Indexing Structure For Handling Uncertain Spatial Data

Bir Bhanu, Rui Li, Chinya Ravishankar, Michael Kurth and Jinfeng Ni
Center for Research in Intelligent Systems
University of California, Riverside, 92521
bhanu, rli@vislab.ucr.edu ; ravi, kurthm, jni@cs.ucr.edu

Keywords: Geographic information system, uncertainty, probability density function, optimized Gaussian mixture hierarchy, mixture model, R-tree, Mojave desert, desert tortoise

Abstract:
Consideration of uncertainty in manipulation and management of spatial data is important. Unlike traditional fuzzy approaches, in this paper we use a probability-based method to model and index uncertain data in the application of Mojave desert endangered species protection. The query is a feature vector describing the habitat for certain species, and we are interested in finding geographic locations suitable for that species. We select appropriate layers of the geo-spatial data affecting species life, called habitat features, and model the uncertainty for each feature as a probability density function (PDF). We partition the geographic area into grids, assign an uncertain feature vector to each cell, and develop a filter-and-refine indexing method. The filter part is a bottom-up binary tree based on the automated clustering result obtained using the EM algorithm. The refine part processes the filtered results based on the "similarity" between the query and properties of each cell. We compare the performance of our proposed indexing structure with the intersection method from Mojave Desert Ecosystem Program (MDEP); our method is more selective and efficient.

1. 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 (Rigaus, Scholl et al. 2001). In spatial database, it is generally agreed that there are several types of error (uncertainty) which result to the overall accuracy of final products. But understanding of the causes of these errors and the relationships between them are not well understood. In (Hunter and Beard 1992), authors present several types of error and their relationships. Certain actions such as data collection, processing and misuse are responsible for causing error in data. The errors are classified into positional errors and attribute errors. These errors then contribute to the final product error which users ultimately wish to determine. Positional error is considered to be the error in the location description of an object, while attribute error is associated with the accuracy of identification, or thematic description, of that object. From an accuracy perspective, errors in location and attributes are often determined by quite different processes. In our work, we mainly focus on positional error. At present there isn't a complete system which can store and operate on uncertain data.

Our work focuses on developing new theory, tools and technology to support uncertainty data and uncertain queries in spatial databases. In this paper, we present a probabilistic method to model uncertain data, in which every object in spatial database is represented by a PDF. We introduce a similarity measure between uncertain (certain) data, and design a new indexing structure, called *Optimized Gaussian Mixture Hierarchy (OGMH)*, based on unsupervised clustering of the feature vectors. We also build a real-world application of our uncertainty model and index.

Section 2 briefly introduces some related works. The uncertainty representation and index construction are explained in Section 3. Section 4 gives the experimental results on Mojave desert endangered species protection. Finally, Section 5 presents the conclusion of the paper.

2. Related works

Most of the existing approaches for management of probabilistic data are based on the relational model and use fuzzy set theory (Schneider 1999; Robinson 2003). 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 function to represent uncertainty, which means every object in a GIS database is a random variable described by its distribution.

For spatial database index, R-tree is the most often used structure. It is a depth-balanced tree whose nodes are represented by *Minimum Bounding Rectangles (MBR)*. It is invented by (Guttman 1984) and (Greene 1989; Beckmann, Kriegel et al. 1990) did some optimization, which reduce margin and MBR overlap. But R-tree family of trees indexes fixed data only. In order to handle the uncertain information related to each object, a filter-refine query strategy can be used. For example, uncertain R-tree is constructed using the fixed data. The uncertain information is attached to every data entry. For a query, in the filter step it is like normal point query, but it stops at a certain node based on some strategy, e.g. when the node contains enough data entry for refinement. All the data entry belonging to this node construct the candidate set. Then refinement is done using the uncertainty attached to each data entry in the candidate set to give the final query result. This uncertain R-tree works for certain query and it has comparable performance with our Optimized Gaussian Mixture Hierarchy presented in this paper, but it has no ability to handle uncertain query. To build a system which handles both certain and uncertain data and query, uncertain R-tree is not sufficient.

