Querying Moving Objects with Uncertainty

Bin Lin Hoda Mokhtar Rafael Pelaez-Aguilera Jianwen Su
Department of Computer Science
University of California
Santa Barbara, CA 93106
{linbin, hmokhtar, rpelaez, su}@cs.ucsb.edu

***Abstract*— Rapid development of mobile devices makes it possible that billions of mobile and personal devices will soon be used in every daily routine work. This emerging environment presents a number of challenging problems in several research areas including databases. The need for managing and querying continuously moving objects is among the new demands of this development. Although global positioning systems (GPS) are widely used nowadays, it is still a fact that the position information obtained is not accurate at every time instant. Thus, in most cases trajectory information is in fact "expected" trajectory with uncertainty. An interesting problem is indexing and querying trajectories with uncertain information. In this paper we introduce a data model for moving object trajectories with uncertainty and develop a new index structure (TPRU-tree) for uncertain trajectories based on TPR-tree. We discuss query evaluation using TPRU-tree and present an empirical study.**

I. Introduction

We are currently facing a rapid change towards a world based on personal and mobile devices. such a wide spread of personal devices enables a wave of new database applications. Moving object databases and location based data management are among the consequences of this new environment. In particular, managing and querying continuously moving objects is becoming increasingly urgent, especially in the presence of devices such as global positioning systems (GPS).

Due to a variety of unpredictable and somewhat random factors such as sudden wind changes, unexpected speed slowdowns, device failures, etc., obtaining future object locations is an interesting research problem. As argued by Pfoser and Jensen [1] and Trajcevski et al [2], the location of a moving object is typically associated with uncertainty due to a variety of factors.

Although many issues have been studied for precise trajectories (see e.g., [3]) for moving objects, modeling uncertain trajectories is fundamentally different. In this paper we propose a model that is partially motivated by and significantly extends earlier models [1], [2]. Pfoser and Jensen [1] focused on sampling errors in determining object locations and modeling such errors with uniform distributions over a disk whose radius is a parameter, they also looked into evaluating window queries. On the other hand, Trajcevski et al [2] studied queries on trajectories. Quantifiers as "always", "possibly", "sometimes", etc. were developed and used to express different degrees of likelihoods for window queries over trajectories. However, their model lacks the ability to provide quantitative answers for the queries. Moreover, their model uses a fixed radius to indicate that at any time instant the object can be within that radius from the mean. Therefore they model uncertainty as a cylinder around the mean with uniform distribution inside it to specify the location of the object. Query evaluation for uncertain trajectories was investigated in [4], [5], where the authors considered time instant queries and presented algorithms for evaluating probabilistic time instant range and nearest neighbor queries and well as some probabilistic aggregate queries.

In this paper we present a general model for moving object trajectories with uncertainty. The framework borrows techniques from probability theory. We take the approach of viewing the location of a moving object in an n-dimensional space at a time instant as a vector of n random variables. And the trajectory of a moving object as a vector of stochastic (random) processes. We use time-parametric uniform and Gaussian distributions to represent coordinates of moving object locations.

Based on this model, we develop a new index structure, "TPRU-tree", for indexing uncertain trajectories. TPRU-tree is extended from TPR-tree [6] to allow uncertainty information to be represented. We also develop algorithms for evaluating moving objects queries such as "find the delivery trucks that will be in Santa Barbara area tomorrow at 5pm with at least 90% probability". We study exact and approximate evaluation of "top k" queries (k most probable objects) and propose 6 search strategies. Experimental results show the following. (1) Evaluation of queries is more efficient with TPRU-tree. (2) Approximate answers can significantly reduce the I/O complexity of queries while producing very accurate answers.

Section 2 presents the model of trajectories with uncertainty. Section 3 introduces the TPRU-tree index structure, including search strategies. Section 4 discusses queries and query evaluation. Section 5 includes the experimental results, and Section 6 concludes the paper.

