Spatial Range Querying for Gaussian-Based Imprecise Query Objects

Yoshiharu Ishikawa [1], Yuichi Iijima [2], Jeffrey Xu Yu [3]

[1]*Information Technology Center/* [2]*Graduate School of Information Science, Nagoya University, Japan*
[1]ishikawa@itc.nagoya-u.ac.jp, [2]iijima@db.itc.nagoya-u.ac.jp

[3]*The Chinese University of Hong Kong, China*
[3]yu@se.cuhk.edu.hk

***Abstract*— In sensor environments and moving robot applications, the position of an object is often known imprecisely because of measurement error and/or movement of the object. In this paper, we present query processing methods for spatial databases in which the position of the query object is imprecisely specified by a probability density function based on a *Gaussian distribution*. We define the notion of a *probabilistic range query* by extending the traditional notion of a spatial range query and present three strategies for query processing. Since the qualification probability evaluation of target objects requires numerical integration by a method such as the Monte Carlo method, reduction of the number of candidate objects that should be evaluated has a large impact on query performance. We compare three strategies and their combinations in terms of the experiments and evaluate their effectiveness.**

I. Introduction

In recent years, much research on the representation and processing of imprecise and uncertain data has been undertaken. For example, information obtained from sensors is often imprecise, meaning that it is often uncertain as to whether the obtained data can be treated as being accurate. Moreover, in the context of heterogeneous information integration, the quality of information contained in the underlying information sources is not necessarily sufficient and may contain vagueness. In response to such problems, several proposed solutions such as data models for representing uncertain data directly and query processing techniques for handling imprecise data have appeared in the literature [10], [16].

In this paper, we propose query processing methods for spatial range queries based on imprecise location information. Specifically, we consider a situation in which the location of a query object is known imprecisely. There has been much interest in query processing techniques based on imprecise location information in recent years (e.g., [5], [9], [20]). Let us consider a database system for use with a mobile sensor environment. A GPS system is often used to detect the location of an object, but it is not always possible to receive a GPS signal in every situation. Moreover, power consumption of GPS is an important factor to consider when the system is powered by a battery. In such a case, it may not be possible to update the location information frequently; consequently, the accuracy with which the location of a moving object is known becomes lower.

In the context of moving object databases, problems due to imprecise location information may occur. For example, when we monitor the movement status of a number of moving objects, frequent updates of locations generate a high processing load. If we use a low update frequency to obtain a satisfactory efficiency, it is not easy to accurately know the position of each object. As another example, consider the problem of *location anonymity*. There are several approaches to making a user's location anonymous for privacy reasons [15]. Even if the system knows the exact location of a user, it may only indicate that the user is within a certain region so as to conceal the exact location of the user.

Furthermore, there is a related problem in robotics research; *localization* is an important issue in mobile robotics [22]. By using sensor data and movement history information obtained during the movement of a robot, we can estimate the location of the robot. However, it is not easy to estimate the position accurately. When we perform localization using a probabilistic approach, the location of a moving object is typically represented using a Gaussian distribution [22].

In light of the above circumstances, we consider here the situation where the locations of objects are imprecisely known. In particular, we consider the case where the location of a *query object* is imprecisely known and is represented by a *Gaussian distribution*, while the target objects to be searched for have exact locations. We extend the traditional notion of spatial range queries, and then define *probabilistic range queries*. We introduce our idea using an example of a moving robot.

Example 1: Figure 1 illustrates the situation in which a moving robot performs localization on the basis of on its movement history. Each of the shown ellipses is an equi-probability contour of the Gaussian distribution obtained by the estimation performed at the movement point. For example, suppose that the location of the robot at time $t = \tau$ can be represented by a Gaussian distribution with the center $\boldsymbol{q} = (q_x, q_y)$. The highest probability is that the robot is located at (q_x, q_y) at time $t = \tau$, but it is also possible that the robot is located elsewhere. Consider the situation in which an object wants to find other nearby objects within a ten-meters range of itself. This is a location-based range query if the location of the object is exactly known. However, we

1084-4627/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDE.2009.93

need to extend the notion of the range query to also allow consideration of imprecisely known locations.

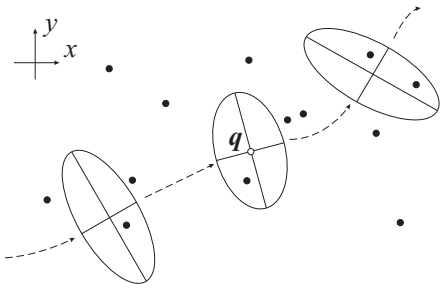

Fig. 1. Localization of a moving robot

In this paper, we propose new query processing approaches to *imprecise location-based spatial queries* for the situation in which the location of the query object is represented by a d-dimensional *Gaussian distribution*. We assume that the target objects are d-dimensional *points* with precisely known locations. The query considered here is an extended version of a conventional location-based range query based on the *Euclidean distance*. In past decades, research into distance-based range queries such as "retrieve all the objects within distance δ from q" was very active. Nowadays, we find that it has been conceptually extended to imprecise range queries [5], [6], [8], [9], [13], [20], [21]. However, most of the existing approaches assume simple uniform probability distributions, or consider arbitrary probability density functions. The latter is good in terms of generality, but the specific features of the target density function are not utilized. To the authors' knowledge, there has been no research performed on query processing methods when the location of a query object is represented by a Gaussian distribution. Since Gaussian distributions are widely used in statistics, pattern recognition [11], and localization in robotics [22], it is important to have an effective query processing method to use. Moreover, in the context of spatio-temporal databases, imprecision of location information of a moving object is also an important issue [27].

The application of our strategies is not limited to 2D or 3D spatial queries. As another example, consider example-based multimedia retrieval. When given some example images, an image retrieval system can make an approximate guess of the user's interests, assuming that the interests can be represented by a Gaussian distribution. After the estimation of the user-specific Gaussian distribution, we can retrieve target images.

An important problem in this context is that, as described below, we need to estimate the qualification probability of a target object using numerical integration. Since Gaussian distributions cannot be integrated analytically, we need to perform numerical integration using a method such as the Monte Carlo method, but this significantly increases the cost. Thus, we propose an approach to reducing the number of candidate objects which require numerical integration as much as possible using sophisticated filtering processes.

In the following discussions, we consider multidimensional cases ($d \geq 2$) because the one-dimensional case is trivial and can be implemented using a simple algorithm.

The organization of the paper hereafter is as follows. Section II describes the related work. Section III introduces the notion of probabilistic range queries. Section IV describes three query processing strategies that use spatial indexes such as R-trees [14]. Sections V and VI show the experimental results and then the conclusion of the paper is given in Section VII.

II. Related Work

For objects with uncertain locations, consideration of query processing techniques is a currently very active area in database research. In particular, Cheng and coworkers have performed intensive studies on this issue. In one study, [7] classified the concept of uncertainty of data and introduced the notions of queries on imprecise data such as sensor data. Among their definitions, a type of query called *probabilistic threshold queries* is related to our notion of probabilistic range queries described below. In their approach, the location of target objects are imprecisely known, but our research focuses on the situation in which the location of a query object is imprecisely known. In another study, [6] considered processing of probabilistic queries on one-dimensional uncertain data. They classified queries into several types and then presented algorithms that can be used to solve them. [9] also proposed query processing techniques to be used with moving objects with imprecisely known locations. The targets of these techniques are range queries and nearest neighbor queries and several query processing strategies corresponding to different movement patterns were presented in their paper. In addition, [8] discussed methods that can be used with one-dimensional probabilistic range queries from a theoretical perspective. They assumed several probabilistic density functions including Gaussian functions and considered averages and distribution of data. Moreover, [5] proposed a range query processing method to be used in a case where the locations of both a query object and target objects are imprecisely known. They assumed that each object exists within a rectangular region.

Tao et al. have shown a probabilistic range query method for the case in which all objects in a database have imprecisely known locations; a probability density function is associated with each object to represent the existence range of the object in the target space in which the location of the object is represented [20], [21]. A query region is specified by a rectangle. When the probability that an object exists within the specified rectangle is greater than a given threshold, the object is added to the query result. They proposed an index structure *U-tree* for evaluating such queries efficiently. Although their probabilistic range queries are similar to those used in our approach, they considered the opposite situation to ours: the locations of the target objects are uncertain. In addition, they considered arbitrary density functions and did not describe efficient query processing that uses the features of a specific probability distribution.

