Toward Managing Uncertain Spatial Information for Situational Awareness Applications

Yiming Ma, Dmitri V. Kalashnikov, and Sharad Mehrotra

Abstract—Situational awareness (SA) applications monitor the real world and the entities therein to support tasks such as rapid decision making, reasoning, and analysis. Raw input about unfolding events may arrive from variety of sources in the form of sensor data, video streams, human observations, and so on, from which events of interest are extracted. Location is one of the most important attributes of events, useful for a variety of SA tasks. In this paper, we consider the problem of reaching situation awareness from textual input. We propose an approach to probabilistically model and represent (potentially uncertain) event locations described by human reporters in the form of free text. We analyze several types of spatial queries of interest in SA applications. We design techniques to store and index the uncertain locations, to support the efficient processing of queries. Our extensive experimental evaluation over real and synthetic data sets demonstrates the effectiveness and efficiency of our approach.

Index Terms—Modeling, retrieval, spatial database, uncertainty indexing, U-grid, probability.

1 Introduction

Recent events (Southeast Asia Tsunamis and Hurricane Katrina) have illustrated the need for accurate and timely situational awareness (SA) tools in emergency response. Developing effective SA systems has the potential to radically improve decision support in crises by improving the accuracy and reliability of the information available to the decision-makers [2], [22], [25], [26], [27]. In this paper, we study the problem of representing and querying uncertain location information about real-world events that are described using free text. As a motivating example, consider the excerpts from two real reports filed by *Port Authority Police Department* (PAPD) Officers who participated in the events of 11 September 2001:

Example 1.1

1. *"... The PAPD Mobile Command Post was located on West St. north of WTC and there was equipment being staged there. ..."*
2. *"... a PAPD Command Truck parked on the west side of Broadway St. and north of Vesey St. ..."*

These two reports refer to the same location of the same command post—a point-location in the New York, Manhattan area. However, the reports neither specify the exact location of the events, nor do they mention the same street names. We would like to represent such reports in a way that it enables efficient evaluation of spatial queries and analysis. For instance, the representation must enable us to retrieve events in a given geographical region (e.g., around World Trade Center). Likewise, it should enable us to determine similarity between reports based on their spatial properties; e.g., we should be able to determine that the above events might refer to the same location.

Our primary motivation in studying the aforementioned problem comes from designing database solutions to support applications where the real world is being monitored (potentially using a variety of sensing technologies) to support tasks such as situation assessment and decision making. Such SA applications abound in a variety of domains including homeland security, emergency response, command and control, process monitoring/automation, and business activity monitoring, to name a few. Our particular interest lies in the domain of emergency response and security. We already alluded to the usefulness of spatial reasoning over free text in the example above. Such solutions are useful in a variety of other application scenarios in emergency response. For instance, such a system could support real-time triaging and filtering of relevant communications and reports among first responders (and the public) during a crisis. In our project, we are building SA tools to enable social scientists and disaster researchers to perform spatial analysis over two such data sets: 1) the transcribed communication logs and reports filed by the first responders after the 9/11 disaster and 2) newspaper articles and blog reports covering the SE Asia Tsunami disaster. We believe that techniques such as ours can benefit a very broad class of applications where free text is used to describe events.

Our goal in this paper is to represent and store uncertain locations specified in reports in the database to allow for efficient execution of analytical queries. Clearly, merely storing location in the database as free text is not sufficient either to answer spatial queries or to disambiguate reports based on spatial locations. For example, spatial query such as "retrieve events near WTC," based on keywords alone, can only retrieve the first report mentioned earlier.

- *Y. Ma is with the Nokia Research Center, 955 Page Mill Road, Palo Alto, CA 94304. E-mail: yiming200x@gmail.com.*
- *D.V. Kalashnikov and S. Mehrota are with the Department of Computer Science, Bren School of Information and Computer Science, University of California at Irvine, Irvine, CA 92617. E-mail: {dvk, sharad}@ics.uci.edu.*

Manuscript received 18 Apr. 2007; revised 6 Oct. 2007; accepted 12 Feb. 2008; published online 25 Feb. 2008.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2007-04-0172.
Digital Object Identifier no. 10.1109/TKDE.2008.49.

free text	*s-expression*
'near WTC'	$\texttt{near}(WTC)$
'on West St., north of WTC'	on(West St.) $\wedge\ \texttt{north}(WTC)$

Fig. 1. Examples of s-expressions.

To support spatial analyses on free text reports, we need to project the spatial properties of the event described in the report onto the domain. In this paper, we model uncertain event locations as random variables that have certain probability density functions (*pdfs*) associated with them. We develop techniques to map free text onto the corresponding pdf defined over the domain.

