Probabilistic Spatial Database Operations

Jinfeng Ni[1], Chinya V. Ravishankar[1], and Bir Bhanu[2]

[1] Department of Computer Science & Engineering
[2] Department of Electrical Engineering
University of California, Riverside
Riverside, CA 92521, USA

Abstract. Spatial databases typically assume that the positional attributes of spatial objects are precisely known. In practice, however, they are known only approximately, with the error depending on the nature of the measurement and the source of data. In this paper, we address the problem how to perform spatial database operations in the presence of uncertainty. We first discuss a probabilistic spatial data model to represent the positional uncertainty. We then present a method for performing the probabilistic spatial join operations, which, given two uncertain data sets, find all pairs of polygons whose probability of overlap is larger than a given threshold. This method uses an R-tree based probabilistic index structure (PrR-tree) to support probabilistic filtering, and an efficient algorithm to compute the intersection probability between two uncertain polygons for the refinement step. Our experiments show that our method achieves higher accuracy than methods based on traditional spatial joins, while reducing overall cost by a factor of more than two.

1 Introduction

The past decade has witnessed a significant increase in work on spatial database management systems. The field has gained significance both in traditional applications such as urban planning, as well as in emerging areas such as mobile ad-hoc networks. The increase in GPS-based has also been a significant boost to this field. A standard assumption in the spatial database domain has been that the positional attributes of spatial objects are known precisely. Unfortunately, this is often not the case, and various kinds of uncertainty may be associated with spatial data. GIS researchers, in particular, have long recognized that spatial data are rarely error-free [8, 30]. Five important components of spatial data quality are categorized in the National Standard for Spatial Data Accuracy(NSSDA) [6]: positional accuracy, attribute accuracy, logical consistency, completeness and lineage.

In this paper, we focus on positional accuracy. The positional accuracy represents how closely the coordinate descriptions of spatial objects match their actual positions (or ground truth). The term *positional uncertainty* or *positional error* is often used to refer the difference between the digital representations of spatial objects and the actual locations. Numerous factors may contribute

T. Hadzilacos et al. (Eds.): SSTD 2003, LNCS 2750, pp. 140–159, 2003.

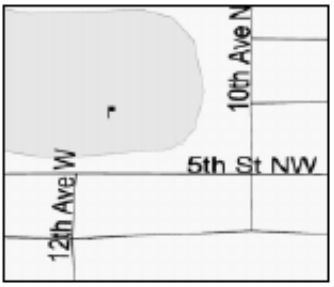


Fig. 1. Data problems: Overlay places building (flag) within lake (grey area), instead of on 5th Street [18]

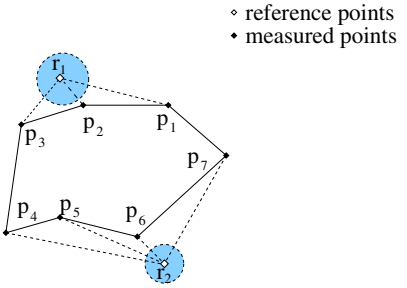


Fig. 2. A surveying network with 2 reference points

to the positional uncertainty in spatial data. In practice, there are a variety of ways to capture spatial data, each having its own sources of errors. Several widely used data sources are land surveys, GPS information, photography, remotely sensed satellite images, and digitizing or scanning a paper map. As indicated in [18], errors may be introduced during these processes by the digitizing methods used, the source material characteristics, generalizations, symbol interpretations, specifications for aerial photography, aerotriangulation techniques, ground control reliability, photogrammetric characteristics, scribling precision, resolution, processing algorithms, printing limitations, and so on.

Traditional techniques for querying spatial data need to be revised, since uncertainties in spatial data may affect the accuracy of the answers to queries. It is in fact possible for traditional database queries to produce wildly incorrect or invalid results based on the measured data. To illustrate this point, consider the example in Figure 1, taken from a document by Minnesota Planning [18]. In this example, the locations of buildings, shown as the black flag, are measured by GPS receivers, and then overlaid over a digital base map containing roads, and lakes. However, this operation results in some buildings being located in the middle of lakes! The error in this case arises from the differences in data sources and their differing accuracies. The base map was accurate to 167 feet, while the GPS data was accurate only to 300 feet.

1.1 Managing Uncertainty

It is generally agreed that positional uncertainties should be reported along with data, so that users may evaluate the suitability of the data for specific applications during decision making. One recent attempt is the widely-accepted standard by NSSDA [6]. NSSDA provides a well-defined statistical method and promotes the use of root-mean-square error (RMSE) to estimate and report the positional accuracy. RMSE is the square root of the average of the set of squared differences between dataset coordinate values and "ground truth" values

of coordinates from an independent source of higher accuracy for the same points. For example, [18] shows how to apply NSSDA to report the positional accuracy in a variety of data. A typical positional accuracy statement is *using the National Standard for Spatial Data Accuracy, the data set tested 0.181meters horizontal accuracy at 95% confidence level.*

Unfortunately, RMSE is merely a gross, global measure of uncertainty, averaged over all points in a given data set. It therefore fails to characterize the local or spatial structure of uncertainty [7], and is inadequate for analysis of uncertainty. It remains a research issue to find an appropriate model to estimate and report the spatial structure of positional uncertainty.

From the perspective of spatial database systems, there are two important issues to address: modeling and reporting the positional uncertainty, and evaluating spatial queries over uncertain spatial data. In our work, we develop a probabilistic spatial data model (PSDM) for polygon data that associates probability distributions with the positional attributes. In PSDM, each polygon is partitioned into k disjoint independent *chunks*. Vertices from the same chunk have fully correlated uncertainties, while vertices from different chunks are independent. Furthermore, each chunk's uncertainty is assumed to follow a circular normal distribution.

Given the inherent positional uncertainties in the spatial data, exact match responses to queries are not meaningful, but probabilistic statements about query results are appropriate. Probabilistic answers to range and nearest-neighbor queries over moving point objects with uncertain locations were used in [29, 25]. In contrast, our work is concerned with probabilistic responses to queries when positional uncertainty exists over polygon boundaries. We also consider how to perform spatial operations in the presence of uncertainty.

In particular, we focus on evaluating *probabilistic spatial joins*. The response to a probabilistic spatial join consists of object pairs and the intersection probability between each pair. For example, consider the set R of state parks in California, and the set S of burned areas from recent forest fires. An example of a probabilistic spatial join query between R and S is: *Find all burned areas that overlap state parks in California with a probability of at least 0.8, and compute their overlap probability.*

As with traditional spatial joins, we evaluate probabilistic spatial join in two steps: filtering and refinement [5]. We propose the Probabilistic R-tree (PrR-tree) index, which supports a probabilistic filter step. We also propose an efficient algorithm to obtain the intersection probability between two candidate polygons for the refinement step.

This paper is organized as follows. Section 2 discusses related work. Section 3 presents PSDM, a probabilistic spatial data model for polygon's uncertainty. Section 4 presents our filtering and refinement algorithms for evaluating probabilsitc spatial joins. Experimental results are given in Section 5. Section 6 concludes the paper.