Böhm et al. proposed an efficient index structure called *Gauss-Tree* for indexing features that obey Gaussian distributions [4]. They assumed that the objects to be indexed are Gaussian distributions with different centers and covariance matrices.

The differences between such previous work and our proposal can be summarized as follows:

- The location of a query object is imprecisely known and it is represented by a Gaussian distribution.
- The target objects have exact locations.

We present an efficient query processing method with consideration of the properties of the Gaussian distribution. Our focus is generalization of the conventional *distance-based* range search, while most of the previous approaches to range queries for imprecise location data [5], [6], [8], [9], [13], [20], [21] have considered rectangle-based range queries.

In the context of moving object databases, approaches to representing the locations of imprecise objects with a Gaussian distribution can be found in [17]. Uncertain location information is also considered in [24], [26], [27].

III. Probabilistic Range Queries

A. Definitions

We first define the location of a query object in a probabilistic manner.

Definition 1: **(Location of Query Object)** *Assume that $\boldsymbol{x}$, the location of a query object q, is represented by a d-dimensional Gaussian distribution [11]*

$$p_q(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{q})^t \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{q})\right], \quad (1)$$

where $\boldsymbol{q}$ is the average of the distribution, $\boldsymbol{\Sigma}$ is a $d \times d$ covariance matrix, and $|\boldsymbol{\Sigma}|$ represents its determinant.

We extend the traditional notion of spatial range queries and define *probabilistic range queries*.

Definition 2: **(Probabilistic Range Query)** *Given the probability density function $p_q(\boldsymbol{x})$, the distance threshold δ ($\delta > 0$), and the probability threshold θ ($0 < \theta < 1$), a* probabilistic range query $PRQ(q, \delta, \theta)$ *returns all the objects such that the probabilities, that their distances from the query object q are less than or equal to δ, are greater than or equal to θ. It is formally defined as follows:*

$$PRQ(q, \delta, \theta) = \{o \mid o \in \mathcal{O}, \Pr(\|\boldsymbol{x} - \boldsymbol{o}\|^2 \leq \delta^2) \geq \theta\}, \quad (2)$$

where $\mathcal{O}$ is the set of the target objects, $\|\cdot\|$ is the length of a vector, and $\|\boldsymbol{x} - \boldsymbol{o}\|^2$ represents the squared Euclidean distance between $\boldsymbol{x}$, the location of the query object q, and $\boldsymbol{o}$, the location of the object o.

The probability threshold should satisfy $0 < \theta < 1$. Since a Gaussian distribution has infinite spread, all the target objects satisfy the query when $\theta = 0$. On the other hand, no object can satisfy the condition when $\theta = 1$.

Note that the location of the query object is probabilistically defined, but the target objects are static points. Therefore, we can use conventional spatial indexes [14] such as R-trees for efficient query processing.

B. Outline of Query Processing

Next, we show an outline of a query process for a probabilistic range query. It is based on a simple idea: we exchange the roles of the query object and the target object. Instead of considering the probability that the object o is within the distance range δ from the query object q, we evaluate *the probability that q is within the distance range δ from o* for each target object o. If the probability is greater than or equal to θ, o is added to the result set. In other words, we investigate whether

$$\int_{\boldsymbol{x} \in R} p_q(\boldsymbol{x}) d\boldsymbol{x} \geq \theta \quad (3)$$

holds, where R is a sphere with center $\boldsymbol{o}$ and radius δ. However, it is very costly to derive the probability since integration of the Gaussian density function (Eq. (1)) requires numerical integration using a method such as the Monte Carlo method.

The generic query processing strategy consists of the following three phases:

1) **Index-Based Search**: Find all the candidate objects that may satisfy the given query using a spatial index. We use the R-tree index family [14] since it is the most widely used one. For the retrieval, we need to determine the rectilinear query region.
2) **Filtering**: Some of the candidate objects are pruned in accordance with the analytical result of the query. In addition, under some conditions, we can safely decide that some objects will satisfy the query without performing probability computation. Such objects are added to the result set without performing numerical integration.
3) **Probability Computation**: For each remaining candidate object, the probability is computed using Eq. (3). If the result is greater than or equal to θ, the object is added to the result set.

In traditional spatial queries, the cost of retrieval process is the main concern. However, in our case, the cost of the probability computation phase is dominant. As will be shown later in the experimental results, phase three occupies most of the processing time. This means that we should reduce the number of candidate objects in the filtering phase so as to minimize the probability computation cost. In the following, we propose three query processing strategies for implementing the above generic query processing strategy.

IV. Query Processing Strategies

A. Rectilinear-Region-Based Approach (RR)

For pruning candidate objects, the notion of an *uncertainty region*, the region in which an object with an uncertain location exists, is often used. Since the object cannot exist outside its uncertainty region, we can reduce the scope of the search. Moreover, Tao et al. [20], [21] use a *probabilistically constrained rectangle* (*PCR*) by extending this notion; it is a rectangle for which the probability that the object is in it is greater than the specified threshold value. We extend this approach for our problem.

1) Definition and Derivation of θ-Region: First we define the notion of a θ-*region*, which is a d-dimensional ellipsoidal region. It should satisfy the condition that the probability that the query object q exists in the region is $1-2\theta$ ($0 < \theta < 1/2$).[1]

Definition 3: **($\boldsymbol{\theta}$-Region)**
Consider integration of the probability density function $p_q(\boldsymbol{x})$ over an ellipsoidal region $(\boldsymbol{x}-\boldsymbol{q})^t\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{q}) \leq r^2$. Given θ ($0 < \theta < 1/2$), let the value of r for which the result of the integration becomes $1-2\theta$ be r_θ:

$$\int_{(\boldsymbol{x}-\boldsymbol{q})^t\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{q})\leq r_\theta^2} p_q(\boldsymbol{x})d\boldsymbol{x} = 1-2\theta. \tag{4}$$

We call the ellipsoidal region

$$(\boldsymbol{x}-\boldsymbol{q})^t\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{q}) \leq r_\theta^2 \tag{5}$$

*defined by r_θ the θ-*region*.*

In other work (e.g., [20], [21]), a similar idea has been used for objects with uncertain locations stored in a database; for such static objects, we can compute the θ-region for each object in advance. In contrast, our problem requires derivation of the θ-region *dynamically* for a given query since it considers an imprecise query object location. The straightforward approach to computing the θ-region for a given query is to perform a binary search to find an appropriate r_θ-value that satisfies Eq. (4); however, this is costly to be performed at run-time.

Next, we show a method for determining the θ-region (and r_θ) for a given query. The approach we take is to transform the problem to an integration for a d-dimensional spherical region. To prepare for a description of this approach, let us introduce the notion of a normalized Gaussian distribution.

Definition 4: **(Normalized Gaussian distribution)**
The normalized Gaussian distribution *is defined as*

$$p_{\mathrm{norm}}(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}} \exp\left[-\frac{1}{2}\|\boldsymbol{x}\|^2\right] \tag{6}$$

by assigning values $\boldsymbol{q} = \mathbf{0}$ and $\boldsymbol{\Sigma} = \mathbf{I}$ in Eq. (1), where $\mathbf{I}$ is the d-dimensional unit matrix.

On the basis of this probability density function, $\tilde{r}_\theta$ is defined as follows.

Definition 5: Consider integration of $p_{\mathrm{norm}}(\boldsymbol{x})$ over the region $\|\boldsymbol{x}\|^2 \leq r^2$, which is a sphere with the origin as its center and the radius r. For the given θ ($0 < \theta < 1/2$), let $\tilde{r}_\theta$ be the the radius with which the integration result becomes $1-2\theta$:

$$\int_{\|\boldsymbol{x}\|^2\leq \tilde{r}_\theta^2} p_{\mathrm{norm}}(\boldsymbol{x})d\boldsymbol{x} = 1-2\theta. \tag{7}$$

The following property is satisfied.

Property 1: For a given θ, $r_\theta = \tilde{r}_\theta$ holds.