Our approach is based on the assumption[1] that people report event locations based on certain *landmarks*. Let $\Omega \subset \mathbb{R}^2$ be a 2D physical space in which the events described in the reports are immersed. Landmarks correspond to significant spatial objects such as buildings, streets, intersections, regions, cities, and areas embedded in the space. Spatial location of events specified in those reports can be mapped into *spatial expressions* (s-expressions) that are, in turn, composed of a set of *spatial descriptors* (s-descriptors) (such as `near`, `behind`, and `infrontof`) described relative to landmarks. Usually, the set of landmarks, the ontology of spatial descriptors, and the precise interpretations of both are domain and context dependent. Fig. 1 shows excerpts of free text referring to event locations and the corresponding spatial expressions. These expressions use ***WTC*** and ***West St.*** as landmarks. While the locations of landmarks are precise, spatial expressions are inherently uncertain: they usually do not provide enough information to identify the exact point-locations of the events.

Our approach to representing uncertain locations described in free text consists of a two-step process, illustrated in Fig. 2. First, a location specified as a free text is mapped into the corresponding s-expression, which in turn is mapped to its corresponding pdf representation. Given such a model, we develop techniques to represent, store, and index pdfs to support spatial analysis and efficient query execution over the pdf representations.

The **primary contributions** of this paper are:

- an approach to mapping and modeling uncertain location information from free text into the corresponding spatial expressions (Section 3);
- identification of the query requirements of SA applications (Section 4);
- a novel **U-grid** indexing framework and algorithms for efficient spatial query processing (Sections 6 and 6.1); and
- extensive empirical evaluation of the indexing (Section 7) and modeling (electronic appendix) approaches using both synthetic and real data sets.

1. We have validated this claim through a careful study of a variety of crisis-related data sets we have collected in the past. This is also addressed in [17] and [33].

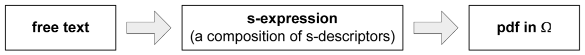


Fig. 2. Free text location $\mapsto$ pdf.

2 Related Work

This paper is based on our previously published work on uncertain spatial information modeling [21], and indexing [20]. However, this paper has the following major differences and improvements:

- This paper presents a coherent framework for managing uncertain spatial information from the data modeling and the query evaluation perspectives. Such framework is critical for building end-to-end SA applications.
- Section 6.1 provides detailed algorithmic descriptions of two new index pruning strategies that were not covered in the past. Due to these two strategies, the results published in this paper outperform those covered in [20] and [21] by a significant margin.
- Section 7 presents a real case study to demonstrate the effectiveness of the proposed modeling approach. It also discusses some analytical results to show its advantages.
- Finally, the paper provides an extensive empirical evaluation to justify our modeling and indexing strategies.

This paper considers various practical issues for building an end-to-end approach for spatial awareness from textual input. Building such a solution requires addressing several challenges, including

1. modeling spatial uncertainty in text,
2. representation,
3. indexing, and
4. query processing.

In this section, we highlight only the most related research.

Modeling. We will use the probabilistic model for spatial uncertainty developed in [5], [6], and [7], because it has the following properties:

- *Formality.* It builds on the formal probability theory.
- *Practicality.* It has been implemented in practice.
- *Generality.* This model is capable of handling (probabilistic versions of) many different types of spatial queries, as opposed to retrieval (selection) queries only.
- *Effectiveness.* Existing solutions that employ it are known to be effective and scalable.

In the probabilistic model, an uncertain location ℓ is treated as a continuous random variable (r.v.) which takes values $(x, y) \in \Omega$ and has a certain pdf $f_\ell(x, y)$ associated with it. Interpreted this way, for any spatial region R, the probability that ℓ is inside R is computed as $\int_R f_\ell(x, y)dxdy$.

In general, spatial uncertainty has been explored both in the GIS and in database literature. The GIS literature has traditionally focused on qualitative approaches to representing uncertain spatial information [11], [12], [19], [30]. Spatial relations are classified as topological relations (e.g., disjoint and overlap), direction relations (e.g., North, South), ordinal relations (e.g., inside and contain), and distance relations (e.g., far and near). Geospatial ontologies have been explored in [1] and [18]. Uncertain spatial information has been explored in the context of moving objects in [36]. In [28], authors proposed a probabilistic spatial data model that captures positional uncertainty arising due to imprecise data collection (e.g., such as GPS). In [37], the data model quantifies uncertainty arising from spatial analysis such as the discretization of thematic attributes. Another related work is on georeferencing and spatial retrieval of documents, e.g., [38]. If we view the spatial domain as a uniform grid of cells, their modeling task can be formulated as follows: Given a document, for each cell in the grid, determine the number of times this cell is covered by the regions mentioned in the document. Our modeling task is different: Given a description of an event, for each cell, determine the probability that the event happened in this cell. Finally, there has been some theoretic work, e.g., [10], on modeling spatial uncertainty in text using heuristics and fuzzy logic techniques. However, it is not clear how to train the fuzzy set models or to answer the spatial queries effectively in a large and practical setting.