2 Related Work

Much work has been done on spatial databases, especially on the evaluation of spatial joins. Typically, spatial objects are approximated and indexed using their minimal bounding rectangle (MBR). A spatial join query is processed in two steps: *filtering*, where all the MBR pairs satisfying the join predicative are retrieved, and *refinement*, where the exact geometry of the objects is used to determine whether the object pair is a true hit or a false hit.

Consequently, previous work on spatial join has focussed on either the filtering or refinement step. For example, [5] proposed an algorithm to perform spatial join of two datasets indexed by the R-tree and its variations [2, 21]. Similarly, [15, 17] proposed algorithms to join one pre-indexed dataset with one unindexed dataset, and [16, 12, 23] proposed algorithms for the cases where neither dataset has existing indices. Other work focuses on how to improve the performance of the refinement step, which is much more expensive than the filtering step in terms of both I/O and computation costs. A common approach for the refinement is to approximate the spatial objects with a better approximation than MBR provides. For instance, [4] proposed a multi-step framework for processing spatial join, using approximations such as convex hulls, and minimum bounding m-corner boxes. In contrast, [9] proposed the Symbolic Intersect Detection (SID) technique to reduce the expensive computation cost of exact geometry test. Raster approximations are proposed for refinement step in [31].

In contrast, little work exists on query evaluation for spatial objects with uncertain position. In [13], the ϵ-*band* model is used to model the probabilistic distribution of polygon boundaries, and the upper bound of the probability for the point-in-polygon query is derived. The ϵ-band model requires one parameter to describe the uncertainty of a database, and simply assumes that all points, including the intermediate points between two vertices have the same uncertainty. However, as [26] indicates, this assumption may not be valid. The work in [26] found that the uncertainty of the middle points is lower than that of the two vertices. Also, [13] showed no experimental results for their approach, so it is unclear how well their approach works. Furthermore, point-in-polygon queries are very simple, and are not be applicable to spatial join in any direct way. The *G-Band* was proposed in [27] to describe the positional uncertainty of line segments, assuming that the uncertainties of the endpoints follow two-dimensional Normal distributions. One good feature of G-Band model is that it accounts for the dependence between the two end-points of segments. However, it may impose high computation cost when being applied to query evaluation.

Due to the complexity of spatial uncertainty, Openshaw [22] recommended the use of *Monte Carlo* techniques to evaluate the spatial uncertainty inherited in the spatial dataset. With this approach, a set of equally probable realizations of the spatial objects are generated, and these realizations can be used to evaluate the uncertainty associated with the spatial data. Although this is a computation-intensive method, Openshaw argued that due to the improvement of computer technology, this approach would find its way in more and more applications. For instance, [10, 7] demonstrated how to evaluate point-in-polygon queries using

Table 1. Summary of notation

Notation	Description
p_i	Vertex i
(x_i, y_i)	The x and y coordinates of vertex p_i
σ_i	The standard dviation of the circular normal associated with p_i
$\langle P \rangle$	A polygon P.
$\langle P : j \rangle$	The j-th *chunk* of polygon P
$\langle p_k, p_{k+1}, \cdots, p_{k+l} \rangle$	Chunk with vertices $p_k, p_{k+1}, \cdots, p_{k+l}$ (numbered anti-clockwise)
$[\|x^{\vdash}, y^{\vdash}, x^{\dashv}, y^{\dashv}\|]$	MBR with lower left corner at $(x^{\vdash}, y^{\vdash})$ and upper right corner $(x^{\dashv}, y^{\dashv})$
$\sigma_x^{\vdash}, \sigma_y^{\vdash}, \sigma_x^{\dashv}, \sigma_y^{\dashv}$	The standard deviations of respective MBR corners X and Y axes.
γ	Confidence threshold for probabilistic queries

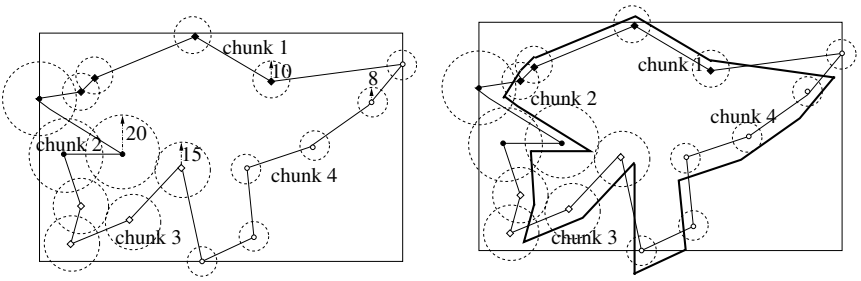
a Monto Carlo method. Despite great improvements in computer technology, Monte Carlo method is still not usable in large spatial databases. One possible use of this approach, suggested by Openshaw, is to serve as a benchmark for the evaluation of new techniques. Indeed, we use the result of Monte Carlo method to generate the ground truth values in our experiments.

The notion of probabilistic answers to queries over uncertain data was introduced in [29, 25] for range queries and nearest-neighbor queries when handling with the uncertainty associated with the location of moving point objects. Each query returns a set of tuples in the form of (O, p), where O is the object, and p is the probability that O satisfies the query predicate. The uncertainty of moving objects also was discussed in [28], where the uncertainty of object locations was modeled as a 3D cylindrical body along the trajectory, and semantics *sometime*, *always*, *everywhere*, *somewhere*, *possibly* and *definitely* were added into the traditional range queries. In [24], Pfoser et. al reported how to represent the uncertainty of moving objects introduced by the sampling technique and interpolation, and presented a filter-and-refine approach for probabilistic range queries.

3 Probabilistic Spatial Data Model

Traditional spatial databases typically use three primitives to represent spatial extent: points, lines, and polygons. Consequently, the uncertainty of the spatial extent can be described via the probability distributions of the vertices defining these three primitive objects. We recognize three requirements for an ideal probabilistic spatial data model: *simplictity, accuracy*, and *efficiecny*. Simplicity dictates that few parameters must be needed to describe the uncertainty, making it easier to incorporate probabilistic spatial data models into current spatial databases. Accuracy requires that the model should characterize the uncertainty with the greatest fidelity. Efficiency dictates that cost of processing the uncertainty during query execution should be reasonable.

In this section, we will present a probabilsitc spatial data model (PSDM) which meets these requirements. Table 1 is the list of notations used in this paper.



(a) A polygon with four chunks

(b) A possible realization of the uncertain polygon

Fig. 3. An uncertain polygon with 4 chunks, and a possible realization for this polygon

3.1 Positional Uncertainty Model

Our model begins with a partitioning of polygons into contiguous series of vertices called *chunks*. The positions of vertices from the same chunk are perfectly correlated, and positions of vertices from different chunks are independent. This model is reasonable in many practical scenarios since positions of vertices on polygons defining features are frequently assigned locations with respect to some standard reference points.

