of I/O and CPU costs.

k-NN [9, 10] has been studied to search for nearest neighbor objects at a lower cost than NN. However, the number of objects defined as *k* does not decrease the overall computational cost. Therefore, k-NN does not overcome the drawbacks of NN.

NN and *k*-NN do not consider the visible or invisible objects in their algorithm and they have the highest searching time comparing all abjects. Hence, they are not suitable for searching VNs.

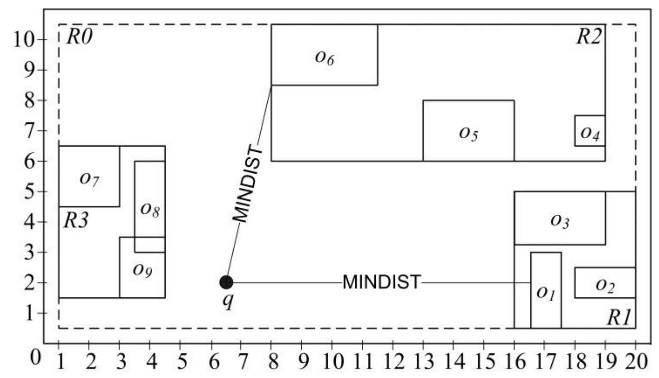

Figure 2: Nearest neighbor search

VISIBLE NEIGHBOR SKYLINE SEARCH

The VNs are spatial objects that can be viewed or accessed directly at the user's location and are not hidden by other objects. Since the mobile environment navigates around the spatial objects according to the user's movements, all identifiable objects in the location must be provided at any particular instance. Visible objects contain interval objects between different objects, either as visible objects directly at the user's location, or as spatial objects that the user can access. These objects are different from obscured objects. Interval objects are important because they are one of the visible objects that users can access directly. Thus, VNs include objects that are seen and the interval objects around the query point.

In this scheme, we find objects around the query point and then check for hidden or interval objects using angles. Angles are either constructed from an x- or y-axis to an object or calculated between two objects.

Initially, we identify the angles that are used in relations and the algorithm of VN. The notations used in the paper are θ for an angle and θ_{qi} for an angle from q to any object in i^{th}-quadrant.

> **Definition 1.** (Quadrant angle of search space).
> Let R0 be the search area of the dataset, and q be a query point in the search area. If we divide R0 around q into quadrants, each quadrant is divided by 90°. In this case, the angle of a quadrant is defined as θ_{qi} (i=1, ..., 4-quadrant) and the initial and final angles are $\theta_{qi}^{\square}$, and $\theta_{qi}^{\square}$, respectively. The angle of each quadrant is defined in equation (1), where the square brackets indicate the closed interval from the initial to the final angle.

$$\theta_{qi} = [\theta^{\square}_{qi}, \theta^{\square}_{qi}] = \frac{\pi}{2}$$
$$where: \theta^{\le}_{qi} = \frac{\pi}{2}(\mathrm{i}-1), \theta^{\ge}_{qi} = \frac{\pi}{2}\mathrm{i}, (\mathrm{i}=1,...,4) \quad . \tag{1}$$

Definition 2. (Angle of spatial objects).
Let o_i (i=1, ..., n) be an object in the first quadrant, and *rs* and *re* (s: start, e: end) be two points of the diagonal of an MBR. Currently, *rs* and *re* of o_i are $(o^{rs}_{ix}, o^{rs}_{iy})$ and $(o^{re}_{ix}, o^{re}_{iy})$, respectively. *ix* and *iy* represent an object of each axis. The angle θ_{oi} from the query point *q* in the first quadrant to objects o_i can be defined as ${}_{q1}\theta_{oi}$. In equation (2), the starting and end angles of an object in quadrant *i* are differentiated using □ and □, respectively. The angles are described below.

$$\theta_{oi} = [\theta^{\square}_{oi}, \theta^{\square}_{oi}] = (\tan {}_{q1}\theta^{re}_{oi} - \tan {}_{q1}\theta^{rs}_{oi}), (i = 1, 2, ..., n)$$
$$where: \tan\theta^{re}_{oi} = \frac{\overline{o^{re}_{ix}o^{re}_{iy}}}{\overline{\overline{qo^{re}_{ix}}}}, \theta^{re}_{oi} = \tan^{-1}\cdot\frac{\overline{o^{re}_{ix}o^{re}_{iy}}}{\overline{\overline{qo^{re}_{ix}}}},$$
$$\tan\theta^{rs}_{oi} = \frac{\overline{o^{rs}_{ix}o^{rs}_{iy}}}{\overline{\overline{qo^{rs}_{ix}}}}, \theta^{rs}_{oi} = \tan^{-1}\cdot\frac{\overline{o^{rs}_{ix}o^{rs}_{iy}}}{\overline{\overline{qo^{rs}_{ix}}}} \quad . \tag{2}$$

Fig. 3 shows the first quadrant of R0. The angle of the object o_1 that intersects the first and the fourth quadrant is divided into two parts. The angle is divided by a horizontal line from the query point *q* to the right end of R0 as the base line. The two angles are ${}_{q1}\theta_{o1}$ for the first quadrant and ${}_{q4}\theta_{o1}$ for the fourth quadrant. The angle of o_1 have to calculate twice in the upper and lower part of x-axis.

