

Indexing the Imprecise Positions of Moving Objects

Xiaofeng Ding

Department of Computer Science
Huazhong University of Science and
Technology
Wuhan, China

dxf@smail.hust.edu.cn

Yansheng Lu

Department of Computer Science
Huazhong University of Science and
Technology
Wuhan, China

lys@mail.hust.edu.cn

ABSTRACT

In moving object environments, it is rarely possible for the database to record the exact positions of moving objects at all times. Very often, the database does not have the objects' precise positions due to update delay, sampling error, or limitation of measuring equipments. Information retrieval directly based on uncertain data is meaningless, since the result does not have any quality guarantee. Current database management systems with the exception of ORION do not provide a convenient means for representing or manipulating this type of uncertainty. In this paper, along with the concept of qualification probability and probability density function about moving objects, we introduce a general framework of monitoring and indexing moving objects with uncertainty region. Our goal is to quantify the query answer validity by efficiently computing the qualification probability of an object for satisfying kinds of queries.

Categories and Subject Descriptors

H.2.4 [Database Management]: systems

General Terms

Algorithms, Performance

Keywords

Moving objects, imprecise data, uncertainty management, range queries, location-based services

1. INTRODUCTION

In recent years, positioning technologies like the Global Positioning Systems, GSM, and the wireless LAN have received rapid development [1]. These techniques enable a new class of applications for continuous monitoring or tracking of mobile objects as they move in space [2]. Due to limitations of bandwidth and the battery power of the mobile devices, it is infeasible for the database to contain the exact position of each object at each point in time. For example, if there is a time delay between the capture of the location and its receipt at the database, the location values

received by the object may be different from the actual location values. An inherent property of these applications is that object locations cannot be updated continuously. Following an update, the position of the object is unknown until the next update is received. Under these conditions, the data in the database is only an estimate of the actual location at most points in time—an object can be anywhere in an uncertainty region (the grey area in Figure 1). This inherent uncertainty affects the accuracy of the answers to queries.

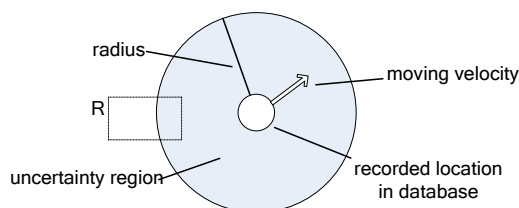


Figure 1. An example of uncertain moving object.

Due to the inherent uncertainty in the data, providing meaningful answers seems impossible, since the result does not have any quality guarantee. Consider the query “find the cabs currently in the downtown area”. Returning the cabs whose last updates satisfy the query is inadequate, because many cabs may have entered or left the search region since they contacted the server last time. Moreover, due to the lack of support for storing such uncertainty, many applications have to construct their own complex data models for managing the uncertainty. Similar issues exist in the location-based service applications and the domain of integrating unstructured text information with structured databases, such as the customer relationship management (CRM) databases [13], and email search databases.

According to the above situations, it is reasonable to use a location uncertainty model to describe the imprecision of the data value, and evaluate the location uncertainty in order to provide probabilistic guarantee over the validity of the query answers. The notion of probabilistic answers to queries over uncertain data was introduced in [3] for range queries, where the answer consists of objects along with the probability of each object lying in the query range. A common model for characterizing location uncertainty of an object is a closed region together with a probability distribution of the object in the region [4, 5]. Some previous work, such as [4, 5, 7], used this model to compute probabilities of each location object for satisfying a query, including the range and nearest-neighbor queries. The probability values provide confidence guarantees about the query answer, and allow quality of service metrics to be defined [6, 9].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Proceedings of SIGMOD2007 Ph.D. Workshop on Innovative Database Research 2007(IDAR2007), June 10, 2007, Beijing, China.

In summary, there are many applications for which the data exhibits uncertainty in both tuple and attribute values. Several researchers have recently proposed the development of uncertain relational data models to manage uncertain data [14, 15, 16]. The ORION project [15, 22] is a recent effort aimed at providing native support for uncertain data. The current version of the system, which is developed based on PostgreSQL, supports the storage and querying of uncertain attributes. In particular, each data value in ORION is represented as an interval and a probability distribution function, and it can be processed with probabilistic query operators to produce probabilistic answers. Traditional database index structures are not directly applicable for uncertain data. Much of the recent interest in uncertain data management has focused on the development of models for representing uncertainty and query processing semantics [14, 17]. Indexing support for uncertain data has only been developed for real-valued attributes of stationary objects. These index structures are inapplicable for uncertain moving objects.