Consider Figure 2, in which a surveyor is mapping out a polygonal region with seven vertices. Because of site characteristics such as landmark visibility and distance, the surveyor has chosen to obtain the positions of vertices p_1, p_2, p_3 with respect to a reference point r_1, and the positions of p_4, p_5, p_6, p_7 with respect to a different reference point r_2. One approach is to obtain the locations of vertices relative to reference points via trigonometric calculations using measurements of distances and angles, the accuracy associated with which approach can be quite high. It is therefore reasonable to assert that the positional uncertainties in the locations of these vertices is largely determined by the uncertianties in the locations of the reference points. In this case, our chunks are $\langle p_1, p_2, p_3 \rangle$ and $\langle p_4, p_5, p_6, p_7 \rangle$. The errors in the postions of these chunks are uncorrelated as long the positions of r_1 and r_2 are.

Errors can occur both systematically and randomly [11, 3]. Generally, systematic errors, in the form of outliers, biases, blunders, do not follow a distribution function, and can be removed through a posteriori techniques such as calibration. Random errors commonly arise during measurement and tend to be distributed Normally [3]. We consider only random errors, and assume that the positional uncertainty of a point follows a circular Normal distribution, as in previous work [13, 14].

Definition 1. ***Uncertainty of point****: Let p be a point in 2 dimensional space whose position is uncertain. If σ is the uncertainty parameter associated with p, then the probability that p is located within a circle of radius r centered at p is given by the circular Normal distribution* $\Pr_p(r) = 1 - e^{-\frac{r^2}{2\sigma^2}}$.

Now, consider the uncertainty associated with a polygon $\langle P \rangle$ with n vertices $p_1, p_2, \ldots, p_n$. If we partition these vertices into the k disjoint chunks $\langle P:1 \rangle$, $\langle P:2 \rangle, \ldots, \langle P:k \rangle$, all the vertices in any chunk move as a unit. That is, if chunk $\langle P:j \rangle$ comprises the contiguous vertices $\langle p_{s_j}, p_{s_{j+1}}, \cdots, p_{s_{j+l}} \rangle$, the positions of any two of these vertices are perfectly correlated. Consequently, if any vertex in this set has associated uncertainty parameter σ, every other vertex in the chunk also has uncertainty σ. On the other hand, the vertices from different chunk are independent.

Figure 3(a) shows a polygon $\langle P \rangle$ with four chunks, where the dotted circles have radius σ, the uncertainty parameter of the corresponding chunk. In Figure 3(b) the dark polyon is a possible realization for $\langle P \rangle$. Notice that all the vertices in a chunk move in tandem.

Various types of queries over the uncertain spatial data can be defined under this model. It is also natural to assign a probability value to each query result. A similar approache is adopted in [29, 25], where methods are presented for evaluating probabilistic range queries and nearest neighbour queries over moving points with location uncertainty. Probabilistic range, distance, and spatial join queries may now be defined as follows.

Definition 2. ***Probabilistic Range Query****: Given a rectangular region R, a set of uncertain polygons S, and a constant γ, $0 \leq \gamma \leq 1$, a Probabilistic Range Query (PRQ) returns a set of pairs of the form (s_i, π_i), where $s_i \in S$, and $\pi_i \geq \gamma$ is the intersection probability between s_i and R. We call γ the confidence threshold for the range query.*

Definition 3. ***Probabilistic Distance Query****: Given a query point q, a query distance d, a set of uncertain polygons S, and a constant γ, $0 \leq \gamma \leq 1$, a Probabilistic Distance Query (PDQ) returns a set of pairs of the form (s_i, π_i), where $s_i \in S$ is within distance d from q with probability $\pi_i \geq \gamma$. We call γ the confidence threshold for the distance query.*

Definition 4. ***Probabilistic Spatial Join****: Given two sets of uncertain polygons S_1, S_2, a query distance d, and a constant γ, $0 \leq \gamma \leq 1$, a Probabilistic Spatial Join (PSJ) returns a set of triples of the form (s_i, s_j, π_i), where $s_i \in S_1$ and $s_j \in S_2$ are within distance d of each other with probability $\pi_i \geq \gamma$. We call γ the confidence threshold for the join query.*

A question common to such queries is how to compute the intersection probability between two polygons, given that at least one of them has an uncertain boundary. In the following sections, we will focus on PSJ and show how to evaluate it for the case when $d = 0$, that is for overlap queries. The other queries are answerable using the resulting methods.

4 Query Evaluation for PSJ

As the shapes of spatial objects are rarely regular, they are usually approximated by their Minimum Bounding Rectangles (MBR). Traditional spatial joins are executed in two steps [5]. In the first or *filtering step*, the join predicate is evaluated on the set of MBRs, and a result set of candidate pairs is produced. The MBRs are indexed with indices such as the R-tree and its variations [2, 21], or Seeded Trees [15], and a *tree matching* algorithm is used to find matching pairs of MBRs. In the second or *refinement step*, the actual spatial objects are matched using the join predicate.

This filtering-and-refinement strategy can also be applied to evaluate probabilistic spatial joins. However, traditional spatial indices do not support any notion of uncertainty or probabilistic matching. We must hence modify the MBR approximations and index structures to support probabilistic filtering. In this section, we present the PrR-tree, an R-tree based index structure for uncertain polygons.

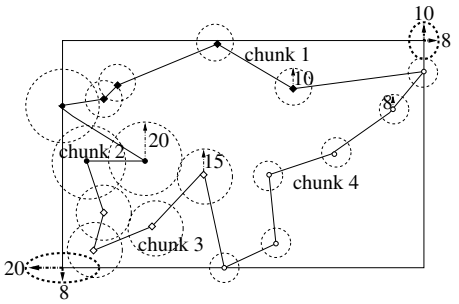
4.1 PrR-Tree and Probabilistic Filtering

The PrR-tree is a disk-based, balanced and multi-way tree with the structure of an R-tree. To support probabilistic filtering, we augment the MBR approximation with the probability distribution of MBR's boundary. As in an R-tree, the entry in a leaf node has the form (MBR, oid) tuples, and contains an object's MBR and a pointer to its exact representation. Intermediate node entries have the form (MBR, ptr), where ptr points to a lower level node, and MBR *covers* all the MBRs in this node in the sense explained below.

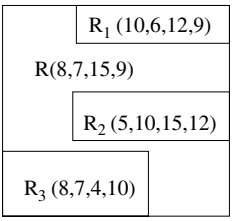
Consider a polygon $\langle P \rangle$, as in Figure 4, and let its MBR be defined by the lower-left and upper-right vertices being $(x^{\vdash}, y^{\vdash})$ and $(x^{\dashv}, y^{\dashv})$, respectively. We will represent this MBR as $[|x^{\vdash}, y^{\vdash}, x^{\dashv}, y^{\dashv}|]$. Observe that $x^{\vdash}$ is determined by the vertex of $\langle P \rangle$ with the lowest X-coordinate, and that $y^{\vdash}$ is determined by the vertex with the lowest Y-coordinate. These may be different vertices, so the uncertainty distribution of the lower-left corner is not a circular Normal, even when the vertices of $\langle P \rangle$ are associated with uncertainties that are circular Normals.

