Querying Imprecise Data in Moving Object Environments *

Reynold Cheng Sunil Prabhakar Dmitri V. Kalashnikov
Department of Computer Science, Purdue University.
Email: {ckcheng,sunil,dvk}@cs.purdue.edu

Abstract

In moving object environments it is infeasible for the database tracking the movement of objects to store the exact locations of objects at all times. Typically the location of an object is known with certainty only at the time of the update. The uncertainty in its location increases until the next update. In this environment, it is possible for queries to produce incorrect results based upon old data. However, if the degree of uncertainty is controlled, then the error of the answers to certain queries can be reduced. More generally, query answers can be augmented with probabilistic estimates of the validity of the answer. In this paper we study the execution of such probabilistic nearest-neighbor queries. The imprecision in answers to the queries is an inherent property of these applications due to uncertainty in the data, unlike the techniques for approximate nearest-neighbor processing that trade accuracy for performance.

1 Introduction

Systems for continuous monitoring of moving objects receive updated locations of objects as they move in space. 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 are updated periodically. 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. This uncertainty affects the accuracy of the answers to queries.

Due to the inherent uncertainty in the data, it seems impossible to provide meaningful answers. However, one can argue that for most moving objects, the locations of objects cannot change drastically in a short period of time. In fact, the degree and rate of movement of an object is constrained in practice. Such information can help solve the problem. For example, given a point q, we would like to know which of the two objects, x and y, is the nearest neighbor of q. Suppose we can provide a guarantee that at the time the query is evaluated, x and y could be no further than some distances d_x and d_y from their locations stored in the database. We can then state with confidence that x is the nearest neighbor of q if the longest possible distance of x from q is shorter than the shortest distance of y from q. In general, the uncertainty of the objects may not allow us to determine a single object as the nearest neighbor. Instead, each object has a probability of being the nearest neighbor.

The notion of probabilistic answers to queries over uncertain data was introduced in [3] for range queries, where the answer consists of objects and the probability that each object is in the specified range. We extend this idea to answer nearest-neighbor queries – the answer consists of not a single object that is closest to the object, but a set of objects each of which could be the nearest neighbor. The probability of each object being the nearest-neighbor is evaluated. The probabilities allow the user to place appropriate confidence in the answer as opposed to having an incorrect answer or no answer at all. Note that, depending upon the application, one may choose to report only the object with the highest probability as the nearest neighbor, or only those objects whose probability exceeds a minimum threshold.

Providing probabilistic answers to nearest-neighbor queries is much more difficult than range queries. For range queries, the probability for each object can be determined independent of the other objects – it depends only upon the query and the uncertainty of the object in question. However, for nearest-neighbor queries, the interplay between objects is critical, and the probability that an object is the closest to the query is greatly influenced by the position and uncertainty of the other objects. In this paper, we present a novel technique for providing probabilistic guarantees to answers of nearest-neighbor queries. Our algorithm is generic, in the sense that the geometric shape of

*Portions of this work were supported by NSF CAREER grant IIS-9985019, NSF grant 0010044-CCR and NSF grant 9972883

the region where each object is possibly located is not important. The algorithm can thus be applied to any practical object movement model.

2 Uncertainty Model and Probabilistic Queries

One popular model for uncertainty is that at any point in time, the location of the object is within a certain distance, d, of its last reported position. If the object moves further than this distance, it reports its new location and possibly alters the distance d to a new value (known to both the object and the server) [3]. Another model is one that has no uncertainty [2] where the exact speed and direction of movement are known. This model requires updates at the server whenever the objects speed or direction change.

For the purpose of our discussion, the exact model of movement is unimportant. All that is required is that at the time of query execution, the location (with uncertainty) of each object be known. The uncertainty of an object can be characterized as follows:

Definition 1: An **uncertainty region** of an object O_i at time t, denoted by $U_i(t)$, is a closed region such that O_i can be found only inside this region.

Definition 2: An **uncertainty probability density function** of an object O_i, denoted by $f_i(x,y,t)$, is a probability density function of O_i's location (x,y) at time t, that has a value of 0 outside $U_i(t)$.

