R. Kotagiri, P. R. Krishna, M. Mohania, E. Natajeewarawat (Eds.):
In Proc. 12th International Conference on Database Systems
for Advanced Applications (DASFAA 2007), LNCS 4443, pp. 337–348, 2007.

Probabilistic Nearest-Neighbor Query on Uncertain Objects

Hans-Peter Kriegel, Peter Kunath, Matthias Renz
University of Munich, Germany
{kriegel, kunath, renz}@dbs.ifi.lmu.de

Abstract. Nearest-neighbor queries are an important query type for commonly used feature databases. In many different application areas, e.g. sensor databases, location based services or face recognition systems, distances between objects have to be computed based on vague and uncertain data. A successful approach is to express the distance between two uncertain objects by probability density functions which assign a probability value to each possible distance value. By integrating the complete probabilistic distance function as a whole directly into the query algorithm, the full information provided by these functions is exploited. The result of such a probabilistic query algorithm consists of tuples containing the result object and a probability value indicating the likelihood that the object satisfies the query predicate. In this paper we introduce an efficient strategy for processing probabilistic nearest-neighbor queries, as the computation of these probability values is very expensive. In a detailed experimental evaluation, we demonstrate the benefits of our probabilistic query approach. The experiments show that we can achieve high quality query results with rather low computational cost.

1 Introduction

In many modern application ranges, e.g. spatio-temporal query processing of moving objects [4], sensor databases [3] or personal identification systems [13], usually only uncertain data is available. In the area of multimedia databases, e.g. image or music databases, or in the area of personal identification systems based on face recognition and fingerprint analysis, there often exists the problem that a feature vector cannot exactly be determined. This "positional" uncertain data can be handled by assigning confidence intervals to the feature values, by specifying probability density functions indicating the likelihood of certain feature values, or by specifying confidence values for a set of discrete feature values. The advantage of the latter form of representation of uncertain data is that distances between the uncertain objects can be processed more easily than object distances based on smooth probability density functions. Furthermore, positional uncertainties of objects are often given in form of discrete values, in particular, if potential object locations are derived from different observations. Even when the uncertainty of the objects are specified by means of smooth probability density functions, we can achieve our preferred discrete data representation by means of sampling techniques. With this concept, we can find a good trade-off between accuracy and query performance.

The approach proposed in [9] which uses probabilistic distance functions to measure the similarity between uncertain objects seems very promising for probabilistic similarity queries, in particular for the probabilistic distance-range join. Contrary to traditional

approaches, they do not extract aggregated values from the probabilistic distance functions but enhance the join algorithms so that they can exploit the full information provided by these functions. The resulting probabilistic similarity join assigns a probability value to each object pair indicating the likelihood that the pair belongs to the result set, i.e. these probably values reflect the trustability of the result. In this paper, we adopt the idea to use probabilistic distance functions between positional uncertain objects in order to assign probability values to query results reflecting the trustability of the result. In applications where wrong results have fatal consequences, e.g. medical treatment, users might only look at very certain results, whereas in commercial advertising, for instance, all results might be interesting. Based on this concept, we propose a solution for probabilistic nearest neighbor queries which are practically very important in many application areas.

2 Related Work

In the last decade, a lot of work has been done in the field of similarity query processing with the focus on management and processing of uncertain data. Thereby, the development of efficient and effective approaches providing probabilistic query results were of main interest. A survey of the research area concerning uncertainty and incomplete information in databases is given in [1] and [11]. Recently a lot of work has been published in the area of management and query processing of uncertain data in sensor databases [3] and especially in moving object environments [4, 12]. Similar to the approach presented in this paper, the approaches in [2, 3, 4, 12] model uncertain data by means of probabilistic density functions (*pdf*s). In [12], for instance, moving objects send their new positions to the server, iff their new positions considerably vary from their last sent positions. Thus, the server always knows that an object can only be a certain threshold value away from the last sent position. The server, then, assigns a pdf to each object reflecting the likelihood of the objects possible positions. Based on this information the server performs probabilistic range queries. Likewise, in [4] an approach is presented for probabilistic nearest neighbor queries. Note that both approaches assume non-uncertain query objects, and thus, they cannot be used for queries where both query and database objects are uncertain. Queries that support uncertain database objects as well as uncertain query objects are very important as they build a foundation for probabilistic join procedures. Most recently, in [9] a probabilistic distance range join on uncertain objects was proposed. Instead of applying their join computations directly on the pdfs describing the uncertain objects, they used sample points as uncertain object descriptions for the computation of the probabilistic join results.

Furthermore, most recently [5] an approach was proposed dealing with spatial query processing not on positionally uncertain data but on existentially uncertain data. This kind of data naturally occurs, if, for instance, objects are extracted from uncertain satellite images. The approach presented in this paper does not deal with existentially uncertain data but with positionally uncertain data which can be modelled by probability density functions or are already given as probabilistic set of discrete object positions similar to the approach presented in [9].

