Handling Uncertain Spatial Data: Comparisons between Indexing Structures

Bir Bhanu, Rui Li, Chinya Ravishankar and Jinfeng Ni

***Abstract*— Managing and manipulating uncertainty in spatial databases are important problems for various practical applications. Unlike the traditional fuzzy approaches in relational databases, in this paper we propose a probability-based method to model and index uncertain spatial data where every object is represented by a probability density function (PDF). To index PDFs, we construct an optimized Gaussian mixture hierarchy (OGMH) and two variants of uncertain R-tree. We provide a comprehensive comparison among these three indices and plain R-tree on TIGER/Line Southern California landmark point dataset. We find that uncertain R-tree is the best for fixed query and OGMH is suitable for both certain and uncertain queries. Moreover, OGMH is suitable not only for spatial databases, but also for multi-dimensional indexing applications like content based image retrieval, where R-tree is inefficient in high dimensions.**

***Index Terms*—index structures, R-tree, spatial database, uncertainty.**

I. Introduction

Geographic information system (GIS) is a system of computer software, hardware and data, and personnel to help manipulate, analyze and present information that is tied to a spatial location. Spatial database is the system which organizes spatial information in GIS [1]. In GIS applications, it is generally agreed that there are several types of error (uncertainty) which determine the overall accuracy of final products. A biennial Spatial Accuracy Symposium is held specifically on this topic [2]. However, in the filed of information technology scant attention has been paid for handling uncertainty in spatial databases [3]. Even though there is more awareness and some understanding of uncertainty sources in spatial data, at present there is not a complete system which can store and operate on uncertain spatial data. ArcInfo and Oracle Extensions are only suited for *certain* spatial data [1]. In this paper, we present our approaches on indexing uncertain spatial data.

Most of the existing approaches for management of probabilistic data are based on the relational model and use fuzzy set theory [4, 5]. They are useful for representing uncertainty at the symbolic level. However, in addition to symbolic uncertainty, sensor-processing tasks involve uncertainties at both numeric and existence levels. Supporting these types of uncertainty in the current relational model using fuzzy logic is fundamentally difficult. So in our approach, we use *probability density functions (PDFs)* to represent uncertain data, which means every object is a random variable.

In spatial databases, R-tree is the most often used indexing structure, which is a depth-balanced tree whose nodes are represented by *Minimum Bounding Rectangles (MBR)*. Guttman [6] first proposed R-tree and Greene and Beckmann [7, 8] optimized it by reducing margin and MBR overlap. But R-tree family indexes fixed data only. In order to handle the uncertain information related to each object, we have to use R-tree variant or design new index. In this paper, we build two variants of uncertain R-tree and an *optimized Gaussian mixture hierarchy (OGMH)* based on *Gaussian mixture model (GMM)*.

The rest of the paper is organized as follows. The uncertainty representation and index construction are explained in Section II. Section III gives the experimental results on TIGER/Line Southern California landmark point dataset. Section **IV** concludes the paper.

II. Technical Approach

A. PDF representation

In conventional spatial databases, each object is represented by a feature vector in n dimensional feature space. But when the data are uncertain, a different representation is required.

In out approach, we use a PDF to represent each uncertain object. If an object needs n features to describe, then it is an n dimensional (feature vector) random variable, as given in (1).

$$\mathbf{f}^i = \left[f_1^i, f_2^i, \cdots, f_n^i\right]^T \quad (1)$$

where $i = 1, \ldots, N$, N is the number of objects

For simplicity, we assume that the features are independent of each other and each feature's PDF (f_j^i, $j = 1,..., n$) is known. The process of getting the PDFs is called *uncertainty*

This work was supported by NSF Information Technology Research Grant 0114036. The contents of the information do not reflect the position or policy of the U.S. Government.

B. Bhanu is with the Center for Research in Intelligent System, University of California, Riverside, CA 2521 (e-mail: bhanu@cris.ucr.edu).

R. Li is with the Department of Electrical Engineering, University of California, Riverside, CA 92521 (e-mail: rli@vislab.ucr.edu).