Mapping text into probabilistic model. While we employ an existing spatial probabilistic model, the process of *mapping* textual locations into the corresponding representations in a probabilistic model has not been studied before. Such a mapping is one of the pivotal steps in developing the end-to-end awareness system, we cover it in Section 3.

Representation. In our context, we need to be able to represent pdfs of complex shapes in the database. There are several known methods for such a representation, such as histograms and modeling pdf as a mixture of Gaussians or of other distributions. In our approach, we will represent pdfs as histograms, which are stored in the database as quad-trees. Using quad-trees allows us to achieve fast query response time and also to compress those histograms (Section 5). It is interesting to note that the existing solutions that also deal with probabilistic spatial queries [5], [6], [7] do not address the representation issues directly. The reason is that their empirical evaluation is carried out using only simple densities such as uniform and Gaussian.

Indexing and query processing. In Sections 5 and 6.1, we introduce novel indexing and query processing techniques. Indexing pdfs to support efficient query processing has been explored in previous literature as well [6], [7]. A typical approach is to restrict possible values of an uncertain location ℓ to be inside an *uncertainty region* U_ℓ such that $f_\ell(x, y) = 0$ if $(x, y) \notin U_\ell$. Fig. 5a illustrates that concept by showing that locations of events a_1, a_2, a_3, and a_4 are restricted to their respective uncertainty regions (shaded). The uncertain location is indexed using data structures such as R-tree based on their uncertainty region. This allows answering certain spatial queries without performing costly integration operations. For instance, by analyzing the uncertainty regions in Fig. 5a, it is clear that the location of a_1 is guaranteed to be inside the region R.

The x-bounds and U-tree are the current state-of-the-art techniques studied in [7] and [34] for 1D and n-dimensional cases. They build on the ideas of indexing uncertainty regions by also taking into account the pdf part of locations to improve pruning. Both techniques have been implemented as variations of R-tree. For 1D case, the idea is to store in each internal node $\mathcal{N}$ of an R-tree extra information: left and right "x-bounds" for several values of x, e.g., $x = 0.1$ and $x = 0.2$. A "left x-bound" of $\mathcal{N}$ is any value $l(x) \in \mathbb{R}$ such that for any pdf $f(z)$ covered by $\mathcal{N}$ it follows that $\int_{-\infty}^{l(x)} f(z)dz < x$. A "right x-bound" is defined similarly. So, if a probabilistic threshold query (see Section 4) with range R and threshold p_τ overlaps the minimum bounding rectangle (MBR) of $\mathcal{N}$, but $\mathcal{N}$ stores a left x-bound such that 1) $R \subset (-\infty, l(x)]$ and 2) $p_\tau > x$, then $\mathcal{N}$ is pruned since it cannot contain any object (pdf) that would satisfy the τ-RQ.

The idea of x-bounds generalizes to 2D case, but now, in addition to left and right x-bounds, top and bottom y-bounds should be maintained as well. While x-bounds can be used for general region-based query with arbitrary shape. A special case can be formed for the axis parallel range query. In [34], authors formalize the U-PCR, which is a bounding rectangle (BR) formed based on x-bounds. Given an axis parallel range query, a new set of pruning and validation strategies has been introduced. U-PCR can be extended to handle the region query (RQ). In [35], authors have proposed to convert an RQ to a range query. So, the U-PCR bounding strategies can be applied. However, in our problem domain, RQ can be complex. For example, a first responder might want to retrieve the events that are in a particular region and are outdoor events. Such query may lead to a very irregular shaped query. Converting such query to a range query might reduce the effectiveness of the bounding strategies. Keeping x-bounds or multiple U-PCRs can increase the height of the R-tree due to the extra information kept in nodes of the R-tree, negatively affecting performance. In [7] and [34], the authors show that the gains of x-bounds outweigh their disadvantages. Notice, unlike traditional methods, the x-bound-based approaches employ for pruning not only spatial part (i.e., MBR) but also values of the pdf.

Very recently, in [35], the authors have proposed, among other things, enhanced versions for some of the heuristics covered in [7] and [34]. Both this paper and [35] have done a comparison to R-tree, and the results of our framework appear to be significantly better than those of [35]. But, since the data sets that we use (and perhaps some system parameters) are different, the validity of such an indirect comparison is questionable. A direct comparison to the techniques proposed in [35] is an interesting direction of future work.

General purpose uncertain/probabilistic database. There are also research efforts on developing Probabilistic Relational Algebra (PRA) [16]. PRA has been widely viewed as an extension to the relational algebra. Most of the extensions [4], [8], [32] assume the multinomial row distribution model (i.e., possible-worlds model). The goal is to develop suitable database query semantics under uncertain environments. Instead of going after general database solutions, this paper focuses on the spatial domain, in which continuous probability distributions are needed. Furthermore, we provide an end-to-end solution to model and process a general class of spatial queries. Due to

TABLE 1
Examples of S-Descriptors

Relation Class	Descriptors			
Topological	`indoor/inside`	`outdoor/disjoint`	`meet`	`at/equal`
Cardinal Direction	`north`	`east`	`south`	`west`
Orientation	`behind`	`in_front_of`	`to_the_left_of`	`to_the_right_of`
Distance	`within_dist`	`near`	`far`	`around`

TABLE 2
Examples of Landmark Objects

EXAMPLES OF S-DESCRIPTORS.