3 Probabilistic Nearest Neighbor Query on Uncertain Data

As already mentioned, a non-probabilistic similarity query on positional uncertain data has some limitations which are overcome by our probabilistic approach introduced in this section. It is based on a direct integration of the probabilistic distance functions rather than using only aggregated values. Our new query type assigns to each result object a probability value reflecting the likelihood that the object fulfills the query predicate.

Definition 1 (probabilistic similarity query)

Let q be an uncertain object and DB denote a database, and let θ_d denote any similarity query predicate based on a given distance function d. Furthermore, let $P(q\ \theta_d\ o)$ denote the probability that $q\ \theta_d\ o$ is true for the object pair $(q, o) \in q \times DB$. Then, the *probabilistic similarity query* Q_θ^{prob} consists of result pairs $(o, P(q\ \theta_d\ o)) \in DB \times [0,1]$ for which $P(q\ \theta_d\ o) > 0$ holds, i.e. $Q_\theta^{\text{prob}} = \{(o, P(q\ \theta_d\ o)) \mid P(q\ \theta_d\ o) > 0\} \subseteq DB \times [0,1]$

3.1 Probabilistic Nearest-Neighbor Query Based on Smooth Probabilistic Distance Functions

In this section, we shortly show how we can theoretically compute the probability value $P(q\ \theta_d^{nn}\ o)$ underlying the probabilistic nearest-neighbor query.

Lemma 1. For a given uncertain query object q each uncertain database object o, we can compute $P(q\ \theta_d^{nn} o)$ reflecting the probability that o is the nearest neighbor of q as follows:

$$P(q\ \theta_d^{nn} o) = \int_{IR^d} q(v) \cdot \left(\int_{-\infty}^{+\infty} f_d(\delta(\tau - v), o)(\tau) \cdot \prod_{x \in DB \setminus o} \left(1 - \int_{-\infty}^{\tau} f_d(\delta(\tau - v), x)(t)dt \right) d\tau \right) dv$$

Proof: First, we fix a certain position v for the uncertain object representation q. Then, we weigh the probabilistic distance function $f_d(\delta(\tau - v), o)(\tau)$ between our uncertain object o and our "certain" position v with a probability value P_{weight} indicating the likelihood that all database objects $x \in DB \setminus o$ have a distance higher than τ from v. Integrating, over all distance values τ yields the probability that o is the nearest neighbor of q under the condition that the position of q is equal to v. Finally, integrating over all possible positions of q yields the probability that o is the nearest neighbor of q. □

Note that we can extend Lemma 1 so that it can be used as foundation for the probabilistic nearest-neighbor query, by substituting the probability value P_{weight} by the following expression:

$$\sum_{\substack{A \subseteq DB \setminus o \\ |A| = k-1}} \prod_{\substack{y \in A \\ x \in (DB \setminus o) \setminus A}} \left(\int_{-\infty}^{\tau} f_d(\delta(\tau - v), y)(t)dt \right) \cdot \left(1 - \int_{-\infty}^{\tau} f_d(\delta(\tau - v), x)(t)dt \right)$$

3.2 Probabilistic Nearest-Neighbor Query Based on Discrete Probabilistic Distance Representations

Although for some uncertain object representations it would be possible to compute the probabilistic similarity queries directly on Lemma 1, we propose to compute them

based on the generally applicable concept of monte-carlo sampling. In many applications the uncertain objects might already be described by a discrete probability density function, i.e. we have the sample set already. If the uncertain object is described by a continuos probability density function, we can easily sample according to this function and derive a set of samples. In the following, we assume that each object o is represented by a set of s sample points, i.e. o is represented by s different representations $\{o_1, ..., o_s\}$. After having described how to organize these discrete object representations within a database (cf. Section 3.2.1), we show how to compute the probabilistic nearest-neighbor query (cf. Section 3.2.2) based on these discrete object representations.

3.2.1 Database Integration of Uncertain Data. In order to reduce the complexity of the query computation, we introduce an efficient query algorithm which is based on groups of samples. Thereby two samples o_i and o_j of the same object o are grouped together to one cluster, if they are close to each other. We can generate such a clustering on the object samples by applying the partitioning clustering algorithm k-means [10] individually to each sample set $\{o_1, ..., o_s\}$. Thus, an object is no longer approximated by s samples, but by k clusters containing all the s sample points of the object.

Definition 2 (clustered object representation)

Let $\{o_1, ..., o_s\}$ be a discrete object representation. Then, we call the set $\{\{o_{1,1}, ..., o_{1,n_1}\},..., \{o_{k,1}, ..., o_{k,n_k}\}\}$ a *clustered object representation* where $\bigcup_{i=1...k, j=1...n_i} o_{i,j} = \{o_1, ..., o_s\}$ and $n_1+...+n_k = s$.

Similar to [9], we store these clustered object representations in R-tree [6] like index structures.

3.2.2 Nearest-Neighbor Query Algorithm. A straightforward approach for an efficient probabilistic nearest-neighbor query is based on the *minimal maximum distance* d_{minmax} (cf. Definition 3). Based on this distance it is possible to exclude many database objects o from the probabilistic nearest-neighbor search of a query object q (cf. Lemma 2).

