DEIM Forum 2010 F3-3

方向に基づく空間スカイライン問合せ

郭　　茜[†]　石川　佳治[††]

[†] 名古屋大学情報科学研究科，〒 464–8601 名古屋市千種区不老町
[††] 名古屋大学情報基盤センター，〒 464–8601 名古屋市千種区不老町
E-mail: [†]guoxi@db.itc.nagoya-u.ac.jp, [††]ishikawa@itc.nagoya-u.ac.jp

あらまし　位置に基づくサービスはモバイル技術の一種であり，典型的な応用としては，地理情報を分析・利用してモバイルユーザのためにアイテム（例：レストラン，駐車場）を推薦するものがある．本論文では，最も近いという観点だけでなく，可能な移動方向を考えたときに近いと考えられるアイテムを推薦する位置ベースのサービスを提案する．よりよいアイテムを選ぶために，距離でなく方向の属性も考慮して比較を行う．この問題は，複数の要素を考えて決定を行う，意思決定のために新たなタイプのスカイライン問合せである．本研究ではこの問題を**位置に基づく空間スカイライン**（direction-based spatial skyline, DSS）問題と呼ぶ．あるアイテムが他のアイテムを**支配**するのは，それらが同じ方向にあり，前者が後者よりもユーザに近い場合である．あるアイテムがどのアイテムにも支配されないとき，それは DSS 上にあることになる．本研究では，ユーザに対しすべての DSS オブジェクトを提示することを目的としている．本稿では DSS を見つけるための効率的なアルゴリズムを示す．

キーワード　空間データベース，スカイライン問合せ，方向

DirectionBased Spatial Skyline Queries

Xi GUO[†] and Yoshiharu ISHIKAWA[††]

[†] Graduate School of Information Science, Nagoya University, Furo-cho, Chikusa-ku, Nagoya, 464–8601 Japan
[††] Information Technology Center, Nagoya University, Furo-cho, Chikusa-ku, Nagoya, 464–8601 Japan
E-mail: [†]guoxi@db.itc.nagoya-u.ac.jp, [††]ishikawa@itc.nagoya-u.ac.jp

Abstract　A location-based service is a kind of mobile technologies which suggests items (restaurants, car parks, etc.) to the mobile device users by utilizing and analyzing the geographical information. In this paper, we propose a new location-based service to suggest the user nearer items in all directions rather than only the nearest item. We compare the items and find better ones considering both the distance and the direction attributes. This problem is a new skyline problem which takes into account two or more attributes to make decisions. We name this new problem the *direction-based spatial skyline* (DSS) problem. One item *dominates* another item if it is closer to the user and they are in the same direction. If an item cannot be domianted by any other items, it is on the DSS. The items on the DSS are the nearer ones around the user. We also propose efficient algorithms to find the objects which are on the DSS.

Key words　Spatial databases, skyline queries, directions

1. Introduction

A location-based service is a mobile technology which gives the mobile device user suggestions by using the geographical information. It is an application or a system running in the user's mobile device. It positions the user, analyzes the geographical information, and presents suggestions to help the user make a reasonalbe decision. The automotive navigation system is a typical location-based service. This system uses a GPS navigation device to position the user and guides the user to his destination through analyzing the maps in its database. The automotive navigation system has been studied for about twenty yeares. Recently, with the development of mobile technologies, many new location-based services are springing up. In this paper, we put forward a new location-based service for helping the user to find nearer places or objects all around.

Finding nearer places or objects in all directions is a familiar problem in our daily life. In most cases, we would like to know the nearer surrounding targets rather than only the nearest one. A tourist prefers acquiring the nearer sight spots in all directions. Because it is not enough for her to make a sightseeing plan if she only knows the nearest one. A shopper would like to know all the nearby discount shops around to make her shopping plan. Only the nearest one seems too few for a shopper. A soldier has to be fully aware of the nearby enemies in all directions. Because it will be dangerous if she only notices the nearest enemie in a battle field. On the grounds of these practical requirements, we develop a new location-based service to help people find nearer targets in all directions by utilizing the mobile device facilities.