Proof. Although this is based on a well-known fact, we will show a proof for the following discussion. Let the spectral decomposition of the covariance matrix $\boldsymbol{\Sigma}^{-1}$ be

$$\boldsymbol{\Sigma}^{-1} = \sum_{i=1}^{d} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^t, \tag{8}$$

[1]The reason why we use $1-2\theta$ instead of $1-\theta$ is explained later.

where λ_i and $\boldsymbol{v}_i$ are the i-th eigenvalue and its corresponding eigenvector. Let us define

$$\lambda^\top = \min\{\lambda_i\} \tag{9}$$
$$\lambda^\perp = \max\{\lambda_i\} \tag{10}$$

for the following discussion. Note that all the eigenvalues of a covariance matrix are greater than zero. Then we define $d \times d$ matrices $\boldsymbol{\Lambda}$ and $\mathbf{E}$ as follows:

$$\boldsymbol{\Lambda} = \mathrm{diag}(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots, \sqrt{\lambda_d}) \tag{11}$$
$$\mathbf{E} = [\boldsymbol{v}_1\, \boldsymbol{v}_2\, \cdots\, \boldsymbol{v}_d], \tag{12}$$

where $\mathrm{diag}(\ldots)$ denotes a diagonal matrix. Using the above, we define a vector $\boldsymbol{x}$ as

$$\boldsymbol{x} = \boldsymbol{\Lambda}\mathbf{E}^t\boldsymbol{y}. \tag{13}$$

Then we can rewrite Eq. (7) as follows:

$$\int_{\boldsymbol{y}^t\boldsymbol{\Sigma}^{-1}\boldsymbol{y}\leq \tilde{r}_\theta^2} \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}\boldsymbol{y}^t\boldsymbol{\Sigma}^{-1}\boldsymbol{y}\right] d\boldsymbol{y} = 1-2\theta. \tag{14}$$

If we replace $\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{q}$, we get

$$\int_{(\boldsymbol{x}-\boldsymbol{q})^t\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{q})\leq \tilde{r}_\theta^2} p_q(\boldsymbol{x})d\boldsymbol{x} = 1-2\theta. \tag{15}$$

By comparing this formula with Eq. (4), we can notice that $r_\theta = \tilde{r}_\theta$. ■

2) Deriving Search Region: Unfortunately, we cannot directly use a θ-region to retrieve target objects. We assume the use of an R-tree, but the ellipsoidal shape of a θ-region is not suited to its use. Thus, we derive the bounding box for the given θ-region. As shown in Fig. 2, let the width of the box from the query center $\boldsymbol{q}$ along the i-th dimension axis be w_i.

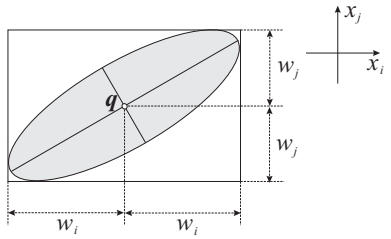

Fig. 2. Using bounding box

The width w_i satisfies the following property.

Property 2: The value of w_i ($i = 1, 2, \ldots, d$) is given as

$$w_i = \sigma_i r_\theta, \tag{16}$$

where σ_i corresponds to the standard deviation for the i-th dimension.

$$\sigma_i = \sqrt{(\boldsymbol{\Sigma})_{ii}}, \tag{17}$$

where $(\boldsymbol{\Sigma})_{ii}$ represents the (i,i) entry of $\boldsymbol{\Sigma}$.

Proof. We extend the idea of Ankerst et al. [1] for this problem. Given an ellipsoidal distance $d^2_{\mathrm{ellip}}(\boldsymbol{p},\boldsymbol{q}) = (\boldsymbol{p}-\boldsymbol{q})^t\mathbf{A}(\boldsymbol{p}-\boldsymbol{q})$, they showed that a box-shape distance function

that tightly bounds $d^2_{\mathrm{ellip}}(\boldsymbol{p},\boldsymbol{q})$ is given by $d^2_{\mathrm{MBB}}(\boldsymbol{p},\boldsymbol{q}) = \max_{i=1}^{d}\left[\frac{(p_i-q_i)^2}{(\mathbf{A}^{-1})_{ii}}\right]$, where $(\mathbf{A}^{-1})_{ii}$ denotes the (i,i) entry of $\mathbf{A}^{-1}$. [1] also proved that $d^2_{\mathrm{MBB}}(\boldsymbol{p},\boldsymbol{q}) \leq \varepsilon \Leftrightarrow \forall i : q_i - \sqrt{\varepsilon(\mathbf{A}^{-1})_{ii}} \leq p_i \leq q_i + \sqrt{\varepsilon(\mathbf{A}^{-1})_{ii}}$ holds. If we make substitutions $\mathbf{A} = \boldsymbol{\Sigma}^{-1}$ and $\varepsilon = r^2_\theta$, we get

$$\forall i : q_i - \sigma_i r_\theta \leq x_i \leq q_i + \sigma_i r_\theta. \tag{18}$$

This means that $w_i = \sigma_i r_\theta$. ■

Consider Fig. 3, where a and b are target objects. The shaded ellipse represents the θ-region and its bounding rectangle is the bounding box derived above. Note that object a does not satisfy the condition of $PRQ(q,\delta,\theta)$. The reason is as follows. First, the probability that the query object is located outside of the bounding box is $1-(1-2\theta) = 2\theta$ since the probability that the query object is located inside of the θ-region is $1-2\theta$. Second, since the Gaussian distribution has point symmetry, if we draw a point symmetry object a' for a in terms of $\boldsymbol{q}$, the integration result for the circular region centered at a' with radius δ has the same probability as that for a. This means that the integration result (the probability) for a (and a') is less than θ.

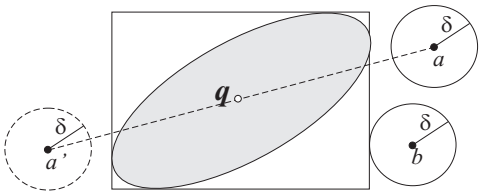

Fig. 3. MBB and target objects

On the other hand, object b, for which the enclosing circle touches the MBB, has a *possibility* that the integration result is greater than or equal to θ. Of course, it is clear that, if we look at the figure, the probability for object b in Fig. 3 is smaller than θ, but it is not easy to judge that with a simple condition. On the basis of the above consideration, the objects inside the rounded solid-line box shown in Fig. 4 will become candidates that may satisfy the query condition. The region corresponds to the notion of *Minkowski Sum*, often used in similarity-based retrieval [2].

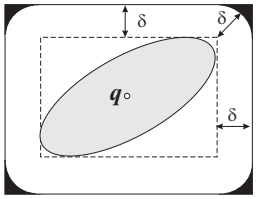

Fig. 4. Minkowski-sum

3) Query Processing: In Subsection III-B, the generic query processing strategy was shown. The correspondence between the approach shown here and the generic one is as follows. In Phase 1 (Index-Based Search), the R-tree is searched using the bounding box that tightly bounds the Minkowski-sum region in Fig. 4. In Phase 2 (Filtering), the objects in the fringe part (the black regions in Fig. 4) are deleted from the candidate set. Unfortunately, computation of fringe part is not easy for $d \geq 3$. Thus, we apply this filtering method only for $d = 2$. Finally, in Phase 3 (Probability Computation), the probabilities are computed for all the remaining objects inside the Minkowski-region using numerical integration. If the probability is greater than or equal to θ, the object satisfies the query.

We have one issue to consider in regard to query processing: how to obtain r_θ from the given θ. Referring back to Property 1, to obtain θ-region (Eq. (5)) for the given θ, we can use the normalized formula Eq. (7). Since the function $p_{\mathrm{norm}}(\boldsymbol{x})$ of Eq. (6) cannot be integrated analytically, it is not possible to compute r_θ directly from θ. To solve this problem, we construct a table that contains θ and its corresponding r_θ for each representative value of r. When a query is given, we can search this table to find r_θ for the given θ to derive the θ-region. A similar table-based approach was used in other work. For example, [20], [21] called such a table a *U-catalog*. We should note that the corresponding entry for the given θ-value (e.g., $\theta = 0.06$) may not exist in the table. For this case, we find the entry r^*_θ which is the maximal entry in the table that satisfies $\theta^* < \theta$ and use the corresponding value r^*_θ. Although this approach may increase the number of target objects for numerical integration, the correctness of the result is retained.

On the basis of the above consideration, the query processing algorithm can be given as Algorithm 1.