Definition 3 (minimal maximum object distance)

Let q be an uncertain query object. Then the *minimal maximum object distance* of q is computed by: $d_{minmax} = min\ \{maxdist\ (\mathrm{MBR}(q), \mathrm{MBR}(o))| o \in DB\}$

Lemma 2. Let q be an uncertain query object and DB be a set of uncertain objects. Then, the following statement holds:

$\forall o \in DB$: $mindist\ (\mathrm{MBR}(q), \mathrm{MBR}(o)) > d_{minmax} \Rightarrow P(q\ \theta_d^{nn}\ o) = 0$

Proof. Let $o' \in DB$ be the object in the database for which $maxdist\ (\mathrm{MBR}(q), \mathrm{MBR}(o')) = d_{minmax}$ holds. Then, for all sample points $q_{i,j}$, $o_{i,j}$, $o'_{i',j'}$ the following statement holds: $d(q_{i,j}, o_{i,j}) > d(q_{i,j}, o'_{i',j'})$. Therefore, the probability that o is the nearest neighbor of q is equal to 0. □

Based on the candidate sets $C = \{o \in DB \mid mindist\ (\mathrm{MBR}(q), \mathrm{MBR}(o)) \leq d_{minmax}\}$, we can introduce a straightforward approach which computes the probability value $p_{nn}(q, o)$ indicating the likelihood that the object $o = \{o_1, ..., o_s\}$ is the nearest neighbor of $q = \{q_1, ..., q_s\}$.

$p_{nn}(q_1, o_1) = (1\text{-}1/2)\cdot(1\text{-}2/2) = 0/4$
$p_{nn}(q_1, o_2) = (1\text{-}2/2)\cdot(1\text{-}2/2) = 0/4$
$p_{nn}(q_2, o_1) = (1\text{-}0/2)\cdot(1\text{-}0/2) = 4/4$
$p_{nn}(q_2, o_2) = (1\text{-}0/2)\cdot(1\text{-}1/2) = 2/4$

$p_{nn}(q, o) = (4/4 + 2/4)/4 = 6/16 = 37{,}5\ \%$

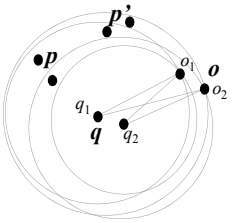


Fig. 1. Computation of nearest-neighbor probabilities (s=2).

Lemma 3. Let $\{o_1, ..., o_s\} \in C$. Then, the probability value $p_{nn}(q, o)$ indicating the likelihood that o is the nearest neighbor of q can be computed by:

$$p_{nn}(q, o) = \frac{\sum_{i,j \in 1\ldots s} p_{nn}(q_i, o_j)}{s^2},$$

where $p_{nn}(q_i, o_j)$ is equal to

$$\prod_{\substack{p \in C \\ p \neq q \wedge p \neq o}} \left(1 - \frac{\left|\{(q_i, p_l) \middle| d(q_i, p_l) < d(q_i, o_j) \wedge l \in 1\ldots s\}\right|}{s}\right)$$

Proof. First, we compute the probability $p_{nn}(q_i, o_j)$ that o_j is the closest sample to the sample q_i, by computing for each database object $p \in C$ the probability $P(p, q_i, o_j)$ that no sample of p is closer to the sample q_i than the sample o_j. Note that for objects $p \in DB \backslash C$ $P(p, q_i, o_j)$ is 1. The combination $\prod P(p, o_i, o_j)$ of these independent probability values yields the probability that the sample point o_j is the nearest sample point for the sample point q_i. The average of these s^2 many probability values $p_{nn}(q_i, o_j)$ is equal to $p_{nn}(q, o)$. □

In the following, show that the pruning distance for the uncertain query object q can further be decreased. The basic idea is that we do not use the minimal maximum object distance of q, i.e. d_{minmax}, but the minimal maximum distance of each single sample point.

Definition 4 (minimal maximum sample distance)

Let DB be a set of uncertain objects and let $q = \{\{q_{1,1}, ..., q_{1,n_1}\},..., \{q_{k,1}, ..., q_{k,n_k}\}\}$ be a clustered query object representation. Then, the *minimal maximum sample distance* of each sample point $q_{i,j}$ and the *minimal maximum cluster distance* of each cluster $C_i = \{q_{i,1}, ..., q_{i,n_i}\}$ are computed as follows:

$d_{minmax}(q_{i,j}) = min\ \{maxdist\ (q_{i,j}, \mathrm{MBR}(o)) |\ o \in DB\}$

$d_{minmax}(C_i) = min\ \{maxdist\ (\mathrm{MBR}(C_i), \mathrm{MBR}(o)) |\ o' \in DB\}$

Lemma 4. Let DB be a set of uncertain objects. Then, the following statement holds for an uncertain query object $q = \{\{q_{1,1}, ..., q_{1,n_1}\},..., \{q_{k,1}, ..., q_{k,n_k}\}\}$:

$\forall i \in 1..k\ \forall j \in 1..n_i : d_{minmax}(q_{i,j}) \leq d_{minmax}(C_i) \leq d_{minmax}$

In our final approach, we exploit the above lemma. Basically, our probabilistic nearest-neighbor query computes for the query object $q = \{\{q_{1,1}, ..., q_{1,n_1}\},..., \{q_{k,1}, ..., q_{k,n_k}\}\}$ the possible nearest neighbors in the set DB by carrying out the following two steps for each $o \in DB$, an example is depicted in Figure 1:

ALGORITHM 1. Probabilistic-Nearest-Neighbor Query.

