IEICE TRANS. INF. & SYST., VOL.E97–D, NO.4 APRIL 2014

PAPER Special Section on Data Engineering and Information Management

Probabilistic Range Querying over Gaussian Objects

Tingting DONG[†a)], **Chuan XIAO**[†b)], ***Nonmembers***, ***and*** **Yoshiharu ISHIKAWA**[†c)], ***Member***

SUMMARY Probabilistic range query is an important type of query in the area of uncertain data management. A probabilistic range query returns all the data objects within a specific range from the query object with a probability no less than a given threshold. In this paper, we assume that each uncertain object stored in the database is associated with a multi-dimensional Gaussian distribution, which describes the probability distribution that the object appears in the multi-dimensional space. A query object is either a certain object or an uncertain object modeled by a Gaussian distribution. We propose several filtering techniques and an R-tree-based index to efficiently support probabilistic range queries over Gaussian objects. Extensive experiments on real data demonstrate the efficiency of our proposed approach.

key words: *uncertain data, probabilistic databases, Gaussian distribution, range queries*

1. Introduction

In the area of uncertain data management, *probabilistic range query* is an important problem for processing uncertain data in real-world applications such as mobile robotics and sensor networks. A probabilistic range query returns all the data objects that appear within the given search region with probabilities no less than a given probability threshold.

For instance, consider a self-navigated mobile robot moving in a wireless environment. The robot builds a map of the environment by observing nearby landmarks via devices such as sonars and laser range finders. Due to the inherent limitation brought about by sensor accuracy and signal noises, the location information acquired from measuring devices is not always precise. At the same time, the robot also conducts probabilistic localization [1] to estimate its own location autonomously by integrating its movement history and the landmark information. This may cause impreciseness in the location of the robot, too. In consequence, probabilistic queries have evolved to tackle such impreciseness as "find landmarks lying within 5 meters from my current location with a probability at least 80%".

Typically for such applications, uncertain objects are stored in the database and associated with probability distributions. Multi-dimensional *Gaussian distribution* is one of the commonly used distributions for such a purpose. It is widely adopted in statistics, pattern recognition [2], and localization in robotics [1].

Manuscript received July 4, 2013.

[†]The authors are with the Graduate School of Information Science, Nagoya University, Nagoya-shi, 464–8601 Japan.
a) E-mail: dongtt@nagoya-u.jp
b) E-mail: chuanx@nagoya-u.jp
c) E-mail: ishikawa@is.nagoya-u.ac.jp

DOI: 10.1587/transinf.E97.D.694

In this paper we study the case where the locations of data objects are uncertain, whereas the location of the query object is either exact or uncertain. Specifically, data objects are described by Gaussian distributions with different parameters to indicate their respective uncertainty. A query object can be either at a certain point in the multi-dimensional space or an uncertain location represented by a multi-dimensional Gaussian distribution. We solve the probabilistic range query problem according to the above setup.

There have been solutions to probabilistic range queries that can handle Gaussian-based uncertain data, yet based on specific assumptions. For example, U-tree [3] assumes that each uncertain object exists within a *pre-defined uncertainty region*. It constructs an index for all the objects based on this region to reduce the number of candidates that require the expensive numerical integration. However, for some application scenarios it is not easy to decide a suitable extent of the uncertainty region for a real world object. Gauss-tree [4] assumes that the Gaussian distribution must be *independent* in each dimension. When these assumptions are violated, the solutions no longer work. In this paper, we solve these problems with generic Gaussian distributions without any of these assumptions, i.e., an object can locate in an infinite space as opposed to U-tree, and have correlations between dimensions as opposed to Gauss-tree.

A straightforward approach to this problem is to compute the *appearance probability* [5] for each data object and output it if this probability is no less than the threshold. However, the probability computation usually requires costly numerical integration for the accurate result [3], rendering it prohibitively expensive to compute for all the data objects and check if the query constraint is satisfied. Thus, such computations should be reduced as much as possible.

To this end, we propose filtering techniques to generate a set of candidate data objects and compute integrations only for these candidates. Equipped with the filtering techniques, an R-tree-based indexing method is proposed to accelerate query processing. The index structure is inspired by the idea of TPR-tree [6], in which the minimum bounding boxes (MBBs) vary with time. The difference is that in our index a parent MBB not only varies with the probability threshold but also tightly encloses all the child MBBs.

In our preliminary work [7], we proposed query processing algorithms for probabilistic range queries, assuming that *only the location of the query object* is uncertain and modeled by a Gaussian distribution, but data objects are

certain multi-dimensional points. An R-tree can be used to manage these certain data points to process queries, which is different from the situation here. In this paper, we extend the uncertainty to data objects and propose novel solutions. A precedent report of this work has appeared in [8]. The approach proposed in [8] approximates the Gaussian distribution with an upper-bounding function. An R-tree-like hierarchical index structure was proposed and an exponential summary function was defined to cover multiple upper-bounding functions. Nevertheless, the summary function is so sensitive to the child functions that it will become drastically large if the bounded Gaussian distributions are sparsely distributed in the space or one of them has large variances, leading to loose bounding in the index and weak filtering power. This paper is an extension of our previous work [9]. We have extended the algorithms and conducted additional experiments.

Our contributions are summarized as follows:

1. We formalize two types of probabilistic range queries with respect to the query object: a certain point and an uncertain location represented by a Gaussian distribution, while data objects are represented by Gaussian distributions with different parameters.
2. For the two types of queries, we propose several effective filtering techniques to identify promising data objects and prune unpromising ones.
3. We design a novel R-tree-based index structure to support probabilistic range queries on Gaussian objects.
4. We demonstrate the efficiency of our approach through comprehensive experimental performance study.

The rest of the paper is organized as follows. Section 2 defines our problem. We present our filtering strategies in Sect. 3. Section 4 describes our index structure. Experimental results and analyses are covered by Sect. 5. Section 6 reviews related work. Section 7 concludes the paper.

2. Problem Definition

In this section, we first define Gaussian objects, and then define probabilistic range queries on two types of query objects: certain point objects and uncertain Gaussian objects.

2.1 Gaussian Objects

The Gaussian distribution, also known as the normal distribution, is a continuous probability distribution defined by a bell-shaped probability density function in the one-dimensional case. In this paper, we assume that data objects are modeled by Gaussian distributions in a d-dimensional space. A point $\boldsymbol{x}$ is in a d-dimensional numerical space, namely, $\boldsymbol{x} = (x^1, \ldots, x^d)^t$.

Definition 1 (Gaussian objects): A Gaussian object o is represented by its possible locations (points) and the probability density it appears at each location. Formally, the probability density that o is located at $\boldsymbol{x}_o$ is captured by a d-dimensional Gaussian probability density function

$$p_o(\boldsymbol{x}_o) = \frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}_o|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\boldsymbol{x}_o - \boldsymbol{\mu}_o)^t \boldsymbol{\Sigma}_o^{-1}(\boldsymbol{x}_o - \boldsymbol{\mu}_o)\right]. \quad (1)$$

$\boldsymbol{\mu}_o$ is the mean location (center) of o. $\boldsymbol{\Sigma}_o$ is a $d \times d$ covariance matrix. $|\boldsymbol{\Sigma}_o|$ and $\boldsymbol{\Sigma}_o^{-1}$ are the determinant and the inverse of $\boldsymbol{\Sigma}_o$, respectively.