Landmark Class	Name+	Shape	Type	Area	Length	Height	City+
Building	Y	Polygon	Housing/business/government	m^2	NA	story/meter	Y
Street	Y	Polyline	Highway/major/minor	NA	meter	NA	Y
Street Intersection	Y	Point	Highway/major/minor	NA	NA	NA	Y

the complex nature of our modeling tasks, the resulting probability distributions may not necessarily have parametric representations and can be arbitrarily complex. Therefore, our processing techniques are very different from the ones proposed for the general uncertain databases. Although, our technique can be potentially integrated into that framework, such integration is out of the scope of this paper. We plan to study its feasibility as our future work.

We have above summarized the existing body of research on spatial uncertainty most related to this paper. Other concepts/techniques (e.g., histograms, quad-trees, and indexing), which are related to our work as well, will be discussed in this paper when the need arises.

3 Modeling Location Uncertainty

We model uncertain locations as continuous random variables that have certain pdfs associated with them. When processing a report about an event, our goal is to determine $f(x, y|report)$: the location of the event, given the information contained in that report. A report might contain several types of information that can influence $f(x, y|report)$. We focus on a frequent case where this density is context-invariant and in the form $f(x, y|s, t)$. Here, s is an s-expression and t is the type of the event. We first consider how to compute $f(x, y|s)$. After that, we will consider how $f(x, y|s, t)$ can be computed.

For instance, in the report *"A traffic accident near World Trade Center"*, we have $s = \texttt{near}(WTC)$ and $t =$ "$traffic\ accident$." Let us observe that, among all types of information mentioned in the report, s narrows down the possible location of the event most significantly. Then, we can employ the event type, "traffic accident," to refine our answer further by observing that an event of that type is more likely to occur on a road than somewhere else.

Our approach first extracts s and t from the report (Section 3.1). S-expression s is a composition of s-descriptors $\mathcal{D}_1, \ldots, \mathcal{D}_n$. S-descriptors are less complex than s-expressions and can be mapped into the corresponding pdfs (Section 3.2). The desired pdf $f(x, y|s, t)$ is computed by combining the pdfs $f(x, y|\mathcal{D}_i)$ and $f(x, y|t)$ (Section 3.3).

3.1 Mapping Free Text onto S-Expression

Mapping of free-text locations into s-expressions has been studied before in the context of spatial ontologies. Even though spatial ontologies are *not* a focus of this paper, we summarize some of the related concepts to explain our approach.

The basic idea is that each application domain $\mathcal{A}$ has, in general, its own spatial ontology $\mathcal{D}(\mathcal{A})$. The ontology defines what constitutes the landmarks in $\mathcal{A}$. It also defines the set of basic s-descriptors $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n\}$ and ways to compose them, such that any free-text location from $\mathcal{A}$ can be mapped onto a composition of s-descriptors. The four major classes of s-descriptors are topological relations (e.g., disjoint and inside) [11], cardinal direction relations (e.g., north and west) [13], orientation relations (e.g., left of and right of) [15], and distance relations (e.g., near and around) [14]. Examples of landmarks and s-descriptors are provided in Tables 1 and 2. Each s-descriptor is of the form $\mathcal{D}_i(\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_m)$: it takes as input $m \in \mathbb{N}$ landmarks, where m is determined by the type of s-descriptor. Fig. 1 shows examples of free text referring to event locations and the corresponding s-expressions. Some descriptors may not take any parameters, e.g., an ontology may use the concepts of `indoor` and `outdoor`, to mean "in some building" and "not in any building."

An s-expression consists of a set of instantiated s-descriptors. An s-expression arises when the same location ℓ is described using n different s-descriptions $\{s_1, s_2, \ldots, s_n\}$. As an example, assume a person is asked "where are you?" to which he replies "I am near building A *and* near building B", which corresponds to the s-expression: $\{\texttt{near}(A), \texttt{near}(B)\}$.

Let us note that representing event location using s-expression requires first extracting them from text. Although extracting spatial properties is complex in general, when ontologies and domains are fixed, the task becomes relatively simpler.

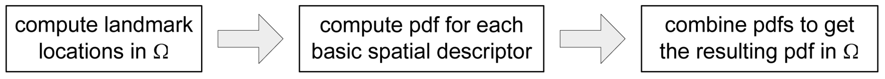

compute landmark locations in Ω
compute pdf for each basic spatial descriptor
combine pdfs to get the resulting pdf in Ω

Fig. 3. Combination of s-descriptors $\mapsto$ pdf.