This paper addresses the problem of managing uncertain data in a constantly-evolving environment. We proposed data models to capture temporal and spatial uncertainty, where a data item is represented as an interval with a probability distribution function instead of a single value. These models can be applied to a wide range of sensor-based applications, and can be extended to multiple dimensions. To the best of our knowledge, all existing work considers only uncertain stationary objects and the uncertainty of moving objects has not been studied before.

2. RELATED WORK

There has been a great deal of work on the development of models for representing uncertainty in databases [18]. In section 2.1, we first review the existing techniques for query processing in uncertain database. Then, section 2.2 introduced the TPR*-tree, which is an effective multi-dimensional access method for moving objects. Finally, section 2.3 discussed the velocity constrained indexing method, which is fundamental to our solutions.

2.1 Imprecise queries

Various data models for accurately capturing the locations of moving objects have been widely studied before. These models require update whenever an object's velocity changes and the corresponding query algorithms aim at minimizing the amount of data transmission to ensure the precision of database values. Cheng et al. [6] are the first to use probability values to provide confidence guarantees about the answer set. They present an interesting taxonomy of novel query types, together with the evaluation and quality of different probabilistic queries.

Cheng et al. [8] develop several efficient strategies for probabilistic range queries in one-dimensional data space. Their discussions are limited to range queries and they show that range query in uncertain databases is inherently more difficult than that on traditional precise objects, and support their conclusions by providing two theoretical approaches that achieve asymptotically optimal performance. Moreover, an I/O efficient algorithm for nearest neighbor search is proposed in [7, 11].

Tao et al. [10] present the U-tree, a multi-dimensional access method on uncertain data with arbitrary probability density functions. This index structure tries to minimize the amount of

appearance probability computation in probability range search. Actually, it achieves this by pre-computing some "auxiliary information" for each object, which can be used to prune the non-qualifying object or to validate it as a result without calculating its accurate appearance probability. Such information is maintained thoroughly through the tree structure to avoid accessing the subtrees that do not contain any query results. Furthermore, the U-tree is well organized for that the objects can be inserted or deleted easily.

Another related area deals with probabilistic modeling of uncertainty which is the focus of [14]. The majority of this work has focused on tuple-uncertainty, which refers to the degree of likelihood that a given tuple is part of a relation. Uncertainty tuples constitute the "probabilistic databases", where each record is attached with an existential probability. For example, the results of a text search predicate can be considered to be a relation where the various tuples have some degree of match with the predicate. This match can be treated as a probabilistic value [14, 22]. Probabilistic database does not capture the type of uncertainty addressed in [10] where the concrete value of a given attribute within a tuple is uncertain (the topic of this paper), whereas the tuple is definitely part of the relation.

2.2 The TPR*-tree

The TPR*-tree [12] is regarded as a variance of the TPR-tree for multi-dimensional moving objects; it follows the general update methodology of TPR- (and R-) trees, but includes some enhanced heuristics. When an object o is inserted, the TPR*-tree first identifies the leaf N that will accommodate o with the choose path algorithm, which, instead of the greedy traversal of the R*- and TPR-trees, follows the path that leads to the minimization of the penalty metrics in a branch-and bound manner. If N is full, a set of entries, selected by pick worst algorithm, are removed from N and re-inserted. These objects are such that, their removal minimizes the minimum bounding rectangle (MBR) and velocity bounding rectangle (VBR) of the parent node. Any node that overflows during the re-insertion is split using sorting split, which decides the entry distribution by sorting all the spatial and velocity dimensions.

In the TPR*-tree, a moving object is represented with (i) an MBR that denotes its extent at last updated time, and (ii) a VBR that describes the velocity of the lower (upper) boundary of moving object along each dimension. Figure 2 shows a 2-dimensional example where 4 moving objects a, b, c, d that are grouped into nodes N_1, N_2 that become the entries of the root.