We could, in principle, apply the techniques of *Order Statistics* [20] to derive the probability distribution of $x^{\vdash}$, from the probability distributions of the individual vertices. However, these derived distributions tend to have very complicated forms, and would impose a high computation cost during the filter step. Instead, we obtain approximations for the distributions for $x^{\vdash}, y^{\vdash}, x^{\dashv}, y^{\dashv}$ in terms of Normal distributions. This approach is reasonable since the means for the vertex positions are likely to be large compared to their variances.

Therefore, the parameters for the MBR $[|x^{\vdash}, y^{\vdash}, x^{\dashv}, y^{\dashv}|]$ implicitly include both the mean positions $x^{\vdash}$, $y^{\vdash}$, $x^{\dashv}$, $y^{\dashv}$ as well as the corresponding variances $\sigma_x^{\vdash}$, $\sigma_y^{\vdash}$, $\sigma_x^{\dashv}$, $\sigma_y^{\dashv}$. One obvious benefit of this approach is that the representation of MBR is quite simple, since we only need add four more values to the traditional representation of MBRs.



(a) leaf MBR



(b) non-leaf MBR

Fig. 4. The MBR at leaf node and the MBR at intermediate node

Specifically, we order the vertices of $\langle P \rangle$ by the mean values of their X and Y coordinates, and take lowest of the X-coordinates to be the mean position of $x^{\vdash}$. The value of $\sigma_x^{\vdash}$ is the maximum σ value among all vertices whose mean positions coincide with the MBR's left edge. For example, Figure 4 (a) shows a polygon with four chunks, whose sigma values are 10, 20, 15, 8, respectively. The left edge of the MBR is defined by one vertex from chunk 2, so that the mean value of $x^{\vdash}$ is defined by the mean position of this vertex, and $\sigma_x^{\vdash}$, the sigma value for $x^{\vdash}$, is equal to the sigma value of chunk 2, that is, $\sigma_x^{\vdash} = 20$. We can similarly obtain the mean values for $y^{\vdash}$, $x^{\dashv}$ and $y^{\dashv}$, and the sigma values $\sigma_y^{\vdash} = 8$, $\sigma_x^{\dashv} = 8$, and $\sigma_y^{\dashv} = 10$.

Covering MBRs. The MBR at the intermediate level of the PrR-Tree can be derived as follows. We represent MBRs as $[|x^{\vdash}, y^{\vdash}, x^{\dashv}, y^{\dashv}|]$, and use $\sigma_x^{\vdash}$, $\sigma_y^{\vdash}$, $\sigma_x^{\dashv}$, $\sigma_y^{\dashv}$ to represent the standard deviation of $x^{\vdash}$, $y^{\vdash}$, $x^{\dashv}$, $y^{\dashv}$, respectively.

We say MBR $[|x^{\vdash}, y^{\vdash}, x^{\dashv}, y^{\dashv}|]$ *covers* the MBRs at an intermediate node in the PrR-tree if it overlays these MBRs tightly. First, the mean position of $x^{\dashv}$ must correspond to the farthest left of the lower-left corners of the covered MBRs.

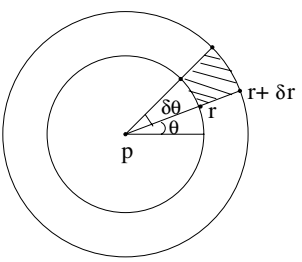


Fig. 5. Shadow area is $\triangle_p(r, \theta)$ for vertex p

Also, the value of $\sigma_x^{\vdash}$ of the covered MBR must also correspond to the $\sigma_x^{\vdash}$ value of this corner. (In the case where several covered MBRs have their lower-left corners at the extreme left position, we pick the corner with the highest $\sigma_x^{\vdash}$ value.) We proceed similarly for the values of $y^{\dashv}$, $x^{\dashv}$, $y^{\dashv}$, and the uncertainty parameters $\sigma_y^{\vdash}$, $\sigma_x^{\dashv}$ and $\sigma_y^{\dashv}$.

For example, Figure 4 (b) shows a MBR R at the intermediate node covering the three MBRs R_1, R_2, and R_3. For simplicity, we only show the σ values of these MBRs. Since R_3 defines the lower X-bound of R, the $\sigma_x^{\vdash}$ for R is equal to the sigma value of R_3's lower X-bound, that is, 8. Computing $\sigma_x^{\dashv}$ is a bit more involved, since the mean positions of the right edges of R_1 and R_2 coincide, so that they both define the upper X-bound of R. In this case, we must take R's $\sigma_x^{\dashv}$ as equal to the higher of the $\sigma_x^{\dashv}$ values of R_1 and R_2, that is, as equal to 15. In a similar fashion, we can calculate $\sigma_y^{\vdash}$ and $\sigma_y^{\dashv}$.

Insertion and Deletion. The insertion and deletion algorithm of the R-tree can be directly applied to the PrR-tree, as the mean positions of MBRs are used to compute their areas, intersection areas, and margins. However, we must maintain the σ values for the MBRs as described above during insertions, deletions, or node split processing.

Filtering Algorithm. The basic idea behind the filtering step for PSJ is to simultaneously traverse the two PrR-trees, checking whether the intersection probabilities between pairs of MBRs exceeds γ. If the intersection probability of two MBRs is less than γ, there can be no PSJ candidate pairs among the MBRs covered by them. Otherwise, there could be candidate pairs whose intersection probability equals or exceeds γ, in the corresponding subtrees.

While our algorithm is similar to the traditional tree-matching algorithm for spatial join in [5], it is also different since it must compute the intersection probability between two MBRs. Given two MBRs $R_1 = [|x_1^{\vdash}, y_1^{\vdash}, x_1^{\dashv}, y_1^{\dashv}|]$, and $R_2 = [|x_2^{\vdash}, y_2^{\vdash}, x_2^{\dashv}, y_2^{\dashv}|]$, the intersection probability between them can be computed as follows:

$$\Pr[R_1 \cap R_2] = (1 - \Pr[x_2^{\dashv} \leq x_1^{\vdash}] - \Pr[x_1^{\dashv} \leq x_2^{\vdash}]) \times (1 - \Pr[y_2^{\dashv} \leq y_1^{\vdash}] - \Pr[y_1^{\dashv} \leq y_2^{\vdash}]). \quad (1)$$

If X, Y are two independent Normal random variables, then $X - Y$ is also Normal. Therefore, it is quite easy to get $Pr(X \leq Y)$ by computing $Pr(X - Y <= 0)$. Since $x_1^{\vdash}$, $y_1^{\vdash}$, $x_1^{\dashv}$, $y_1^{\dashv}$, $x_2^{\vdash}$, $y_2^{\vdash}$, $x_2^{\dashv}$ all follow Normal distributions, we can easily compute $Pr(x_2^{\dashv} \leq x_1^{\vdash})$, $Pr(x_1^{\dashv} \leq x_2^{\vdash})$, $Pr(y_2^{\dashv} \leq y_1^{\vdash})$, and $Pr(y_1^{\dashv} \leq y_2^{\vdash})$, and thus derive $\Pr[R_1 \cap R_2]$.