C. Ravishankar is with the Department of Computer Science & Engineering, University of California, Riverside, CA 92521 (e-mail: ravi@cs.ucr.edu).

J. Ni is with the Department of Computer Science & Engineering, University of California, Riverside, CA 92521 (e-mail: jni@cs.ucr.edu).

modeling and it is a part of our on going related work. Our system needs to handle both certain and uncertain data. So we do not give a specific name to uncertain data. Feature vector is the name for both certain and uncertain data in this paper.

For feature vectors, metrics like Euclidean distance, Manhattan distance etc. are used to measure similarity. Since our uncertain objects are random variables represented by PDFs, we define the similarity of feature vectors as the probability that two random variables are the same, as shown in (2). In this equation, **D** and **Q** are two objects. **D** is the object in the database and **Q** is the query. Δ is a threshold vector describing the maximal error the system can tolerate to still regard **D** as "similar" to **Q**. Paper [9] presents the similarity measure in detail.

$$similarity(\mathbf{D},\mathbf{Q}) = Pr\left(|\mathbf{D}-\mathbf{Q}| < \Delta\right) \quad (2)$$

D, **Q** are two feature vectors

B. System diagram

In our method, we assume that the uncertainty is small [10], therefore the disturbed objects roughly keep the original distribution. Thus, we can use the feature vector means to construct the index, and attach uncertain information to the data entries. At this step, an index which handles uncertainty is constructed, as shown by the dash-line box in Fig. 1.

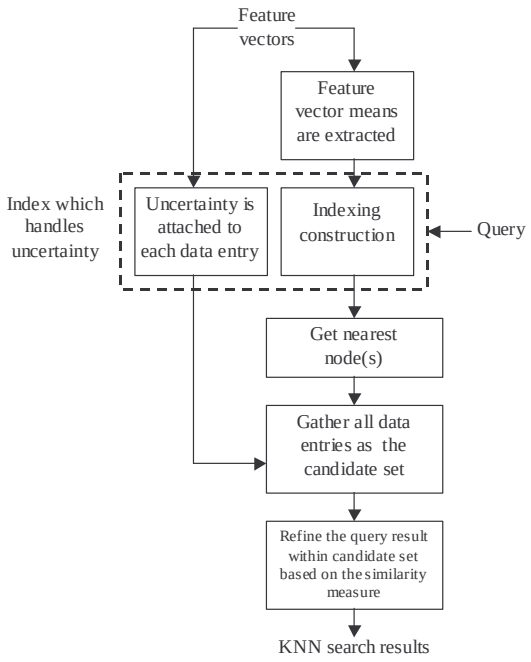

Fig. 1. Index structure and KNN search procedure.

When a query comes in, the nearest node(s) in the index is found. All the data entries belonging to these nodes (with uncertainty) are gathered as the candidate set. This process is called "filter". In the "refine" step, the similarity between the query and each data entry in the candidate set is calculated and sorted. The K entries corresponding to the K largest similarity are the query result. This is called *K nearest neighbors (KNN)* search. Here, we are only interested in KNN because it is the basis for other comprehensive queries.

The filter step differs in different index constructions. We construct two index systems:

(a) *Optimized Gaussian mixture hierarchy (OGMH)* based on *Gaussian mixture model*, and

(b) Two R-tree variants called *uncertain R-tree 1 (UR1)* and *UR2*.

These are explained in part C and D in this section.

C. Optimized Gaussian Mixture Hierarchy (OGMH)

We assume all the objects in the n dimensional feature space follow some distribution. From the probability theory, any distribution can be approximated by a weighted sum of several Gaussian distributions [11], as shown in (3), where distribution $p(x)$ is approximated by C Gaussian distributions: $f_i(x)$, $i = 1,\cdots,C$. α_i is the weight for each Gaussian distribution represented by mean vector u_i and covariance matrix Σ_i:

$$p(x) = \sum_i^C \alpha_i f_i(x) = \sum_i^C \alpha_i N(u_i, \Sigma_i) \quad (3)$$

Figueriedo and Jain [12] proposed a variant of *Expectation Maximization (EM)* algorithm to automatically find the number of clusters and to perform clustering. We use this algorithm to get the Gaussian components. It involves the following steps:

In the first step, the entire dataset is clustered into several groups. In the subsequent steps, a bottom-up binary tree is built based on these Gaussian components.