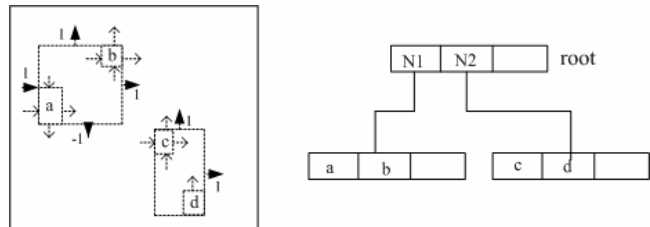


Figure 2. Entry representations in a TPR*-tree.

2.3 Velocity constrained indexing

The velocity constrained indexing (VCI) method [20], which is particularly suited for handling uncertainty of moving objects,

allows an index to be useful even when it does not reflect the accurate locations of moving objects. The velocity constrained index remains effective for large periods of time without updating the positions of objects, independent of the real movement of moving objects. For the case of VCI, the only restriction imposed on the moving objects is that they have a maximum velocity that they will not exceed. If the object wants to move faster than its current maximum speed, this maximum value can be changed.

The maximum speeds of all moving objects are used in the construction of the VCI. The VCI is an R-tree like index structure. It differs from the R-tree in that each node has an additional velocity field: V_{max} . This field stores the maximum speed over all objects covered by that node in the index. The V_{max} entry for an internal node is simply the maximum of the V_{max} entries of its children. The V_{max} entry for a leaf node is the maximum allowed speed among the objects that fall under the node. Construction is similar to the R-tree except that the velocity field is always adjusted to ensure that it is the largest value Figure 3 shows an example of a velocity constrained index structure. The V_{max} entry contained in each node is maintained in the way similar to the MBR of each entry in the node, except that there is only one V_{max} entry per node compared to several MBRs of the entries in the node. Upon the split of an old node, the V_{max} entry for each of the new nodes is simply copied from the old one.

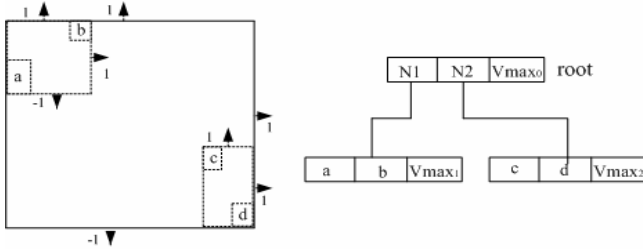


Figure 3. Entry representations in VCI.

It should be pointed out that, although the extension of MBRs in VCI is similar to the time varying MBRs proposed in TPR*-tree, the two are quite different in terms of indexing of moving objects. The key difference between the two is the model of object movement. In the TPR*-tree, the moving objects report their movements according to their velocities. In VCI, the movement of object is unreported as long as the maximum velocity is not exceeded. Thus, the VCI requires no updates to the index as objects move, but for query processing some more operations are necessary to take into account the actual movement of the moving objects.

3. PROBLEM DEFINITION

In this section we describe the general uncertainty model of moving objects, and definitions of imprecise location-dependent queries studied in this paper.

3.1 Uncertainty Model

Formally, an “uncertain moving object” O is associated with (i) a d -dimensional uncertainty region $O.ur$, (ii) a moving velocity $O.v$, and (iii) a probability density function $O.pdf(x)$, where x is an arbitrary d -dimensional point:

Definition 1 The uncertainty region of O_i at time t , denoted by $U_i(t)$, is a closed region where O_i is located.

Definition 2 The moving velocity of O_i at time t , denoted by $V_i(t)$, is a d -dimensional vector which decides the moving direction.

Definition 3 The uncertainty probability density function (pdf) of object O_i , denoted by $pdf_i(x, t)$, is a pdf of O_i 's location x at time t , that has a value of 0 outside $U_i(t)$.

Since $pdf_i(x, t)$ is a probability density function, it has the property that:

$$\int_{U_i(t)} pdf_i(x, t) dx = 1 \quad (1)$$

We do not know the exact probability density function of an object inside the uncertainty region. The formula for the uncertain pdf is different for kinds of applications. The only restriction for this function is that its value is 0 outside the uncertainty region. The problem would be much easier if the probability density function is specified. Wolfson et al. propose that the object location follows the Gaussian distribution inside the uncertainty region [5]. An important case of uncertainty probability density function is a uniform distribution [4]; essentially, this indicates a scenario where we have no idea of which point in the uncertainty region possesses a higher probability. We are planning to make the solutions applicable to any form of uncertainty probability density functions.