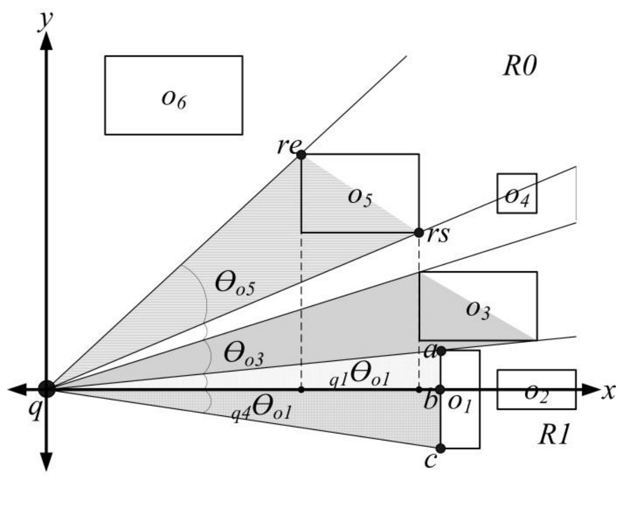

Figure 3: Spatial objects' angle in 1-quadrant

In this case, since point *a* is on the line that meets the right side of R0 from *q*, it constructs an angle $\angle aqb$. Similarly, point c constitutes an angle $\angle cqb$ and is calculated by Definition 1. The angle of tan ${}_{q1}\theta_{o1}$ is obtained by the arc tangent (See equation (3)).

Based on Definitions 1 and 2, the heuristic of the hidden and the visible objects are presented below.

$$\tan {}_{q1}\theta_{o1} = \frac{\overline{ab}}{\overline{\overline{qb}}}, {}_{q1}\theta_{o1} = \tan^{-1}\cdot\frac{\overline{ab}}{\overline{\overline{qb}}},$$
$$\tan {}_{q4}\theta_{o1} = \frac{\overline{bc}}{\overline{\overline{qb}}}, {}_{q4}\theta_{o1} = \tan^{-1}\cdot\frac{\overline{bc}}{\overline{\overline{qb}}} \quad . \tag{3}$$

Heuristic 1. (Identifying invisible objects).
Let o_i and o_{i+1} be spatial objects. If o_i is closer than other objects to *q* and an angle θ_{i+1} is included in θ_{oi}, then o_{i+1} is a hidden object behind o_i. The hidden object set o_{hid} is defined as follows.

$$o_{hid} = \{o_i \mid (MINDIST(o_i) < MINDIST(o_{i+1})) \wedge (\theta_{oi} \supset \theta_{oi+1})\} \quad . \tag{4}$$

The VN search is performed in a spatial index R-tree. There are two ways to search for VNs in R-tree. One involves determining whether the angle of the new MBR, R, is included in the angle of the object o that has already been found in the visible neighbor. Objects that are obscured like this are not considered further.

In addition, it searches for interval objects by judging the interval angle with other objects. If this is performed from the root of the R-tree, the entries of the node removed in the search order can search faster because there is no need to search the sub-entries.

```
Procedure: VN_Query(q, treefile)
Input: q is a query point
       treefile is a R-tree
Output: vnList is visible neighbors
Begin
1 RTree *rt ← new RTree(treefile);
2 rt->load_root();
3 do{
4  int n ← rt->entry_unums;
5  for(int entry = 0; entry < n; entry++)
6    vnList += TraverSubMbr(rt->node_ptr);
7  rt++;
8 }while(rt != NULL);
9 delete rt;
10 return vnList;
End
```

Figure 4: Visible neighbor algorithm

Fig. 4 shows the algorithm for the VN query. This algorithm accepts query point *q* and *treefile* which has spatial data. The R-tree index reads and constructs the spatial data from the file as given in the first line. The algorithm has a loop from third line to eighth line that do-while statement find visible neighbors from R-tree and accumulate VNs to *vnList*, which is a list of the visible neighbors. The fourth line reads the number of entries of a node and navigates the sub-nodes to search VNs by the *TraverSubMbr* function in line six. The discovered VNs are accumulated in the *vnList* and become skyline objects for the VNs.