3. Technical approach

3.1. Uncertainty modeling

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

We use PDF to represent each uncertain object. If it needs n features to describe, then it is an n dimensional (feature vector) random variable, as given by equation 1.

$$\mathbf{f}^i = \left[f_1^i, f_2^i, \cdots, f_n^i \right]^T \quad \text{where } i = 1, \ldots, N \text{ , } N \text{ is the number of objects} \tag{1}$$

As of now, for simplicity, we assume the features are independent of each other, and we know PDFs of each $f_j^i, j = 1,...,n$. Getting the PDFs is called uncertainty modeling. Our system can handle both certain and uncertain data. So we don't give a specific name to uncertain data. Feature vector is the name for both certain and uncertain data in this paper.

3.2. Similarity measure

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 as the **probability that two random variables are the same** , as given below:

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

In this equation, **D** and **Q** are two objects. Usually **D** stands for data 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**. $\mathbf{D} = [D_1, \ldots, D_n]$, $\mathbf{Q} = [Q_1, \ldots, Q_n]$ and $\Delta = [\Delta_1, \ldots, \Delta_n]$.

Since both data and query can be certain and uncertain, there are four combinations. We construct a system which handles all these four scenarios. So we define different similarity functions for them, which is shown in Figure 1. In this figure, F indicates for the *cumulative distribution function (CDF)* and f is the PDF.

Query \ Data	Certain	Uncertain
Certain	**Physical meaning:** |D - Q| < Δ ? True / False D Q **Math Model:** P = Pcc(Q,D, Δ) = |D - Q| < Δ ? 1:0	**Physical meaning:** |D-Q| < Δ with probability P D P Q **Math Model:** P = Puc(Q,D, Δ) = Pr{|D-Q| < Δ} = $F_D(Q+\Delta)-F_D(Q-\Delta)$
Uncertain	**Physical meaning:** |D-Q| < Δ with probability P Q P D **Math Model:** P = Puc(Q,D, Δ) = Pr{|Q-D| < Δ} = $F_Q(D+\Delta)-F_Q(D-\Delta)$	**Physical meaning:** |D-Q| < Δ with probability P D Q **Math Model:** P = Puu(Q, D, Δ) = $\int$ Pr{ |Q-t|< Δ} · $f_D(t)$ dt = $\int [F_Q(t+\Delta)-F_Q(t-\Delta)] \cdot f_D(t)dt$

Figure 1. Physical meaning and math model for four scenarios in our system.

When all the features are independent of each other, the right part of equation 2 is explained in equation 3. So the similarity is the probability that all the features of **D** is within a specified vicinity of **Q**.

$$Pr(|\mathbf{D}-\mathbf{Q}| < \Delta) \equiv Pr[|D_1 - Q_1| < \Delta_1] \cdot [\cdots], Pr[|D_n - Q_n| < \Delta_n] \qquad (3)$$

When the distributions happen to be Gaussian, Puc, Pcu and Puu can be replaced by Mahalanobis distance and Bhattacharyya distance (Duda, Hart et al. 2000). But in our experiment, we calculate the similarities using these definitions directly.

3.3. System overview

Our system is shown in Figure 2. There are four major parts in this system: the uncertainty modeling, indexing, query pre-processing and KNN search.

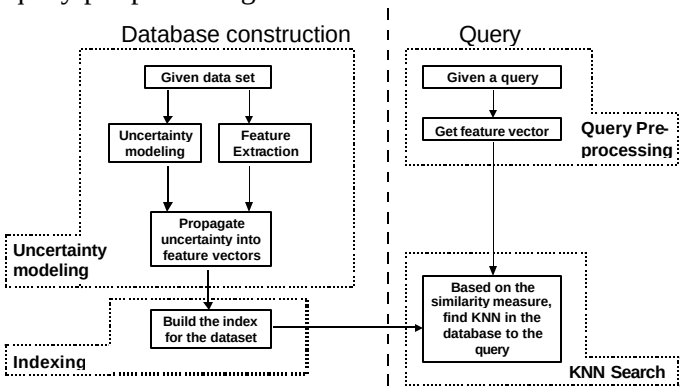

Figure 2. System diagram.