3.2 Imprecise Location-Dependent Queries

Based on the above model, different types of location-related queries, such as range queries and nearest-neighbor queries, can be issued. The imprecision of location values imply that some objects may satisfy a query. Therefore, it is natural to assign a probability value to each object that satisfies the query result. In [3], a probabilistic method for capturing the uncertainty information in range queries is presented. Each query returns a set of tuples in the form (O_i, P_i) where O_i is the object and P_i is the qualification probability that O_i is in the “range query region” specified by the user. Only the tuples where P_i is greater than some minimum threshold are returned.

Given a range query region R and a set of n moving objects O_i ($0 < i \leq n$), the definition of an imprecise range query can be generalized as follows:

Definition 4 An Imprecise Range Query (IRQ) returns a set of tuples in the form of (O_i, P_i) where P_i is the nonzero qualification probability that O_i is located inside R at a specific time t .

Sometimes the user is more likely to be interested in the answers with sufficiently high probability values. In fact, it is useful to define a probability threshold constraint, which restricts queries to return answers with probability values higher than a certain pre-defined value. We call this parameter the probability threshold (P_t), which is a real value between 0 and 1. The following query can then be defined:

Definition 5 A Constrained Imprecise Range Query (CIRQ) returns a set of tuples in the form of (O_i, P_i) such that $P_i \geq P_t$, where P_i is the qualification probability that O_i is located inside R at a specific time t .

It should be noted that, the answer to the same probabilistic query executed at two different time instants over the same database can be different even if no updates are received by the

database during the time intervals. This is reasonable because the uncertainty regions of moving objects may change over time.

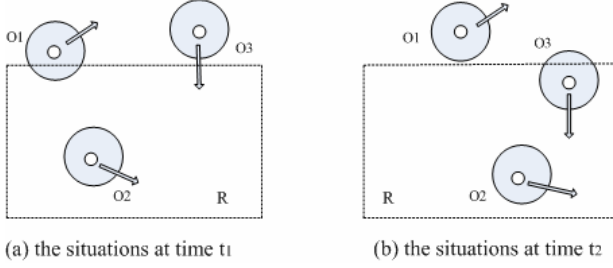


Figure 4. An example of imprecise range query.

As in Figure 4, moving objects O_1 , O_2 , O_3 , each with different uncertainty regions, are being queried. Assume that each object has an even chance of being located in its uncertainty region. An imprecise range query with query region R is invoked at time t_1 . Object O_2 is inside the rectangle R , so its qualification probability value is 1. Object O_3 is outside the rectangle, thus it has no chance of being located inside R at time t_1 , so the result of the IRQ at time t_1 is: $\{(O_2, 1), (O_1, 0.4)\}$. In the same example, if a IRQ is issued at time t_2 , as shown in Fig.3 (b). Object O_2 is always inside the rectangle R , so its qualification probability value is still 1. Object O_1 is moving outside the rectangle, thus it has no chance of being located inside R at time t_2 , while Object O_3 is moving inside the rectangle, so the result of the IRQ at time t_2 is: $\{(O_2, 1), (O_3, 0.7)\}$.

4. APPLIED METHODOLOGY

In this section, we first examine the basic solutions for evaluating IRQ and CIRQ; they form the foundation for further discussion. Then our proposed preliminary methods and other issues to be worked are discussed.

4.1 Basic Methods

For an IRQ is a special case of CIRQ, we will only examine how CIRQ can be answered under the uncertainty model described in section 3.

Given a CIRQ with a rectangle R and a probability threshold $P_t \in [0, 1]$, we examine the qualification probability P_i that an uncertain moving object satisfies the query. This is calculated by integrating the uncertainty probability density functions of O_i in the overlapping area of U_i and R , i.e.,

$$P_i = \int_{U_i(t) \cap R} p df_i(x, t) dx \quad (2)$$

where $U_i(t) \cap R$ denotes the intersection of $U_i(t)$ and R . Object O_i is a result if $P_i \geq P_t$ at the issued time t .

In practice, equation 2 is costly to implement especially for the unknown probability density functions of moving objects. The problem would be much easier if all the probability density functions were known to be of the same type. For example, if only Gaussian functions were present, specialized methods could be developed based on their means and variances [8]. These methods, however, are not useful for other types of pdfs, which in turn require specified solutions based on their own characteristics. Actually, any unified solutions that could handle probabilistic

queries regardless of their own probability density functions should be considered [10].