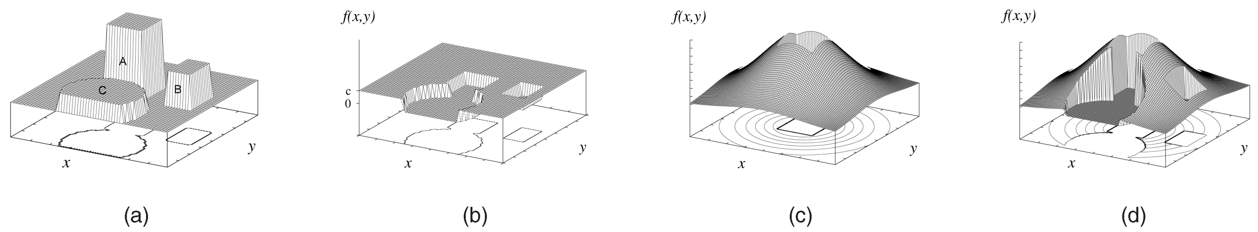


Fig. 4. Part of campus and various pdfs. (a) Part of campus. (b) PDF: `outdoor`. (c) PDF: $\texttt{near}(A)$. (d) PDF: $\{\texttt{outdoor}, \texttt{near}(A)\}$.

3.2 Pdf for a Single S-Descriptor

Merely having locations represented as spatial expressions is still not sufficient. We also need to be able to *project* the meaning of each s-expression onto the domain Ω. We achieve this by 1) computing the projection (i.e., the pdf) of each individual s-descriptor in the s-expression and 2) combining the projections, as illustrated in Fig. 3.

Let us first understand how a basic s-descriptor can be projected into Ω in an automated fashion, i.e., not manually. In Section 3.3, we will demonstrate how to compose those projections to determine the pdfs for s-expressions. Let us note that other components of the overall approach for creating spatial awareness from text are *independent* of a particular algorithm for mapping basic s-descriptors into pdfs. This section presents only one such algorithm, which can be treated as a guideline for creating customized density functions suited for a particular domain.

To illustrate the steps of the algorithm more clearly, consider a simple scenario demonstrated in Fig. 4a. This figure shows a portion of a university campus with three buildings A, B, and C. We will illustrate the concepts with the help of two descriptors: `outdoor` and $\texttt{near}(A)$. Notice that, in general, at run time, the algorithm might need to compute the pdf for $\texttt{near}(\mathcal{L})$ for any landmark $\mathcal{L}$. If it is desirable to automate this pdf computation, intuitively, one should avoid manually predefining a separate pdf per each known landmark $\mathcal{L}$ in the domain in advance, since there can be many landmarks. Instead, a more preferable approach is to design a single generic pdf-generating procedure for all possible landmarks. Given any landmark $\mathcal{L}$, such a procedure will generate the desired pdf based only on the relevant *properties* of the landmark, such as its type, footprint, and height.

That is, one method for determining the pdf $f(x, y|\mathcal{D})$ for any s-descriptor $\mathcal{D}(\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_m)$ is to make reasonable assumptions about the functional form of $f(x, y|\mathcal{D})$ based on the properties of the landmarks it takes as input. Those assumptions can be refined or rejected later on, e.g., using Bayesian framework [9].

For instance, we can define the pdf $f(x, y|\texttt{outdoor})$ for the s-descriptor `outdoor` as having the uniform distribution everywhere inside the domain Ω except for the footprints of the buildings that belong to Ω, as illustrated in Fig. 4b. That is $f(x, y|\texttt{outdoor}) = c$ for any point $(x, y) \in \Omega$ except when (x, y) is inside the footprint of a building, in which case $f(x, y|\texttt{outdoor}) = 0$. The real-valued constant c is determined from the constraint $\int_\Omega f(x, y|\texttt{outdoor})dxdy = 1$.

Another example of an s-descriptor is $\texttt{near}(A)$, which means somewhere close to the landmark A (the closer the better), but not inside A. Let us observe that, unlike the density for `outdoor`, the pdf for $\texttt{near}(A)$ is clearly *not uniform*. Rather, a more reasonable density can be a variation of the truncated-Gaussian density, centered at the center of the landmark, with variance determined by the spatial properties of the landmark A (its height, the size of its footprint). Also, since the location cannot be inside A, the values of that density should be zero for each point inside the footprint of the landmark, as illustrated in Fig. 4c. Using the above procedure, we can construct pdfs for arbitrarily complex s-descriptors in an automated fashion.