2.2 Probabilistic Range Queries on Gaussian Objects

Given a dataset of Gaussian objects $\mathcal{D}$, a query object q, a distance threshold δ, and a probability threshold θ, a *probabilistic range query (PRQ) on Gaussian objects* retrieves all the data objects $o \in \mathcal{D}$ such that the distance between o and q is no more than δ with a probability no less than θ. In this paper, we consider two types of query objects for q:

1. The query object is a point object, namely,
$$\boldsymbol{q} = (x_q^1, x_q^2, \ldots, x_q^d)^t.$$
2. The query object is a Gaussian object, namely,
$$p_q(\boldsymbol{x}_q) = \frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}_q|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\boldsymbol{x}_q - \boldsymbol{\mu}_q)^t \boldsymbol{\Sigma}_q^{-1}(\boldsymbol{x}_q - \boldsymbol{\mu}_q)\right].$$

The *probabilistic range query with a point query object* (PRQ-P) is formally defined as

$$\text{PRQ-P}(\mathcal{D}, q, \delta, \theta) = \{o \mid o \in \mathcal{D}, \Pr(\|\boldsymbol{x}_o - \boldsymbol{q}\| \le \delta) \ge \theta\},$$

where $\|\boldsymbol{x}_o - \boldsymbol{q}\|$ represents the Euclidean distance between $\boldsymbol{x}_o$ and $\boldsymbol{q}$. We call the region consisting of the points with distances no more than δ from the query object *the query region*, *QR* for short. $\Pr(\|\boldsymbol{x}_o - \boldsymbol{q}\| \le \delta)$, the probability that o lies within *QR*, is computed by

$$\Pr(\|\boldsymbol{x}_o - \boldsymbol{q}\| \le \delta) = \int \chi_\delta(\boldsymbol{x}_o, \boldsymbol{q}) \cdot p_o(\boldsymbol{x}_o) d\boldsymbol{x}_o, \quad (2)$$

where

$$\chi_\delta(\boldsymbol{x}_o, \boldsymbol{q}) = \begin{cases} 1, & \|\boldsymbol{x}_o - \boldsymbol{q}\| \le \delta; \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

The integration in Eq. (2) is not in a closed-form and hence cannot be computed directly. Numerical solutions such as Monte Carlo methods can be employed to evaluate the probability. We use the *importance sampling* [10] in the interest of efficiency. Specifically, we generate $\boldsymbol{x}_o$ as per the probability function $p_o(\boldsymbol{x}_o)$, and increment the count when Eq. (3) is satisfied. Finally, the probability can be obtained by dividing the count by the number of samples generated. Generally speaking, however, Monte Carlo methods are accurate only if the number of samples is sufficiently large (at the order of 10^6) [3]. Therefore, the probability computation

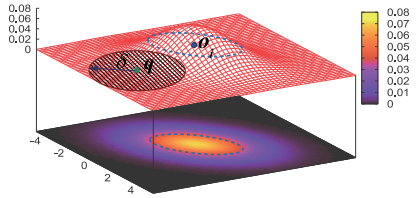


Fig. 1 PRQ-P query.

induces expensive runtime cost.

Figure 1 illustrates a PRQ-P query in a 2-dimensional space. The Gaussian object o exists in the space with decreasing probability densities as it spreads from the center $\boldsymbol{\mu}_o$. A PRQ-P query finds the Gaussian objects located in the proximity of the query point with a required probability. Computing the probability using Eq. (2) corresponds to integrating the probability density function of o within the shaded area around q.

Similar to PRQ-P, the *probabilistic range query with a Gaussian query object* (PRQ-G) is defined as

$$\text{PRQ-G}(\mathcal{D}, q, \delta, \theta) = \{o \mid o \in \mathcal{D}, \Pr(\|\boldsymbol{x}_o - \boldsymbol{x}_q\| \le \delta) \ge \theta\},$$

where $\Pr(\|\boldsymbol{x}_o - \boldsymbol{x}_q\| \le \delta)$ is computed by

$$\Pr(\|\boldsymbol{x}_o - \boldsymbol{x}_q\| \le \delta) = \iint \chi_\delta(\boldsymbol{x}_o, \boldsymbol{x}_q) \cdot p_o(\boldsymbol{x}_o) \cdot p_q(\boldsymbol{x}_q) d\boldsymbol{x}_o d\boldsymbol{x}_q, \quad (4)$$

where

$$\chi_\delta(\boldsymbol{x}_o, \boldsymbol{x}_q) = \begin{cases} 1, & \|\boldsymbol{x}_o - \boldsymbol{x}_q\| \le \delta; \\ 0, & \text{otherwise.} \end{cases}$$

To compute the integration in Eq. (4), although we can simply generate random numbers for the two Gaussian distributions $p_o(\boldsymbol{x}_o)$ and $p_q(\boldsymbol{x}_q)$, respectively, a more efficient method is shown in [8]. It constructs a $2d$-dimensional Gaussian distribution by combining the two d-dimensional Gaussian distributions. Then the probability can be computed by running the Monte Carlo method.

3. Filtering Based on Approximated Region

A naïve algorithm to answer PRQ-P or PRQ-G queries is to pair the query object with each data object and perform integration check with either Eq. (2) or Eq. (4). However, due to the overhead of the integration, the algorithm becomes prohibitively expensive for large datasets. So we develop our approach based on the filter-and-refine paradigm, i.e., to obtain a set of candidate objects and then compute the integration only for the candidates.

In this section, we first introduce the notion of ρ-region that leverages the two thresholds δ and θ, and then propose the ρ-region-based filtering techniques to handle PRQ-P and PRQ-G queries.

3.1 ρ-Region

Definition 2 (ρ-region): Consider a Gaussian object o and the integration of its probability density function $p_o(\boldsymbol{x}_o)$ over an ellipsoidal region $(\boldsymbol{x}_o - \boldsymbol{\mu}_o)^t \boldsymbol{\Sigma}_o^{-1} (\boldsymbol{x}_o - \boldsymbol{\mu}_o) \le r^2$. Let r_ρ be the value of r within which the integration equals ρ:

$$\int_{(\boldsymbol{x}_o - \boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_o^{-1} (\boldsymbol{x}_o - \boldsymbol{\mu}_o) \le r_\rho^2} p_o(\boldsymbol{x}_o) d\boldsymbol{x}_o = \rho.$$

We call the ellipsoidal region

$$(\boldsymbol{x}_o - \boldsymbol{\mu}_o)^t \boldsymbol{\Sigma}_o^{-1} (\boldsymbol{x}_o - \boldsymbol{\mu}_o) \le r_\rho^2$$

the *ρ-region* of o.

In Fig. 1, the area encompassed by the dotted ellipsoidal curve shows a ρ-region. Because the probability density of a Gaussian distribution decreases as we move away from the center of the object, if the query object is distant enough from the center, its probability within QR will not reach θ. In other words, it is possible to determine whether a data object can satisfy the query condition by deriving a suitable ρ-region with θ (the method is introduced in Sect. 3.3 and 3.4) and examining whether its ρ-region intersects QR.

To compute r_ρ with a given ρ, we borrow the approach proposed in our previous work [7]. It transforms the integration over an ellipsoidal region to an integration over a d-dimensional spherical region. By assigning $\boldsymbol{\mu}_o = \mathbf{0}$ and $\boldsymbol{\Sigma}_o = \mathbf{I}$ in Eq. (1), we have the *normalized Gaussian distribution*

$$p_{\text{norm}}(\boldsymbol{x}) = \mathcal{N}(\mathbf{0}, \mathbf{I}) = \frac{1}{(2\pi)^{d/2}} \exp\left[-\frac{1}{2}\|\boldsymbol{x}\|^2\right].$$

Based on this probability density function, the following property can be derived.