Since every leaf node is represented by a Gaussian distribution, we represent every inner node by the Gaussian mixture of all its offspring leaves.

When a query comes in, we start from the root node, calculate the query's similarity with the left child and the right child, and then select the branch with higher similarity. This process is repeated until a leaf node is reached. Then all the data entries belonging to this leaf are gathered as the candidate set. If there are not enough data entries in the candidate set, all data belonging to the sibling of this leaf node are added.

Refinement is performed within the candidate set to get the KNN search result.

Detail explanation of tree construction and KNN search algorithm are given in [9].

D. Uncertain R-tree

The uncertain R-tree construction is based on all the feature vector means and uncertain information is attached to each data entry. To support KNN searches, two filter strategies: UR1 and UR2 are developed.

1) ***UR1:*** Nearest leaves are returned, even if they do not belong to the same parents. The number of leaves is decided by the required candidate set size. All data entries of these leaf nodes are gathered together as the candidate set for our uncertain query.

2) ***UR2:*** The nearest leaf is found, then its ancestors are backtracked until the one that has enough data entry

offspring for refinement is met. All the data entries of these offspring construct the candidate set.

The candidate set in both UR1 and UR2 is refined to get the KNN for the query. UR2 and OGMH have the same candidate set extraction strategy, so only their comparison is fair, whereas UR1 will theoretically achieve higher precision. The comparison in the next section will testify this result.

III. Experimental Results

A. Dataset and uncertainty assignment

We take the TIGER/Line southern California landmark point dataset in the experiment, as shown in Fig. 2 and Fig. 3. It contains 8703 2D coordinates (longitude and latitude in degrees) and its precision (uncertainty) is 167 feet [10], which is 0.0005°. Uncertainty is added as a 2D Gaussian noise to each point, as shown in (4):

$$Noise \sim N\left([0,0]^T, \begin{bmatrix} \delta_x^2 & 0 \\ 0 & \delta_y^2 \end{bmatrix}\right) \quad (4)$$

Fig. 2. Counties in Southern California.

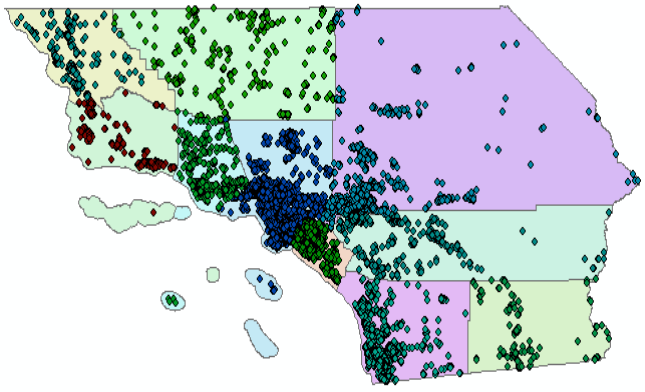

Fig. 3. Landmarks in Southern California.

In all the experiments, the test data are fixed (the original data) and the training data are noisy, where δ_x, δ_y for each point are randomly selected from [0, 0.0005°] or [0, 0.005°] for different uncertainty cases. There is a parameter defining the minimum candidate set size, called MCS_size. 1~15 nearest neighbor(s) are returned as the result to the query.

B. Comparison of OGMH and uncertain R-tree

The KNN search is made on OGMH, UR1, UR2 and plain R-tree. Their performance measures are precision, I/O cost and CPU cost. The experiments are performed under two different uncertainties (0.0005°, 0.005°) and two MCS_sizes (40, 60).