INPUT: $q = \{\{q_{1,1}, ..., q_{1,n_1}\},...,\{q_{k,1}, ..., q_{k,n_k}\}\}$,
R-tree containing clustered uncertain objects from DB
OUTPUT:$(o,p_{nn}(q,o))$ for all objects $o \in DB$ where $p_{nn}(q,o) > 0$
BEGIN

```
1  FOR ALL i ∈ 1...k DO
2    FOR ALL j ∈ 1...n_i DO
3      LIST nnlist(q_{i,j}); // manages entries of the form (o, p_nn(q_{i,j},o), sample_cnt_o)
4  PriorityQueue queue; // sorted in ascending order according to the mindist value of the entries
5  queue.insert (mindist(MBR(q), MBR(R-tree.root)), (q, R-tree.root));
6  WHILE NOT (queue.isempty() OR ProbDoNotChange ({nnlist(q_{i,j}) | i ∈ 1...k ∧ j ∈ 1...n_i})) DO
7    Element first = queue.pop();
8    CASE type(first.db) // of what type is the R-tree element first.db?
9      DirNode, DataNode: // type of first.query is Object
10       FOR EACH element IN first.db DO {// element is a tree node
11         d = mindist (MBR(first.query), MBR(element));
12         queue.insert (d, (first.query, element)); }
13     Object: // type of first.query is also Object
14       IF SplitFurtherObject(first, queue) THEN
15         FOR EACH C_i(q) IN first.query DO
16           FOR EACH C_i'(o) IN first.db DO {
17             d = mindist (MBR(C_i(q)), MBR(C_i'(o));
18             queue.insert (d, (C_i(q), C_i'(o))); }
19       ELSE UpdateProbValues(first, {nnlist(q_{i,j}) | i ∈ 1..k ∧ j ∈ 1..n_i});
20     ObjectCluster: // type of first.query is also ObjectCluster
21       IF SplitFurtherCluster(first, queue) THEN
22         FOR EACH q_{i,j} IN first.query DO
23           FOR EACH o_{i',j'} IN first.db DO
24             queue.insert (dist(q_{i,j}, o_{i',j'}), (q_{i,j}, o_{i',j'}));
25       ELSE UpdateProbValues(first, {nnlist(q_{i,j}) | C_i(q) = first.query ∧ j ∈ 1..n_i});
26     ObjectSample: // type of first.query is also ObjectSample
27         UpdateProbValues(first, {nnlist(q_{i,j}) | q_{i,j} = first.query});
28   END;
29 END DO;
30 ReportResults ({nnlist(q_{i,j}) | i ∈ 1...k ∧ j ∈ 1...n_i});
END.
```

- First, we compute simultaneously for each sample point $q_{i,j}$ the probability $p_{nn}(q_{i,j}, o)$ that an object o is the nearest neighbor of the sample point $q_{i,j}$.
- Second, we combine the s probability values $p_{nn}(q_{i,j}, o)$ to an overall probability value $p_{nn}(q, o)$ which indicates the likelihood that the object o is the nearest neighbor of q.

The second task can be carried out straightforward based on the following lemma, whereas the first task is more complex and is explained in the remainder of this section.

Lemma 5. Let DB be a set of uncertain objects and let $q = \{\{q_{1,1}, ..., q_{1,n_1}\},..., \{q_{k,1}, ..., q_{k,n_k}\}\}$ be an uncertain query object. Then, the following statement holds.

$$\forall o \in DB: p_{nn}(q, o) = \frac{1}{s} \cdot \sum_{i = 1 \ldots k, j = 1 \ldots n_i} p_{nn}(q_{i,j}, o)$$

Thus, the remaining question is how to compute the values $p_{nn}(q_{i,j}, o)$ efficiently. The approach proposed in this paper can be regarded as an extension of the nearest-neighbor search algorithm presented in [7]. Contrary to [7], our approach deals with complex clustered uncertain object representations instead of simple feature vectors. Furthermore, we do not compute a distance ranking for the query object q but a probability value $p_{nn}(q_{i,j},o)$ to each sample point $q_{i,j}$ indicating the likelihood that object $o \in DB$ is nearest neighbor of $q_{i,j}$.