In order to find the nearer objects in all directions, we describe the objects considering both their distances and their directions. We assume the positions of the objects $p_i \in P$ and the user q are points in the two-dimensional Euclidean space $\mathcal{R}^2$. The object's vector, which starts from the user and ends in the object, indicates the distance and direction of the object. The distance is the vector's quantity which is the Euclidean distance between the user and the object. The direction is the vector's direction pointing to the object from the user's position. We compare the distance of two objects if they are in the same direction. And the nearer one is better than the further one. However, whether two objects are in the same direction or not is a matter of opinion. Intuitively two objects are in the same direction if the included angle between their vectors is smaller than an acceptable angle. In our work, this acceptable angle is given by the user and we denote it as θ.

Finding the nearer objects in all directions is a new skyline problem. The skyline problem is selecting better objects by comparing them in two or more attributes. If one object is better than another object in all attributes, it dominates another object. If one object cannot be dominated by any other objects, it is on the skyline. So the objects on the skyline are better objects considering all the attributes. In our case, we compare objects in the distance and direction attributes. One object can dominate another object if it is nearer and they are in the same direction. Our direction-based spatial skyline (DSS) is to find out the objects which are not dominated by any other objects considering both the distance and direction attributes.

Example 1 Figure. 1 and Figure. 2 shows an example of finding neaer objects in all directions. There are 9 objects (a to i) and the user q. The θ given by the user is $\frac{\pi}{4}$. As Figure. 1 shows, we describe objects a and b by using vector $\overrightarrow{QA}$ and vector $\overrightarrow{QB}$. The distances of a and b are $|\overrightarrow{QA}|$ and $|\overrightarrow{QB}|$. And the directions of a and b are the directions of vector $\overrightarrow{QA}$ and vector $\overrightarrow{QB}$. Objects a and b are in the same direction because the included angle between $\overrightarrow{QA}$ and $\overrightarrow{QB}$ is smaller than θ. Figure. 2 shows the nearer objects around the user (the red points). And they are the objects on the direction-based spatial skyline. Objects b, h and f are dominated by object a because they are in the same direction with object a but further than a. Likewise, objects g and i is dominated by objects c and e respectively. The objects a, c, d and e cannot be dominated by any other objects and they are on the direction-based spatial skyline. ■

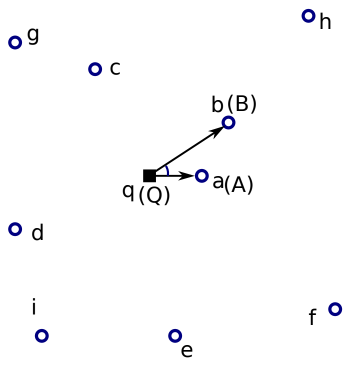

Figure. 1 Describe objects by vectors($\theta = \frac{\pi}{4}$)

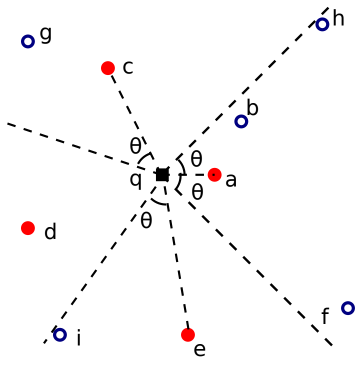

Figure. 2 Direction-based spatial skyline($\theta = \frac{\pi}{4}$)

2. Preliminaries

We define our problem in the two-dimensional Euclidean space $\mathcal{R}^2$. The objects $P = \{p_i | i \in [1,\ n]\}$ and the user q are all points with xy-coordinates in $\mathcal{R}^2$. For simple, we create the xy-coordinates by using the user's position as the origin $O(0,\ 0)$ and the direction of the vector $\overrightarrow{ON}$ as the x-axis. (N is the nearest neighbor of the user.) The vector from the user q ($O(0,\ 0)$) to the object p ($P(x_p,\ y_p)$) indicates the distance and the direction of this object.