II. A Model for Uncertain Trajectories

In this section we introduce a data model for moving object trajectories with uncertainty. Intuitively, a trajectory in this model is represented as a vector of stochastic (random) processes. More specifically, our model treats a coordinate of a trajectory as a stochastic process with either a time-parametric uniform or Gaussian distribution. For a given time instant, a coordinate of an object location is a random variable that

Support in part by NSF grants IIS-9817432 and IIS-0101134.

ranges over a (closed) interval of likely positions. The model is based on the following two assumptions: (1) trajectories of different objects are independent, and (2) different coordinates of the same trajectory are independent.

We assume the time to be continuous (represented by real numbers) and fix t to be the time variable. Also, objects move in some $n > 0$ dimensional continuous physical space.

Consider an object o moving towards northeast in a 2-dimensional plane as shown in Fig. 1. Due to uncertainty, the location of o at a time instant t cannot be precisely known. One way to model this uncertainty is to treat the coordinates of o's location as a pair of random variables, $\bar{x}_t$ and $\bar{y}_t$, with some given distributions.

Fig. 1. The coordinate $\bar{x}_t$ at a time instant t

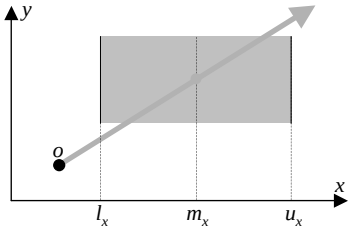


Fig. 1 also shows the range $[l_x, u_x]$ of possible values for $\bar{x}_t$ with a mean m_x. We can easily associate a distribution with $\bar{x}_t$ to model the uncertainty of x_t. $\bar{y}_t$ can be similarly represented.

Under these assumptions a trajectory is treated as a vector of stochastic processes with time-parametric distributions, e.g, uniform distribution with the distribution bounds being functions of time or Gaussian distribution with the mean and variance being functions of time. Uniform distributions are interesting since they are simple. Gaussian distributions, on the other hand, are often used to model random variables that are affected by many factors [7], for example airplane trajectories [8]. We use a variant of the traditional Gaussian distribution. A "modified Gaussian distribution" is a Gaussian distribution with time-parametric distribution bounds. In this paper we consider linear functions of time and hence linear time-parametric uniform and Gaussian distributions.

We represent each *(time-parametric) Gaussian distribution* by a triple of linear continuous (real) functions over time (μ, σ, δ) where $\mu(t)$ is the mean of the distribution at time t, $\delta(t) > 0$ for all time instants t such that $[\mu(t) - \delta(t), \mu(t) + \delta(t)]$ is the range of possible values, and $\sigma(t)$ is the variance of the Gaussian distribution at time t. Similarly, a *(time-parametric) uniform distribution* is represented by the pair (μ, δ), where μ and δ are as the same as in a Gaussian distribution.

For each Gaussian distribution (μ, σ, δ), its distribution function $\mathcal{G}_{\mu,\sigma,\delta}$ is modified from a standard Gaussian distribution function so that for all time instants t

$$\int_{\ell}^{u} \mathcal{G}_{\mu,\sigma,\delta} dx = 1 \text{ and } \int_{-\infty}^{\ell} \mathcal{G}_{\mu,\sigma,\delta} dx = \int_{u}^{\infty} \mathcal{G}_{\mu,\sigma,\delta} dx = 0$$

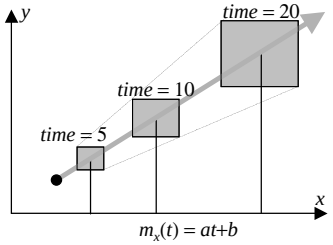
where $u = \mu + \delta$ and $\ell = \mu - \delta$. Note that u and ℓ are functions of time t.