Algorithm 1 Rectilinear-Region-Based Approach

```
1: procedure PRQ-RR(q, Σ, δ, θ)
2:     /* Preparation */
3:     Calculate σ_i (i = 1, ..., d) and λ^⊤ from Σ
4:     From the U-catalog, obtain r*_θ that corresponds to θ*,
       which is the maximal value satisfying θ* ≤ θ
5:     /* Phase 1: Index-Based Search */
6:     Using {σ_i}_{i=1}^d, r*_θ, δ, derive the search region shown
       in Fig. 4
7:     Perform R-tree-based search to get candidate objects
8:     /* Phase 2: Filtering (only for d = 2) */
9:     Delete o ∈ C if it is in the fringe regions of Fig. 4
10:    /* Phase 3: Probability Computation */
11:    For each o ∈ C, compute ∫_{x∈R} p_q(x)dx. If the result
       is greater than or equal to θ, o is output.
12: end procedure
```

B. Oblique-region-based approach (OR)

The second approach to query processing is an *oblique-region*-based one, as shown in Fig. 5. The oblique box is parallel to the axes of the θ-region ellipsoid, and the distance between the bounding box and the ellipsoid is greater than or equal to δ. On the basis of the same consideration as in the rectilinear-region-based approach, the objects inside of the

rectangle become the candidates of the query. In Fig. 5, a is not a candidate but b, c, and d are candidates.

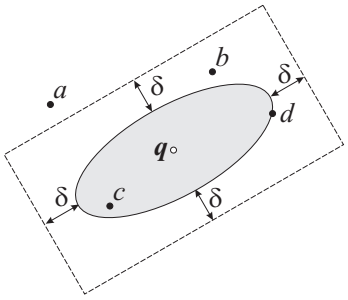

Fig. 5. Oblique region

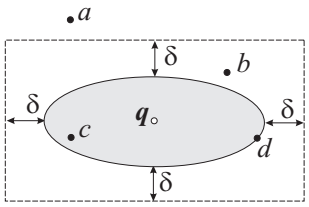

Fig. 6. Axis transformation

However, it is not easy to directly apply filtering using the oblique region. To make the problem easier to solve, we perform an axis transformation. As shown in Fig. 6, we can transform the oblique rectangle into a parallel-axis rectangle. The filtering process can be easily implemented with this transformation. The idea is formalized as follows:

Property 3: Now we refer back to the matrix $\mathbf{E} = [\boldsymbol{v}_1 \boldsymbol{v}_2 \cdots \boldsymbol{v}_d]$ *in Eq. (12) that consists of the eigenvectors of* $\boldsymbol{\Sigma}^{-1}$. *For the given* d*-dimensional point* $\boldsymbol{x}$*, we consider the vector* $\boldsymbol{y}$ *that satisfies*

$$\boldsymbol{x} = \mathbf{E}\boldsymbol{y}. \tag{19}$$

The point $\boldsymbol{y}$ *corresponds to the transformed point.*

Proof. Suppose that $\boldsymbol{x}$ is located on the ellipsoid $(\boldsymbol{x} - \boldsymbol{q})^t \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{q}) = r^2$. We subtract $\boldsymbol{q}$ from $\boldsymbol{x}$ beforehand ($\boldsymbol{x} \leftarrow \boldsymbol{x} - \boldsymbol{q}$) and let $\boldsymbol{q} = \mathbf{0}$. We have the following result.

$$\begin{aligned} \boldsymbol{x}^t \boldsymbol{\Sigma}^{-1} \boldsymbol{x} &= \boldsymbol{x}^t \left(\sum_{i=1}^{d} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^t \right) \boldsymbol{x} = \sum_{i=1}^{d} \lambda_i \boldsymbol{x}^t \boldsymbol{v}_i \boldsymbol{v}_i^t \boldsymbol{x} \\ &= \sum_{i=1}^{d} \lambda_i (\mathbf{E}\boldsymbol{y})^t \boldsymbol{v}_i \boldsymbol{v}_i^t (\mathbf{E}\boldsymbol{y}) = \sum_{i=1}^{d} \lambda_i y_i^2 = r^2 \end{aligned}$$

This means that $\boldsymbol{x}$ is translated to the point $\boldsymbol{y}$ on the ellipsoid $\sum_{i=1}^{d} \lambda_i y_i^2 = r^2$. ■

The filtering region for this strategy is illustrated as Fig. 7. For the i-th dimension, the box is described by the range

$$-\frac{r}{\sqrt{\lambda_i}} - \delta \le y_i \le \frac{r}{\sqrt{\lambda_i}} + \delta. \tag{20}$$

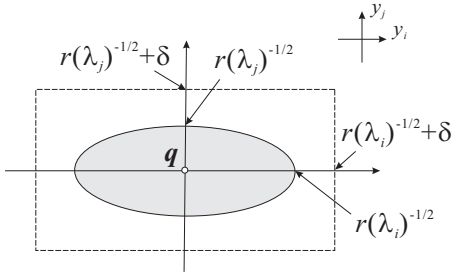

Fig. 7. Filtering region for OR

How can we use the above property? We can compute a bounding box for the oblique rectangle (shown in Fig. 5) for index-based retrieval, but the size of the bounding box is generally large. Therefore, it is possible to use the property as an additional filtering step (Phase 2). For example, we can enhance Algorithm 1 by adding the oblique-region-based filtering to Phase 2. The filtering process used is quite simple: given a candidate object o with the corresponding vector $\boldsymbol{x}$, we derive the transformed vector $\boldsymbol{y}$ using Eq. (19). If $\boldsymbol{y}$ is not inside of the box expressed by Eq. (20), we can delete it from the candidate set.

C. Bounding-Function-Based Approach (BF)

The third strategy takes a different approach: it uses upper- and lower-bounding functions for the probability density function.

1) Basic Idea: First, consider a special case $\boldsymbol{q} = \mathbf{0}$ and $\boldsymbol{\Sigma} = \mathbf{I}$. Namely, we are considering the probability density function $p_{\mathrm{norm}}(\boldsymbol{x})$ shown in Eq. (6). The idea is illustrated in Fig. 8. An equi-distance surface of $p_{\mathrm{norm}}(\boldsymbol{x})$ and two target objects a and b are shown. Suppose that there is a sphere R with the radius δ which has a circumference the distance α from the origin. The shaded region in Fig. 8 corresponds to R. Then, suppose that α satisfies the condition

$$\int_{\boldsymbol{x} \in R} p_{\mathrm{norm}}(\boldsymbol{x}) d\boldsymbol{x} = \theta. \tag{21}$$

This means that the integration result of $p_{\mathrm{norm}}(\boldsymbol{x})$ for region R is equal to θ.

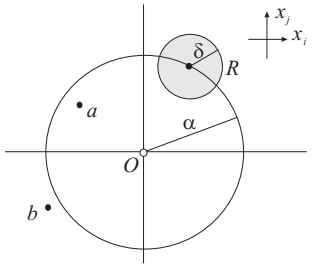

Fig. 8. Basic idea of BF

If we are given an appropriate α for the given query, the query can be directly processed: we retrieve all the objects

within the sphere with radius α. In Fig. 8, only object a satisfies the condition and we can determine that the probability is greater than θ without numerical integration.

Since it is not possible to obtain α analytically from the given δ and θ, we need to apply the U-catalog approach. In the preparation step, we perform numerical integration for different combinations of δ and θ, and then create a table containing entries with the form (δ, θ, α).

2) General Case:

a) Upper- and Lower-bounding Functions: For the general case, we cannot use the simple strategy shown above since the probability distribution is not isotropic. Thus, we define the upper- and lower-bounding functions.

Definition 6: **(Bounding Functions)**
We define the two matrices $\mathbf{M}^\top$ and $\mathbf{M}^\perp$ as follows:

$$\mathbf{M}^\top = \lambda^\top \sum_{i=1}^{d} \boldsymbol{v}_i \boldsymbol{v}_i^t = \lambda^\top \mathbf{I} \quad (22)$$

$$\mathbf{M}^\perp = \lambda^\perp \sum_{i=1}^{d} \boldsymbol{v}_i \boldsymbol{v}_i^t = \lambda^\perp \mathbf{I}. \quad (23)$$

Then, we define the following functions obtained by substituting $\mathbf{\Sigma}^{-1}$in Eq. (1) with $\mathbf{M}^\top$ and $\mathbf{M}^\perp$:

$$p_q^\top(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}|\mathbf{\Sigma}|^{1/2}} \exp\left[-\frac{\lambda^\top}{2}\|\boldsymbol{x}-\boldsymbol{q}\|^2\right] \quad (24)$$

$$p_q^\perp(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}|\mathbf{\Sigma}|^{1/2}} \exp\left[-\frac{\lambda^\perp}{2}\|\boldsymbol{x}-\boldsymbol{q}\|^2\right]. \quad (25)$$

The equi-probable surfaces of $p_q^\top(\boldsymbol{x})$ and $p_q^\perp(\boldsymbol{x})$ have spherical shapes.