Definition 1 (Distance) The quantity of the vector $\overrightarrow{OP}$ is the *distance* d_p of the object p.

$$d_p = |\overrightarrow{OP}| = \sqrt{x_p^2 + y_p^2} \qquad (1)$$

. □

Definition 2 (Direction) The *direction* of the object p is the direction of vector $\overrightarrow{OP}$. We describe the direction by using the counter-clockwise angle from the x-axis to the vector $\overrightarrow{OP}$. And we denote this angle as α_p.

$$\alpha_p = \begin{cases} \arctan \frac{y_p}{x_p}, & x_p > 0,\ y_p \geqq 0 \\ \arctan \frac{y_p}{x_p} + \pi, & x_p < 0 \\ \arctan \frac{y_p}{x_p} + 2\pi, & x_p > 0,\ y_p \leqq 0 \\ \frac{\pi}{2}, & x_p = 0,\ y_p \geqq 0 \\ \frac{3\pi}{2}, & x_p = 0,\ y_p \leqq 0 \end{cases} \qquad (2)$$

□

where the arctangent value ranges in its usual principal value $[-\frac{\pi}{2},\ \frac{\pi}{2}]$.

There are two objects p_i (at the position $P_i(x_{p_i}, y_{p_i})$) and p_j (at the position $P_j(x_{p_j},\ y_{p_j})$). If the included angle between the vector $\overrightarrow{OP_i}$ and the vector $\overrightarrow{OP_j}$ is smaller than θ (given by the user), the two objects are in the *same direction*.

Definition 3 (Same Direction) For two objects p_i and p_j, we denote their included angle $\angle P_iOP_j$ as $\beta_{p_ip_j}$.

$$\beta_{p_ip_j} = \arccos \frac{\overrightarrow{OP_i} \cdot \overrightarrow{OP_j}}{|\overrightarrow{OP_i}||\overrightarrow{OP_j}|} = \arccos \frac{x_{p_i}x_{p_j} + y_{p_i}y_{p_j}}{d_{p_i}d_{p_j}}. \quad (3)$$

If $\beta_{p_ip_j} < \theta$, p_i and p_j are in the *same direction*. □

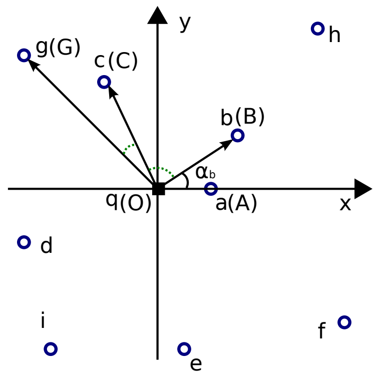

Figure. 3 The directions of the objects ($\theta = \frac{\pi}{4}$)

Example 2 Figure. 3 shows an example. We create the xy-coordinates by using the user's position as the origin $O(0, 0)$ and the direction of the vector $\overrightarrow{OA}$ as the x-axis. The direction of the object b is the direction of the vector $\overrightarrow{OB}$. We use the counter-clockwise angle α_b which starts from the x-axis and ends in $\overrightarrow{OB}$ to denote this direction. Likewise, we can also denote the directions of objects g and c. If $\theta = \frac{\pi}{4}$, the objects c and g are in the same direction becasue $\angle COG < \theta$. On the contrary, the objects c and b are in the different direction because $\angle BOC > \frac{\pi}{4}$. ■.

We compare two objects considering both the distance attribute and the direction attribute. If one object p_i is nearer than another object p_j and they are in the same direction, the object p_i is better than p_j. We also say that the object p_i dominates the object p_j.

Definition 4 (Dominate) For two objects p_i and p_j, if $d_{p_i} < d_{p_j}$ and $\beta_{p_ip_j} < \theta$, the object p_i *dominates* the object p_j. We denote the dominance relationship as $p_i \prec p_j$ (or $p_j \succ p_i$). □

Definition 5 (Direction-Based Spatial Skyline) If one object p_i cannot be dominated by any other objects, it is on the *d̲irection-based s̲patial s̲kyline (DSS) S*.

$$S = \{p_i | p_i \nsucc p_j,\ \forall p_j \in P - \{p_i\}\}. \qquad (4)$$

□

3. Direction-Based Spatial Skyline Query Algorithms

A naïve solution to this problem is to compare every object p_i with all the other objects in both of the direction and distance attributes. If p_i is closer than any other points which are in the same direction with it, it is on the DSS S. The time complexity of this solution is $O(n^2)$. Obviously, we can improvement the procdure considering when an object has already been dominated by another object, it is unseated to be on the DSS. However, the cost is still quite large.