Similarly, for each uniform distribution (μ, δ), its distribution function $\mathcal{U}_{\mu,\delta}$ is defined as follows.

$$\mathcal{U}_{\mu,\delta}(x) = \begin{cases} \frac{1}{2\delta} & \text{for } \mu - \delta \leqslant x \leqslant \mu + \delta \\ 0 & \text{otherwise} \end{cases}$$

Fig. 2 shows a trajectory with uniform distribution for both coordinates. In particular, the distribution for the x dimension is (m_x, δ).

Fig. 2. A trajectory with uniform distribution



Note that both types of distributions can degenerate to the special case when the range becomes a point, i.e., $\delta(t) = 0$.

In general, introducing uncertainty in the representation of a trajectory creates an uncertainty region (volume) around the mean (expected) trajectory. Our approach to modeling uncertainty using stochastic processes, allows the uncertainty volume to change over time. Such flexibility in the uncertainty region captures the nature of uncertain information. At the beginning of motion exact information about the starting location is available, as time goes, this precise information can no longer be guaranteed and hence the uncertainty region widens. As the object approaches its final destination which is again a defined location, the uncertainty region shrinks to reflect the available amount of precise information.

III. An Index Structure for Trajectories with Uncertainty

In this section we introduce a new index structure, TPRU-tree (Time Parameterized R-tree with Uncertainty), which is an extension of TPR-tree [6]. TPRU-tree efficiently indexes uncertain trajectories of moving objects. A TPRU-tree treats each dimension independently. In the following, we focus on one dimension in describing TPRU-tree.

We start with describing leaf nodes. Suppose that an uncertain trajectory in this dimension is given by a Gaussian distribution (μ, σ, δ), where μ, σ, δ are linear functions. (A uniform distribution only needs μ and δ and will be dealt with similarly.) Let $\mu(t) = a + bt$, $\sigma(t) = c + dt$, and $\delta(t) = e + ft$, where a, b, c, d, e, f are real numbers. Then each entry in a leaf node contains (1) a, b, c, d, e, f as floating point numbers for this dimension, and (2) t_{ref}, where t_{ref} is a floating point number denoting the last update time of this object (having the trajectory). For a uniform distribution, we set $c = d = 0$. This flag allows the same data structure to be used for both types of distributions. We choose to have an implicit flag since larger entry sizes will reduce the number entries stored in a node (page), i.e., the fan-out factor.

In the above description, a leaf entry contains six floating point numbers for each dimension, which increases the storage requirement of an entry significantly and consequently

reduces the fan-out factor for the leaf level. To overcome this problem, we let c, d, e, and f be k-bit numbers, and specify M_c, M_d, M_e, M_f (defined globally for the entire tree) as their maximum values, respectively. Their actual values are given by $\frac{x}{2^k}M_i$, where $x \in \{c, d, e, f\}$. This method however sacrifices some precision in order to improve the performance of TPRU-tree. For different applications, we can adjust parameter k to get a satisfactory trade-off between precision and performance.

To describe the data structure for internal nodes, we first view each trajectory as an "expanding box" which coincides with its uncertainty region. Note that the boundaries of this box are linear functions of time.

For internal nodes of TPRU-tree, the key is to compute the bounding rectangle of a set of expanding boxes. In our design, a bounding rectangle in an entry contains all expanding boxes in the corresponding child node. Let $o_1, ..., o_n$ be a set of expanding boxes (from a child node). We define $t_{\text{ref}} = \max\{o_i.t_{\text{ref}} \mid 1 < i < n\}$ to be the latest update time of the child nodes. Let $[z_l, z_u]$ be the minimal bounding interval for all o_i's at time t_{ref}, and v_l, v_u be the minimum and maximum velocities (resp.) of the boundaries of o_i's. Thus, the entry in the internal node contains z_l, z_u, v_l, v_u for this dimension and t_{ref}.