Property 1: [7] Consider integration of $p_{\text{norm}}(\boldsymbol{x})$ over $\|\boldsymbol{x}\|^2 \le r^2$. For a given ρ $(0 < \rho < 1)$, let $\tilde{r}_\rho$ be the radius within which the integration becomes ρ:

$$\int_{\|\boldsymbol{x}\|^2 \le \tilde{r}_\rho^2} p_{\text{norm}}(\boldsymbol{x}) d\boldsymbol{x} = \rho.$$

Then $r_\rho = \tilde{r}_\rho$ holds.

The preceding property indicates that we can compute $\tilde{r}_\rho$ and hence r_ρ for a given ρ value. Therefore, we can construct a (ρ, r_ρ)-table offline (numerical integration is necessary) and obtain the ρ-region by looking up the corresponding r_ρ from this table. If there is no matched entry for a given ρ, we conservatively return the corresponding r_ρ of the smallest value greater than ρ, so the correctness of the result can be guaranteed.

The ellipsoidal shape of a ρ-region renders it difficult to quickly examine whether the ρ-region intersects QR as well as develop an indexing scheme based on prevalent spatial indexes such as R-tree. Hence we will study the minimum bounding box (MBB) which tightly bounds the ρ-region.

3.2 Minimum Bounding Box of ρ-Region

Figure 2 shows the MBB of a ρ-region of a 2-dimensional

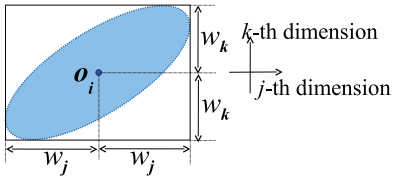


Fig. 2 MBB of ρ-region.

Gaussian object o. Let w_j denote the length of its edge along the j-th dimension. The following property holds [7].

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

$$w_j = \sigma_j r_\rho, \tag{5}$$

where σ_j corresponds to the standard deviation in the j-th dimension

$$\sigma_j = \sqrt{(\boldsymbol{\Sigma}_o)_{jj}},$$

where $(\boldsymbol{\Sigma}_o)_{jj}$ represents the (j, j)-th element of $\boldsymbol{\Sigma}_o$.

For a data object o, since σ_j can be calculated from the covariance matrix $\boldsymbol{\Sigma}_o$, the scale of the MBB is determined uniquely by r_ρ, and hence ρ. Consequently, in order to establish the filtering conditions utilizing the MBBs, it is essential to explore the relation between ρ and the probability threshold θ. Next we will present our filtering techniques for PRQ-P and PRQ-G, respectively.

3.3 Filtering Policies for PRQ-P Queries

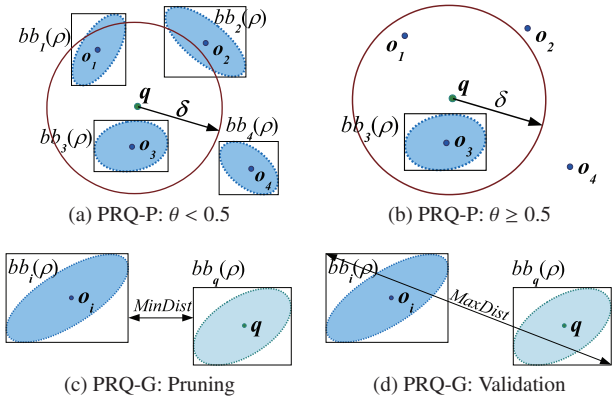
Our filtering policies to process PRQ-P queries are divided into two cases: $\theta < 0.5$ and $\theta \geq 0.5$.

Case 1 ($\theta < 0.5$)**:** Consider the four data objects o_1, o_2, o_3, o_4 shown in Fig. 3 (a). $bb_i(\rho)$ denotes the MBB of o_i's ρ-region. First, let's consider o_4. Since the probability that o_4 is located inside its ρ-region is ρ, the probability of being outside the ρ-region's MBB, is definitely less than $1 - \rho$. Furthermore, given the line symmetry of the Gaussian distribution, the probability that o_4 exists inside QR is at most $(1 - \rho)/2$. Hence, if $\rho = 1 - 2\theta$, and $bb_4(\rho)$ and QR do not overlap, the probability that o_4 lies within QR must be less than θ. Then, for objects o_1, o_2 and o_3, we check and find their MBBs intersect QR, so they become candidates. In summary, when $\theta < 0.5$, a data object is a candidate only if its $bb_i(1 - 2\theta)$ intersects QR.

Case 2 ($\theta \geq 0.5$)**:** We show our idea in Fig. 3 (b). For the probability that a data object exists inside QR to reach θ, its mean location should lie within QR. Thus, o_2 and o_4 can be pruned, whereas o_1 and o_3 are considered as candidates.

Moreover, for all candidates, let $\rho' = \theta$ and compute their $bb_i(\rho')$'s. If QR fully contains a $bb_i(\rho')$ (e.g., o_3), the probability that this object lies within QR is definitely greater than θ. We validate it as a result without computing the numerical integration.

Summarizing the two cases, the filtering condition for PRQ-P is: (1) $\theta < 0.5$, $bb_i(\rho)$ ($\rho = 1 - 2\theta$) intersects QR for pruning, and $bb_i(\rho')$ ($\rho' = \theta$) is contained by QR for validation. (2) $\theta \geq 0.5$, $\|\boldsymbol{\mu}_o - \boldsymbol{q}\| < \delta$ for pruning, and $bb_i(\rho')$ ($\rho' = \theta$) is contained by QR for validation.



(a) PRQ-P: $\theta < 0.5$ (b) PRQ-P: $\theta \geq 0.5$

(c) PRQ-G: Pruning (d) PRQ-G: Validation

Fig. 3 Filtering policies for PRQ-P and PRQ-G.

3.4 Filtering Policies for PRQ-G Queries

For PRQ-G queries, since both the query object q and the data object o_i are Gaussian distributions, we obtain both of their MBBs of ρ-regions. We also consider two cases for filtering: $\theta < 0.75$ and $\theta \geq 0.75$.

Case 1 ($\theta < 0.75$)**:** As shown in Fig. 3 (c), assume q and o_i are independent in the space. When the minimum distance between the two MBBs is exactly δ, the maximal probability of $\|\boldsymbol{x}_o - \boldsymbol{x}_q\| \leq \delta$ is given by the following lemma (the proof is in Appendix B):

Lemma 1: If $MinDist(bb_i(\rho), bb_q(\rho)) \geq \delta$, $Pr(\|\boldsymbol{x}_o - \boldsymbol{x}_q\| \leq \delta) < (3 - 2\rho - \rho^2)/4$.

Let $(3 - 2\rho - \rho^2)/4 = \theta$, i.e., $\rho = 2\sqrt{1 - \theta} - 1$. o_i can be excluded from the candidate set if the minimum distance between $bb_i(\rho)$ and $bb_q(\rho)$ is no less than δ.

Lemma 2: If $\|\boldsymbol{\mu}_o - \boldsymbol{\mu}_q\| \geq \delta$, $Pr(\|\boldsymbol{x}_o - \boldsymbol{x}_q\| \leq \delta) < 0.75$.

Case 2 ($\theta \geq 0.75$)**:** Based on Lemma 2 (the proof is in Appendix B), an object can be pruned if the distance between its center and that of the query object is no less than δ.

Moreover, let $\rho' = \sqrt{\theta}$, if the maximum distance of $bb_i(\rho')$ and $bb_q(\rho')$ is less than δ, as shown in Fig. 3 (d), the probability of $\|\boldsymbol{x}_o - \boldsymbol{x}_q\| \leq \delta$ is greater than ρ'^2, i.e., θ. In this case, we validate it as a result without computing the exact probability by numerical integration.