Given a training dataset, first we need to model errors originated from source data and caused by data processing (see Section 1). At the same time, we need to decide which features we are going to use, and then propagate the errors layer by layer in the GIS system to the feature level so

that an object is represented as an n dimensional feature vector. After uncertainty modeling, the dataset is ready, as given by equation 1 and we construct an index based on these feature vectors. A Gaussian mixture hierarchy is constructed to index all these feature vectors (see Section 3.4). The query object after pre-processing is also a feature vector. It is indexed into the database to find the *K nearest neighbors (KNN)*. That is, given a query, find the K closest objects in the database to the query. The "closeness" is decided by the certain similarity measure explained in section 3.2. Here, we are only interested in KNN because it is the basis for other comprehensive queries. The detail of index construction and KNN are explained in the next section.

3.4. Indexing structure – Optimized Gaussian Mixture Hierarchy

The index construction and filter-refinement query are shown in Figure 3. Clustering, tree construction and KNN search are explained in the following.

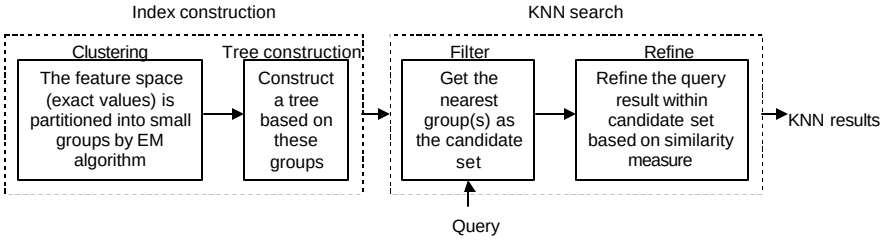

Figure 3. Indexing structure and filter-refinement query procedure.

1. Clustering: Given a set of feature vectors, in the first step of index construction shown in Figure 3, we only consider the mean for each of them. So it is similar to a common multi-dimensional indexing problem. The means of the feature vectors in the database follow some distribution. From the probability theory view point, any distribution can be approximated by a weighted sum of several Gaussian distributions (Duda, Hart et al. 2000), as shown in equation 4, where distribution $p(x)$ is approximated by C Gaussian distributions: $f_i(x)$, $i = 1, \cdots, C$. a_i is the weight for each Gaussian distribution represented by mean vector u_i and covariance matrix Σ_i.

$$p(x) = \sum_{i}^{C} a_i f_i(x) = \sum_{i}^{C} a_i N(u_i, \Sigma_i) \qquad (4)$$

   (Figueriedo and Jain 2002) 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. In another word, the whole dataset is clustered into several groups.

2. Tree Construction: Figure 4 is the algorithm to construct a bottom-up binary tree based on the clustering results. In this algorithm, the groups obtained from clustering algorithm using object means build the leaf level. To construct the level right above, we find a partition so that every two groups are merged into a new one. If the component number is odd or some components are too far away from each other, this left over component or the separated components are called " isolated" and they directly move up into the above level. This process is repeated level by level until all components are merged into one group, the root.

   In this paper, we do the agglomeration exhaustively: get all the possible partitions, find the Bhattacharyya distance within each new group (including two components), then sum up all the distances for each group as a partition criterion, which is called the total Bhattacharyya distance. The "best" partition is the one that minimizes total Bhattacharyya distance.