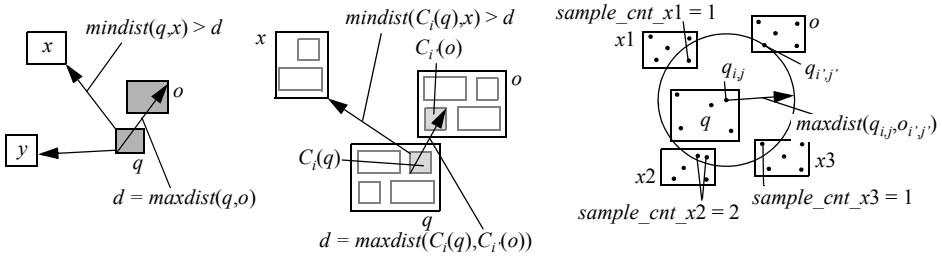
Algorithm 1 depicts our proposed *probabilistic nearest-neighbor query* algorithm. Like in the approach presented in [7], we use a priority queue *queue* as main data structure. In our case, the priority queue contains tuples *entry*=(d, (q, o)), where q is a part of the query object *entry.query*, o is a part of a database object *entry.db,* and d indicates the minimum distance between q and o. The distance values d determine the ordering of the priority queue. We have to store pairs of objects instead of simple objects because the query object itself consists of different parts, i.e. s sample objects $q_{i,j}$ and k clusters C_i (called $C_i(q)$ in the algorithm for clarity reasons). The priority queue is initialized with the pair (*mindist*(MBR(q), MBR(*Rtree.root*)), (q, *Rtree.root*)). We always take the first element from the priority queue and test of what type the stored elements are. Then we decide for the first element of the priority queue whether it must be further refined or whether we can already use this first element to change the probability values of the probabilistic nearest neighbors of the query sample points $q_{i,j}$. Three cases are distinguished (cf. Figure 2):

- Assume the elements contained in the first element *first* of the priority queue are complete uncertain objects q and o. Then we test whether there exists an entry (d, (p, p')) in *queue* for which the value d is smaller than *maxdist*(q, o), using the function *SplitFurtherObject*(*first*, *queue*). If this is the case, we split q and o into their cluster elements $C_i(q)$ and $C_{i'}(o)$ and store the k^2 many combinations of these clusters in *queue*. If there does not exist such an entry (d, (p, p')) (cf. Figure 2a), we update the lists *nnlist*($q_{i,j}$) which contain all information about the up-to-now found probabilistic nearest neighbors of the sample point $q_{i,j}$. In the function *UpdateProbValues* (*first*, {*nnlist*($q_{i,j}$) | $i \in 1...k \wedge j \in 1...n_i$}), the entries ($o$, $p_{nn}(q_{i,j},o)$, *sample_cnt_o*) are updated. The values $p_{nn}(q_{i,j},o)$ indicating the likelihood that o is the nearest neighbor of $q_{i,j}$ are set to (cf. Figure 2a):

$$\prod_{\substack{(x,\, p_{nn}(q_{i,j},\, x),\, sample\_cnt\_x) \in nnlist(q_{i,j}) \\ x \neq o}} \left(1 - \frac{sample\_cnt\_x}{s}\right)$$

  Furthermore, the values *sample_cnt_o* are set to s.

- Assume the elements contained in *first* are clusters, i.e. cluster $C_i(q)$ corresponds to the query object and cluster $C_{i'}(o)$ corresponds to the database object. Then, in the function *SplitFurtherCluster*(*first*, *queue*), we first test whether there exists an entry (d, (p, p')) in *queue* for which the value d is smaller than *maxdist*($C_i(q)$, $C_{i'}(o)$) and for which the following two conditions hold. First, p has to be equal to q, to $C_i(q)$, or to an object sample $q_{i,j}$. Second, p' must not be a part of o, i.e. another cluster of o or a sample point of o. If an entry (d, (p, p')) exists for which these conditions hold, we split $C_i(q)$ and $C_{i'}(o)$ in its sample points $q_{i,j}$ and $o_{i',j'}$ and store the $|C_i(q)| \cdot |C_{i'}(o)|$ many combinations of the sample points in *queue*. If there does not exist such an entry



a) case 1: object pair (q,o) does not have to be refined

b) case 2: cluster pair $(C_i(q),C_{i'}(o))$ does not have be refined

c) case 3: update of the value $p_{nn}(q_{i,j},o)$ $p_{nn}(q_{i,j},o) = p_{nn}(q_{i,j},o) + 1/5 \cdot (4/5 \cdot 3/5 \cdot 4/5)$

Fig. 2. Three cases of the probabilistic nearest-neighbor query algorithm.