The program is written in C++ and the system configuration is as below:

System: Sun Microsystems sun4u
OS: Solaris 2.8
Memory: 2048MB

1) Precision:

$$Precision = \frac{\#\ of\ correct\ results\ actually\ returned}{\#\ of\ results\ should\ be\ returned}$$

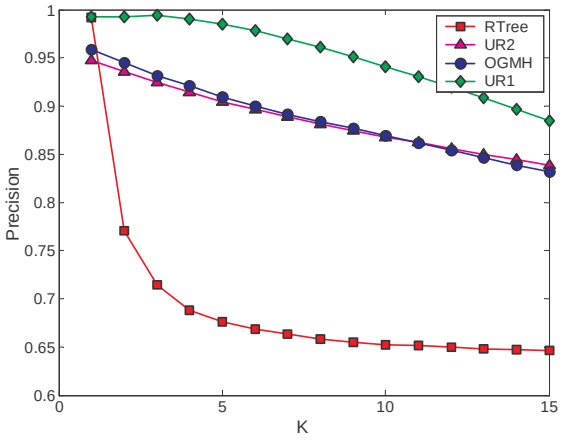

Fig. 4. σ= 0.0005, MCS_size = 40.

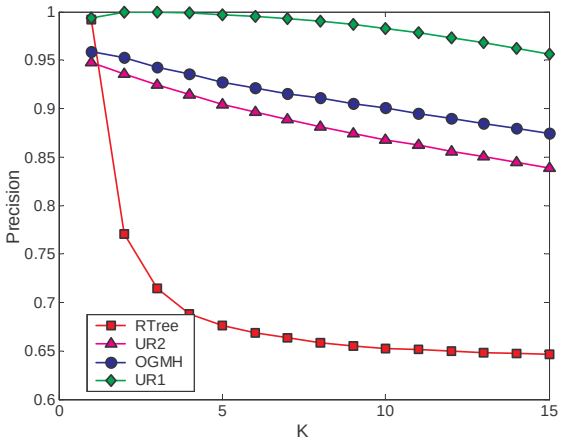

Fig. 5. σ= 0.0005, MCS size = 60.

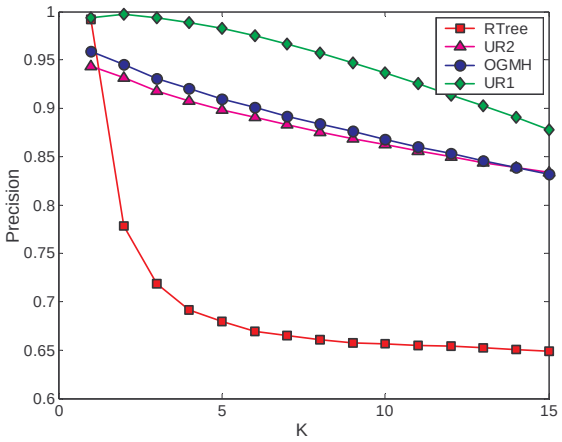

Fig. 6. σ= 0.005, MCS_size = 40.

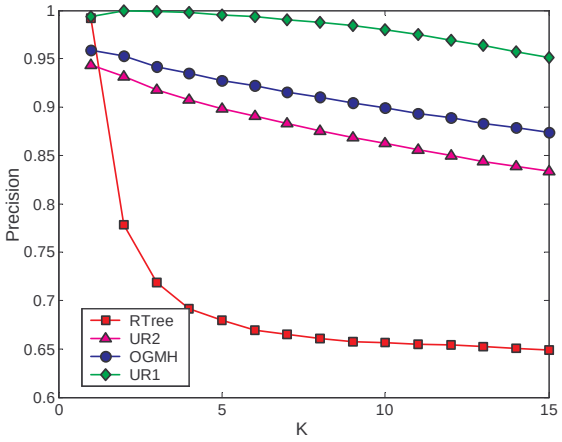

Fig. 7. σ= 0.005, MCS_size = 60.