Note that $p_q^\top(\boldsymbol{x})$ and $p_q^\perp(\boldsymbol{x})$ are *not* probability density functions since their integration results for the whole space are not equal to one.

The functions $p_q^\top(\boldsymbol{x})$ and $p_q^\perp(\boldsymbol{x})$ have the following lower- and upper-bounding properties:

Property 4: For any point $\boldsymbol{x}$, $p_q^\perp(\boldsymbol{x}) \le p_q(\boldsymbol{x}) \le p_q^\top(\boldsymbol{x})$ holds. This means that they are the lower- and upper-bounding functions of $p_q(\boldsymbol{x})$.

In fact, $p_q^\top(\boldsymbol{x})$ and $p_q^\perp(\boldsymbol{x})$ are the optimal functions with spherical shapes that have the bounding properties.

Figure 9 illustrates $p_q^\top(\boldsymbol{x})$ and $p_q^\perp(\boldsymbol{x})$. The outside and inside spheres correspond to $p_q^\top(\boldsymbol{x})$ and $p_q^\perp(\boldsymbol{x})$, respectively. This figure shows the isosurface of equal probability for each function.

b) Query Processing: We next consider the query processing method using the above properties. As shown in Fig. 10, let $R^\top$ be a spherical region with radius δ and its center relative to $\boldsymbol{q}$ is $\alpha^\top$, but assume that $R^\top$ satisfies the following constraint for the given θ:

$$\int_{\boldsymbol{x}\in R^\top} p_q^\top(\boldsymbol{x}) d\boldsymbol{x} = \theta \quad (26)$$

The value of $\alpha^\top$ can be determined by the following property.

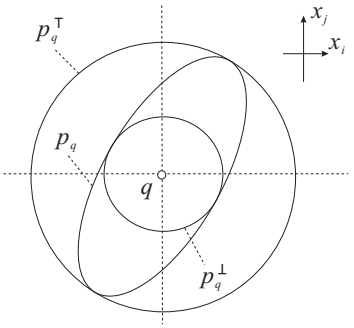

Fig. 9. $p_q^\top(\boldsymbol{x})$ and $p_q^\perp(\boldsymbol{x})$

Property 5: Let $S^\top$ be a spherical region with radius $\sqrt{\lambda^\top}\delta$ and its center relative to the origin is $\beta^\top$, and assume that $S^\top$ satisfies the following equation:

$$\int_{\boldsymbol{x}\in S^\top} p_{\mathrm{norm}}(\boldsymbol{x}) d\boldsymbol{x} = (\lambda^\top)^{d/2}|\mathbf{\Sigma}|^{1/2}\theta. \quad (27)$$

We can determine $\beta^\top$ on the basis of this constraint. If we get $\beta^\top$, we can derive $\alpha^\top$ as

$$\alpha^\top = \frac{\beta^\top}{\sqrt{\lambda^\top}}. \quad (28)$$

This property can be proved by transforming Eq. (26). The proof is shown in the appendix.

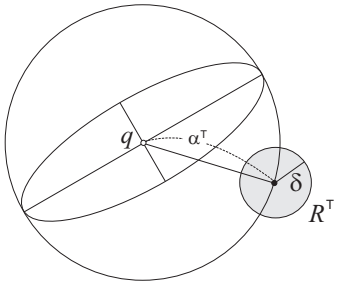

Fig. 10. $R^\top$ and $\alpha^\top$

Remember the basic case $p_q(\boldsymbol{x}) = p_{\mathrm{norm}}(\boldsymbol{x})$ given in the previous subsection. Given a probability threshold θ and a search distance δ, if we have a table that returns α which satisfies the constraint of Eq. (21), we can perform the query using a range query with the radius α. Similarly, the properties shown here indicate that we can obtain the $\alpha^\top$ value using a table. When we process a query, we search the table to find an appropriate α for the given δ and θ. We denote this function as $\alpha = ucatalog\_lookup(\delta, \theta)$.

To derive $\alpha^\top$ in Eq. (28) using Property 5, first we derive $\beta^\top$ using

$$\beta^\top = ucatalog\_lookup(\sqrt{\lambda^\top}\delta, (\lambda^\top)^{d/2}|\mathbf{\Sigma}|^{1/2}\theta), \quad (29)$$

then obtain $\alpha^\top$ based on Eq. (28). We can obtain $\alpha^\perp$ in a similar manner. First, we derive $\beta^\perp$ using

$$\beta^\perp = ucatalog\_lookup(\sqrt{\lambda^\perp}\delta, (\lambda^\perp)^{d/2}|\mathbf{\Sigma}|^{1/2}\theta), \quad (30)$$

then obtain $\alpha^{\perp}$ using

$$\alpha^{\perp} = \frac{\beta^{\perp}}{\sqrt{\lambda^{\perp}}}. \tag{31}$$

Next, we describe the meaning of the above properties. Figure 11 shows a conceptual figure in which the horizontal axis represents the x_i axis and the vertical axis represents the value of the functions. The query center $\boldsymbol{q}$ is located on the origin. We can notice that the curve of the probabilistic density function $p_q(\boldsymbol{x})$ is located between functions $p_q^{\top}(\boldsymbol{x})$ and $p_q^{\perp}(\boldsymbol{x})$. The shaded left region represents the integration of $p_q^{\top}$; the integration range is a sphere with radius δ and its center is separated from $\boldsymbol{q}$ by $\alpha^{\perp}$. As described above, the integration result is θ. Similarly, the right shaded region represents the integration of $p_q^{\top}(\boldsymbol{x})$ for a sphere with radius δ and the distance between the center and $\boldsymbol{q}$ is $\alpha^{\top}$. Its volume is also θ.

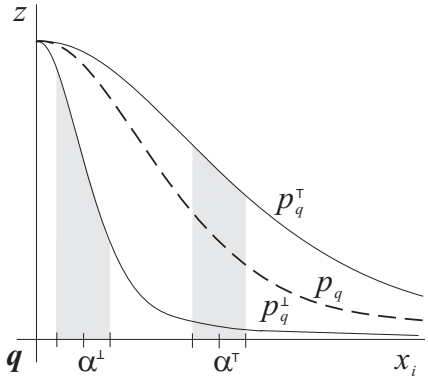

Fig. 11. Roles of $\alpha^{\top}$ and $\alpha^{\perp}$

From this figure, we note that if the distance from point $\boldsymbol{q}$ is greater than $\alpha^{\top}$, even using the upper estimate function $p_q^{\top}(\boldsymbol{x})$, the result of integration is less than θ. Thus, an object whose distance from $\boldsymbol{q}$ is greater than $\alpha^{\top}$ is not a candidate. On the other hand, if the distance from q is less than $\alpha^{\perp}$, even using the lower estimate function $p_q^{\perp}(\boldsymbol{x})$, we obtain a result greater than θ. If the distance from the origin is between $\alpha^{\top}$ and $\alpha^{\perp}$, we need to compute the probability of the object using numerical integration.

c) Query Processing Algorithm: In this case, the same problem as with the rectilinear-region-based approach also occurs: we may not be able to find the entry corresponding to the values $\alpha^{\top}$ and $\alpha^{\perp}$. The solution is as follows. When we derive $\alpha^{\top}$, we look up the U-catalog based on Eq. (29), and then find the entry for the pair $(\sqrt{\lambda^{\top}}\delta, (\lambda^{\top})^{d/2}|\boldsymbol{\Sigma}|^{1/2}\theta)$. If we cannot find the entry, we use the next best entry $\beta_*^{\top}$ defined as follows:

$$\begin{aligned}\beta_*^{\top} \quad = \quad & \min\{\alpha \mid (\delta,\theta,\alpha) \in U \wedge \delta \geq \sqrt{\lambda^{\top}}\delta \wedge \\ & \theta \leq (\lambda^{\top})^{d/2}|\boldsymbol{\Sigma}|^{1/2}\theta\},\end{aligned} \tag{32}$$

where U represents the U-catalog. Using $\beta_*^{\top}$, we can derive $\alpha^{\top}$ from Eq. (28). We call the resulting value $\alpha_*^{\top}$. Since $\alpha_*^{\top} > \alpha^{\top}$ holds, the search based on $\alpha_*^{\top}$ may retrieve additional objects compared to the case of $\alpha^{\top}$, and increase the probability computation cost. Similarly, for $\alpha^{\perp}$, we derive

$$\begin{aligned}\beta_*^{\perp} \quad = \quad & \max\{\alpha \mid (\delta,\theta,\alpha) \in U \wedge \delta \leq \sqrt{\lambda^{\perp}}\delta \wedge \\ & \theta \geq (\lambda^{\perp})^{d/2}|\boldsymbol{\Sigma}|^{1/2}\theta\},\end{aligned} \tag{33}$$

and then use Eq. (31) to obtain $\alpha_*^{\perp}$ instead of $\alpha^{\perp}$. The usage of $\alpha_*^{\perp}$ means that we need to compute the probabilities for some additional objects. On the basis of the above discussion, we can derive the query processing algorithm Algorithm 2.