$(d, (p, p'))$, we update the lists $nnlist(q_{i,j})$ (cf. Figure 2b). In the function *UpdateProb-Values* (*first*, $\{nnlist(q_{i,j}) \mid j \in 1...n_i\}$), the entries $(o, p_{nn}(q_{i,j},o), sample\_cnt\_o)$ are updated. The values $p_{nn}(q_{i,j},o)$ indicating the likelihood that o is the nearest neighbor of $q_{i,j}$ are set to:

$$p_{nn}(q_{i,j}, o) + \frac{|C_{i'}(o)|}{s} \cdot \prod_{\substack{(x,\, p_{nn}(q_{i,j},\, x),\, sample\_cnt\_x) \in nnlist(q_{i,j}) \\ x \neq o}} \left(1 - \frac{sample\_cnt\_x}{s}\right)$$

Furthermore, the values *sample_cnt_o* are set to $sample\_cnt\_o + |C_{i'}(o)|$.

- Assume the elements in *first* are sample points, i.e. $q_{i,j}$ is the query object and $o_{i',j'}$ is the database object. Then, we call the function *UpdateProbValues* (*first*, $\{nnlist(q_{i,j})\}$) which updates the entries $(o, p_{nn}(q_{i,j},o), sample\_cnt\_o)$. The values $p_{nn}(q_{i,j},o)$ indicating the likelihood that o is the nearest neighbor of $q_{i,j}$ are modified as follows (cf. Figure 2c):

$$p_{nn}(q_{i,j}, o) + \frac{1}{s} \cdot \prod_{\substack{(x,\, p_{nn}(q_{i,j},\, x),\, sample\_cnt\_x) \in nnlist(q_{i,j}) \\ x \neq o}} \left(1 - \frac{sample\_cnt\_x}{s}\right)$$

Furthermore, the values *sample_cnt_o* are set to *sample_cnt_o* + 1.

The algorithm terminates, if either the priority queue is empty or if in all s lists $nnlist(q_{i,j})$ there exists an entry $(o, p_{nn}(q_{i,j},o), sample\_cnt\_o)$ for which $sample\_cnt\_o = s$ holds. If this is the case, the probability values of all elements in the database do not change anymore. Thus, we can stop processing any further elements from *queue.* After the algorithm terminates, the values $p_{nn}(q_{i,j}, o)$ contained in the lists $nnlist(q_{i,j})$ indicate the probability that o is the nearest neighbor of $q_{i,j}$. Finally, in accordance with Lemma 5, the probability values that o is the nearest neighbor of q are computed in the function *ReportResults* ($\{nnlist(q_{i,j}) \mid i \in 1...k \wedge j \in 1...n_i\}$).

4 Experimental Evaluation

In this section, we examine the effectiveness, i.e. the quality, and the efficiency of our proposed probabilistic nearest-neighbor query approach. The efficiency of our approach

was measured by the average number of required distance computations per query object which dominate the overall runtime cost.

The following experiments are based on the same datasets as used in [9]. We used artificial datasets, each consisting of a set of 3- and 10-dimensional uncertain feature vectors. Additionally, we also applied our approaches to two distributed real-world datasets PLANE and PDB where the feature vectors were described by multi-dimensional boxes according to [8]. The following table summarizes the characteristics of the datasets:

Table 1. Characteristics of the datasets.

dataset	ART3 (low)	ART3 (high)	ART10 (low)	ART10 (high)	PLANE	PDB
dimensions *d*	3	3	10	10	42	120
uncertainty *u*	3%	5%	3%	4%	1%	4%

For the sampling of the possible object positions we assumed an equal distribution within the corresponding uncertainty areas. All *d*-dimensional datasets are normalized w.r.t. the unit space $[0,1]^d$. As distance measure we used the L_1-distance (Manhattan distance). We split all datasets into two sets containing 90% respectively 10% of all objects. For the nearest neighbor queries, we used the objects from the smaller set as query objects and summarized the results. If not stated otherwise, the size of the sample set of each uncertain object is initially set to 25 samples which are approximated by 7 clusters.

4.1 Experiments on the Sample Rate

First, we turned our attention to the quality of our probabilistic nn-query approach by varying the number of used samples per object. We noticed that for sample rates higher than 100 the resulting probability values do not change any more considerably. Therefore, we used the probabilistic nn-query result $R_{exact} = \{(o, P_{exact}(q\ \theta_d\ o))|\ P_{exact}(q\ \theta_d\ o) > 0\}$ (cf. Definition 1) based on 100 samples as reference query result for measuring the error of the probabilistic nn-query results $R_{approx} = \{(o, P_{approx}(q\ \theta_d\ o))\ |\ P_{approx}(q\ \theta_d\ o) > 0\}$ based on sample rates $s < 100$. The used error measure Err_{nn} for the nearest-neighbor query is defined as follows:

$$Err_{nn}(R_{approx}, R_{exact}) = \sum_{q \in Q}\left(\frac{1}{2} \cdot \sum_{(q,o) \in Q \times DB}\left|P_{approx}(q\ \theta_d^{nn}\ o) - P_{exact}(q\ \theta_d^{nn}\ o)\right|\right)$$

Figure 3a shows the error of the probabilistic nearest-neighbor query for a varying sample rate *s*. It can be clearly seen that the error decreases rapidly with increasing sample rate *s*. At a sample rate $s = 10$ the error is less than half the size compared to the error at $s = 1$ for some datasets. Furthermore, comparing the artificial datasets with high uncertainties (ART*d*(*high*)) to those with low uncertainties (ART*d*(*low*)), we can observe that a higher uncertainty leads to a higher error.