We propose an efficient method to answer the DSS query efficiently. The method is based on nearest neighbor queries in spatial databases and it can utilize spatial indexes such as R-tree efficiently. The idea is simple. At first, we find the nearest neighbor p_1 for the user's position O. p_1 is on the DSS because it is better than any other objects in the distance attribute. The object p_1 dominates the objects which are in the same direction with it. In other words, it becomes the dominator of its 2θ area. Next we get the second nearest neighbor p_2. If it falls into the area dominated by p_1, it is not on the DSS. Otherwise, it becomes another object on the DSS. No matter it is on the DSS or not, it can dominate a certain area which are not dominated by p_1. We repeat this procedure for the remaining objects from the nearest one. As the process goes, the area dominated by the observed objects enlarges. The process continues when all the angles are covered.

Example 3 Fig. 4 shows an example of finding the DSS based on the idea, where $\theta = \frac{\pi}{4}$. Fig. 4(a) shows the initial step. The object a is the nearest neighbor and is on the DSS. It dominates a 2θ angle $\angle A_1OA_2$. Next we check the

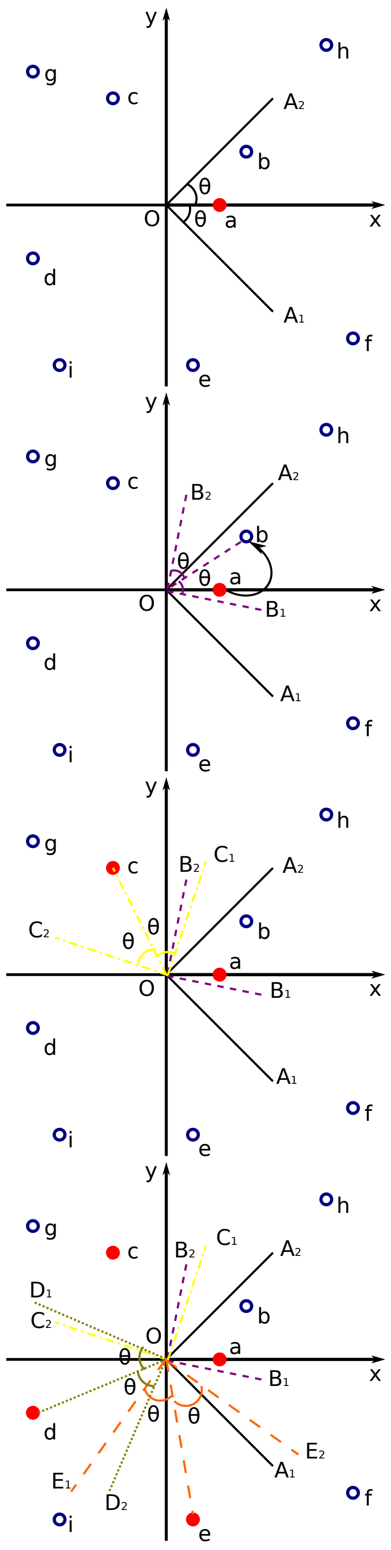

Figure. 4 Processing a DSS query ($\theta = \frac{\pi}{4}$)

second nearest neighbor b as in Fig. 4(b). It falls into the dominated angle $\angle A_1OA_2$ of the object a and cannot be on the DSS. Because the object b also dominates a 2θ angle, the dominated area is enlarged as $\angle A_1OB_2$. Then we check the third nearest neighbor c as in Fig. 4(c). It is outside of the dominated area and becomes an another object on the DSS. And the dominated angle is enlarged to the angle $\angle A_1OC_2$. Likewise, we check the fourth nearest neighbor d and the fifth nearest neighbor e as in Fig. 4(d). They do not fall into the dominated angles and become the objects on the DSS. And the full angles 2π are covered. The process terminates and we have found the final DSS points set $S = \{a, c, d, e\}$. Note that we do not have to access other objects g, h, i and f in the query process. ■

The procedure is summarized in Algorithm 1. The algorithm assumes that we can use the nearest neighbor query facility. The symbol C represents a set of ranges of angles such as $C = \{[10, 20], [30, 50]\}$, where $10, \ldots, 50$ are degrees. The algorithm is fast because it does not check all the points.