From Fig. 4 - Fig. 7, we can make the following observations:

(a) UR1 always gives the best performance, followed by OGMH and UR2. As mentioned in Section II.D (Uncertain R-tree), UR2 and OGMH have the same filter strategy.

(b) OGMH has higher precision performance than UR2, especially when MCS_size is larger. So Gaussian Mixture model is more appropriate than MBRs in PDF indexing.

(c) Plain R-tree gives the worst precision and it is not acceptable, so in the following comparisons, plain R-tree is removed.

From Fig. 8 and Fig. 9 we can see that when the uncertainty increases from 0.0005° to 0.005° (MCS_size = 60), the precision performance of UR1 degrades much more than that of OGMH (5% vs. 0.15%). So OGMH is more stable than uncertain R-tree.

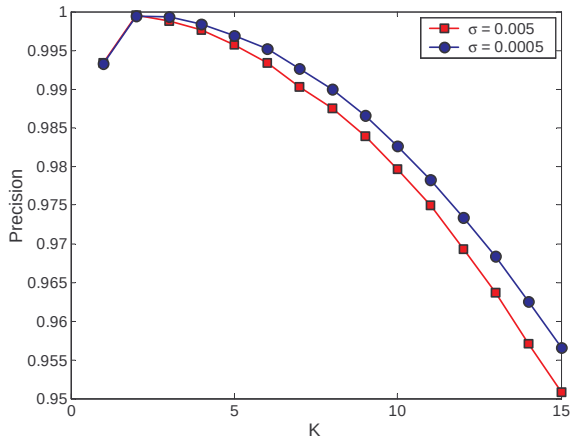

Fig. 8. UR1 precision on different σ.

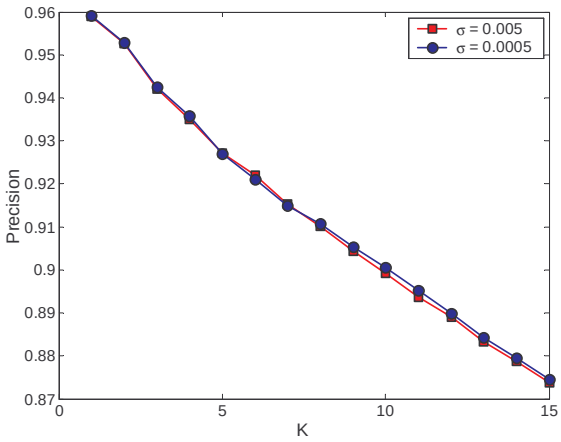

Fig. 9. OGMH precision on different σ.

*2) **I/O cost** -- the average page read/write for 1-NN query.*

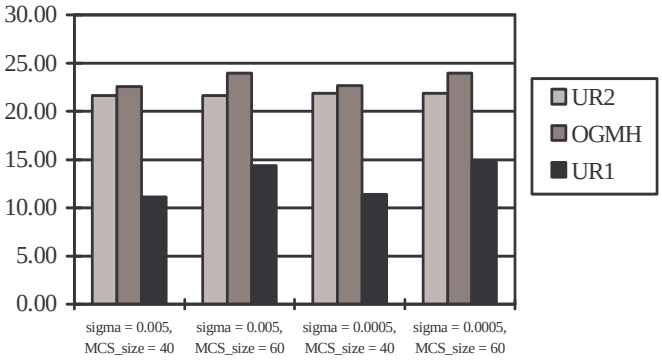

Fig. 10. I/O cost comparison among all indices.

As shown in Fig. 10, OGMH and UR2 need more I/O cost than UR1. The OGMH tree has more levels than R-tree (17 vs. 4). The more I/O cost of UR2 comes from the back tracking, but it is still comparable with OGMH.