In the next experiment, we investigated how the sample rate influences the cost of the query processing. Figure 3b shows the number of distance computations required to per-

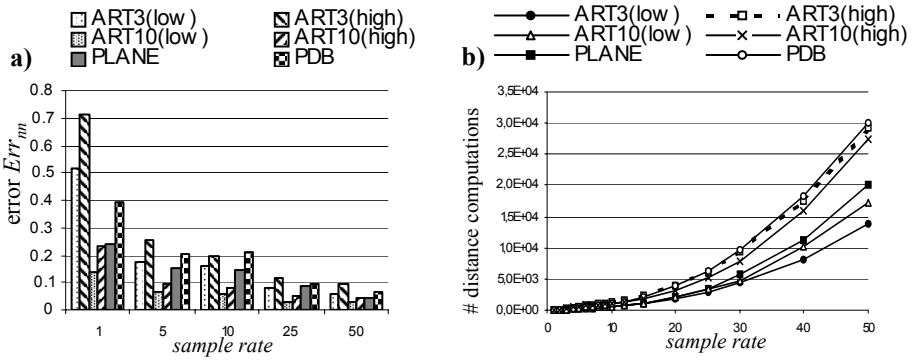


Fig. 3. Influence of the sample rate**. a)** error Err_{nn}, **b)** number of distance computations

form the nn-query for varying sample rates. We set the number k of clusters to 5 for a sample rate s higher than 5, otherwise we set $k = s$. The cost increase superlinear with increasing sample rates s. For high sample rates, the good quality (cf. Figure 3a) goes along with high query cost (cf. Figure 3b). In particular, the query processing on datasets with high uncertainty (ARTd(*high*)) does not only lead to a lower quality of the results but is also more expensive than the processing on more accurate datasets (ARTd(*low*)). In the case of very uncertain datasets the computational cost are higher because the pruning distances, i.e. the *minimal maximum object distance*s (cf. Definition 3), for very uncertain objects are much higher than for non-uncertain objects. Altogether, we achieve a good trade-off between the quality of the results and the required cost when using a sample rate of $s = 25$.

4.2 Experiments on the Efficiency

Next, we examine the runtime performance of our probabilistic nearest-neighbor query approach. Figure 4 shows how the runtime performance depends on the number k of sample clusters. On the one hand, when using only one cluster per object ($k = 1$), we have only a few clusters for which we must compute the distances between them. This is due to the fact that the cluster covers the entire uncertain object, i.e. it has a large extension.

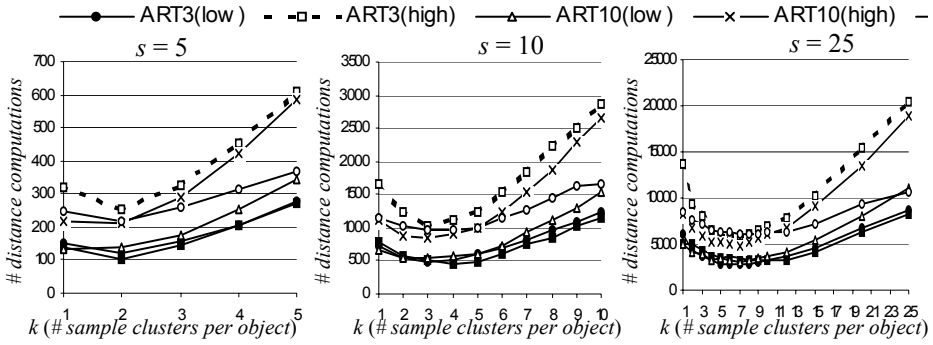


Fig. 4. Runtime performance for varying number of sample clusters.

On the other hand, very small clusters ($k = s$) also lead to an expensive query processing, because we have to compute a lot of distances between pairs of clusters when refining the object pairs. The best trade-off for k can be achieved somewhere in between these two extremes. As depicted in Figure 4, the optimal setting for k depends on the used sample rate. Generally, the higher the used sample rate s, the higher is the optimal value for k. Note that the *maxdist* values of the cluster pairs are very high when using $k = 1$ sample clusters. In this case, we often have to investigate the corresponding sample points of the clusters which leads to a high number of distance computations. Table 2 shows the ratio between the cost required for $k = 7$ and $k = 1$ for the probabilistic nearest-neighbor query (θ_d^{nn}) ($s = 25$). We can conclude that the clustering of the object samples pays off when using an adequate choice of the parameter k.