Algorithm 1 DSS Query

```
1:  procedure DSSQuery($Q, \theta$)
2:      $S \leftarrow \emptyset$;                        ▷ Set of DSS points
3:      init_NN_query($Q$);                   ▷ Initialize the NN query
4:      $C \leftarrow \emptyset$;                 ▷ Initialize the covered angle set
5:      repeat
6:          $p \leftarrow$ get_next();                  ▷ Get the next NN point
7:          $\alpha_p \leftarrow$ direction($p$);        ▷ Caculate the direction of $p$
8:          if $\alpha_p \notin C$ then       ▷ Object $p$ is not dominated by $S$
9:              $S \leftarrow S \cup \{p\}$;           ▷ Object $p$ is on the DSS
10:         end if
11:         $C \leftarrow C \cup \{[\alpha_p - \theta, \alpha_p + \theta]\}$;     ▷ Update the covered
    angles
12:     until $C = \{[0, 360]\}$          ▷ all the $2\pi$ angles are covered
13:     output $S$;
14: end procedure
```

3.1 Discussion

Specially if every nearest neighbor is in the same direction with its previous nearest neighbor, there is only the first nearest neighbor on the DSS. Because every nearest neighbor can be dominated by its previous nearest neighbor if they are in the same direction. This is the worst case for our direction-based spatial skyline. Intuitively, in this case the user will not satisfy with the query results because the objects suggested are too few but actually the nearer objects in different directions exist.

Example 4 Figure. 5 shows an example of this worst case. The distribution of the objects is like a spire. Considering the distance attribute, $d_a < d_b < d_c < d_g < d_h < d_d < d_i < d_e < d_f$. Considering the direction attribute, $\beta_{ab} < \theta$, $\beta_{bc} < \theta$, $\beta_{cg} < \theta$, $\beta_{gh} < \theta$, $\beta_{hd} < \theta$, $\beta_{di} < \theta$, $\beta_{ie} < \theta$ and $\beta_{ef} < \theta$. Therefore, $a \prec b$, $b \prec c$, $c \prec g$, $g \prec h$, $h \prec d$, $d \prec i$, $i \prec e$ and $e \prec f$. So all the objects are dominated by their previous nearest neighbor except the object a. In this case there is only one object a on the skyline. However, intuitively the other objects also seem to be good choices for the user. ■

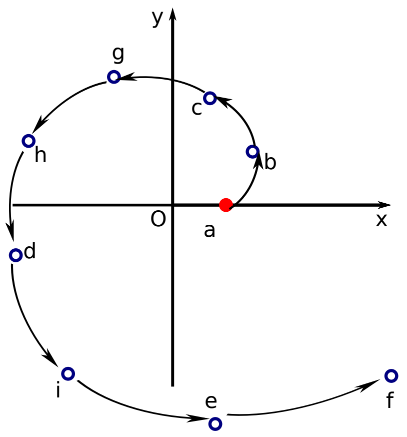

Figure. 5 The worst case ($\theta = \frac{\pi}{4}$)

In order to solve this problem, we extend our original definitions of the DSS and propose the *comparable scope DSS*.

4. Comparable Scope DSS

In the worst case of our DSS definition, if every nearest neighbor is in the same direction with its previous nearest neighbor, there is only the first nearest neighbor on the DSS. In order to solve this problem, we extend our original definitions of the DSS and propose the *comparable scope DSS*.

Definition 6 (Comparable Scope) The *comparable scope* of the object p is its 2θ angles $[\alpha_p - \theta, \alpha_p + \theta]$. We denote the *comparable scope* as λ_p. □

If another object p_j falls into the comparable scope λ_{p_i} of the object p_i, their included angle $\beta_{p_ip_j}$ is smaller than θ. If $d_{p_i} < d_{p_j}$, strictly speaking, the object p_i is better than the object p_j only in their common comparable scopes.