The filtering condition for PRQ-G is summarized as: (1) $\theta < 0.75$, $MinDist(bb_i(\rho), bb_q(\rho)) < \delta$ ($\rho = 2\sqrt{1 - \theta} - 1$) for pruning, and $MaxDist(bb_i(\rho'), bb_q(\rho')) < \delta$ ($\rho' = \sqrt{\theta}$) for validation. (2) $\theta \geq 0.75$, $\|\boldsymbol{\mu}_o - \boldsymbol{\mu}_q\| < \delta$ for pruning, and $MaxDist(bb_i(\rho'), bb_q(\rho')) < \delta$ ($\rho' = \sqrt{\theta}$) for validation.

4. Indexing Data Objects

4.1 The Index Structure

The filtering conditions introduced in the previous section

need to know the value of θ and hence ρ to generate candidates. In order not to scan all the data objects and compute the MBBs of the ρ-regions on the fly with the given θ, an immediate solution is to index the MBBs for a sufficiently large $\rho_{\max}$. Because the MBB with a larger ρ always subsumes the one with a smaller ρ, it can support all the queries if the ρ values computed from θ satisfy the condition $\rho \le \rho_{\max}$. However, the efficiency of the index is compromised for small ρ values. This method serves as a baseline algorithm (we use an R-tree to index MBBs and name it FR-tree), and will be compared in the experiment with the indexing technique we are going to present.

Inspired by the TPR-tree [6], we propose an R-tree-based index structure which stores the MBBs in a parametric fashion. It works for arbitrary probability thresholds and range thresholds, and there is no need to assume the two thresholds are given prior to index construction. The MBBs can be dynamically computed as we traverse the index. Furthermore, the MBB of a node (at both leaf and non-leaf levels) tightly encloses all its children's MBBs regardless of the θ value, as opposed to the TPR-tree within which all child MBBs are bounded in a loose manner.

Our index is a balanced, multi-way tree organized in the structure of an R-tree. Each entry in a leaf node contains a data object in the form of $o_i = (id_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, where id_i is the data object id, $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the mean location and the covariance matrix of the Gaussian object, respectively. In a non-leaf node, each entry has a pointer to a child node and an MBB enclosing all the MBBs within the child node.

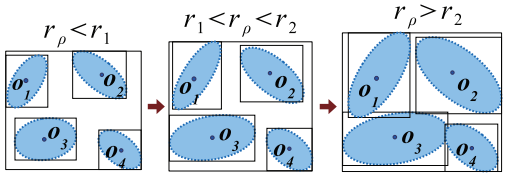
Consider an object o_i with mean location $(x_i^1, \ldots, x_i^d)^t$. Its MBB is a bounding box parameterized with r_ρ (MBB is denoted as the exactly computed bounding box at a specific r_ρ hereafter). From Eq. (5), the extent of the bounding box (BB) in the j-th dimension can be represented by

$$[x_i^j - w_i^j, x_i^j + w_i^j] = [x_i^j - \sigma_i^j r_\rho, x_i^j + \sigma_i^j r_\rho]. \quad (6)$$

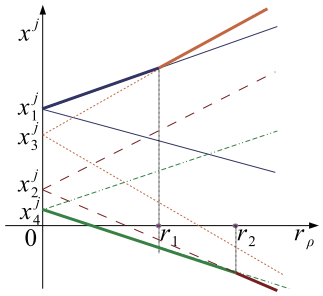
Seeing the BBs grow with r_ρ, in order to tightly bound the BBs of child nodes at every r_ρ, it is necessary to search each dimension for the leftmost and the rightmost BBs under varying ρ. We illustrate this problem in Fig. 4 (a). It shows the changing BB that encloses the BBs of four 2-dimensional objects' ρ-regions as r_ρ increases. When r_ρ is less than r_1, the left edge is determined by o_1, and it becomes o_3 when r_ρ exceeds r_1. The right bound is determined by o_4 when $r_\rho < r_2$, and o_2 otherwise.

Figure 4 (b) shows how the four BBs change horizontally with r_ρ. For each object, a pair of symmetrical lines describe the left and the right coordinates of its BB. The lines have different slopes due to the difference in their standard deviations. The bold polylines illustrate the left and right coordinates of the outer enclosing BB. Hence, the problem becomes how to find the bold polylines. To this end, a BB can be represented by several segments with respect to r_ρ.

We store in the index the j-th dimension of a BB in the form of $(\langle x_1^j, \sigma_1^j, r_1\rangle, \ldots, \langle x_k^j, \sigma_k^j, +\infty\rangle)$. For example, for the four objects in Fig. 4 (a), the left and the right



(a) Node BB of Four Child BBs



(b) Left and Right Edges of the BB

Fig. 4 BB with varying ρ.

coordinates of the BB are $(\langle x_1^j, \sigma_1^j, r_1\rangle, \langle x_3^j, \sigma_3^j, +\infty\rangle)$ and $(\langle x_4^j, \sigma_4^j, r_2\rangle, \langle x_2^j, \sigma_2^j, +\infty\rangle)$, respectively †.

We can find all the segments in the j-th dimension through a sort on the coordinates first and then a linear scan from the object whose standard deviation has a larger value in the j-th dimension. The time complexity is $O(n \log n)$, where n is the number of its child nodes. The number of segments in a BB is at most n.

We note the difference between our index and TPR-tree: (1) The bounding boxes of TPR-tree change towards one direction in a rate (velocity), while our bounding boxes change towards two opposite directions symmetrically with r_ρ. (2) The bounding boxes of TPR-tree are tight only if an update occurs, while our bounding boxes are always tight.

4.2 Query Processing with the Index

Algorithm 1 and Algorithm 2 show the procedures of query processing of PRQ-P (the algorithms for PRQ-G are similar and thus omitted due to the space limitation). To process a query, θ is converted to ρ and ρ', and then r_ρ and $r_{\rho'}$ with the pre-computed (ρ, r_ρ) table (line 1–2). A first-in-first-out queue Q is employed to maintain the nodes we are going to expand. Starting with the root node, we compare QR (MBB of q for PRQ-G) with the node MBB, and check the filtering condition. To compute a node MBB, given r_ρ, we scan the stored jth-dimension of its BB and find α such that $r_{\alpha-1} \le r_\rho < r_\alpha$. Then the extent of its MBB in the j-th dimension can be computed through Eq. (6) using x_α, σ_α, and r_ρ.

At the non-leaf level (line 7–18), if a node MBB at $r_{\rho'}$ satisfies the validation condition, i.e., it is contained by QR (for PRQ-G, the maximum distance of two MBBs is less

†Our experiments show the average number of segments in a BB is 2–3. In case of many segments, we allow users to specify a query probability threshold range $[\theta_{min}, \theta_{max}]$ to reduce the number of segments and hence the index size.

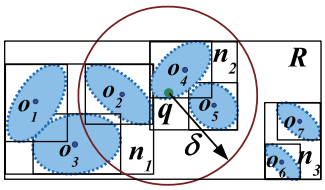


Fig. 5 An illustration of PRQ-P query processing.

than δ), we retrieve all the objects within this node and push them into the result set directly; otherwise, we probe each child entry. If $\theta < 0.5$ (0.75 for PRQ-G), we check whether each child MBB at r_ρ intersects QR (for PRQ-G, whether the minimum distance between each child MBB and the MBB of q is less than δ). On the other hand, when $\theta \geq 0.5$ (0.75 for PRQ-G), we obtain the MBB which bounds leftmost and rightmost means in each dimension, and use this more compact MBB as the new child MBB for filtering checking for both PRQ-P and PRQ-G.