We now briefly discuss the search, insertion, and deletion operations for TPRU-tree. TPRU-tree employs three different search strategies based on the use of depth-first (DF), breadth-first (BF) traversals, and a priority queue. Due to the need to explore multiple paths in a TPRU-tree, the different strategies explore the multiple paths in different orders. The DF and BF explore the paths in the depth-first and breadth-first manners.

For the priority queue strategy, TPRU-tree maintains a priority queue of entries whose bounding boxes intersect the query window. The ordering used in the priority queue may depend on some cost function (e.g. the area of intersection between each the tree node and the query window). Initially, there is only one entry, namely, the root in the priority queue. At each step, the top (root) entry of the priority queue is removed. If the entry is a leaf node, it is evaluated to produce possible answers. Otherwise we expand the entry and evaluate its children against the query window. Children entries whose bounding boxes intersect the query window are then inserted back into the priority queue. The process is repeated until either the priority queue is empty or some pre-condition of the query is satisfied (e.g. enough number of objects are found).

In addition to using different search algorithms, we also consider different ordering metrics. We investigate 2 metrics, namely, distance and overlap area. The former measures the distance between the center of the box in consideration and the center of the query window, while the latter the intersection area of the box and the window.

The metrics are used for not only the priority queue based strategy, but also for DF and BF strategies in the following sense. When a child node is chosen to explore, we may use the distance metric, by choosing the child whose bounding box center is closest to the center of the query window, or the overlap area metric by choosing the child whose bounding box has the largest overlap area with the query window.

By combining the three different search strategies together with the two ordering metrics we have a total of 6 different algorithms for the search operation.

Fig. 3. Summary of Search Strategies

Metric	Depth-first	Breadth-first	Priority queue
Distance	DFD	BFD	PQD
Area	DFA	BFA	PQA

Insertions are similar to that in TPR-tree with the following exception: entries in leaf nodes are not moving points, but uncertain trajectories. So during the split procedure, when we need to sort the uncertain trajectories, we sort them by their current expected positions. For deletions, TPRU-tree locates the trajectory and deletes it. When a node underflows, it is eliminated and its entries are reinserted.

IV. Query Evaluation

In this section we present an approach towards query evaluation over moving objects. We focus on studying "probabilistic range queries" and investigate different query evaluation techniques.

A *range* consists of a closed rectangular spatial region and a time instant. A *range query* is a pair (W, t) where W is a range and t a time instant. Given a moving object o and a range query $Q = (W, t)$, let $P(o, Q)$ denote the probability of the event "at time t, the object o is inside the range W."

In this paper, we consider "top-k" queries. For each integer $k > 0$, a *top k range query* involves a range query Q and a real number p between 0 and 1 representing the probability threshold. An answer to the top k range query consists of the k moving objects o who have the highest probabilities $P(o, Q) > p$.

Example 1: Consider the query "retrieve the top 5 VONS delivery trucks in Santa Barbara area at 5pm today that have probability at least 90%". The answer of this query is basically the 5 trucks set S that satisfy the following condition: the area of the intersection between the uncertainty region of each truck $s \in S$ with the query region is at least 90%, assuming uniform distributions are used.

Evaluating the query in Example 1 proceeds by computing the intersection area between each objects and the query window, after computing this probability and with the help of the TPRU-tree index structure, the objects with probability $\geqslant 0.9$ can be efficiently retrieved.

The use of intersection area is valid for uniform distribution. And thus probability computation for uniform distributions is straightforward. For Gaussian distributions (μ, σ, δ), let $\mathcal{G}_{\mu,\sigma,\delta}$ be the distribution function. The probability $Pr[x \leqslant a]$ can be computed as follows:

$$\int_{-\infty}^{a} \mathcal{G}_{\mu,\sigma,\delta} dx = \frac{\int_{-\infty}^{a} \mathcal{G} dx - \int_{-\infty}^{-\delta} \mathcal{G} dx}{\int_{-\delta}^{\delta} \mathcal{G} dx}$$

where $\mathcal{G}$ is the corresponding standard Gaussian distribution function (i.e., with μ, σ), $[-\delta, \delta]$ specifies the distribution range. The probability can be then be computed by first transforming each term into standard Gaussian form (i.e.,

$\mu = 0$ and $\sigma = 1$), then we use the Q-function that has an excellent approximation formula due to Börjesson and Sundberg (see [9]), with maximum error of 0.27% for any $x \geqslant 0$.