3.3 The pdf of a Spatial Expression

Now that we know how to map s-descriptors into the corresponding pdfs, let us consider how to compute the pdf for an s-expression. Let us first assume an s-expression s consists of only two subexpressions: $s = \{s_1, s_2\}$, as in $\{\texttt{near}(A), \texttt{near}(B)\}$. Our goal is to derive $f(x, y|s_1, s_2)$ from the already known $f(x, y|s_1)$, $f(x, y|s_2)$, and $f(x, y)$. Here, $f(x, y|s_1, s_2)$ is the pdf of the event location, given the report contains s_1 and s_2. The density $f(x, y)$ is the *global prior* which tells us where an event is likely to occur in the absence of any knowledge about the event. To derive $f(x, y|s_1, s_2)$, we first apply Bayes formula:

$$f(x, y|s_1, s_2) = \frac{\mathrm{P}(s_1, s_2|x, y)f(x, y)}{\mathrm{P}(s_1, s_2)},$$

where $\mathrm{P}(s_1, s_2|x, y)$ is the probability to observe s_1 and s_2 in a report, given the location is (x, y). While the presence of s_1 in a report is clearly not independent from the presence of s_2, it is reasonable to assume that they are *conditionally independent given the location.* In other words,

$$\mathrm{P}(s_1, s_2|x, y) = \mathrm{P}(s_1|x, y)\mathrm{P}(s_2|x, y),$$

but $\mathrm{P}(s_1, s_2) \neq \mathrm{P}(s_1)\mathrm{P}(s_2)$. For example, if buildings A and B are very close to each other, things that are "near A" will also tend to be "near B" and thus the two are dependent. However, once we know the location (x, y), we do not need to know whether this location is "near A" to decide whether it is "near B" and vice versa. Using the assumption of conditional independence, we have

$$f(x,y|s_1,s_2) = \frac{\mathrm{P}(s_1|x,y)\mathrm{P}(s_2|x,y)f(x,y)}{\mathrm{P}(s_1,s_2)}.$$

By applying Bayes formula, we compute $\mathrm{P}(s_1|x,y) = \frac{f(x,y|s_1)\mathrm{P}(s_1)}{f(x,y)}$ and $\mathrm{P}(s_2|x,y) = \frac{f(x,y|s_2)\mathrm{P}(s_2)}{f(x,y)}$. Thus,

$$f(x,y|s_1,s_2) = \frac{f(x,y|s_1)f(x,y|s_2)}{f(x,y)} \cdot \frac{\mathrm{P}(s_1)\mathrm{P}(s_2)}{\mathrm{P}(s_1,s_2)}. \tag{1}$$

We can assume that the global prior $f(x,y)$ is uniform, or make a weaker assumption that it is locally uniform, that is, it is not uniform in general but looks uniform inside smaller regions in Ω. Let us observe that if U_1 is an uncertainty region for $f(x,y|s_1)$ and U_2 for $f(x,y|s_2)$, then an uncertainty region for $f(x,y|s_1,s_2)$ can be computed as $U_{1\wedge 2} = U_1 \cap U_2$. For the global prior that is uniform, or locally uniform in $U_{1\wedge 2}$, (1) can be written as $f(x,y|s_1,s_2) = f(x,y|s_1)f(x,y|s_2) \cdot c$. Here, c is a real-valued constant that depends on s_1 and s_2 but does not depend on x and y. To compute c, observe that by definition of an uncertainty region, the true event location is somewhere inside $U_{1\wedge 2}$. Consequently, $f(x,y|s_1,s_2)$ integrates to 1 over $U_{1\wedge 2}$, and thus, the value of c is $1/\int_{U_{1\wedge 2}} f(x,y|s_1)f(x,y|s_2)dxdy$. Let us note that if the integral in the denominator integrates to zero, this constant is undefined. The latter corresponds to an inconsistent definition of a location, such as in "near Los Angeles and London." Thus,

$$\begin{cases} f(x,y|s_1,s_2) = f(x,y|s_1)f(x,y|s_2) \cdot \frac{1}{I} & \text{if } I \neq 0, \\ f(x,y|s_1,s_2) \text{ is undefined} & \text{if } I = 0, \end{cases}$$
$$\text{where } I = \int_{U_{1\wedge 2}} f(x,y|s_1)f(x,y|s_2)dxdy. \tag{2}$$

Similarly, for a general s-expression $s = \{s_1, \cdots, s_n\}$, it holds that $f(x,y|s_1,\ldots,s_n) = f(x,y|s_1) \times \cdots \times f(x,y|s_n) \cdot c$.

Incorporating event type. Let us observe that the event type t can be viewed as another AND condition in $f(x,y|s,t)$ and we can apply the above deduction to derive that $f(x,y|s,t) = f(x,y|s)f(x,y|t) \cdot c$. If the event type does not provide any new information where the event could occur, then we assume $f(x,y|t)$ is (locally) uniform. However, often $f(x,y|t)$ is not uniform and can help us to reduce the uncertainty further. For example, we know that $t =$ "*home robbery*" implies an event happened at a home and not in the middle of a street. It is interesting to observe that $f(x,y|t)$, in essence, serves as a *local prior*: it tells us where an event of a *given type* is likely to occur in the absence of other knowledge.