Notice that since $f_i(x,y,t)$ is a probability distribution function, it has the property that $\int_{U_i(t)} f_i(x,y,t)dxdy = 1$. Based on the definitions of $U_i(t)$ and $f_i(x,y,t)$, we can now define the probabilistic nearest-neighbor query as follows:

Definition 3: **Probabilistic Nearest-Neighbor Query (PNNQ)** For a set of n objects $O_1, O_2, \ldots, O_n$ with uncertainty regions and probability density functions given at time t_0, a PNNQ for a point q is a query that returns a set of tuples of the form (O_i, p_i), where p_i is the non-zero probability that O_i is the nearest-neighbor of q at time t_0.

3 Evaluation of PNNQ

In this section we present an algorithm for answering a PNNQ. Let $S = \{O_1, O_2, \ldots, O_{|S|}\}$ be the set of objects to be considered by q in evaluating the query, and let R be the set of tuples returned by the query. The solution presented here consists of 4 steps: the *projection*, *pruning*, *bounding* and *evaluation* phases.

1. **Projection Phase.** In this phase, for each object O_i in S, we evaluate $U_i(t_0)$ based on the uncertainty model used by the application.

2. **Pruning Phase.** Consider two uncertainty regions $U_1(t_0)$ and $U_2(t_0)$. If the shortest distance of $U_1(t_0)$ to q is longer than the longest distance of $U_2(t_0)$ to q, we can immediately conclude that O_1 is not an answer to the PNNQ: Even if O_2 moves to the point farthest from q, O_1 still has no chance to be closer than O_2. Based on this observation, we can eliminate objects from S by the algorithm shown below:

1. For $i = 1, \ldots, |S|$ do
   (a) Let n_i be the shortest distance of $U_i(t_0)$ from q
   (b) Let f_i be the longest distance of $U_i(t_0)$ from q
2. Let $f = min_{i=1,\ldots,|S|} f_i$
3. For $i = 1, \ldots, |S|$ do
   (a) If $n_i > f$, then $S \leftarrow S - O_i$
4. Return S

After this phase, S contains the (possibly fewer) objects which must be considered by q. This is the minimal set of objects which must be considered by the query since any of them could be the nearest-neighbor of q. Note that with the use of indexing techniques [1], pruning can be done faster.

3. **Bounding Phase.** For each element in S, there is no need to examine all portions in the uncertainty region. We only have to look at the regions that are located no farther than f from q. We do this conceptually by drawing a *bounding circle* C of radius f, centered at q. Any portion of the uncertainty region outside C can be ignored.

4. **Evaluation Phase.** Let $C_q(r)$ denote a circle with center q and radius r. Let $P_i(r)$ be the probability that O_i is located inside $C_q(r)$, and $pr_i(r)$ be the probability density function of r such that O_i is located at the boundary of $C_q(r)$. The following is the algorithm for this phase.

1. $R \leftarrow \emptyset$
2. Sort the elements in S in ascending order of n_i, and rename the sorted elements in S as $O_1, O_2, \ldots, O_{|S|}$
3. $n_{|S|+1} \leftarrow f$
4. For $i \leftarrow 1$ to $|S|$ do
   (a) $p_i \leftarrow 0$
   (b) For $j \leftarrow i$ to $|S|$ do
      i. $p \leftarrow \int_{n_j}^{n_{j+1}} pr_i(r)dr \cdot \prod_{k=1 \wedge k \neq i}^{j} (1 - P_k(r))$
      ii. $p_i \leftarrow p_i + p$
   (c) $R \leftarrow R \cup (O_i, p_i)$
5. Return R

Note that if zero uncertainty is involved i.e., $U_i(t_0)$ is the recorded location of O_i, the evaluation phase algorithm needs to be modified. Readers are referred to our technical report [1] for details on how the above algorithm can be changed to handle zero uncertainty. In the rest of this section, we will explain how the evaluation phase works, assuming non-zero uncertainty.

Evaluation of $P_i(r)$ and $pr_i(r)$. The computation of $P_i(r)$ is illustrated below:

1. If $r \leq n_i$ return 0
2. If $r \geq f_i$, return 1
3. $A \leftarrow$ Overlapping area of $U_i(t_0)$ and $C_q(r)$
4. Return $\int_A f_i(x,y,t_0)dxdy$

If $r \leq n_i$, we are assured that $C_q(r)$ cannot cover any part of $U_i(t_0)$, so O_i cannot lie inside $C_q(r)$ and $P_i(r)$ equals 0 (Step 1). On the other hand, if $r > f_i$, we can be certain that $C_q(r)$ covers all parts of $U_i(t_0)$ i.e., O_i must be inside $C_q(r)$, and $P_i(r)$ equals 1 (Step 2). Steps 3 and 4 return a non-zero $P_i(r)$ value.