Tree construction:
Input: feature vectors
Output: a binary tree
for all components in each level of the tree
1. *partition the component to groups containing 1~2 components each to minimize the total Bhattacharyya distance*
2. *assign a merged group as a new node right above current level*

end
repeat above process until all the components are agglomerated into the root
for all leaves in the tree
repeat the tree construction if it is too big
end

Tree parameter assignment:
Input: the built binary tree
Output: parameters of each node
for each node other than leaf nodes
Put Gaussian mixture of all its leaf offspring as its description
end

Figure 4. Index construction algorithm.

function KNN(OGMH, q)
Input: index OGMH; query q
Output: KNN to q
```
curr_node = OGMH.root;
while curr_node is not a leaf node
   P_L = GM probability (curr_node→ left, q);
   P_R = GM probability (curr_node→ right, q);
   if P_L > P_R  curr_node = curr_node→ left;
   else  curr_node = curr_node→right;
end
for i = 1:curr_node → size
   P[i] = curr_node → f_i(q)
end
sort(P);
return objectss corresponding to the K largest
elements in P;
end function
```
function GM probability(node, q)
Input: a node in OGMH represented by its Gaussian mixture: $\sum_i \alpha_i f_i(x)$; query q,

Output: the probability that q has to the node distribution
```
P = 0;
for i = 1: node → size
   if q is within  3d of node → f_i
      P += (node→ α_i)(node→ f_i(q))
   return P;
end
end function
```

Figure 5. KNN search algorithm.

Usually the clustering results are heavily unbalanced, which means some leaves have much more objects than the others, say 10000 verses 10. If this is the case, we repeat the clustering and tree construction for the large leaves and replace them by the new trees until all the leaf nodes have comparable sizes.

In our binary tree, every leaf node has a Gaussian distribution, so we represent every inner node by the Gaussian mixture of all its offspring leaves.

3. KNN search: When the tree is ready, users can use it to process a query. The query process is shown in Figure 5. When a query comes in, it parses the tree until a leaf is reached. All the data entry attached with that leaf is called a candidate set. Then the uncertain information of each data entry is used to search for K nearest neighbors.

Using mixture model to represent the inner nodes of a tree is accurate, but calculation of all the components is not efficient because the higher the node, the more components it has. So we calculate the similarity only when the query is with 3σ of a node distribution. This is how the "Optimization" taken place for OGMH.

4. Experiment results

In this section, we provide an application of our uncertain representation and index structure. It is the Mojave desert endangered species (desert tortoise) protection.

4.1. Mojave Desert species protection dataset

The Mojave Desert eco-region extends from eastern California to northwestern Arizona, southern Nevada, and southwestern Utah. There are hundreds of endangered species. Desert tortoise

(Gopherus agassizii) is one of them. It has inhabited this region for over one million years, but its population reduced dramatically during the last two decades due to both people and environment change (Humphrey 1987). It was listed as "threatened" under the California Endangered Species Act in 1989 and the federal Endangered Species Act in 1990.

Scientists have been trying to protect them from their extinction. They are interested in finding what factors impact desert tortoises and finding where desert tortoises live so they can protect the land. *Mojave Desert Ecosystem Program (MDEP)* is the first to organize a detailed, environmentally oriented, digital geographic database set over an entire eco-region. They have some progress on the tortoise protection. Based on some information about the tortoise habitats, MDEP researchers use ArcInfo™ to select the suitable places as the tortoise habitats. After this pre-processing, the area of interest reduces a lot. As the last step, they go to these areas and see whether there are tortoises and try to protect them. This trial and error mode is time consuming, expensive and they have to try this over and over again if they are interested in species other than tortoise.

Using our index system, we can find desert tortoise more efficiently. Our scheme is described in Figure 6. Our approach is to cut the whole Mojave into grids, describe each cell using a feature vector, and index these feature vectors. Then for a given interested species, find their habitat description and use a feature vector to describe it, after that do a KNN search. Here the feature vectors in the database are fixed with a confidence and the query is uncertain.