At the leaf level (line 19–30), we also examine whether a node MBB at $r_{\rho'}$ satisfies the validation condition first, so as to push its child entries into the result set as early as possible and reduce the processing cost. If not, each child entry of this node is processed as discussed in Sect. 3.3 (for PRQ-G, Sect. 3.4). When $\theta < 0.5$ (0.75 for PRQ-G), for each child entry, if its MBB at r_ρ intersects QR (for PRQ-G, the minimum distance between each child MBB and the MBB of q is less than δ), we identify this data object using Algorithm 2. We check whether the validation condition is satisfied. The qualified data objects are inserted directly into the result set. Other data objects are regarded as candidates for further verification through integration. When $\theta \geq 0.5$ (0.75 for PRQ-G), for each child entry, if its mean location is inside QR, it is checked by Algorithm 2 in the same way.

Figure 5 shows a query processing example for PRQ-P with $\theta = 0.3$. $r_{0.4}$ and $r_{0.3}$ are obtained from the (ρ, r_ρ) table. MBBs of all nodes in Fig. 5 are at $r_{0.4}$. At first, the root R is popped from Q. Since the MBB of R cannot satisfy the validation condition, we have to check its child nodes n_1, n_2 and n_3. Only n_1 and n_2 are pushed into Q because the MBB of n_3 does not intersect QR. Then we continue with the next level. For n_1, its MBB at $r_{0.4}$ is not contained by QR, so its child entries cannot be all validated as results. Since its MBB at $r_{0.3}$ intersects QR, we probe into its child entries and find that o_1 can be pruned. Then o_2 is validated as a result object and o_3 becomes a candidate further processed through integration. For n_2, since its MBB at $r_{0.3}$ is completely within QR, its entries o_4 and o_5 are inserted into the result set directly without referring to their MBBs.

Algorithm 1 PRQ-P($\boldsymbol{q}, \delta, \theta$)

```
1: ρ ← 1 − 2θ, ρ′ ← θ
2: r_ρ ← TableLookup(ρ), r_ρ′ ← TableLookup(ρ′)
3: QR ← QueryRegion(q, δ), ResultSet = ∅, CandidateSet = ∅
4: Q ← Root, N ← 0
5: while Q ≠ ∅ do
6:     N ← Q.pop()
7:     if N is an internal node then                          ▷ at the non-leaf level
8:         if bb_N(r_ρ′) IsContainedBy QR then
9:             Insert all the objects within N into ResultSet
10:        else
11:            if θ < 0.5 then
12:                for each child N_i in N do
13:                    if bb_{N_i}(r_ρ) Intersects QR then
14:                        Push N_i into Q
15:            else                                           ▷ θ ≥ 0.5
16:                for each child N_i in N do
17:                    if meanbb_{N_i} Intersects QR then
18:                        Push N_i into Q
19:    else                                                   ▷ at the leaf level
20:        if bb_N(r_ρ′) IsContainedBy QR then
21:            Insert all the objects within N_i into ResultSet
22:        else
23:            if θ < 0.5 then
24:                for each child N_i in N do
25:                    if bb_i(r_ρ) Intersects QR then
26:                        IdentifyObjects(N_i)
27:            else                                           ▷ θ ≥ 0.5
28:                for each child N_i in N do
29:                    if mean(N_i) IsInside QR then
30:                        IdentifyObjects(N_i)
```

Algorithm 2 IdentifyObjects(N_i)

```
1: if bb_{N_i}(r_ρ′) IsContainedBy QR then
2:     Insert N_i into ResultSet
3: else
4:     Integral ← computeIntegral(N_i)
5:     if Integral ≥ θ then
6:         Insert N_i into ResultSet
```

5. Experiments

5.1 Experimental Setup

We design two baseline approaches for experimental evaluation. One baseline approach is to sequentially scan the dataset and compute integrations required for obtaining result probabilities. We name it Scan and evaluate our filtering techniques by comparing query processing time and candidate number with it. The other baseline approach indexes the MBBs of the ρ-region with $\rho_{\max} = 0.99$. Because the MBB with a larger ρ always subsumes the one with a smaller ρ, it can support all the queries if the ρ values computed from θ satisfy $\rho \leq \rho_{\max}$. We equip this index with our basic filtering techniques and name it FR-tree, and evaluate our index structure by comparing filtering time and I/O access with it. Our index is referred to as G-tree.

Three real datasets are used in our experiments. MG and LB are two 2-dimensional datasets of Montgomery and Long Beach road networks (39 K and 52 K, respectively) [†]. Airport is a 3-dimensional dataset containing latitudes, longitudes and elevations of 41 K airports in the world [††]. All datasets are normalized to $[0, 1000]^d$. LB is used by default.

We generate PRQ-P and PRQ-G queries randomly. The probability threshold θ lies within $[0.01, 0.99]$, and the

[†]http://www.census.gov/geo/www/tiger
[††]http://www.ourairports.com/data

Table 1 Query response time on LB (seconds).

Query / Algorithm	Overall	Integration
PRQ-P / G-tree	0.157	0.154
PRQ-P / Scan	120.764	120.692
PRQ-G / G-tree	0.809	0.806
PRQ-G / Scan	236.725	236.577

query range δ is chosen from $[10, 100]$ for MG and LB, and $[100, 200]$ for Airport randomly. We also generate covariance matrices for both data Gaussian objects and query Gaussian objects randomly.

We implement the index structure by extending the spatial index library SaiL [†] [11]. It is a generic framework that integrates spatial and spatio-temporal index structures and supports user-defined datatypes and customizable spatial queries. We conduct experiments using a PC with Intel Core 2 Duo CPU E8500 (3.16 GHz), RAM 4 GB, running Fedora 12. We construct an index of all data objects for both PRQ-P and PRQ-G, and store it in the secondary memory.

5.2 Query Performance Evaluation

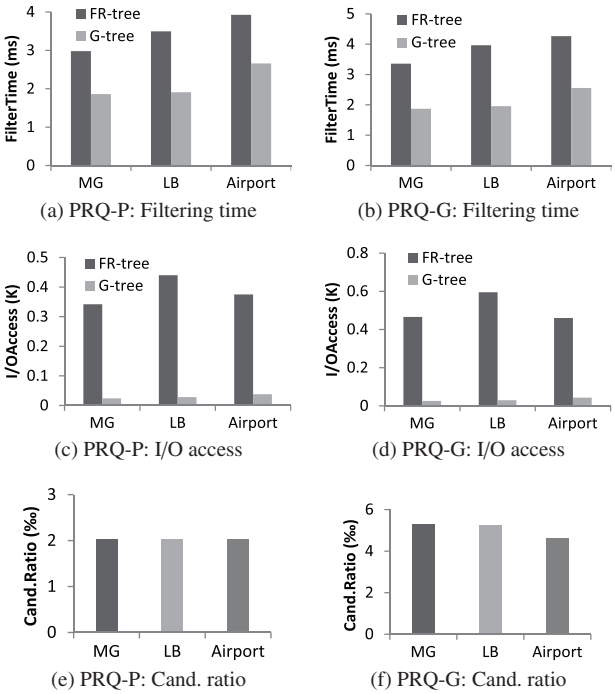
The average query response time of 200 queries on LB is shown in Table 1. It can be seen that the query response time of Scan is 769 times that of G-tree for PRQ-P (293 for PRQ-G). Another observation is that the time spent on probability integration is almost equal to the overall response time. This indicates that the integration dominates the overall query processing and is computationally expensive. Consequently, it is important to reduce candidates which need to perform the integration as much as possible.