There are extra challenges for handling multi-dimensional moving objects. One difficulty in handling multi-dimensional data is that the integral in Equation 2 cannot be solved accurately even for a regular pdf such as Gaussian. To see this, assume that in Figure 1, the object's actual location is described using a Gaussian pdf whose mean falls at the center of the circle (i.e., the uncertainty region). Given an arbitrary query area R , the intersection between R and the uncertainty region has a shape that is not symmetric with respect to the mean. In this case, Equation 2 cannot be derived into a formula without any integrals, and hence, must be evaluated numerically through, for example, the monte-carlo approach [19]. However, monte-carlo is accurate only if the number of sampling points is sufficiently large, and the appropriate number increases with the dimensionality. Tao et al. [10] argued that the monte-carlo method incurs expensive costs, especially when the dimensionality d is high.

4.2 Efficient Solutions

From the last sections we can see that the straightforward calculation of probabilistic for CIRQ can lead to expensive operations. In this section we study how query performance can be improved by only allowing objects with qualification probabilities higher than a pre-defined value to be returned. Our objective is to minimize the cost (including both I/O and CUP time) of CIRQ. The main idea is to use pre-computed uncertain regions for moving objects, based on their moving velocities, in order to achieve better pruning effects. For a further performance improvement, we plan to employ the concept of the p -bound [8, 10, 21], which will be described later.

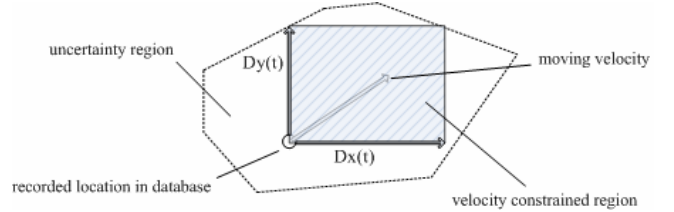


Figure 5. An example of 2D velocity constrained region.

The pre-computed uncertain region for a moving object is a velocity constrained region (VCR). The $VCR_i(t)$ of a moving object O_i takes a parameter V who represents the velocity of object O_i . Figure 5 illustrates a 2-dimensional example, where the polygon represents the uncertainty region of O . The $VCR_i(t)$ is an evolving rectangle (the hatched region in Figure 5) decided by two values $|Dx(t)|$, $|Dy(t)|$ that satisfy:

$$\begin{cases} Dx(t) = V_x(t_0)(t - t_0) \\ Dy(t) = V_y(t_0)(t - t_0) \end{cases} \quad (3)$$

That is to say, the VCR of moving objects could be fixed or may be determined by the velocities of the objects, the last reported location, and time since the last update. We do not know the exact probability density function of an object inside the velocity constrained region. The formula for the uncertain probability density function is application-specific. Further more, we assume that an object has the same chance of being located

anywhere within VCR, i.e., the probability density function $pdfi(x, t)$ of $VCR_i(t)$ is a bounded uniform distribution:

$$pdfi(x, t) = \begin{cases} 1/VCR_i(t) & \text{if } x \in VCR_i(t) \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

As pointed out in [7], the bounded uniform distribution is important in situations when we have no information about the distribution of the object within the velocity constrained region—the worst-case uncertainty. The best guess is then that the object has the same chance of being located in every point in the velocity constrained region. This is also the case when probabilistic queries are the most useful—when querying old data under a high level of uncertainty is prone to error. Due to its simplicity and practicability, we will illustrate how to evaluate CIRQ using uniform distribution in this section. We note that, however, it is possible to extend our generic solution to other kinds of distributions, such as the normal distribution for dead-reckoning policies [3].