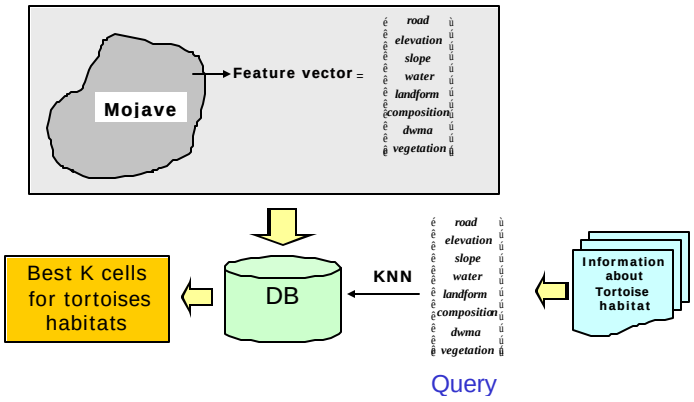

Figure 6. Mojave Desert endangered species protection plan.

4.2. Uncertainty for both data and query

From (Jennings 1993; Esque 1994; Avery, Lovich et al. 1998; Boarman 2002) we found basic information about tortoise habitats and geographic features relevant to these habitats.

Table 1 shows the features we use to represent each cell. First column is the feature type. Second column is the value range of each feature. For experiment convenience, all the feature values are normalized into the range ~[0, 100], as shown in the third column.

Table 1. Features used for Mojave Desert tortoise protection.

	Feature	Value	Normalized Value
1	Major roads	0,1,2	0, 50, 100
2	Elevation	-86~3000m	-4.3~150
3	Slope	0~71.3 degree	0~142.6
4	Water	0, 1, 2, 3, 4, 5	0, 20, 40, 60, 80, 100
5	Landform	1, …, 33	3, …, 99
6	Composition	0, 1	0, 100
7	DWMA*	0, 1	0, 100
8	Vegetation	1, …, 34	3, …, 102

* DWMA: the area that MDEP has right to access, which belongs to the National Park and Bureau of Land Management

Our database is set up for an area of 6379.56 km^2 located at the center of Mojave desert. It is cut into 70884 cells. Each of them is 300m by 300m with a certain feature vector and a confidence. Since all the feature information is given for 30m by 30m cells. So the feature vector for the large cell is the one that has the majority value. The confidence is the occurrence of this majority feature vector. The query PDF is defined based on those habitat information.

4.3. Results – places suitable for desert tortoise

After an OGMH is built, we make a query for the tortoise habitat. The simple intersection result based on the basic information on tortoise habitat is shown by the purple area in Figure 7. The background is the slope. Light color means high slope and dark color indicates low slope.

When $\Delta = [10,10,20,1,40,40,1]$, our query result is shown by the orange areas in Figure 8. All these areas have non-zero probability, which means all these areas are suitable for tortoise. And compared to Figure 7, our query result gives 67.72% improvement (reduction in the area without missing out potential tortoise habitat area) over the simple intersection result shown in Figure 7. The probabilities in Figure 8 extend from 0.000201975 to 0.00040395. When the user are interested in the areas with probability over 0.0001, all these areas meet the requirement. When the user asks for areas with probability over 0.0003, only the red areas in Figure 9 are suitable.

When Δ changes, query result also changes. When $\Delta = [10,15,20,1,40,40,1]$, red areas in Figure 10 have probability over 0.0003. Compared with Figure 8, we can see more areas are suitable when Δ is higher.

Figure 7. Simple intersection result from MDEP.

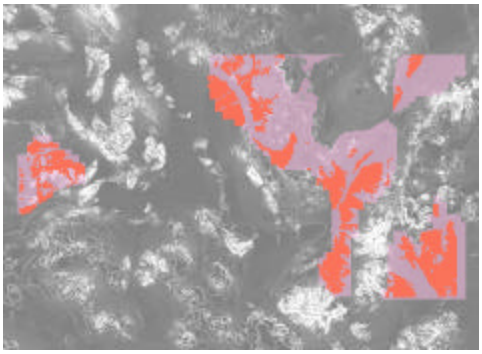

Figure 8. Query result with probability > 0.0001(orange) when $\Delta = [10,10,20,1,40,40,1]$.