Definition 7 (Scope Dominate) If $d_{p_i} < d_{p_j}$, p_i *scope dominates* p_j in their common comparable scopes $\gamma_{p_ip_j} = \lambda_{p_i} \cap \lambda_{p_j}$. We denote this dominant relationship as $p_i \prec_{\gamma_{p_ip_j}} p_j$ (or $p_j \succ_{\gamma_{p_ip_j}} p_i$). □

Definition 8 (Fully Dominate) The object p_i is scope dominated by objects $\{p'_j | j \in 1..k, p'_j \in P - \{p_i\}\}$.

$$p_i \succ_{\gamma_{p_ip'_1}} p'_1;$$
$$p_i \succ_{\gamma_{p_ip'_2}} p'_2;$$
$$...$$
$$p_i \succ_{\gamma_{p_ip'_k}} p'_k,$$

If $\gamma_{p_ip'_1} \cup \gamma_{p_ip'_2} \cup ... \cup \gamma_{p_ip'_k} = \lambda_{p_i}$, the object p_i is *fully dominated.* □

Then we can define the *Comparable Scope DSS* based on the definition of the fully dominate.

Definition 9 (Comparable Scope DSS) If the object p_i cannot be fully dominated, it is on the *Comparable Scope DSS (CDSS)*. We denote the skyline as CS.

$$CS = \{p_i | p_i \nprec_{\eta_{p_i}} p_j,\ \forall p_j \in P - \{p_i\},\ \eta_{p_i} \subseteqq \lambda_{p_i}\} \qquad (5)$$

. □

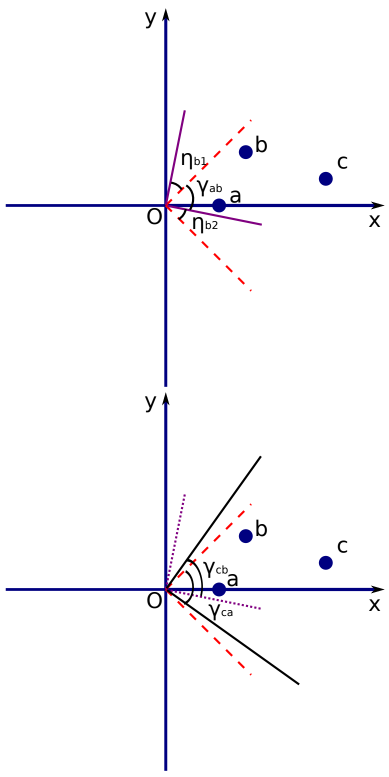

Figure. 6 Comparable Scope DSS ($\theta = \frac{\pi}{4}$)

Example 5 Figure. 6 shows an example of the CDSS. The comparable scope of the object a is the $\lambda_a = [\alpha_a - \theta, \alpha_a + \theta]$. Fig. 6 (a) shows the object a dominates the object b in their common comparable scope $\gamma_{ab} = \lambda_a \cap \lambda_b$. And the object b cannot be fully dominated because it cannot be dominated by neither the object a nor the object c in the scopes η_{b_1} and η_{b_2}. Fig. 6 (b) shows the object c is dominated by the object a in γ_{ca} and is dominated by the object b in γ_{cb}. And the object c is fully dominated because $\gamma_{ca} \cup \gamma_{cb} = \lambda_c$. Therefore, the objects a and b cannot be fully dominated and they are on the CDSS. ■.

We can process the CDSS queries by using the basic idea of the DSS query's solution. At first, we find the nearest object p_1 for the user's position O. The object p_1 is on the CDSS because it is closer to the user than any other objects in its comparable scope λ_{p_1}. Next we get the second nearest neighbor p_2. If it falls into λ_{p_1}, it is dominated by p_1 in their common comparable scope $\gamma_{p_1p_2}$. It is on the CDSS if $\gamma_{p_1p_2} \mp \lambda_{p_2}$. Otherwise, it is not on the CDSS. We repeat this procedure for the remaining objects from the nearest one until the scope covered by the observed objects enlarges to all angles 2π. The procedure is summarized in Algorithm 2 which has some small changes of Algorithm 1.