Among 50,747 objects in LB, the average candidate number of G-tree is 97 for PRQ-P (269 for PRQ-G). The number of validated objects by integration is 65 for PRQ-P (156 for PRQ-G). So for PRQ-P 67% (58% for PRQ-G) of the candidates identified by our approach are real results. This demonstrates the effectiveness of our proposed filtering techniques. In the sequel, we exclude the integration part from query processing and focus on evaluating the filtering and indexing performance of FR-tree and G-tree.

We run the two algorithms to process 10 K queries on the three datasets and show the average filtering time and I/O access of PRQ-P (resp. PRQ-G) in Fig. 6 (a)–6 (c) (resp. Fig. 6 (b)–6 (d)). For PRQ-P, the average filtering time of G-tree is 61.6% of that of FR-tree on the three datasets, because the average I/O access of G-tree is 92.2% less than that of FR-tree, though the segmented bounding boxes in G-tree are more complex to process than those in FR-tree. The reduction on PRQ-G is more substantial. The average filtering time of G-tree on MG, LB and Airport is 45% less than that of FR-tree. The I/O access of G-tree of three datasets is 6.5% that of FR-tree.

As a $\rho_{\max}$ is adopted to process queries with any θ, the bounding boxes in FR-tree are very loose. This causes more I/O accesses and increases filtering time. In contrast, since the bounding boxes in G-tree are constructed in a parametric fashion, they can be calculated dynamically for arbitrary θ and hence are very compact. Another interesting observation is that the I/O access almost resembles the candidate number, indicating most I/Os are spent on retrieving objects.

[†]http://libspatialindex.github.com/



(a) PRQ-P: Filtering time (b) PRQ-G: Filtering time

(c) PRQ-P: I/O access (d) PRQ-G: I/O access

(e) PRQ-P: Cand. ratio (f) PRQ-G: Cand. ratio

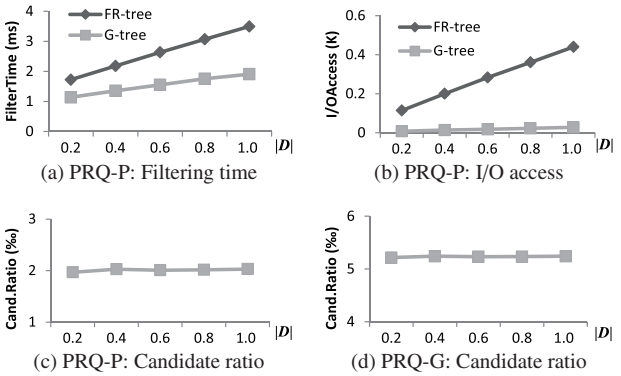
Fig. 6 Filtering and indexing performance.

Figure 6 (e)–6 (f) show the candidate ratio of PRQ-P and PRQ-G, which is calculated by dividing the candidate number by the total number of objects. The candidate number of FR-tree and G-tree is the same since we equip FR-tree with our filtering techniques. The candidate ratio is around 2‰ for PRQ-P and 5‰ for PRQ-G on the three datasets. This reveals that only a very small percentage of data objects will become candidates owing to our filtering techniques.

Varying Dataset Size. To evaluate the scalability of our approach, we extract 20%, 40%, 60%, 80% and 100% of LB dataset randomly and show the filtering time and I/O access of two methods in Fig. 7 (a)–7 (b) on PRQ-P queries. The performance on PRQ-G queries reveals a similar trend and hence is omitted due to the space limitation. As the dataset size becomes larger, the filtering time and I/O access of the two methods almost increase linearly. G-tree displays a steady increasing trend and always outperforms FR-tree.

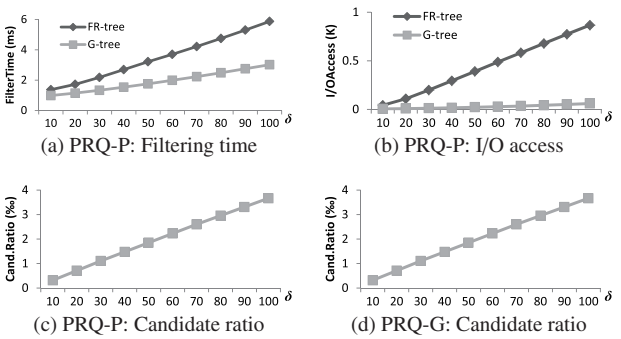
As shown in Fig. 7 (c)–7 (d), in spite of the varying dataset size $|\mathcal{D}|$, the candidate ratio of PRQ-P retains 2‰ and 5.2‰ for PRQ-G, demonstrating the steadiness and scalability of our approach with respect to the dataset size.

Varying Query Range. We vary the query range δ from 10 to 100 and show the performance on PRQ-P queries in Fig. 8 (a)–8 (b). The performance on PRQ-G queries is similar and hence omitted. As δ increases, FR-tree consumes much more time and I/O accesses on filtering. In contrast,



(a) PRQ-P: Filtering time (b) PRQ-P: I/O access (c) PRQ-P: Candidate ratio (d) PRQ-G: Candidate ratio

Fig. 7 Varying $|\mathcal{D}|$.



(a) PRQ-P: Filtering time (b) PRQ-P: I/O access (c) PRQ-P: Candidate ratio (d) PRQ-G: Candidate ratio

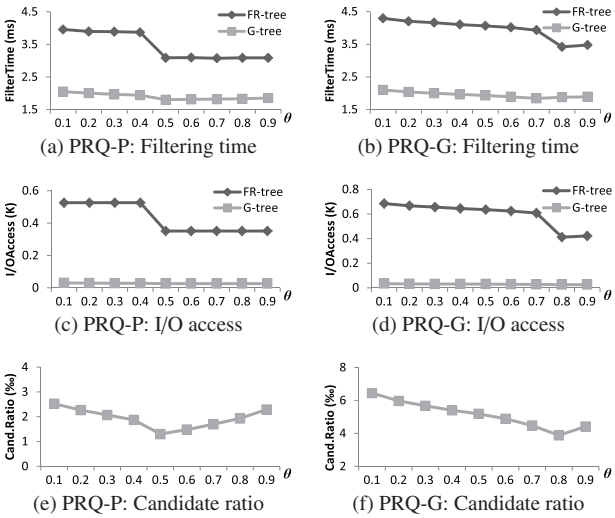
Fig. 8 Varying δ.

G-tree exhibits much slower increasing trends. Figure 8 (c)–8 (d) show that the candidate ratio of both PRQ-P and PRQ-G also increases with δ, but for PRQ-P it is only 3.7‰ (9.4‰ for PRQ-G) even if δ achieves 100.

Varying Probability Threshold. We vary θ from 0.1 to 0.9 and show the performance in Fig. 9 (a)–9 (f) for both PRQ-P and PRQ-G queries. For PRQ-P, the filtering time and I/O access of both FR-tree and G-tree decrease gradually with θ when $\theta < 0.5$ (0.75 for PRQ-G). When θ exceeds 0.5 (0.75 for PRQ-G), the filtering time slightly rebounds. This is consistent with our filtering conditions as discussed in Sect. 3.3–3.4. When $\theta < 0.5$ (0.75 for PRQ-G), $\rho = 1-2\theta$ for PRQ-P (for PRQ-G, $\rho = 2\sqrt{1-\theta} - 1$ if $\theta < 0.75$), so ρ decreases when θ moves towards larger values, and bounding boxes shrink. As a result, most of non-candidates can be filtered quickly and less I/O accesses are needed, and hence it accelerates filtering.