The evaluation phase needs another parameter called $pr_i(r)$, a probability density function of r where O_i lies on an infinitesimally narrow ring of radius r centered at q. If $P_i(r)$ is a continuous function, then $pr_i(r)$ is the derivative of $p_i(r)$.

Evaluation of p_i We can now explain how p_i is computed. Let $Prob(r)$ denote the probability density function that O_i lies on the boundary of $C_q(r)$ and is the nearest-neighbor of q. Then Formula 1 illustrates the structure of our solution:

$$p_i = \int_{n_i}^{f} Prob(r)\, dr \qquad (1)$$

Recall that n_i represents the closest possible distance of $U_i(t_0)$ from q, while f is the radius of the bounding circle, beyond which we do not need to consider. Formula 1 expands $C_q(r)$ with radius n_i to f. Therefore, each point in $U_i(t_0)$ must lie on some circular ring of width dr, center q and radius r, where $r \in [n_i, f]$. For each ring, the formula evaluates the probability that (1) O_i lies on the ring, and (2) O_i is the nearest-neighbor of q. Essentially, we consider all the points in $U_i(t_0)$ that are equidistant from q, and evaluate the chance that they are nearest to q. Using $pr_i(r)$ and $P_k(r)$ (where $k \neq i$), we can change Formula 1 to:

$$p_i = \int_{n_i}^{f} p_i(r)dr \cdot \prod_{k=1 \wedge k \neq i}^{|S|} (1 - P_k(r)) \qquad (2)$$

Observe that each $1 - P_k(r)$ term registers the probability that object O_k (where $k \neq i$) lies at a distance greater than r.

Efficient Computation of p_i We can improve the computation time of Formula 2. Note that $P_k(r)$ has a value of 0 if $r \leq n_k$. This means when $r \leq n_k$, $1 - P_k(r)$ is always 1, and O_k has no effect on the computation of p_i. Instead of always considering $|S| - 1$ objects in the $\prod$ term of Formula 2 throughout $[n_i, f]$, we may actually consider fewer objects in some ranges of values. First, we sort the objects according to their shortest distances (n_i) from q. Next, the integration interval $[n_i, f]$ is broken down into a number of intervals with end points defined by the near distances of the objects. The probability of an object being the closest neighbor of q is then evaluated for each interval in a way similar to Formula 2, except that we only consider the objects with non-zero $P_k(r)$. The sum of the probabilities for these intervals is p_i. The final formula for p_i is:

$$p_i = \sum_{j=i}^{|S|} \int_{n_j}^{n_{j+1}} pr_i(r)dr \cdot \prod_{k=1 \wedge k \neq i}^{j} (1 - P_k(r)) \qquad (3)$$

Here we let $n_{|S|+1}$ be f for notational convenience. Instead of considering $|S| - 1$ objects in the $\prod$ term, Formula 3 only handles $j - 1$ objects in interval $[n_j, n_j + 1]$. Formula 3 is implemented in our evaluation phase algorithm. Step 2 sorts the objects in S in ascending order of the near distances. Step 3 assigns the value of f to $n_{|S|+1}$. Step 4 executes Formula 2 once for every object O_i in S, and puts the tuples (O_i, p_i) into R.

4 Conclusions

In this paper we studied the execution of probabilistic nearest-neighbor queries over uncertain data for moving objects. We define a generic model of uncertainty, and then present algorithms for computing these queries for this model. To the best of our knowledge, with the exception of [3] which addresses probabilistic range queries for objects moving in a straight lines with fixed speed, there is no work on probabilistic queries over uncertain data. We address the problem of the more complicated nearest-neighbor queries under a more relaxed model. A detailed version of this paper including specifics for two common uncertainty models and implementation details is available [1].

References

[1] R. Cheng, S. Prabhakar, and D. V. Kalashnikov. Querying imprecise data in moving object environments. In *TR 02-020*, Purdue University, Oct. 2002.

[2] D.Pfoser and C.S.Jensen. Querying the trajectories of on-line mobile objects. In *MobiDE 2001*, pages 66–73, 2001.

[3] O. Wolfson, P. A. Sistla, S. Chamberlain, and Y. Yesha. Updating and querying databases that track mobile units. *Distributed and Parallel Databases*, 7(3):257–387, 1999.