Algorithm 2 Bounding-Function-Based Approach

```
1: procedure PRQ-BF(q, Σ, δ, θ)
2:     /* Preparation */
3:     Derive λ^⊤, λ^⊥, |Σ| from Σ
4:     Obtain β*^⊤ and β*^⊥ by Eqs. (32) and (33), then compute
       α*^⊤ and α*^⊥ using Eqs. (28) and (31)
5:     /* Phase 1: Index-Based Search */
6:     Using an R-tree, retrieve objects within a range [q_i −
       α*^⊤, q_i + α*^⊤] for each dimension i. The resulting set of
       candidate objects is denoted by C.
7:     /* Phase 2: Filtering */
8:     C ← {o | o ∈ C, dist(o, q) ≤ α*^⊤}
9:     Q ← {o | o ∈ C, dist(o, q) ≤ α*^⊥}
10:    C ← C − Q
11:    /* Phase 3: Probability Computation */
12:    foreach o ∈ C do
13:        Let R be a sphere centered at o with radius δ
14:        if ∫_{x∈R} p_q(x)dx ≥ θ then Q ← Q ∪ {o}
15:    end for
16: end procedure
```

The next example illustrates the above idea.

Example 2: Figure 12 shows an example of query processing. With the first filtering condition, a, b, c become the target objects, but we do not have to calculate the actual probability of a since it is within the distance $\alpha^{\perp}$. In contrast, b and c require numerical integration.

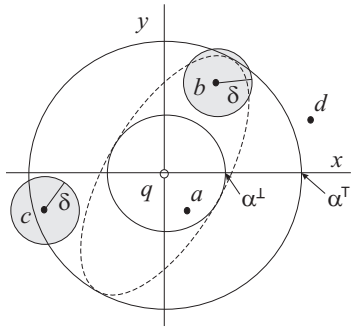

Fig. 12. Query processing for BF

V. Experiments I (2D Data)

A. Experimental Setup

For the experiments, we used road line segment data of Long Beach, California from the TIGER database [23]. We extracted the midpoint for each line segment then made a point set. The point set consisted of 50,747 points and was normalized in a $[0, 1000] \times [0, 1000]$ space. We evaluated the query strategies described in Section IV for $PRQ(q, \delta, \theta)$ using this dataset.

Section IV introduced three strategies 1) Rectilinear-Region-Based (RR), 2) Oblique-Region-Based (OR), and 3) Bounding-Function-Based (BF) ones. We also evaluated their combinations. Since OR is only useful as a filtering method, we considered the following six combinations: 1) RR, 2) BF, 3) RR+BF, 4) RR+OR, 5) BF+OR, and 6) ALL (RR+BF+OR). For RR, RR+BF, RR+OR, and ALL, Algorithm 1 was used as the underlying algorithm. For BF and BF+OR, Algorithm 2 was used. For example, in RR+OR, Algorithm 1 was used to find candidate objects on the basis of the region in Fig. 4, then a filtering was performed for the region shown in Fig. 5. In ALL, all three strategies were combined. In the experiments, we computed accurate $\beta^{\top}$ and $\beta^{\perp}$ values for BF using Eqs. (29) and (30), instead of approximate values.

To define a target query, we needed to specify some parameters. For the distance and probability threshold values, we used $\delta = 25$ and $\theta = 0.01$ as the default ones. The covariance matrix $\mathbf{\Sigma}$ was defined as follows:

$$\mathbf{\Sigma} = \gamma \begin{bmatrix} 7 & 2\sqrt{3} \\ 2\sqrt{3} & 3 \end{bmatrix}, \tag{34}$$

where γ specifies the uncertainty of the distribution. We used $\gamma = 10$ as the default. When γ is large, the spread (uncertainty) of the distribution becomes large. The setting of parameters means that the shape of the isosurface of $p_q(\boldsymbol{x})$ was an ellipse titled at 30° and the major-to-minor axis ratio is 3:1.

We compared the elapsed times (wallclock time) for query processing. We selected one target object randomly as the query center then issued a probabilistic range query. The averaged time of five query trials was used for the comparison. In this experiment, we used the *importance sampling* method [18], a kind of the Monte Carlo method. We generate random numbers that obey a Gaussian distribution and derive the ratio such that random numbers enter the specified region. The ratio corresponds to the probability to be estimated. For our problem, this method converges quickly compared to the standard Monte Carlo method, especially for medium-dimensional cases. For generating random variables, we used RANDLIB [19]. For each numerical integration, 100,000 random numbers were generated and it took about 0.05 seconds for numerical integration for one object. As the spatial index, we used an implementation of the R*-tree index [12]. The page size of an R*-tree node was set as 1KB.

The programs for the experiments were implemented using C language. The experiments were conducted using a PC with an Intel Pentium CPU (2.0 GHz), 1GB of memory, a 143GB hard disk, and Fedora Core 5 OS.

B. Experimental Results

1) Experiments Using Default Parameters ($\gamma = 10, \delta = 25, \theta = 0.01$): Table I shows the query processing time for each combination of strategies with the default parameter setting.[2] Table II shows the number of candidate objects that require numerical integration. For this query, the number of resulting objects (shown in the ANS column) is 546. As shown in Table II, RR and BF required integration of 792 and 683 objects, respectively, but with the combination RR+BF, this could be reduced to 636. Similarly, with other combinations, the number of candidates was reduced and the combination of the three strategies (ALL) was the best one.

TABLE I
Query processing time (seconds) ($\delta = 25, \theta = 0.01$)

γ	RR	BF	RR+BF	RR+OR	BF+OR	ALL
1	18.6	15.9	15.7	17.7	15.1	14.8
10	41.2	35.9	33.5	35.6	29.8	29.4
100	155.3	136.7	123.5	119.3	97.3	93.7

TABLE II
Number of candidates ($\delta = 25, \theta = 0.01$)

γ	RR	BF	RR+BF	RR+OR	BF+OR	ALL	ANS
1	357	302	297	335	285	281	295
10	792	683	636	682	569	558	546
100	2998	2599	2346	2270	1832	1788	1566

For each combination, at least 97% of the total processing time was taken up with numerical integration. This means that the number of candidates for numerical integration directly influence the total cost. Actually, if we compare Tables I and II, we can notice their correspondence.

Figure 13 shows the three regions, for which we needed to perform numerical integration for RR, OR, and BF. If we assume the target objects are uniformly distributed, their areas correspond to the query processing costs. For the combination of three strategies (ALL), we needed to consider only the shaded region of Fig. 14.

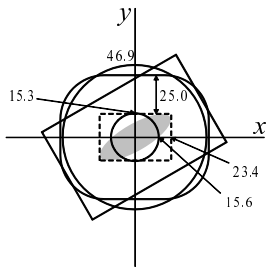

Fig. 13. Three integration regions

[2]The table includes the cases for $\gamma = 1$ and 100. They are explained later.

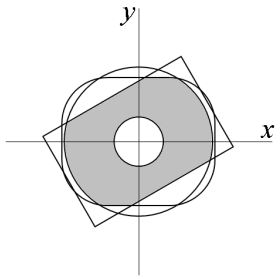

Fig. 14. Integration region for ALL

2) Using Different γ Values ($\gamma = 1$ and $\gamma = 100$): In this experiment, we compared the trends for different γ values. Compared to the default value $\gamma = 10$, $\gamma = 1$ means that the location of the query object is more accurate. In contrast, $\gamma = 100$ represents a more vague location. The results are also shown in Tables I and II. Although large γ value increases the query cost due to the large ambiguity, the overall trends are similar, but note that the combination of the strategies is more effective for $\gamma = 100$.