Heuristic Adjustment of the Threshold γ. In Figure 4(b), let the lower bounds along the X-axis for R, R_1, R_2, and R_3 be $x^{\vdash}$, $x_1^{\vdash}$, $x_2^{\vdash}$, and $x_3^{\vdash}$. Since we have approximated the distribution of $x^{\vdash}$ with the distribution of $x_1^{\vdash}$, we risk missing some true hits if we simply used γ as the threshold in the filtering step.

To minimize this risk, we heuristically adjust γ by estimating the probability that our approximation is correct.

Let $\delta_x^{\vdash}$ be the probability that $x_1^{\vdash}$ is less than both $x_2^{\vdash}$ and $x_3^{\vdash}$. Let $\delta_x^{\dashv}$, $\delta_y^{\vdash}$, and $\delta_y^{\dashv}$ be similarly defined, and let δ be the highest of these values. (Since $x_1^{\vdash}$, $x_2^{\vdash}$ and $x_3^{\vdash}$ follow Normal distributions, these probabilities are easy to compute.) We store δ in R, and use the value $(\delta \times \gamma)$ as the threshold during the filtering step. Our experiments show that this adjustment reduces the number of false negatives at the cost of some increase in false positives. However, the total number of errors is also reduced in the process. Since our problem definition inherently involves uncertainty, not exact matching, this is a reasonable tradeoff.

4.2 The Refinement Step

The filtering step returns a set of candidate pairs whose MBRs intersect with probability at least $(\delta \times \gamma)$. This set, however, may contain false hits which do not satisfy the join predicate. The refinement step must retrieve the exact representations of polygons, compute the intersection probability between two polygons, and evaluate the join predicate. We now consider this problem in greater detail.

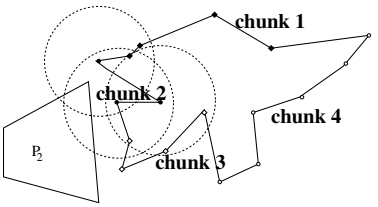
Consider two polygons $\langle P_1 \rangle$ and $\langle P_2 \rangle$. To get the intersection probability between $\langle P_1 \rangle$ and $\langle P_2 \rangle$, we first compute the probability that at least one vertex of $\langle P_1 \rangle$ is located inside $\langle P_2 \rangle$, and the probability that at least one vertex of $\langle P_2 \rangle$ is located inside $\langle P_1 \rangle$. We then use the larger of these values as the intersection probability between $\langle P_1 \rangle$ and $\langle P_2 \rangle$.

Denote the event that at least one vertex of $\langle P_1 \rangle$ is located inside $\langle P_2 \rangle$, by $[\langle P_1 \rangle \succ \langle P_2 \rangle]$. We now consider how to compute the probability $\Pr[\langle P_1 \rangle \succ \langle P_2 \rangle]$. The exact representations of $\langle P_1 \rangle$ and $\langle P_2 \rangle$ are retrieved. Since the uncertainties of any two chunks are independent, the event that the vertices of $\langle P_1 : i \rangle$ are outside of $\langle P_2 \rangle$ is independent of the event that the vertices of $\langle P_1 : j \rangle$ are outside of $\langle P_2 \rangle$. It therefore follows that

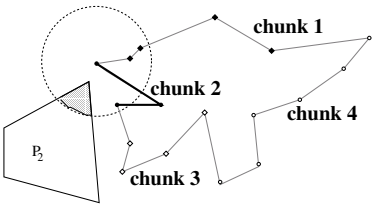
$$\Pr[\langle P_1 \rangle \succ \langle P_2 \rangle] = 1 - \Pr[\langle P_1 \rangle \not\succ \langle P_2 \rangle] = 1 - \prod_{j=1}^{k} (1 - \Pr[\langle P_1 : j \rangle \succ \langle P_2 \rangle]). \quad (2)$$

Now, given the chunk $\langle P_1 : j \rangle$, it remains to show how to compute $\Pr[\langle P_1 : j \rangle \succ \langle P_2 \rangle]$, the probability that any vertex in $\langle P_1 : j \rangle$ is located inside $\langle P_2 \rangle$.

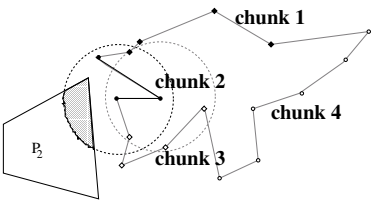
Computing Intersection Probabilities. In Figure 5, let $\triangle_p(r, \theta)$ denote the shaded area between the circles of radii r and $r + \delta r$ at angles θ and $\theta + \delta\theta$ centered around the mean position of p, here p is any vertex from the chunk $\langle P_1 : j \rangle$. From Definition 1, the probability that p is located within a circle of radius r centered at its mean is given by the distribution $\Pr(r) = 1 - e^{-\frac{r^2}{2\sigma^2}}$, where σ is the uncertainty parameter of p and the other vertices in $\langle P_1 : j \rangle$. The corresponding density function is $\mathrm{pr}(r) = \frac{r}{\sigma^2} e^{-r^2/(2\sigma^2)}$. Then, the probability



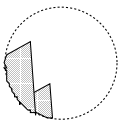
(a) Chunk $\langle P_1 : 2\rangle$ has 3 vertices.



(b) Overlap of $\langle P_2\rangle$ with first ϵ-circle for $\langle P_1 : 2\rangle$



(c) Overlap of $\langle P_2\rangle$ and other ϵ-circles for $\langle P_1 : 2\rangle$



(d) Union of the intersection areas

Fig. 6. Computing $\Pr[\langle P_1, 2\rangle \succ \langle P_2\rangle$

that p is located inside $\triangle_p(r, \theta)$ is obtained from the corresponding density function as $\text{pr}(r) * \delta r * \frac{\delta\theta}{2\pi}$.