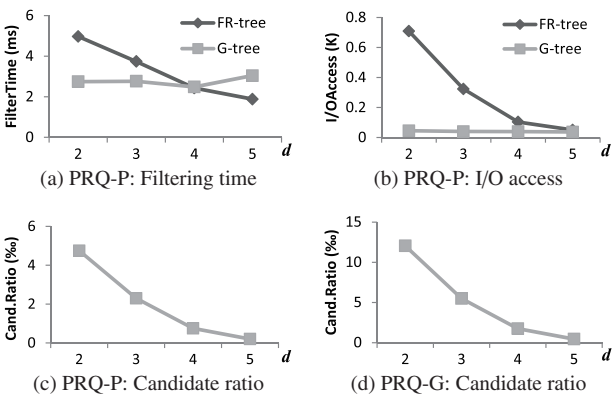
On the contrary, when $\theta \geq 0.5$ (0.75 for PRQ-G), the pruning condition becomes $\|\boldsymbol{\mu}_o - \boldsymbol{q}\| < \delta$ for PRQ-P (for PRQ-G, $\|\boldsymbol{\mu}_o - \boldsymbol{\mu}_q\| < \delta$). So all the figures have turn points at 0.5 for PRQ-P (0.75 for PRQ-G). At the same time, bounding boxes computed for validation which assigns $\rho' = \theta$ for PRQ-P (for PRQ-G, $\rho' = \sqrt{\theta}$) enlarge as θ increases. So more and more nodes and objects will become candidates, leading to the slightly rising of the filtering time and I/O access. The reason also accounts for the trend of G-tree on candidate ratio in Fig. 9 (e)–9 (f).

Despite the variation of θ, G-tree constantly outperforms FR-tree. In the case of PRQ-P, the filtering time of



(a) PRQ-P: Filtering time (b) PRQ-G: Filtering time (c) PRQ-P: I/O access (d) PRQ-G: I/O access (e) PRQ-P: Candidate ratio (f) PRQ-G: Candidate ratio

Fig. 9 Varying θ.



(a) PRQ-P: Filtering time (b) PRQ-P: I/O access (c) PRQ-P: Candidate ratio (d) PRQ-G: Candidate ratio

Fig. 10 Varying d.

FR-tree amounts to 1.8 times that of G-tree on average and 15.8 times on average for I/O access. This contrast is more evident on PRQ-G, where the filtering time of FR-tree is 2 times that of G-tree on average and 20.1 times on average for I/O access.

Varying Dimensionality. We also study the impact of dimensionality d using randomly generated synthetic datasets with the size 20 K and the query range within $[100, 200]$. Figure 10 (a)–10 (d) show the scalability of FR-tree and G-tree against d for PRQ-P. The figures of filtering time and I/O access for PRQ-G have similar trends and are omitted. As shown in Fig. 10 (a), the filtering time of FR-tree reduces constantly with increasing d because the object density decreases with d. This can be confirmed by the decreasing trend of I/O access of PRQ-P in Fig. 10 (b) and the candidate ratio of both PRQ-P and PRQ-G in Fig. 10 (c)–10 (d).

It is also observed that the filtering performance of FR-tree begins to exceed that of G-tree at $d = 5$. The explanation is that, candidates become fewer as object density decreases, and hence the operation of comparing the query region with node bounding boxes dominates the filtering procedure. While FR-tree's MBBs can be obtained directly

from the R-tree, G-tree needs to compute the MBBs with the segments in the index. As d increases, the decreasing object density can lower the processing cost. However, the increasing d can also induce more effort in query processing since the bounding box computation needs to be done for more dimensions. The above two factors result in the fluctuating trend of G-tree's filtering time in Fig. 10 (a).

Index construction. We evaluate the index construction on the Airport dataset. The node capacities of the indexes are selected to optimize query performance for both FR-tree and G-tree. The index size of FR-tree is 6.94 MB, and the construction time is 4.3 seconds on average. G-tree has a size of 6.14 MB, slightly smaller than FR-tree due to the different entry style of an object from FR-tree. At the leaf level of FR-tree, for each object, besides the bounding box at ρ_{max}, the covariance matrix has to be stored additionally in order to carry out our filtering techniques. In contrast, in G-tree the bounding box of an object is formed by its mean and standard deviation in each dimension. In addition to the bounding box, only the covariances between dimensions rather than the whole covariance matrix need to be stored. G-tree takes 59.7 seconds on average to build. Although G-tree needs more construction time, considering the superior query performance and the index construction can be done offline, the index construction is in an affordable manner.

6. Related Work

Uncertain Data Management. We focus on research work in the area of uncertain data management that is closely related to our work. A number of approaches for managing uncertain data have been proposed. Early research primarily focused on queries in a moving object database model [12]–[15]. The solutions to several types of probabilistic queries were proposed in [16], including probabilistic range queries, where their target is merely the one-dimensional case.

A range query processing method for the case where both data objects and query object are imprecise was proposed in [17]. But they assume that each object exists within a rectangular area. Zheng et al. [18] modeled a fuzzy object by a fuzzy set where each element is characterized by its probability of membership (the sum of all probabilities is not necessarily one). For efficient query processing, they proposed the notion of α-cut, the subset of elements whose probabilities are no less than a user-specified probability threshold α, to filter elements of the fuzzy object. Although we also exploit the idea of filtering region (ρ-region), their rationale of computing the α-cut is different from ours.

Gauss-tree [4] was proposed as an index structure for Gaussian distributions. It assumes all Gaussian distributions are probabilistically *independent* in each dimension. This imposes heavy restriction on the generality of the approach and the overall accuracy of the query result is limited. in [19], Lian et al. proposed a generic framework to tackle the local correlations among uncertain data.

Indexing Uncertain Data for Range Queries. Agarwal et al. [20] presented various indexing structures on uncertain data that support range queries in the one-dimensional case. Tao et al. proposed U-tree [3] to process probabilistic range queries in a multi-dimensional space, where uncertain objects are assumed to follow arbitrary probability distributions within uncertainty regions. Zhang et al. proposed a quadtree-based index called U-Quadtree [21] for range searching on multi-dimensional uncertain data. They mainly focused on representing uncertainty by discrete instances inside a minimum bounding box. The difference from our work is that we take advantage of specific properties of Gaussian distribution and index uncertain objects distributed in an infinite space.

Spatial Data Indexing. The traditional spatial database has been well studied and many indexing methods have been proposed [22]–[24] to support spatial query processing. R-tree [23] and its extension R*-tree [22], indexing objects by deriving their minimum bounding rectangles (MBRs), are two of the well-known ones. TPR-tree [6] and TPR*-tree [25] were proposed to index moving objects. However, none of these indexes can be applied directly on the Gaussian objects to support the queries studied in this paper.

7. Conclusion

In this paper, we study probabilistic range queries over uncertain data. We assume that the location of the query object is either fixed or follows a multi-dimensional Gaussian distribution. The locations of data objects are represented by Gaussian distributions. Given these assumptions, we define two types of probabilistic range queries with respect to the query object. We propose filtering techniques and a novel R-tree-based index structure to expedite query processing.

In the current implementation, the node split policy of G-tree follows that of R*-tree and the computation of the four penalty metrics (area, margin, overlap and centroid distance) used for splitting is based on that of TPR-tree. As the future work, the structure of G-tree can be optimized for efficient query processing by taking advantage of its features. Furthermore, we consider extending our approach to support other uncertainty models such as Gaussian Mixture Model and other types of queries such as probabilistic nearest neighbor queries.

Acknowledgments

This research is supported by the FIRST program, Japan and KAKENHI (23650047, 25280039).