It is easy to observe that the different search strategies behave similarly with respect to the query in Example 1 and in fact to all top k range queries. The reason behind this observation is the fact that all search strategies need to find exactly the same set of objects in order to obtain the top k objects in an answer.

On the other hand, in many cases the top k objects in an answer need not be the k best matching objects. In other words, the answer may be *approximated*. While there are many ways to compute approximate answers, in the following we outline one approach.

For each integer $k > 0$, an *any k range query* involves a range query Q and a probability threshold p. Unlike a top k query, an answer to the any k range query consists of any k moving objects o with probabilities $P(o, Q) > p$. Clearly, any k queries are possibly faster to evaluate since there is no need to exhaust all potential objects.

Given a top k range query, we can compute its approximate answer in two steps:

1) Compute the answer to an any k' range query for some $k' \geqslant k$.
2) Select k objects among the k' objects found with the highest probabilities.

Consider the query in Example 1 which asks for the top 5 trucks. We can approximate the answer by finding any 10 VONS delivery trucks in Santa Barbara area at 5pm today that have probability at least 0.9 and then pick the highest five.

V. Experimental Evaluation

In this section we present an empirical study on the relative behaviors of the six search strategies in Fig. 3. In particular, we present experimental results to show the impact of the different search strategies on the performance and accuracy of the TRPU-tree in evaluating both the exact and approximate probabilistic range queries.

For query performances, we focus on the number of I/Os (i.e., page reads and writes) since the I/O is expected to be the dominant factor in the overall complexity.

For exact answers, all six strategies will search through all objects that may possibly be in the answer. As a result, their performances have little differences. We will use "EXT" to denote search strategies for precise answers. For approximate answers, the strategies are different. The reason for this is that we no longer need to find all objects that are possible in the precise answer. We use A*xxx* to denote the strategy *xxx* in Fig. 3 for computing approximate answers.

For accuracy, we measure the percentage of objects in an approximate answer that are in the precise answer. Recall that our approach is to use any k' queries to approximate top k queries for $k' \geqslant k$. In our experiments, we will consider the impact of k' and the probability threshold p on both performance and accuracy.

Our experimental results suggest the following:

1) The TPRU-tree index structure provides efficient access to the moving object positions in roughly logarithmic I/Os.
2) Approximate answers are far more efficient to compute with varying degrees of accuracy depending on k' and p.
   a) When k' is some multiple times of k, the accuracy is 1 or close to 1, while I/O complexity is merely a fraction of that for the exact answer.
   b) High p improves accuracy of approximation and pays more cost on performance, while low p means low accuracy and less cost. Users can get a satisfactory trade-off between accuracy and cost by adjusting p.

Our data set consists of moving points that are generated randomly in a workspace of size 1000×1000 kilometers. A time unit is set to one minute. The number of moving points is $200,000$ for most experiments. About a half of moving points have uniform distributions, and the other half Gaussian distributions. The speeds of moving points range from 0.75 to 2.5 kilometers per minute. For all experiments, the disk page size is set to 1k bytes to get trees with reasonably large size. So the maximum number of entries is 36 for leave nodes and 28 for intermediate nodes. Buffer size is set to 0 to observe queries with small I/Os. All queries are either exact or approximate probabilistic range queries. A parameter QWS is used to describe the size (area) of query windows in terms of a fraction of the total workspace. For most experiments $QWS = 1\%$.