Figures 15 and 16 show the integration regions for $\gamma = 1$ and $\gamma = 100$, respectively. We can easily see that combining the strategies does not improve the query cost very much for $\gamma = 1$. In contrast, combining the strategies can achieve efficient processing for $\gamma = 100$.

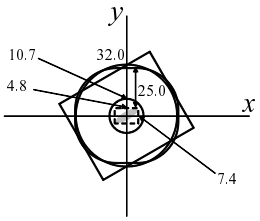

Fig. 15. Integration regions ($\gamma = 1$)

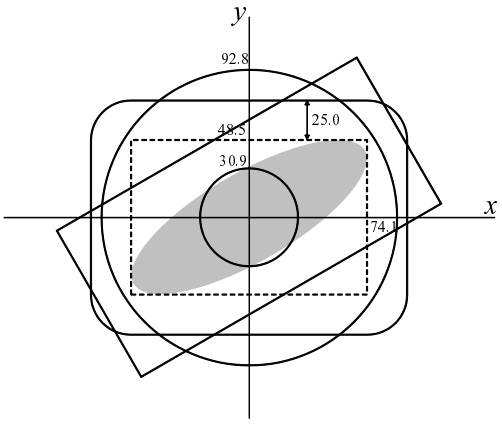

Fig. 16. Integration regions ($\gamma = 100$)

3) Changing Other Parameters: We conducted other experiments by changing the parameters δ, θ and $\mathbf{\Sigma}$. Due to space limitations, we summarize the results here:

- The overall trend does not change if we modify δ, the distance threshold, but for a small δ value, the combination generally becomes more effective. When δ is large, RR and BF have almost the same filtering regions, and the difference between them is rather small.
- Change of θ, the probability threshold, does not influence the trend directly, and the combination approach is better. An interesting observation is that the processing cost does not increase, for example, if we change the threshold value from $\theta = 0.1$ to $\theta = 0.01$. This is due to the exponential feature of the Gaussian distribution: their filtering regions are almost same.
- The internal entries of the covariance matrix $\mathbf{\Sigma}$ determines the shape of the isosurface of the distribution. When the matrix is close to being a unit matrix, the difference between the three strategies becomes small since the isosurface is close to being a sphere. In contrast, if we choose a matrix such that its isosurface has a thin ellipsoidal shape, the difference will increase. In this case, their combination reduces the query cost.

VI. Experiments II (9D Data)

A. Experimental Setup

Next, we perform small experiments for observing the behaviors of our method for medium-dimensional cases. The purpose is not an exhaustive analysis, but to get intuitions for our framework. We used Corel Image Features data set from UCI KDD Archive [25]. Specifically, Color Moments data, which consists of 68,040 nine-dimensional vectors, was used. For this data, the Euclidean distance is assumed for similarity retrieval.

We consider the following scenario that is based on the "pseudo-feedback" technique. First, we select a random object from the dataset and search its k-nearest neighbors (k-NN). In the experiment, we used $k = 20$. Note that k-NN includes the query object itself. We assume that the k-NN objects are *sample images* given by the user, and then derive the covariance matrix as

$$\mathbf{\Sigma} = \tilde{\mathbf{\Sigma}} + \kappa \mathbf{I}, \tag{35}$$

where $\tilde{\mathbf{\Sigma}}$ is the covariance matrix derived from k sample vectors. The additional term $\kappa\mathbf{I}$ is a normalization factor — it is used for avoiding overfitting due to a small number of sample objects. The constant κ is set as $\kappa = |\tilde{\mathbf{\Sigma}}|^{1/9}$ to satisfy $|\tilde{\mathbf{\Sigma}}| = |\kappa\mathbf{I}|$; it means that we blend the sample-based and the Euclidean distance-based approaches with the same importance. As the center of the feedback query $\boldsymbol{q}$, we use the vector of the object selected initially.

The distance parameter of a range query is set as $\delta = 0.7$. If we use this δ value for a standard range query with non-imprecise query location, 15.3 objects are retrieved on average. The probability threshold is set as $\theta = 40\%$. Using Eq. (7), the appropriate r_θ was derived as $r_\theta = 2.32$.

B. Experimental Results

Based on the above procedure, we performed ten random trials. As described in the previous experiments, the cost of Phase 1 (Index-based Search) is negligible so that we focus on Phase 2 (Filtering) and Phase 3 (Probability Computation). Table III shows the averaged number of candidate objects for each method. Compared to RR, BF has a better result. Similar to the previous experiments, combination of three methods gives the best filtering power.

TABLE III
NUMBER OF CANDIDATES ($\delta = 0.7, \theta = 0.4$)

RR	BF	RR+BF	RR+OR	BF+OR	ALL	ANS
3713	3216	2468	1905	1998	1699	3.9

In this situation, OR-based filtering is more effective compared to the 2D case. In this experiment, the number of candidate objects which enter within the filtering region of OR method was 2,620 on average: the value is rather smaller than those of RR and BF. The reason is as follows. The equi-probability isosurfaces of the 9D Gaussian distributions in this experiment have rather narrow shapes. Thus, rectilinear-based bounding (RR) and sphere-based bounding (BF) select many objects as candidates. In contrast, the slanted shape of OR gives more tight regions.

An important problem we can find in Table III is the number of candidate objects is too large compared to the final answer. It may not be surprising that the number of answer objects is small since the probability threshold $\theta = 40\%$ is a rather strict setting. However, it means that we need to perform numerical integration for 1,700 objects on average to derive a tiny answer set. This phenomenon is caused by the increase of dimensions.

To understand the effect of the dimensionality, we show Fig. 17. The figure plots the results of numerical integration of the normalized Gaussian distribution p_{norm} for several different dimensionalities. The x-axis represents the radius of the integration range and the y-axis is the integration result (probability). For example, if a query object obeys 2D p_{norm} distribution, the probability that the object is located within distance one from the origin (the center of the distribution) is 39%. Looking at the figure, we can observe that the radius increases along with the increase of dimensionality for the same probability level. For instance, for the 9D case, the probability that a query object is located within distance two from the query center is only 9%. The reason of this behavior is due to the phenomenon called "curse of dimensionality" [3]. Its effect is critical even for a medium-dimensional case.

The property explains why the number of answers is so small in Table III. For the 9D case, the location of a query object becomes more "imprecise" and it may not be nearby the center of the distribution. Thus, even if a candidate object is close to the distribution center, the result of numerical integration may not be large. Actually, the computed qualification probability of the query object, which is located at the distribution center $\boldsymbol{q}$, was only 70.0% on average for this experiment.

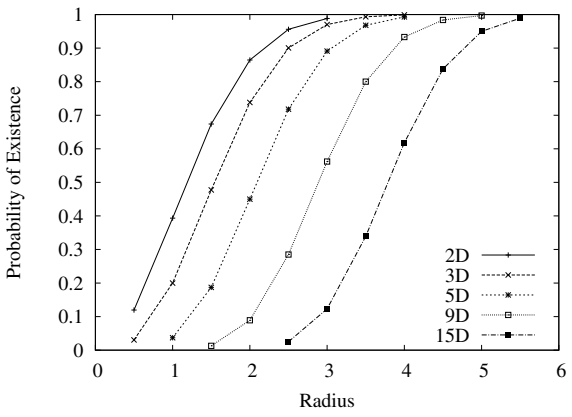

Fig. 17. Probabilities of existence

In other words, when the dimensionality increases, we need to search a larger area if we want to perform queries with the same level of a probability threshold. For example, consider that we are given a query "retrieve objects for the specified distance range with the probability threshold $\theta = 1\%$". Remember that RR and OR need to derive $1 - 2\theta = 98\%$ θ-region based on Eq. (4). In contrast to the corresponding value $r_\theta = 2.79$ for the 2D case, we need to use $r_\theta = 4.44$ for the 9D case. It will increase the volume of the search region.