Recall that our aim is to get the probability that *any* vertex from the chunk $\langle P_1 : j\rangle$ is located inside $\langle P_2\rangle$. That is, we don't really care if more than one such vertex is located inside $\langle P_2\rangle$. We therefore define an indicator function $I(r, \theta)$, such that $I(r, \theta) = 1$ if and only if there is a vertex from $\langle P_1 : j\rangle$, for which $\triangle_p(r, \theta)$ is inside $\langle P_2\rangle$ for some values of r and θ. Now, we can obtain $\Pr[\langle P_1 : j\rangle \succ \langle P_2\rangle]$ by integrating over all possible r and θ to get

$$\Pr[\langle P_1 : j\rangle \succ \langle P_2\rangle] = \int_0^\infty dr \int_0^{2\pi} \frac{1}{2\pi}\text{pr}(r) * I(r, \theta)d\theta. \tag{3}$$

ϵ-Circles and Efficient Computation of the Integral. In practice, it is unnecessary to integrate from 0 to ∞, since the Gaussian distribution is concentrated near its mean. For instance, the probability that p is inside a circle of radius $r = 3\sigma$ is more than 0.99. We can simplify the integral by choosing a probability ϵ, and then finding the r_ϵ such that $\Pr[r_\epsilon] = 1 - \epsilon$. In this case, the probability that p is beyond radius r_ϵ is no more than ϵ. Now, we are safe in computing the above integral by integrating from 0 to r_ϵ. For a given value of ϵ, this resulting circle of radius r_ϵ is called the ϵ-*circle*.

Table 2. Characteristics of the Data Sets Used in the Experiments

Set	Description	Size	Density	Vsize	Csize	CV_{area}	CV_{dist}
LAB	Los Angeles block groups	6357	0.79	35.2	6.9	0.17	
LAL	Los Angeles landmark polygons	5135	0.45	31.8	6.2	0.27	0.0023
UM1	Uniformly distributed monotone polygons	10000	0.34	30.0	5.9	0.35	
UM2	Uniformly distributed monotone polygons	10000	0.31	30.0	5.9	0.35	0.0018
UM3	Uniformly distributed monotone polygons	50000	0.39	29.9	5.9	0.35	
UM4	Uniformly distributed monotone polygons	50000	0.39	30.0	5.9	0.35	0.0010

We simplify the computation of the integral $\int_0^{r_\epsilon} dr \int_0^{2\pi} \frac{1}{2\pi} pr(r) * I(r,\theta) d\theta$. Consider Figure 6 (a), where we need compute $\Pr[\langle P_1 : 2\rangle \succ \langle P_2\rangle]$, the probability that some vertex of the chunk $\langle P_1 : 2\rangle$ of $\langle P_1\rangle$ is located inside $\langle P_2\rangle$. First, we pick a constant ϵ, and for each vertex p in chunk $\langle P_1 : 2\rangle$, we compute the overlap area between $\langle P_2\rangle$ and an ϵ-circle around p (see Figure 6 (b), (c)). If the union of these intersection areas is A (see Figure 6 (d)), we simply compute $\int\int_A \frac{1}{2\pi} \mathrm{pr}(r) dr d\theta$.

5 Experimental Evaluation

We conducted experiments to examine the performance of PSJ with both synthetic and real-life datasets, to determine the accuracy, efficiency, and the scalability of our Probabilistic Spatial Join method.

5.1 Data Sets

In our experiments, we used *quasi-real* and *synthetic* datasets. The quasi-real datasets were generated using real map information as follows. We used the TIGER/LINE data from the U.S Bureau of the Census [19] to determine the mean position of the polygons. We then randomly generated chunks and associated uncertainties (standard deviations σ for the circular normal distribution in our case). For synthetic datasets, both the polygons and chunks with uncertainties were generated randomly.

Table 2 describes several *quasi-real* and synthetic datasets used in the experiments. We use *density*, defined as the total area of the polygons' MBR divided by the total area of the workspace, to measure the coverage of the datasets. The metric *Vsize* represents the average number of vertices per polygon, while *Csize* represents the average number of chunks per polygon. Also, we use the coefficient of variation for the polygons' area(CV_{area}) and the coefficient of variation for the distance between two polygons (CV_{dist}) to measure the degree of uncertainty of the datasets.

The data set LAB contains the block groups in Los Angeles, while LAL contains the landmark polygons in Los Angeles. The synthetic dataset UM1, UM2,UM3,UM4 are created according to a uniformly distributions with 100 clusters. As in [16], we first generated 100 cluster rectangles, whose centers were randomly distributed in the map area. Once the cluster rectangles are generated,

we can randomly generate the MBRs of the polygons, which fixes the size and location of the random polygons.

Given the MBRs, we use the method in [1] to generate random *monotone polygons*, a very common class of polygons, as follows. First, a set of random points is chosen inside the MBR, and a random horizontal line is drawn through them. The points above and below the line are sorted by their x coordinates and connected in the sorted order. The leftmost and rightmost points are moved vertically to the splitting line, and are connected to the other points to form a polygon. Finally, this horizontally aligned polygon is rotated by a random angle.

The ground truth in our experiments is computed using *Monte Carlo* approach. Given two datasets associated with uncertainty, n equally probable realizations are generated and spatial join is applied to these n realizations. Based on the results from n runs, the intersection probability between any polygons are computed. These experimentally obtained intersection probabilities are assumed to be the ground truth and used to measure the accuracy of different approaches. In our experiment, we set $n = 1000$.

5.2 Competing Methods

To demonstrate the efficiency and accuracy of PSJ, we compare it with two other competitors: *Mean Spatial join (MSJ)* and *Random Spatial Join (RSJ)*. PSJ is our algorithm, as described in Section 4. It has two PrR-trees before evaluating join operations. MSJ is the normal R-tree based spatial join using the mean positions of the polygons. Both the two datasets have pre-existing R^*-tree indices and the query is executed using the algorithm in [5]. RSJ is same as the *Monte Carlo Simulation* approach used to get the ground truth, except that RSJ uses much fewer random samples. The notations RSJ3, RSJ5, RSJ10 indicate that the number of random realizations were 3, 5, and 10, respectively.

Each run of RSJ executes the following four steps. First, the the uncertain polygons are read in, and a random sample is generated. Next, we build two R^*-tree for this random sample. The third step is to run the filter step using these two R^*-tree . The final step is to run the refinement step using plane sweep algorithm.

As indicated in [16], it is much more expensive to perform spatial join for two non-indexed datasets using R^*-tree based approach, compared with other approaches like spatial hash join [16], size separation spatial join [12] and PBSM [23]. To make our comparisons with RSJ reasonable, we ignore the costs of tree construction and the costs of the filter step. That is, we counted the CPU and I/O costs of only the first and fourth steps described above. No matter which filtering technique is used, those two steps are essential. In other words, for RSJ, the measured CPU and I/O costs are underestimates, and the real costs should be larger than our measurements.

Table 3. Space and overall construction costs of PrR-tree, normalized to those costs of R^*-tree

Dataset	Ratio of space cost	Ratio of overall construction cost
LAL	1.89	1.90
UM1	1.84	2.04
UM3	1.93	1.85

5.3 Experimental Setup

The experiments were conducted on a Pentium IV 1.7GHz machine with 512MBytes of RAM, running Mandrake Linux 8.2. All the algorithms were implemented in C++ using GNU compilers. To measure I/O costs, we assumed a buffer page size of 8K. The buffer size is set to be 10% of the total size of two datasets. In our experiments, the datasets used have size of 8M, 11M, 55M, respectively. Therefore, the buffer size is set to be 800K(100 pages), 1112K(139 pages), 5536K(692 pages). To compute the I/O cost more precisely, it is desirable to distinguish sequential I/O access from random I/O access. We assumed the ratio of the cost of accessing one disk block randomly to that of accessing one disk block sequentially to be 5 [16], and measured the I/O cost as the number of weighted disk access. The CPU cost can be accurately measured during the program is running. The overall cost is obtained by charging 10ms each random disk access, charging 2ms each sequential access.