After a moving point data set is loaded into a TPRU-tree, a workload composed of both queries and updates is executed over the TPRU-tree. An update in our experiments involves a change in speed, direction and distribution parameters of a moving object. For most experiments, we evaluate the tree performance at time unit 100. It is clear that fixing a time unit has little impact on the relative performance of different strategies and no impact on accuracy results.

Fig. 4 shows performance of TPRU-tree for computing exact answers of *top k range queries* with different sizes of data sets. In this experiment we restrict the uncertainty range parameter δ to prevent uncertainty regions from growing too big and set $QWS = 0.1\%$. Fig. 4 displays that TPRU-tree is efficient for such queries and is scalable.

Fig. 4. Scalability of TPRU-tree ($k = 20, p = 0.5$)

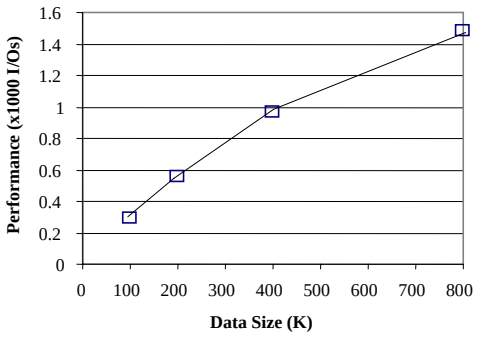


Fig. 5 shows the impact of parameter k' on the performance

and accuracy of different search strategies for approximate queries as well as the performance for corresponding exact queries. All the approximate queries perform much better than the exact query. Depth first and priority queue strategies' perform better than breadth first strategies. In terms of accuracy, area-based search strategies are better than distance-based ones. In general, when k' increases, the accuracy of the approximate queries increases as well. When $k' = 4k$, the accuracy of all approximate queries is close to 1.

Fig. 5. Impact of k' ($p = 0.5, k = 20$)

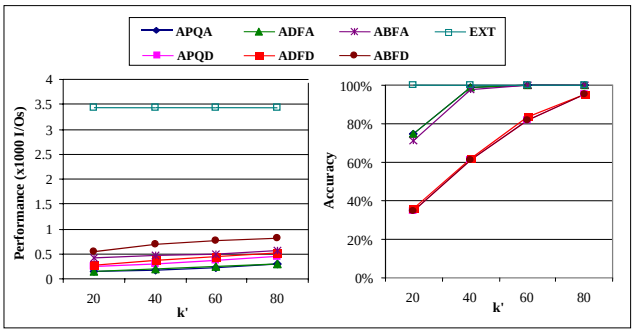
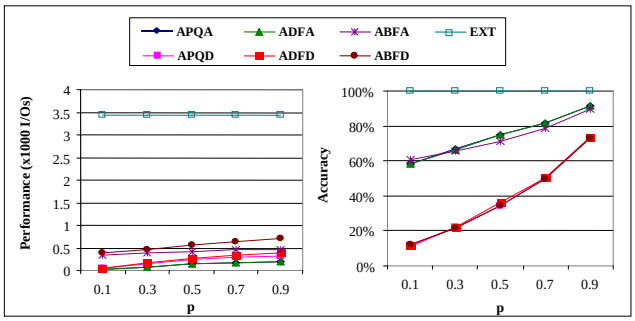


Fig. 6 shows the impact of the threshold p. Performance goes down when p increases. But the improvement by approximate queries in performance is still high. Similar to Fig. 5, area-based search strategies are more accurate than distance-based ones. In general, higher p means higher accuracy. When p approaches 1, the accuracy gets close to 1 as well.