Table 2. Cost ratio between $k = 7$ and $k = 1$ ($s = 25$).

datasets	ART3 (low)	ART3 (high)	ART10 (low)	ART10 (high)	PLANE	PDB
θ_d^{nn}	0.46	0.43	0.61	0.60	0.64	0.71

In the last experiment, we compare our efficient probabilistic nearest-neighbor query approach (*accelerated approach*) as presented in Algorithm 1 to the straightforward solution (*simple approach*) which takes for the pruning of candidate pairs solely the minimal maximum object distance into account (cf. Definition 3 and Lemma 3). Figure 5 depicts the results for the query processing on different datasets and varying sample rates. Figure 5a shows that we achieve a very significant reduction of the query cost using the pruning techniques of our probabilistic nearest-neighbor query algorithm independent of the used sample rate. For the ART10(*high*) dataset the cost were reduced to 20% and for both real-world datasets we even achieved a reduction to 15%. Figure 5b compares the performance of both approaches using the artificial datasets for varying uncertainties of the objects. This experiment shows that the simple approach is not applicable for high uncertainties due to the enormous number of required distance computations. Contrary, our accelerated approach is not very sensitive to the uncertainty of the objects and shows good performance even for very imprecise data.

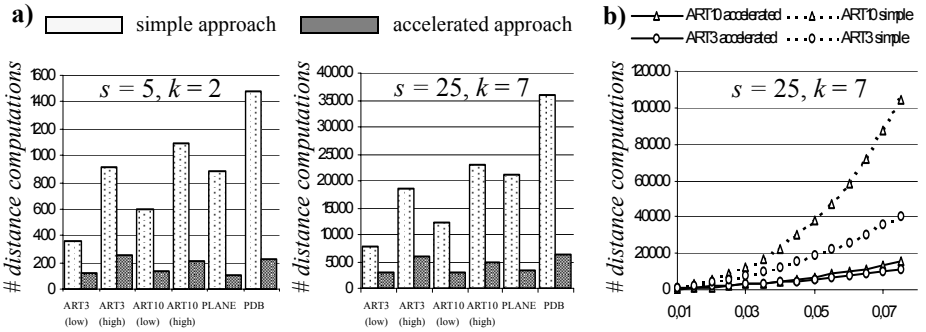


Fig. 5. Runtime performance for different pruning techniques.

5 Conclusions

Probabilistic query processing on uncertain data is an important emerging topic in many modern database application areas. In this paper, we introduced an approach for computing probabilistic nearest-neighbor queries on uncertain objects which assigns to each object a probability value indicating the likelihood that it belongs to the result set. We showed how this probabilistic query can effectively be carried out based on the generally applicable concept of monte-carlo sampling, i.e. each uncertain object is described by a set of sample points. In order to improve the query performance, we determined appropriate approximations of the object samples by means of clustering. Minimal bounding boxes of these clusters, which can be efficiently managed by spatial index structures, are then used to identify and skip unnecessary distance computations in a filter step. In a detailed experimental evaluation based on artificial and real-world data sets, we showed that our technique yields a performance gain of a factor of up to 6 over a straightforward comparison partner.

In our future work, we plan to extend our probabilistic algorithms to join processing, which built a foundation for various data mining algorithms, e.g. clustering and classification on uncertain data.

References

[1] Abiteboul S., Hull R., Vianu V.: *Foundations of Databases.* Addison Wesley, 1995.

[2] Böhm, C., Pryakhin A., Schubert M.: *The Gaus-Tree: Efficient Object Identification of Probabilistic Feature Vectors.* ICDE'06.

[3] Cheng R., Kalashnikov D.V., Prabhakar S.: *Evaluating probabilistic queries over imprecise data.* SIGMOD'03.

[4] Cheng R., Kalashnikov D. V., Prabhakar S.: *Querying imprecise data in moving object environments.* IEEE Transactions on Knowledge and Data Engineering, 2004.

[5] Dai X., Yiu M., Mamoulis N., Tao Y., Vaitis M.: *Probabilistic Spatial Queries on Existentially Uncertain Data.* SSTD'05.

[6] Guttman A.: *R-trees: A Dynamic Index Structure for Spatial Searching.* SIGMOD'84.

[7] Hjaltason G. R., Samet H.: *Ranking in Spatial Databases.* SSD'95.

[8] Kriegel H.-P., Kunath P., Pfeifle M., Renz M.: *Approximated Clustering of Distributed High Dimensional Data.* PAKDD'05.

[9] Kriegel H.-P., Kunath P., Pfeifle M., Renz M.: Probabilistic Similarity Join on Uncertain Data. DASFAA'06.

[10] McQueen J.: *Some Methods for Classification and Analysis of Multivariate Observations*. In 5th Berkeley Symp. Math. Statist. Prob., volume 1, 1967.

[11] Motro A.: *Management of Uncertainty in Database Systems.* In Modern Database Systems, Won Kim (Ed.), Addison Wesley, 1995.

[12] Wolfson O., Sistla A. P. , Chamberlain S., Yesha Y.: *Updating and Querying Databases that Track Mobile Units.* Distributed and Parallel Databases, 7(3), 1999.

[13] Zhao W., Chellappa R., Phillips P.J., Rosenfeld A.: *Face Recognition: A literature survey.* ACM Computational Survey, 35(4), 2000.