The above formulas allow us to derive the exact density $f(x,y|s,t)$ for the types of s-expressions studied in this section, so that we can answer all types of probabilistic spatial queries. As an example of a pdf that results from an s-expression, consider a possible pdf for $\{\texttt{outdoor}, \texttt{near}(A)\}$ shown in Fig. 4d, in the context of the university campus scenario from Fig. 4a. Finally, observe that in SA domains pdfs can be complex and can have highly irregular forms, which do not lend themselves to simple Gaussian or uniform approximation. Thus, special methods for representing and storing pdfs should be devised. The representation of data is normally determined by the nature of queries that are executed on top of the data. That is why we first, in Section 4, take up the types of spatial queries that need to be supported by SA applications and then, in Section 5, we discuss methods for representing pdfs.

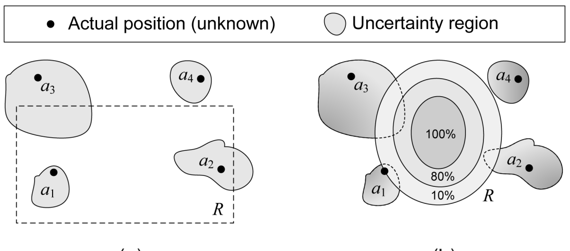


Fig. 5. Examples of $\mathrm{RQ}(R)$ and $\mathrm{PRQ}(q)$. (a) RQ with range R. (b) PRQ with preference region R.

4 Spatial Queries

SA applications, in general, should provide support for all standard types of spatial queries, such as region, NN, spatial join, and so on. We focus on an RQ, such as "find all the events, the location of which can be inside a given region."

Definition 1. *Given a region R and a set of objects $A = \{a_1, a_2, \ldots, a_n\}$, a basic* **RQ** *returns all the elements in A whose probability of being inside R is greater than zero.*

The analytical formula for computing the probability that a location $\ell \sim f_\ell(x,y)$ is located inside a region R is given by $\mathrm{P}(\ell \in R) = \int_R f_\ell(x,y)dxdy$.

Consider the example in Fig. 5a. If we assume the locations of elements $A = \{a_1, a_2, a_3, a_4\}$ are uniformly distributed in the specified (shaded) uncertainty regions, then the probabilities of these elements of being inside R might be 1.0, 0.7, 0.4, 0.0, respectively. Then, the result set for the corresponding RQ is $\{a_1, a_2, a_3\}$.

Having only elements in the result set might not be always sufficient: it is often necessary to be able to get the actual values of the probabilities associated with those elements, which leads us to the next definition:

Definition 2. *A probabilistic query is a* **detached-probability** *query if it returns elements without the probabilities associated with them. A probabilistic query is an* **attached-probability** *query, denoted as p-, if its result is a set of tuples, where each tuple consists of an element and the probability associated with this element.*

By default, any spatial query is a detached-probability query. The answer set for the p-RQ counterpart of the above RQ is $\{\langle a_1, p_1\rangle, \langle a_2, p_2\rangle, \langle a_3, p_3\rangle\}$. Given that we have assumed p_1, p_2, p_3 have specific values, the answer is $\{\langle a_1, 1.0\rangle, \langle a_2, 0.7\rangle, \langle a_3, 0.4\rangle\}$.

To reduce the amount of information, it can be desirable to present only those elements whose associated probabilities exceed a given *probability threshold* p_τ, where $p_\tau \in \mathbb{R}$, $0 \leq p_\tau \leq 1$. To generalize this idea, we can define the following:

Definition 3. *Given a threshold p_τ, query τ-Q is said to be query Q* **with the threshold semantics** *if on the same input as Q it returns all the elements from the result set of Q whose associated probabilities are greater than p_τ.*

To continue with our running example, if we set $p_\tau = 0.5$, then the corresponding τ-RQ will return $\{a_1, a_2\}$ as its result set, because p_1 and p_2 are greater than 0.5, whereas p_3 and p_4 are not. Similarly, $p\tau$-RQ will return $\{\langle a_1, 1.0\rangle, \langle a_2, 0.7\rangle\}$.

Notice, we have deliberately defined each τ-RQ as a single operation, and not as an additional filtering (τ-part) step applied to RQ operation. This is because τ-RQ, as a single query, can be optimized better, which is important for quick query response time.

Finally, observe that a regular region R can be viewed as a function $R(x,y) : \Omega \to \{0,1\}$, such that $R(x,y) = 1$ when $(x,y) \in R$, and $R(x,y) = 0$ otherwise. The formula for $\mathrm{P}(\ell \in R)$ can be written as

$$\mathrm{P}(\ell \in R) = \int_\Omega f_\ell(x,y) R(x,y)\,dxdy. \tag{3}$$

The concept of a regular region generalizes to the concept of a *preference region* $R(x,y) : \Omega \to [0,1]$, which maps a point to a preference value between zero and one. Similarly, the concept of an RQ generalizes to the concept of a **preference RQ (PRQ)**: everything is the same as for RQ, except $\mathrm{P}(\ell \in R)$ is computed using (3). PRQs give the flexibility to specify queries, where, for example, the analyst is primarily interested in objects that are within a certain area of space but also cannot ignore object in a larger area of space. An example of a PRQ with the preference region R is illustrated in Fig. 5b. There, the analyst prefers objects in the inner oval ($R(x,y) = 1$) to object in the middle ring ($R(x,y) = 0.8$) to objects in the outer ring ($R(x,y) = 0.1$). More complex preference regions can be defined, e.g., $R(x,y)$ can be a continuous function.