Answering a CIRQ for our proposed VCR is easy. Assume that a CIRQ is evaluated at time t . First, notice that the probability density function $pdfi(x, t)$ of the uncertainty region $VCR_i(t)$ is uniform and, so, O_i has an equal opportunity of being located anywhere in $VCR_i(t)$. Thus, P_i is simply equal to the fraction of $VCR_i(t)$ that overlaps the query region R :

$$P_i = \frac{VCR_i(t) \cap R}{VCR_i(t)} \quad (5)$$

In particular, we can use the concept of p -bound to further improve the query performance. The idea of using p -bound to facilitate the pruning of object or validating it as a result was first proposed in [8] to answer probabilistic threshold range queries. The p -bound of an uncertain moving object o takes a parameter p whose value is between $[0, 0.5]$. It is composed of 2d line segments for a d -dimensional moving object. Figure 6 illustrates a 2-dimensional example, where the outside rectangle represents the uncertainty velocity constrained region of o and the hatched region represents the p -bound rectangle. The requirement of $L1(p)$ is that the probability of location of o on the left of $L1(p)$ has to be exactly equal to p . Similarly, $U1(p)$ is such that the appearance probability of o on the right of $U1(p)$ equals p . It is obviously that the appearance probability that o lies between $L1(p)$ and $U1(p)$ equals to $1 - 2p$. The remaining line segments $L2(p)$ and $U2(p)$ are obtained in the same way.

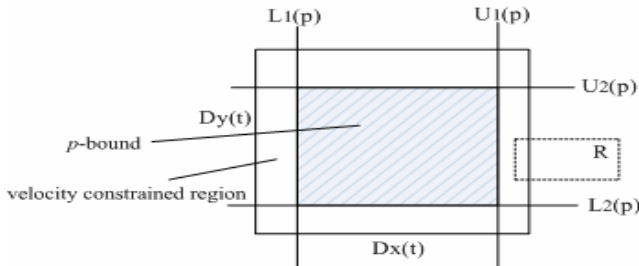


Figure 6. An example of p -bound in 2D.

The major purpose of the p -bound is to facilitate pruning for probabilistic range queries. If the p -bound has been pre-computed and maintained for any value p , better pruning performance without computing the accurate qualification probability can be achieved. To illustrate pruning, assume that the hatched region in

figure 6 is 0.2-bound rectangle, and rectangle R is the query areas of a probability range query q with probability threshold 0.8. Obviously, object o can be pruned for the reason that the appearance probability of o in R must be smaller than 0.2, where 0.2 is the probability of o falling on the right of $U1(p)$. Actually, an object can be asserted to satisfy a query or not using the rules proposed in [10]. We will further specify these rules to make them applicable for our solutions.

How to efficiently compute the answers for the queries is very important. The execution time for the queries is significantly affected by the number of objects that need to be considered. With a large collection of points, it is impractical to evaluate each point for answering the query. As with traditional queries, index structures for facilitating the execution of the pruning phase can be used for this purpose.

One of the most important challenges for any indexing method for moving objects is efficient update processing as the object locations change. Any of the index structures proposed for moving objects can be used for efficiently processing imprecise queries. We plan to use the velocity constrained index, which is particularly suited for handling uncertainty of moving objects. We have described it only briefly in section 2.3; details can be found in [20]. The velocity constrained index (VCI) is an R-tree-like index structure. The maximum speeds of all objects are used in the construction of the index. Also, in the virtue of TPR*-tree, which is an effective multi-dimensional access method for moving objects, a better VCI structure that results in efficient update should be developed in our future work.

We are currently working on development of the velocity constrained index prototype and performance aspects in order to prove the practical viability of the VCR approach.

4.3 Other Issues

There are several interesting issues to be studied for the doctoral work. One important research direction is to optimize the time spent on the calculation phase. An extension is to study how to answer other probabilistic queries like nearest neighbor queries and reverse nearest-neighbor queries. Note that, providing probabilistic answers to nearest neighbor queries and reverse nearest-neighbor queries is much more difficult than range queries. Because an object is the nearest to the query is greatly influenced by the position and uncertainty of the other objects. But for range queries, the qualification probability for an object is independent of the presence of the other objects, it depends only upon the query and the uncertainty of the object considered.

Recently, new types of queries for moving objects have been proposed: a persistent probabilistic query, where the result is updated incrementally to account for updates of the underlying data; and a continuous probabilistic query, where the result as of the changing current time is maintained for a duration of time. Consequently, efficient algorithms for answering these probabilistic queries over imprecise positions of moving object are an interesting future research direction.

Another important issue is to define the suitable metrics for measuring the quality of the results returned by probabilistic queries—how reliable are the results returned by probabilistic queries if we do not know the exact results to the queries? As pointed out in [6] that different metrics are suitable for different