In addition, the increase of dimensionality gives a negative effect to the BF method. In BF, we need to derive $\beta^\top$ based on Eq. (29). In medium-dimensional cases, the value $(\lambda^\top)^{d/2}|\boldsymbol{\Sigma}|^{1/2}\theta$, required for the derivation, tends to be a tiny value. Since the relationship

$$(\lambda^\top)^{d/2}|\boldsymbol{\Sigma}|^{1/2} = \frac{(\lambda^\top)^{d/2}}{|\boldsymbol{\Sigma}^{-1}|^{1/2}} = \left(\frac{(\lambda^\top)^d}{\prod_{i=1}^d \lambda_i}\right)^{1/2} \tag{36}$$

holds, if the shape of a Gaussian distribution is narrow (it means the ratio $\lambda^\perp/\lambda^\top$ is large), the value $(\lambda^\top)^{d/2}|\boldsymbol{\Sigma}|^{1/2}\theta$ may become too small. That means the estimation of $\beta^\top$ requires more accurate computation, which needs a large number of random samples. Moreover, for an ill-shaped Gaussian distribution, the value

$$(\lambda^\perp)^{d/2}|\boldsymbol{\Sigma}|^{1/2} = \left(\frac{(\lambda^\perp)^d}{\prod_{i=1}^d \lambda_i}\right)^{1/2}, \tag{37}$$

required to compute $\beta^\perp$, may become larger than one. That means we cannot find an internal "hole" shown in Fig. 9. Note that BF is still effective for a sphere-like distribution, in which $\lambda^\top \simeq \lambda^\perp$ holds. Particularly, if $\lambda^\top = \lambda^\perp$ is satisfied — that means that the distribution is completely spherical — BF is the best method since it can directly select answer objects and does not require numerical integration.

VII. Conclusions and Future Work

In this paper, we described a probabilistic range query processing technique for a query object with imprecise location information obeying a Gaussian distribution. The query process consisted of three phases: index-based searching, filtering, and probability computation. Since the last phase dominates the processing cost due to time-consuming numerical integration, the filtering phase for pruning the candidate objects plays an important role. We proposed three filtering strategies, Rectilinear-Region-Based (RR), Oblique-Region-Based (OR), and Bounding-Function-Based (BF), and combinations of these approaches that effectively use the underlying properties of the Gaussian distribution. Their effectiveness was evaluated by performing experiments.

In our context, reduction of candidate objects for numerical integration is the most critical factor because the integration process dominates the overall query processing time. For low-dimensional cases, combination of three strategies (ALL) provides the best performance in general since it can filter many objects. If the location of a query object is not so imprecise, RR (or BF) and ALL will provide no remarkable difference. For medium-dimensional cases, the problem itself becomes difficult due to "curse of dimensionality"; since search area increases, we need to perform numerical integration for many candidates for selecting the result objects. For the efficient processing of medium- or high-dimensional cases, we need further development by considering the nature of Gaussian distributions.

Planned future work is as follows. First, we are planning to expand use of our technique to other types of queries such as probabilistic nearest neighbor queries. Second, we would like to extend the framework to environments where the target objects also have uncertain locations. Further future work will include the effective use of the proposed technique in real-world situations such as moving robotics applications and GPS sensor environments.

Acknowledgments

This research is partly supported by the Grant-in-Aid for Scientific Research, Japan (#19024037, #19300027).

References

[1] M. Ankerst, B. Braunmüller, H.-P. Kriegel, and T. Seidl, "Improving adaptable similarity query processing by using approximations," in *Proc. VLDB*, 1998.

[2] S. Berchtold, C. Böhm, D. A. Keim, and H.-P. Kriegel, "A cost model for nearest neighbor search in high-dimensional data space," in *Proc. PODS*, 1997.

[3] C. Böhm, S. Berchtold, and D. A. Keim, "Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases," *ACM Comput. Surv.*, vol. 33, no. 3, pp. 322–373, 2001.

[4] C. Böhm, A. Pryakhin, and M. Schubert, "The Gauss-tree: Efficient object identification in databases of probabilistic feature vectors," in *Proc. ICDE*, 2006.

[5] J. Chen and R. Cheng, "Efficient evaluation of imprecise location-dependent queries," in *Proc. ICDE*, 2007.

[6] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, "Evaluating probabilistic queries over imprecise data," in *Proc. ACM SIGMOD*, 2003.

[7] R. Cheng and S. Prabhakar, "Managing uncertainty in sensor databases," *SIGMOD Record*, vol. 32, no. 4, pp. 41–46, 2003.

[8] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. S. Vitter, "Efficient indexing methods for probabilistic threshold queries over uncertain data," in *Proc. VLDB*, 2004.

[9] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, "Querying imprecise data in moving object environments," *IEEE TKDE*, vol. 16, no. 9, 2004.

[10] N. Dalvi and D. Suciu, "Management of probabilistic data: foundations and challenges," in *Proc. ACM PODS*, 2007.

[11] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley, 2000.

[12] N. Katayama, "HnRStar-1.0," http://research.nii.ac.jp/~katayama/homepage/research/srtree/HnRStar-1.0.tar.gz.

[13] V. Ljosa and A. K. Singh, "APLA: Indexing arbitrary probability distributions," in *Proc. ICDE*, 2007.

[14] Y. Manolopoulos, A. Nanopoulos, A. N. Papadopoulos, and Y. Theodoridis, *R-Trees: Theory and Applications*. Springer, 2005.

[15] M. F. Mokbel, "Towards privacy-aware location-based database servers," in *Proc. Intl. Workshop on Privacy Data Management (PDM)*, 2006.

[16] J. Pei, M. Hua, Y. Tao, and X. Lin, "Query answering techniques on uncertain and probabilistic data (tutorial)," in *Proc. SIGMOD*, 2008.

[17] D. Pfoser and C. S. Jensen, "Capturing the uncertainty of moving-object representations," in *Proc. 6th Intl. Symp. on Advances in Spatial Databases (SSD'99)*, 1999.

[18] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipies: The Art of Scientific Computing*, 3rd ed., Cambridge University Press, 2007.

[19] "RANDLIB," http://biostatistics.mdanderson.org/SoftwareDownload/.

[20] Y. Tao, R. Cheng, X. Xiao, W. K. Ngai, B. Kao, and S. Prabhakar, "Indexing multi-dimensional uncertain data with arbitrary probability density functions," in *Proc. VLDB*, 2005.

[21] Y. Tao, X. Xiao, and R. Cheng, "Range search on multidimensional uncertain data," *ACM TODS*, vol. 32, no. 3, 2007.

[22] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2005.

[23] http://tiger.census.gov/.

[24] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain, "Managing uncertainty in moving objects databases," *ACM TODS*, vol. 29, no. 3, pp. 463–507, 2004.

[25] "UCI KDD Archive," http://kdd.ics.uci.edu/.

[26] O. Wolfson, S. Chamberlain, S. Dao, J. Jiang, and G. Mendez, "Cost and imprecision in modeling the position of moving objects," in *Proc. ICDE*, 1998.

[27] O. Wolfson, A. P. Sistla, S. Chamberlain, and Y. Yesha, "Updating and querying databases that track mobile units," *Distributed and Parallel Databases*, vol. 7, no. 3, pp. 257–287, 1999.

Appendix

Proof of Property 5: Suppose $\boldsymbol{q} = \mathbf{0}$. It does not lose generality. Expanding Eq. (26), we get

$$\int_{\boldsymbol{x}\in R^\top} \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{\lambda^\top}{2}\|\boldsymbol{x}\|^2\right] d\boldsymbol{x} = \theta. \quad (38)$$

We consider a sample sphere $(x_1 - \alpha^\top)^2 + x_2^2 + \cdots + x_d^2 = \delta^2$ that satisfies the target problem. We perform the variable transformation $x_i = u_i/\sqrt{\lambda^\top}$ $(i = 1, \ldots, d)$. Then $R^\top$ is transformed into $(u_1 - \sqrt{\lambda^\top}\alpha^\top)^2 + u_2^2 + \cdots u_d^2 = \lambda^\top\delta^2$. Let the spherical region be $S^\top$. The distance between the center of $S^\top$ and the origin is $\beta^\top = \sqrt{\lambda^\top}\alpha^\top$ and the radius of $S^\top$ is $\sqrt{\lambda^\top}\delta$. Using variable transformation, the formula above can be converted into

$$\int_{\boldsymbol{u}\in S^\top} \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}\|\boldsymbol{u}\|^2\right] |J(\boldsymbol{u})| d\boldsymbol{u} = \theta, \quad (39)$$

where $|J(\boldsymbol{u})|$ is a Jacobian. In this case, $|J(\boldsymbol{u})| = (\lambda^\top)^{-d/2}$ holds. Thus, we get

$$\int_{\boldsymbol{u}\in S^\top} \frac{1}{(2\pi)^{d/2}} \exp\left[-\frac{1}{2}\|\boldsymbol{u}\|^2\right] d\boldsymbol{u} = (\lambda^\top)^{d/2}|\boldsymbol{\Sigma}|^{1/2}\theta. \quad (40)$$