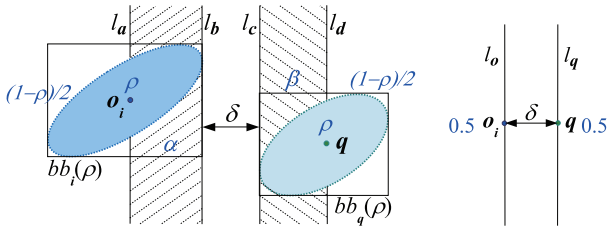
References

[1] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics, The MIT Press, 2005.
[2] R.O. Duda, P.E. Hart, and D.G. Stork, Pattern Classification, 2nd ed., Wiley, 2000.
[3] Y. Tao, R. Cheng, X. Xiao, W.K. Ngai, B. Kao, and S. Prabhakar, "Indexing multi-dimensional uncertain data with arbitrary probability density functions," Proc. VLDB, pp.922–933, 2005.
[4] C. Böhm, A. Pryakhin, and M. Schubert, "The Gauss-tree: Efficient

object identification in databases of probabilistic feature vectors," Proc. ICDE, pp.9:1–9:12, 2006.
[5] Y. Tao, X. Xiao, and R. Cheng, "Range search on multidimensional uncertain data," ACM TODS, vol.32, no.3, pp.15:1–15:54, 2007.
[6] S. Šaltenis, C.S. Jensen, S.T. Leutenegger, and M.A. Lopez, "Indexing the positions of continuously moving objects," Proc. ACM SIGMOD, pp.331–342, 2000.
[7] Y. Ishikawa, Y. Iijima, and J.X. Yu, "Spatial range querying for Gaussian-based imprecise query objects," Proc. ICDE, pp.676–687, 2009.
[8] K. Kodama, T. Dong, and Y. Ishikawa, "An index structure for spatial range querying on Gaussian distributions," Proc. Fifth International Workshop on Management of Uncertain Data (MUD 2011), pp.1–7, 2011.
[9] T. Dong, C. Xiao, X. Guo, and Y. Ishikawa, "Processing probabilistic range queries over Gaussian-based uncertain data," Proc. SSTD, pp.410–428, 2013.
[10] 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.
[11] M. Hadjieleftheriou, E. Hoel, and V.J. Tsotras, "Sail: A spatial index library for efficient application integration," GeoInformatica, vol.9, pp.367–389, 2005.
[12] R. Cheng, D.V. Kalashnikov, and S. Prabhakar, "Querying imprecise data in moving object environments," IEEE Trans. Knowl. Data Eng., vol.16, no.9, pp.1112–1127, 2004.
[13] D. Pfoser and C.S. Jensen, "Capturing the uncertainty of moving-object representations," Proc. 6th Intl. Symp. on Advances in Spatial Databases (SSD'99), pp.111–132, 1999.
[14] 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.
[15] 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.
[16] R. Cheng, D.V. Kalashnikov, and S. Prabhakar, "Evaluating probabilistic queries over imprecise data," Proc. ACM SIGMOD, pp.551–562, 2003.
[17] J. Chen and R. Cheng, "Efficient evaluation of imprecise location-dependent queries," Proc. ICDE, pp.586–595, 2007.
[18] K. Zheng, X. Zhou, P.C. Fung, and K. Xie, "Spatial query processing for fuzzy objects," VLDB Journal, vol.21, no.5, pp.729–751, 2012.
[19] X. Lian and L. Chen, "A generic framework for handling uncertain data with local correlations," PVLDB, vol.4, no.1, pp.12–21, 2010.
[20] P.K. Agarwal, S.W. Cheng, and K. Yi, "Range searching on uncertain data," ACM Trans. Algorithms, vol.8, no.4, pp.43:1–43:17, 2012.
[21] Y. Zhang, W. Zhang, Q. Lin, and X. Lin, "Effectively indexing the multi-dimensional uncertain objects for range searching," EDBT, pp.504–515, 2012.
[22] N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An efficient and robust access method for points and rectangles," Proc. ACM SIGMOD, pp.322–331, 1990.
[23] A. Guttman, "R-trees: A dynamic index structure for spatial searching," Proc. ACM SIGMOD, pp.47–57, 1984.
[24] Y. Manolopoulos, A. Nanopoulos, A.N. Papadopoulos, and Y. Theodoridis, R-Trees: Theory and Applications, Springer, 2005.
[25] Y. Tao, D. Papadias, and J. Sun, "The TPR*-tree: An optimized spatio-temporal access method for predictive queries," Proc. VLDB, pp.790–801, 2003.

Appendix A: Proof of Lemma 1

As shown in Fig. A· 1 (a), consider the case where the minimum distance between $bb_i(\rho)$ and $bb_q(\rho)$ is δ. The space



(a) $\theta < 0.75$ (b) $\theta \geq 0.75$

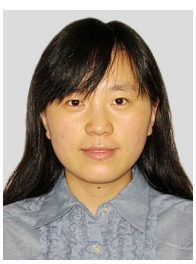
Fig. A· 1 Proof of maximal probability for PRQ-G.

outside the ρ-region of o_i (resp. q) is divided into two parts by the line l_a (resp. l_d) equally, both with an existence probability of $(1-\rho)/2$. Assuming the probability that o_i (resp. q) is located in the dashed area between l_a and l_b (resp. l_c and l_d) excluding the half part of ρ-region is α (resp. β), the probability that o_i lies within the left part of l_b and q lies within the left part of l_c is $((1-\rho)/2+\rho+\alpha)((1-\rho)/2-\beta)$. When o_i lies within the right part of l_b (q can be located both in the right and left part of l_c), the probability is $((1-\rho)/2-\alpha)$. Thus, the maximal probability of $\|\boldsymbol{x}_o-\boldsymbol{x}_q\| \leq \delta$ can be calculated by summing up the two probabilities, resulting in

$$\begin{aligned} &Pr(\|\boldsymbol{x}_o-\boldsymbol{x}_q\| \leq \delta) \\ &< ((1-\rho)/2+\rho+\alpha)((1-\rho)/2-\beta) \\ &+ ((1-\rho)/2-\alpha) \\ &= (3-2\rho-\rho^2)/4-(\alpha+\beta)(1+\rho)/2-\alpha\beta \\ &< (3-2\rho-\rho^2)/4. \end{aligned}$$

Appendix B: Proof of Lemma 2

Assume that the distance between the two mean locations is exactly δ as illustrated in Fig. A· 1 (b). o_i (resp. q) has a probability of 0.5 to be located in both left and right part of the line l_o (resp. l_q). $\|\boldsymbol{x}_o-\boldsymbol{x}_q\| \leq \delta$ happens in three cases: (1) Both o_i and q distribute in the left part of l_o and l_q. (2) Both o_i and q distribute in the right part of l_o and l_q. (3) o_i distribute in the right part of l_o and q distribute in the left part of l_q. Each case has a probability of $0.5*0.5$. Hence, the maximal probability of $\|\boldsymbol{x}_o-\boldsymbol{x}_q\| \leq \delta$ is $0.5*0.5*3 = 0.75$.



Tingting Dong is a PhD candidate at Graduate School of Information Science, Nagoya University. She received her MS degree in Information Science from Nagoya University in 2013, and both BS and BE degrees in Mathematics and Applied Mathematics and Software Engineering from Dalian Jiaotong University, China in 2010. Her research interests include uncertain data management, spatio-temporal databases, and sensor databases.

Chuan Xiao is a research associate in Graduate School of Information Science, Nagoya University. He received bachelor's degree from Northeastern University, China in 2005, and PhD degree from The University of New South Wales in 2010. His research interests include similarity search, textual databases, and graph databases.

Yoshiharu Ishikawa is a professor in Graduate School of Information Science, Nagoya University. His research interests include spatio-temporal databases, mobile databases, sensor databases, data mining, information retrieval, and Web information systems. He is a member of the Database Society of Japan, IPSJ, IEICE, JSAI, ACM, and IEEE.