SIMULATION

In this section, we evaluate the performance of the proposed algorithm. We conducted a simulation on a XEON server (2 CPU, 2.4 GHz, Quad core) with 32 GB RAM. The datasets are listed in Table 1.

Table 1: Summary of datasets and node size

Category	Real dataset	Synthetic dataset
MBR	65,536	100,000
Distribution	Skewed	Uniform
Region	Seoul/Korea	Generated
Node size	4096 Bytes	

The simulation evaluated the query performance on the skyline using real and synthetic spatial data. In Fig. 5, real data shows 65,536 buildings in Seoul of Korea [11], and 100,000 units of virtual data [12] are used for the experiments (Fig. 6).

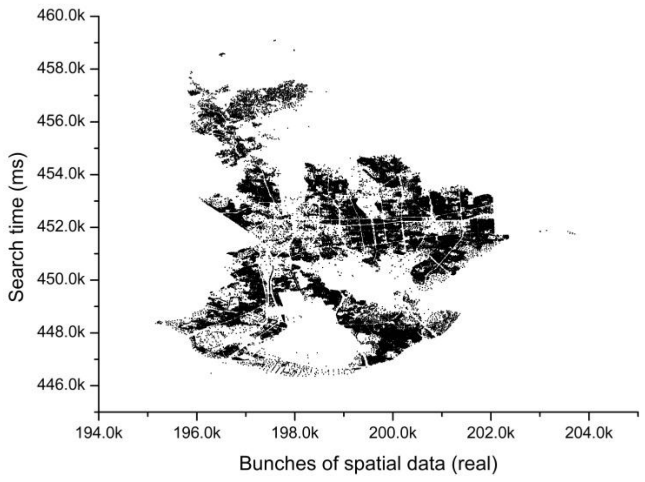

Figure 5: Spatial data of building in Seoul

We used real and synthetic datasets with skewed and standard distributions, respectively, to evaluate the performance of the proposed method. The skyline compares the two attributes of price and distance, and the price attribute is added by modifying each entry of the R-tree. Spatial data were divided into data sets of 10K, 20K, 40K, 60K, 80K, and 100K, randomly, and the experiments were conducted for each dataset.

Fig. 7 shows the results of searching visible objects with NN and VN by applying distances and angles respectively, in real data. NN takes more computation time than VN in data bunches over 60K. This is because of the skewed distribution of the real data. Thus, the comparison operations are concentrated in a particular section.

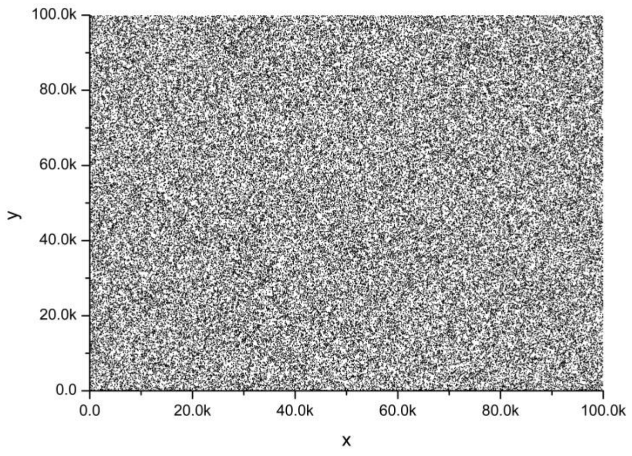

Figure 6: Synthetic spatial data

Fig. 8 shows the results for the experimental data with the standard distribution, and NN and VN show similar computation time. In each data set, NN takes a longer time to search for visible objects than VN. Because VN determines VN objects only when two objects intersect, it has much cheaper than NN in terms of computational cost.