Algorithm 2 CDSS Query

```
 1: procedure CDSSQUERY($Q, \theta$)
 2:     $CS \leftarrow \emptyset$;                          ▷ Set of CDSS points
 3:     init_NN_query($Q$);                       ▷ Initialize the NN query
 4:     $C \leftarrow \emptyset$;                  ▷ Initialize the covered angle set
 5:     repeat
 6:         $p \leftarrow$ get_next();              ▷ Get the next NN point
 7:         $\alpha_p \leftarrow$ direction($p$);      ▷ Caculate the direction of $p$
 8:         $\lambda_p \leftarrow [\alpha_p - \theta, \alpha_p + \theta]$; ▷ Set the comparable scope of $p$
 9:         $\eta_p \leftarrow \lambda_p - \lambda_p \cap C$;     ▷ Set the undominated scope $\eta_p$
10:         if $\eta_p \neq \emptyset$ then                    ▷ Scope $\eta_p$ exists
11:             $CS \leftarrow CS \cup \{p\}$;          ▷ Object $p$ is on the CDSS
12:             $C \leftarrow C \cup \eta_p$;          ▷ Update the covered angles
13:         end if
14:     until $C = \{[0, 360]\}$           ▷ all the $2\pi$ angles are covered
15:     output $CS$;
16: end procedure
```

5. Experiments

The experiments were implemented by using a PC with an Intel Pentium CPU (3.00 GHz), 2GB of memory, and Fedora 11 OS. The dataset was made by extracting the midpoints for each line segment of the LBeach dataset. (LBeach datasets are road line segments of Long Beach from the TIGER database http://tiger.census.gov/.) This dataset consisted of 50747 points and was normalized in $[0,\ 1000]^2$ space.

We evaluated the performance of the intantaneous DSS query by the number of the processed nearest neighbors and the number of the DSS points. Fig. 7 shows the average number of the processed nearest neighbors. The number decreases with the increase of θ and is far smaller than the total number of points. When the θ is larger, the nearest neighbor can dominate larger angle ranges and the algorithm terminates quickly. Fig. 8 shows the number of DSS points with repect to the different θs. The number also decreases with the increase of θ because when the θ is larger, one DSS point can cover larger area and dominate much more points.

We also captured some images of the DSS points with different θs and different user positions. As Fig. 9 shows, the larger point is the user's position, the solid points are the DSS points and the hollow points are the other points.

6. Related Work

Traditionally, the skyline query problem is known as the maximum vector problem [2] in mathmatical field. Skyline query has attracted more and more attentions in database