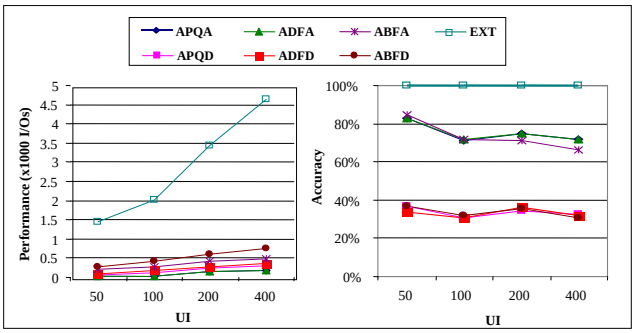
Fig. 6. Impact of p ($k = 20$)



To study the impact of the data freshness on the performance and accuracy, a parameter called Update Interval (UI) is used in data generation [6]. It represents the approximate time interval between two consecutive updates for the same object. Smaller UI means more data freshness. For most experiments $UI = 200$.

Fig. 7 shows the impact of the parameter UI (i.e. Update Interval). When UI is small, this means the update frequency is high (i.e. data is refreshed more frequently), and consequently the uncertainty regions are small. In general, approximate queries perform better than exact queries. When UI increases, the performance of exact queries goes down quickly, but that of approximate queries goes down relatively slower. On the other hand, accuracy increases as UI decreases. However, to achieve better performance and higher accuracy we have to pay more on the update maintenance. In general, area-based search strategies are more accurate than distance-based ones.

Fig. 7. Impact of Update Interval ($k = 20, p = 0.5$)



VI. Conclusions

In this paper we developed a data model for trajectories of moving objects with uncertainty, and a new index structure for moving objects with uncertainty. We also studied evaluation of top k range queries with exact and approximate answers and considered 6 evaluation strategies in conjunction with TPRU-tree. Experimental results show that the index structure TPRU-tree can efficiently answer top k range queries. In addition, they show that different search strategies behave differently towards accuracy and I/O performance.

We think that querying moving objects with uncertainty is far from mature. It is especially interesting to study evaluation techniques for other types of queries, and to improve index structures and search strategies to speed up query evaluation in presence of uncertainty.

References

[1] D. Pfoser and C. Jensen, "Capturing the uncertainty of moving-object representations," in *Advances in Spatial Databases, 6th International Symposium, SSD'99, Hong Kong, China, July 20-23, 1999, Proceedings*, ser. LNCS, R. H. Güting, D. Papadias, and F. H. Lochovsky, Eds., vol. 1651. Springer, 1999, pp. 111–132.

[2] G. Trajcevski, O. Wolfson, F. Zhang, and S. Chamberlain, "The geometry of uncertainty in moving objects databases," in *Advances in Database Technology - EDBT 2002, 8th Int. Conf. on Extending Database Technology*. Springer, March 2002, pp. 233–250.

[3] C. S. Jensen, M. Schneider, B. Seeger, and V. J. Tsotras, Eds., *Advances in Spatial and Temporal Databases, 7th International Symposium, SSTD 2001, Redondo Beach, CA, USA, July 12-15, 2001, Proceedings*, ser. Lecture Notes in Computer Science, vol. 2121. Springer, 2001.

[4] R. Cheng, S. Prabhakar, and D. V. Kalashnikov, "Querying imprecise data in moving object environments," in *Proc. Int. Conf. on Data Engineering*, Bangalore, India, Mar. 2003.

[5] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, "Evaluating probabilistic queries over imprecise data," in *Proc. ACM SIGMOD Int. Conf. on Management of Data*, San Diego, CA, USA, June 9–12 2003.

[6] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez, "Indexing the positions of continuously moving objects," in *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2000, pp. 331–342.

[7] A. Faradjian, J. Gehrke, and P. Bonnet, "GADT: A probability space ADT for representing and querying the physical world," in *Proc. Int. Conf. on Data Engineering*, 2002.

[8] M. Prandini, J. Hu, J. Lygeros, and S. Sastry, "A probabilistic approach to aircraft conflict detection," *IEEE Trans. Intelligent Transportation Systems*, vol. 1, no. 4, pp. 199–220, Dec. 2000.

[9] P. Peebles Jr., *Probability, Random Variables, and Random Signal Principles*, 4th ed. McGraw-Hill, July 2000.