5.4 Performance Metrics and Evaluation

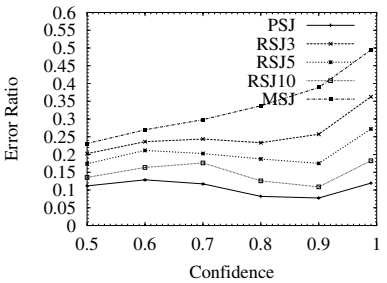
The first set of experiments is to evaluate the space and overall construction cost of PrR-tree, compared with those of R^*-tree . Table 3 presents the ratio of the size of PrR-tree to the size of R^*-tree for the three groups of datasets, and the ratio of the overall cost of building PrR-tree to the overall cost of building R^*-tree . Note here the buffer size is set to be 10% of the size of PrR-trees.

We then evaluate the accuracy of the three methods. Accuracy covers two aspects: *completeness*, or how well the method is able to find the object pairs that should be found, and *fidelity*, which measures how close the returned probability is to the actual probability defined by ground truth.

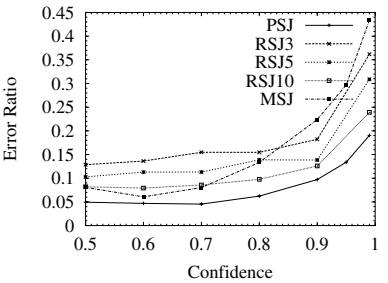
Two metrics are used to measure the completeness aspect of the accuracy of the three methods: *ratio of false negative*(R_{fn}) and *ratio of false positive*(R_{fp}). Let GT represent the ground true result, QR represents the query result. R_{fn} is defined as $|GT - QR|/|QT|$, and R_{fp} is defined as $|QR - GT|/|QR|$. The accuracy of competing methods is compared in terms of $errorRatio$, defined as the sum of R_{fn} and R_{fp}. The comparisons are shown in Figure 7(a) - (c).

As for fidelity, we use the root mean square error (square root of the average of the set of squared differences) between the returned probability and the actual probability (RMSE). These comparisons are shown in Figure 7(d)-(f).

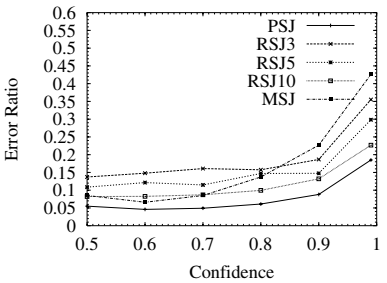
Finally, we evaluate the efficiency of the three methods, in terms of CPU cost, weighted I/O cost and overall cost. The results are shown in Figure 8.



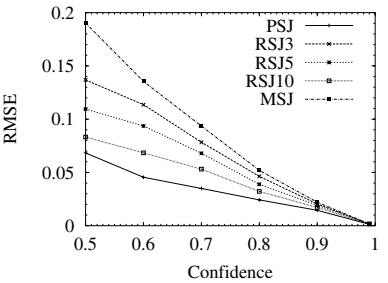
(a) LAB vs. LAL



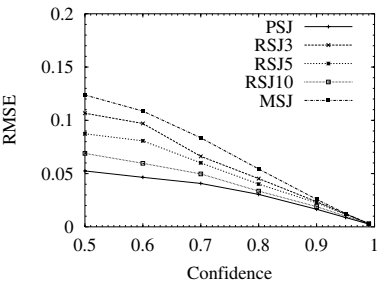
(b) UM1 vs. UM2



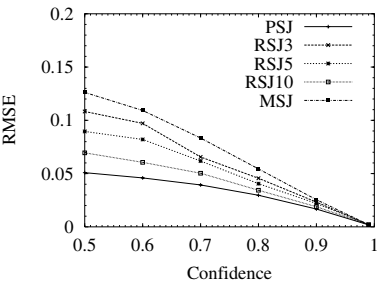
(c) UM3 vs. UM4



(d) LAB vs. LAL

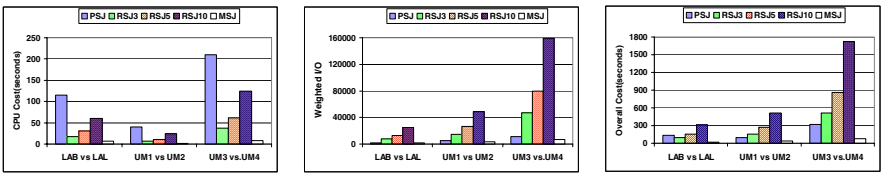


(e) UM1 vs. UM2



(f) UM3 vs. UM4

Fig. 7. Experimental comparison of error ratio and RMSE



(a) CPU Cost (b) Weighted I/O Cost (c) Overall Cost

Fig. 8. Experimental comparison of Evaluation Cost

5.5 Comparisons

As Table 3 shows, the space and construction costs for PrR-tree is roughly 80% - 100% higher than that of R^*-tree . This is expected, since each entry of PrR-tree records the mean positions as well as the standard deviations, and the insertion procedure must maintain both these values.

Figures 7 and 8 show that MSJ has the lowest CPU cost, weighted I/O cost and overall cost for all of the datasets. However, MSJ's *errorRatio* is among one of the highest. In particular, when the confidence threshold ranges between 0.85 - 0.99, its *errorRatio* increased dramatically to about 0.3 - 0.5. We conclude that MSJ does not make much sense when the confidence threshold is relatively high, although it has low CPU cost and I/O cost.

RSJ has lower CPU cost than PSJ, but much higher I/O cost. RSJ's I/O costs and accuracy increase with the number of samples since it must retrieve the representations of uncertain polygons and generate random samples for each run. Also, RSJ's I/O cost dominates the overall cost. With the gap between computation speed and I/O speed continuing to increase, the overhead of RSJ's I/O cost makes it unsuitable for large spatial databases.

In contrast, PSJ achieves the lowest *errorRatio* and RMSE among the three competitors, with the overall cost being reduced by a factor of 2.4–5.3 over RSJ10, which has the highest accuracy among the three variations of RSJ. PSJ has the highest CPU cost, since it must compute integrals during the refinement step. However, this drawback is compensated for by its weighted I/O cost, which is much lower than RSJ's. It is worth noting that PSJ outperforms RSJ10 in proportional to dataset size, since RSJ10 has much higher I/O overhead. We believe that with the continuing increases in CPU speed, PSJ's computation overhead will not be a great problem.