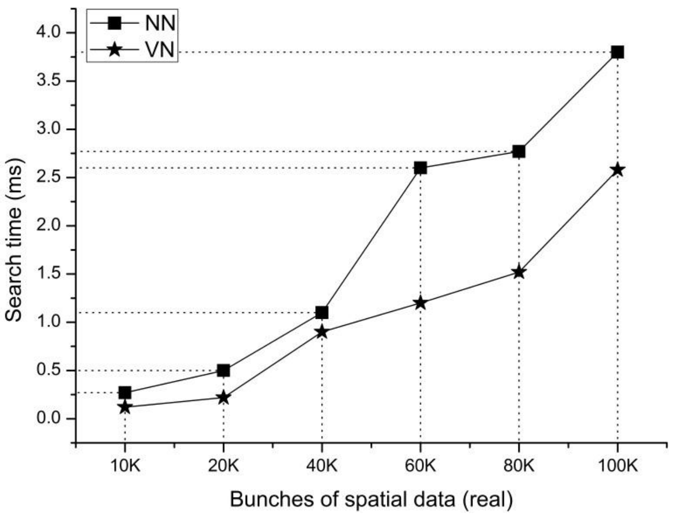

Figure 7: Search time in real dataset

CONCLUSION

Spatial information should make it easy for users to access spatial objects such as parks, cars, or hospitals. However, since conventional search systems search for objects based on distance, it is not easy to access objects that are hidden behind other objects. we search for data that is easy for users to access in this study. We search for surrounding objects that can be identified by the user, and exclude the objects hidden behind other objects.

Experimental results show that VN is faster than NN when searching for the same number of nearby objects. Therefore, this scheme is expected to expand its application in data retrieval and recommendation systems.

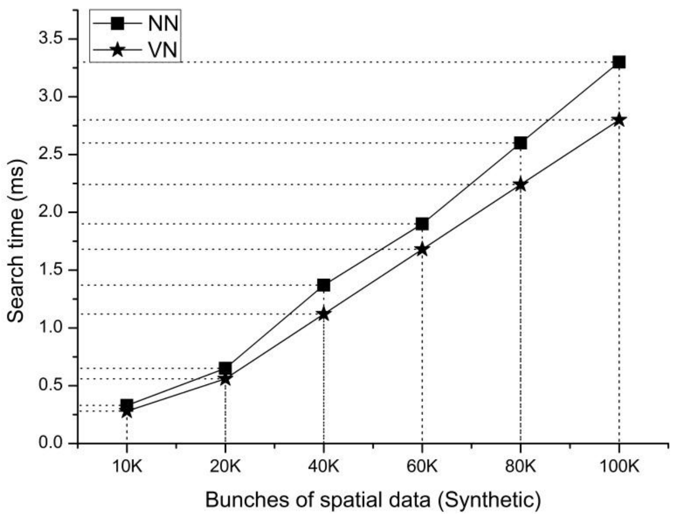

Figure 8: Search time in synthetic dataset

ACKNOWLEDGEMENTS

This work was supported by a National Research Foundation of Korea Grant funded by the Korean Government [NRF-2011-357-D00190] and [NRF-2014R1A1A2059902].

REFERENCES

[1] Jongwan K. (2007). SQR-tree: A Spatial Index Using Semi-quantized MBR Compression Scheme in R-tree*. Journal of Information Science and Engineering, Vol. 23, pp. 1541-1563.

[2] Stephanie R. Matthias M. and Markus B. (2016). Location-Based Services. Business & Information Systems Engineering, Vol. 58 Issue 3, pp. 233-237.

[3] Stephan B., Donald K. and Konrad S. (2001). The Skyline Operator. ICDE, pp. 421-430.

[4] Eleftherios, T., Apostolos, N. P. and Yannis M. (2015) Skyline Queries: an Introduction. Information, Intelligence, Systems and Applications (IISA), pp. 1-6.

[5] Hekang C., Shuigeng Z. and Jihong G. (2007). Towards Energy-Efficient Skyline Monitoring in Wireless Sensor Networks. EWSN, LNCS 4373, pp. 101-116.

[6] Yohan J. R., Inchul S., Joo H. J., Kyoung G. W. and Myoung H. K. (2013). Energy-Efficient Two-Dimensional Skyline Query Processing in Wireless Sensor Networks. 10th Annual IEEE-CCNC Smart Spaces and Sensor Networks, pp. 294-301.

[7] Yufei T., Dimitris P. and Qiongmao S. (2002). Continuous Nearest Neighbor Search. VLDB, pp. 287-298.

[8] Nick R., Stephen K. and Frederic V. (1995). Nearest Neighbor Queries. SIGMOD, pp. 71-79.

[9] Ugur D., Farnoush B. K. and Cyrus S. (2010). Efficient K-Nearest Neighbor Search in Time-Dependent Spatial Networks. DEXA, pp. 432-449.

[10] Huu V. L. C., Trong N. P., Minh Q. T., Thanh L. H. and Minh N. Q. T. (2016). Processing All k-Nearest Neighbor Query on Large Multidimensional Data. International Conference on Advanced Computing and Applications, pp. 11-17.

[11] Building location information of Seoul, Korea. (2016). http://data.seoul.go.kr/openinf/sheetview.jsp?infId=OA-13224&tMenu=11.

[12] Spatial Data Generator. DaVisual Code1.0.