3) ***CPU cost*** *-- average time (second) for 1-NN query.*

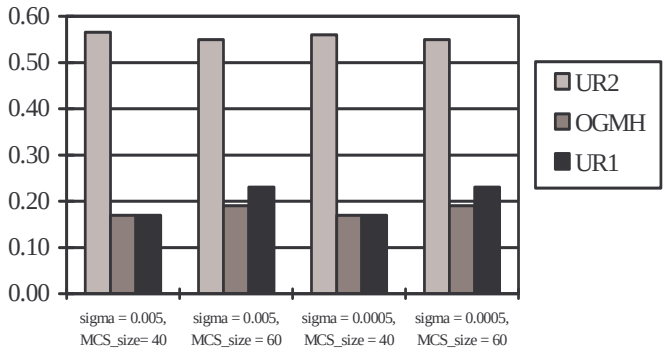

Fig. 11. CPU cost comparison among all indices.

Fig. 11 indicates that UR1 and OGMH are comparable on time complexity and they are more efficient than UR2, this is because the back tracking of UR2 is time consuming.

From the three comparisons above, we see that when the query is fixed, UR1 is the best in general, followed by OGMH and UR2. Plain R-tree is not acceptable with respect to precision performance.

But as mentioned in Section I, a complete system should be able to handle both fixed and uncertain queries. Thus, the choice of index structure depends on the application. If no uncertain query is asked, UR1 is the best choice, otherwise only OGMH is suitable.

IV. Conclusions

Uncertainty in spatial databases is getting more and more attention, but most of the existing approaches are based on relational models using fuzzy set theory. This method is only suitable for handling uncertainty in symbolic level. In order to support uncertainties at numeric and existence levels, we proposed a new uncertainty model and new indexing schemes.

In this paper, we represented uncertain objects with PDFs and constructed an *optimized Gaussian mixture hierarchy* based on Gaussian mixture model and an uncertain R-tree. After a comprehensive comparison based on KNN search precision, I/O cost and CPU cost, uncertain R-tree (UR1) is the best for fixed queries. But OGMH is suitable for both certain and uncertain queries. Moreover, the OGMH is not suitable for only spatial databases, but also for other multi-dimensional index applications like *content based image and video retrieval*.

References

[1] Rigaus, P., M. Scholl, and A. Voisard, Spatial Databases: with Application to GIS. San Francisco, California: Morgan Kaufmann, 2001.

[2] International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences. http://www.nrcse.washington.edu/ties/events/ties2004/default.html

[3] Zaniolo, C., et al., Introduction to Advanced Database Systems: Morgan-Kaufmann, 1997.

[4] Schneider, M. Uncertainty Management for Spatial Data in Databases: Fuzzy Spatial Data Types. in *6th Int. Symposium on Advances in Spatial Databases (SSD)*. 1999: Springer Verlag.

[5] Robinson, V.B., A Perspective on Managing Uncertainty in Geographic Information System with Fuzzy Sets. *IEEE Transactions in Geographic Information System*, 2003. 7(1): p. 211-215.

[6] Guttman, A. R-trees: A Dynamic Index Structure for Spatial Searching. in *SIGMOD*. 1984. Boston, MA,.

[7] Greene, D. An Implementation and Performance Analysis of Spatial Data Access Methods. in *Proceedings of the Fifth International Conference on Data Engineering*. 1989: IEEE Computer Society Washington, DC, USA.

[8] Beckmann, N., et al. The R*-tree: an efficient and robust access method for points and rectangles. in *ACM SIGMOD International Conference on Management of Data*. 1990. Atlantic City, NJ.

[9] Bhanu, B., R. Li, C. Ravishankar, M. Kurth, and J. Ni. Indexing Structure for Handling Uncertain Spatial Database. in *6th International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences*. 2004. Portland, Maine, USA.

[10] Brown, R.H. and E. Ehrlich, TIGER/Line(TM) Files, 1992. 1993: Washington, D. C. http://www.census.gov/geo/www/tiger/content.html (03/30/04)

[11] Duda, R.O., Hart, P.E., and D.G. Stork, Pattern Classification. New York: A Wiley-Interscience Publication, 2000.

[12] Figueriedo, M.A. and A. Jain, Unsupervised Learning of Finite Mixture Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002. 24(3): p. 381-396.