6 Conclusions

The modeling of spatial objects with uncertain boundaries and evaluating spatial queries over them is an open issue. We have presented a method for performing spatial joins over uncertain spatial data. We have also proposed a probabilistic

spatial data model to model the positional uncertainty of polygons. Based on this model, we present the PrR-tree, a probabilisitc index structure to support probabilistic filtering, and an algorithm for the refinement step. Our experiments demonstrate that our approach achieves higher accuracy for probabilistic spatial join queries, while reducing the overall cost by a factor of more than two.

Acknowledgement

This work was supported in part by grants from Tata Consultancy Services, Inc., the Digital Media Innovations program of the University of California, the National Science Foundation under contract 0114036, and by the Fault-Tolerant Networks program of the Defense Advanced Research Projects Agency, under contract F30602-01-2-0536. Thanks are due to M. Hadjieleftheriou for providing the R^*-tree library, and to Sandeep Gupta for helpful discussions.

References

[1] A. Aboulnaga and J.F. Naughton. Accurate estimation of the cost of spatial selections. In *Proceedings of International Conference on Data Engineering(ICDE)*, pages 123–134, San Diego, California, March 2000. 153

[2] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM-SIGMOD Conference*, pages 47–57, Boston, Massachusetts, June 1984. 143, 147

[3] M. Azouzi. Introducing the concept of reliability in spatial data. In Kim Lowell and Annick Jaton, editors, *Spatial Accuracy Assessment: land information uncertainty in Natural Resources*, pages 139–144. Ann Arbor Press, 1999. 145

[4] T. Brinkhoff, H. Kriegel, R. Schneider, and B. Seeger. Multi-step processing of spatial joins. In *Proceedings of the 1994 ACM-SIGMOD Conference*, pages 197–208, Minneapolis, Minnesota, May 1994. 143

[5] T. Brinkhoff, H. Kriegel, and B. Seeger. Efficient processing of spatial joins using r-trees. In *Proceedings of the 1993 ACM-SIGMOD Conference*, pages 237–246, Washington, D.C., May 1993. 142, 143, 147, 149, 153

[6] FGDC. National standard for spatial data accuracy, fgdc-std-007.3-1998. `http://www.fgdc.gov/standards/status/sub1_3.html`, March 1998. 140, 141

[7] M.F. Goodchild, A.M. Shortridge, and P. Fohl. Encapsulating simulation models with geospatial data sets. In Kim Lowell and Annick Jaton, editors, *Spatial Accuracy Assessment: land information uncertainty in Natural Resources*, pages 123–129. Ann Arbor Press, 1999. 142, 143

[8] G.B.M. Heuvelink. *Error Propagation in Environmental Modeling with GIS*. Taylor & Francis, London, UK, 1998. 140

[9] Y.W. Huang, M. Jones, and E. A. Rundensteiner. Symbolic intersect detection: A method for improving spatial intersect joins. *GeoInformatica*, 2(2):149–174, June 1998. 143

[10] G. Hunter and M.F. Goodchild. Application of new model of vector data uncertainty. In Kim Lowell and Annick Jaton, editors, *Spatial Accuracy Assessment: land information uncertainty in Natural Resources*, pages 203–208. Ann Arbor Press, 1999. 143

[11] J.P. King. Modeling boundaries of influence among positional uncertainty fields. Master's thesis, University of Maine, December 2002. 145
[12] N. Koudas and K. C. Sevcik. Size separation spatial join. In *Proceedings of the 1997 ACM-SIGMOD Conference*, pages 324–335, Tucson, Arizona, May 1997. 143, 153
[13] Y. Leung and J. Yan. Point-in-polygon analysis under certainty and uncertainty. *GeoInformatica*, 1(1):93–114, Apr 1997. 143, 145
[14] Y. Leung and J. Yan. A locational error model for spatial features. *Int. J. Geographical Information Science*, 12(6):607–620, Nov 1998. 145
[15] M.L. Lo and C. V. Ravishankar. Spatial joins using seeded trees. In *Proceedings of the 1994 ACM-SIGMOD Conference*, pages 209–220, Minneapolis, Minnesota, June 1994. 143, 147
[16] M.L. Lo and C. V. Ravishankar. Spatial hash join. In *Proceedings of the 1996 ACM-SIGMOD Conference*, pages 247–258, Montreal, Canada, June 1996. 143, 152, 153, 154
[17] N. Mamoulis and D. Papadias. Slot index spatial join. *IEEE Transaction on Knowledge and Data Engineering*, 15(1):211–231, Jan 2003. 143
[18] Minnesota Planning. Positional accuracy handbook. `http://www.mnplan.state.mn.us/press/accurate.html`, October 1999. 141, 142
[19] U.S. Bureau of the Census. *Census 2000 TIGER/Line Data*. Washington DC, 2000. 152
[20] N. Balakrishnan and C.R. Rao. *Order statistics: theory and methods*. Elsevier, New York, 1998. 147
[21] N.Beckmann, R. Schneider H.P. Kriegel, and B. Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM-SIGMOD Conference*, pages 322–331, Atlantic City, NJ, May 1990. 143, 147
[22] S. Openshaw. Learning to live with errors in spatial databases. In M. Goodchild and S. Gopal, editors, *Accuracy of Spatial Databases*, pages 263–276. Taylor & Francis, 1989. 143
[23] J. M. Patel and D. J. DeWitt. Partition based spatial-merge join. In *Proceedings of the 1996 ACM-SIGMOD Conference*, pages 259–270, Montreal, Canada, June 1996. 143, 153
[24] D. Pfoser and C. S. Jensen. Capturing the uncertainty of moving-object representations. In *Proceeding of the 6th International Symposium on Large Spatial Databases(SSD)*, pages 111–132, Hongkong, China, July 1999. 144
[25] R. Cheng, D.V. Kalashnikov, and S. Prabhakar. Querying imprecise data in moving object environments. Poster Session, International Conference on Data Engineering(ICDE), 2003. To appear. 142, 144, 146
[26] W. Shi. A generic statistical approach for modeling error of geometric features in gis. *Int. J. Geographical Information Science*, 12(2):131–143, March 1998. 143
[27] W. Shi and W. Liu. A stochastic process-based model for the positional error of line segments in gis. *Int. J. Geographical Information Science*, 14(1):51–66, Jan 2000. 143
[28] G. Trajcevski, O. Wolfson, F. Zhang, and S. Chamberlain. The geometry of uncertainty in moving object databases. In *Proceedings of the 8th International Conference on Extending Database Technology(EDBT)*, pages 233–250, Prague, Czech Republic, March 2002. 144

[29] O. Wolfson, P.A. Sistla, S. Chamberlain, and Y. Yesha. Updating and querying databases that track mobile units. *Distributed and Parallel Databases*, 3(7):257–387, July 1999. 142, 144, 146

[30] J.X. Zhang and M.F. Goodchild. *Uncertainty in Geographical Information System.* Taylor & Francis, Erewhon, NC, 2002. 140

[31] G. Zimbrao and J.M. Souza. A raster approximation for the processing of spatial joins. In *Proceedings of the 24th VLDB Conference*, pages 311–322, New York City, New York, August 1998. 143