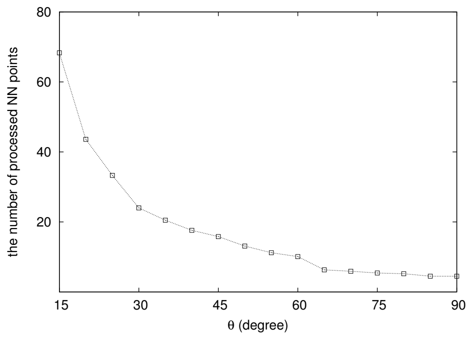

Figure. 7 The Number of Processed Nearest Neighbors

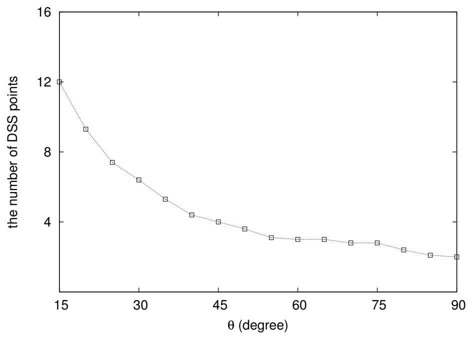

Figure. 8 The Number of DSS Points

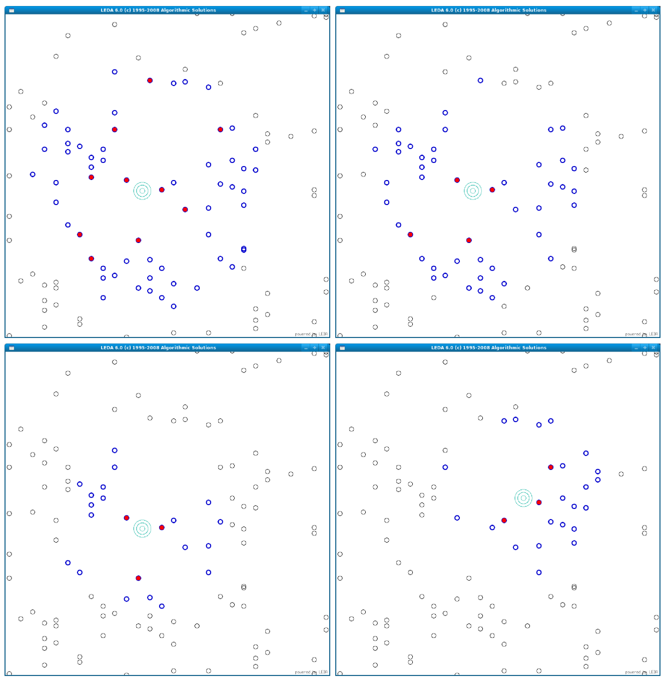

Figure. 9 The Images of DSS Points

area since 2001 when the first paper [1] considering skyline queries in relational databases appeared. Afterwards, many subsequent algorithms [3] [5] [6] appeared to optimize it.

Moreover, skyline queries in spatial database becomes a hot issue accompanied with the development of mobile technology. Most spatial skyline queries [7] [8] are based on the distance aspect. [7] considers how to find out better targets with regard to several possible locations of the query. [8] proposed the method to find out the skyline considering distance aspect and non-spatial aspects.

7. Conclusion

In this paper, we propose the definiton of the *direction-based spatial skyline*(DSS). And we design an efficient algorithm to answer the DSS query. Then we extend our basic definition to the *comparable scope DSS* and we also design an algorithm to answer the CDSS query.

Acknowledgments

This research was partly supported by a Grant-in-Aid for Scientific Research (#19300027, #21013023) from the Japan Society for the Promotion of Science (JSPS).

文　献

[1] Stephan Börzsönyi and Donald Kossmann and Konrad Stocker, *The Skyline Operator*, ICDE, 2001.
[2] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, New York: Springer-Verlag, 1985.
[3] Dimitris Padadias and Yufei Tao and Greg Fu and Bernhard Seeger, *An Optimal and Progressive Algorithm for Skyline Queries*, SIGMOD, 2003.
[4] Chee-Yong Chan and Pin-Kwang Eng and Kian-Lee Tan, *Stratified Computation of Skylines with Partially-Ordered Domains*, SIGMOD, 2005.
[5] J. Chomicki and P. Godfrey and J. Gryz and D. Liang, *Skyline with presorting*, ICDE, 2003.
[6] Parke Godfrey Ryan and Ryan Shipley and Jarek Gryz, *Maximal Vector Computation in Large Data Sets*, VLDB, 2005.
[7] Mehdi Sharifzadeh and Cyrus Shahabi, *The Spatial Skyline Queries*, VLDB, 2006.
[8] Baihua Zhang and Ken C. K. Lee and Wang-Chien Lee, *Location-Dependent Skyline Query*, MDM, 2008.
[9] Zhiyong Huang and Hua Lu and Beng Chin Ooi and Anthony K. H. Tung, *Continuous Skyline Queries for Moving Objects*, TKDE, 2006.
[10] Katerina Raptopoulou and Apostolos Papadopoulos and Yannis Manolopoulos, *Fast Nearest-Neighbor Query Processing in Moving-Object Databases*, GeoInformatica, 2003.
[11] YuFei Tao and Dimitris Papadias and Qiongmao Shen, *Continuous Nearest Neighbor Search*, VLDB, 2002.
[12] http:en.wikipedia.orgwikiQuartic_function