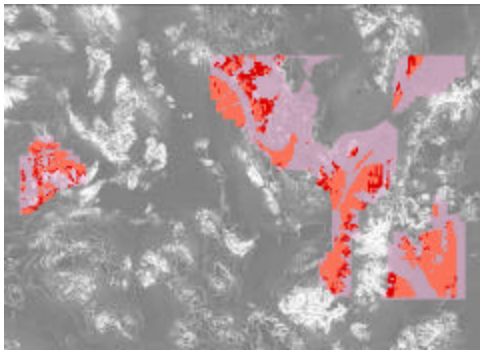

Figure 9. Query result with probability > 0.0003(red) and <0.0003 (orange) when $\Delta = [10,10,20,1,40,40,1]$.

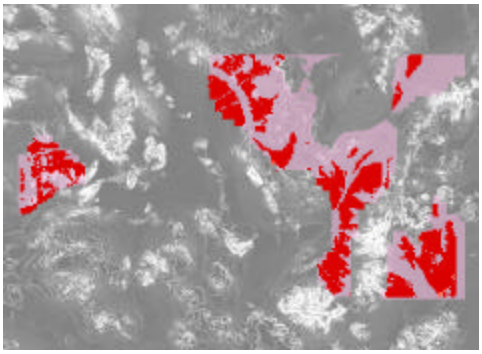

Figure 10. Query result with probability >0.0003 (red) when $\Delta = [10,15,20,1,40,40,1]$.

From Figure 8, we find tortoises does not like high slope. This is consistent with the information about tortoise habitats, and further, when lower Δ is set, the results are more selective. Thus, the use of our scheme saves time and expenses.

5. Conclusions

Uncertainty is related to the data quality and decision making. In this paper, we presented a way to represent, index and query uncertain spatial data.

We presented an application on Mojave desert endangered species protection. Compared with the result from MEDP on desert tortoise, our method is more selective and more efficient. This method is not only for the desert tortoise, but can be applied for other species. Using our method, we can easily find habitats for a certain species and define a confidence for each result.

The uncertainty modeling is a key part in the uncertainty handling work and it is our future research. Handling comprehensive queries is another work we are currently investigating.

By adding support for uncertainty in the database management system this research will substantially increase the power and flexibility of GIS databases.

Acknowledgements:

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.

References:

Avery, H. W., J. E. Lovich, et al. 1998. in. prep. Effects of Cattle Grazing on Vegetation and Soils in the Eastern Mojave Desert. U.S.Geological Survey. Riverside, CA,

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

Boarman, W. I. 2002. Desert Tortoises. U.S.Geological Survey. W. I. Boarman and K. Bearman. Sacramento, CA, Western Ecological Research Center

Duda, R. O., Hart, P.E., et al. 2000. *Pattern Classification*. New York, A Wiley-Interscience Publication.

Esque, T. 1994. Diet and Diet Selection of the Desert Tortoise (Gopherus agassizzii) in the Northeast Mojave Desert. Fort Collins, CO,, Colorado State University

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

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

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

Humphrey, R. R. 1987. *90 Years and 535 Miles: Vegetation Changes Along the Mexican Border*, University of New Mexico Press.

Hunter, G. J. and K. Beard 1992. "Understanding Error in Spatial Databases." *The Austrailan Surveyor* 37(2): 108-19.

Jennings, W. 1993. Foraging Ecology of the Desert Tortoise (Gopherus agassizzii) in Western Mojave Desert. Arlington, University of Texas

Rigaus, P., M. Scholl, et al. 2001. *Spatial Databases: with Application to GIS.* San Francisco, California, Morgan Kaufmann.

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

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