5 Efficient Disk-Based pdf Representations

In order to represent and manipulate pdfs with complex shapes, we first quantize the space by viewing the domain Ω as a fine uniform grid G with cells of size $\delta \times \delta$. The grid G is *virtual* and is never materialized. We use the same notation G_{ij} for both the cell in ith row and jth column of G, i.e., $G = \{G_{ij}\}$, and for the spatial region it occupies. We refer to cells of the virtual grid G as *vcells*. A *vcell* is treated as the finest element of space granularity and each spatial region can be defined by specifying the set of cells it occupies. A region R cannot occupy only part of a cell: either it occupies the whole cell or it does not occupy the cell at all.[2] We will use $vcellsin(R)$ to denote the number of cells R occupies.

The pdf $f_\ell(x,y)$ for any location ℓ is first viewed as a *histogram*: For each cell G_{ij}, the probability p^ℓ_{ij} of ℓ to be inside this cell is computed as $p^\ell_{ij} = \int_{G_{ij}} f_\ell(x,y)dxdy$. For an event with location ℓ, we are naturally interested in the set of all the cells in which this event can be located: $U_\ell = \{G_{ij} : p^\ell_{ij} \neq 0\}$. We will call all cells for which $p^\ell_{ij} = 0$ the *zero-cells* (for ℓ). Since $f_\ell(x,y)$ is a pdf, $\sum_{G_{ij} \in U_\ell} p^\ell_{ij} = 1$. We will use the notation U_ℓ to refer to both the set of cells as well as the region they occupy. Notice that the latter simply defines an uncertainty region for ℓ. The uncertainty region U_ℓ along with p^ℓ_{ij} for each $G_{ij} \in U_\ell$ defines the histogram H_ℓ for $f_\ell(x,y)$: $H_\ell = \{U_\ell, \{p^\ell_{ij} : G_{ij} \in U_\ell\}\}$.

At this point, a naive solution is to represent (and store on disk) each pdf as a histogram, e.g., by first identifying the MBR for the histogram and then treating the cells inside the MBR as a 2D array of real values, which can be stored sequentially on disk.

2. We do not consider the requirement that R is aligned to vcells to be a serious limitation. First, G can be a very fine grid. Second, a nonaligned query can be expanded to an aligned query that fully covers it. Querying with the expanded query would result in extra objects that need to be pruned, but we do not anticipate any major performance decrease due that.

5.1 Quad-Tree Representation of pdfs

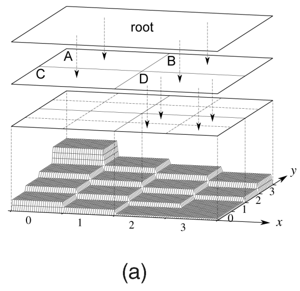
We might be able to improve the histogram representations of pdfs further by making two observations.

First, a histogram as a representation of pdf may require large storage overhead for the pdfs that cover large areas, especially if G is a very fine grid. To reduce the storage overhead as well as the number of I/Os required to retrieve a pdf from disk (thus, improving the execution time), the analyst must be able to specify that any representation of a given pdf must fit in $s_\tau \in \mathbb{N}$ amount of space. For instance, the analyst might decide that certain types of locations are just not important, and representing each finest detail of them is not required. Let us note that the text of all the original reports is stored in the database as well. Therefore, if the analyst decides to represent certain location in more detail at a later time, that can be achieved from the stored reports.

Second, keeping certain aggregate information about probabilities in a given histogram, may allow for more efficient query processing. For instance, consider a τ-RQ with a threshold p_τ and a region R, whose overlap with the MBR of a given histogram H_ℓ for a location ℓ consists of only n cells. Assume, for each histogram, we keep the maximum value p^ℓ_{max} among all the p^ℓ_{ij} probabilities in the histogram. Then, we might be able to answer the τ-RQ more efficiently. Specifically, if $np^\ell_{max} < p_\tau$, then since $\mathrm{P}(\ell \in R) \le np^\ell_{max} < p_\tau$, it immediately follows that ℓ does not belong to the answer set of the τ-RQ, without performing costly computations and many disk I/Os. We can generalize the idea of keeping aggregate information to multiple levels of resolution, not only at the MBR level.

To address the above observations, we can index each histogram H_ℓ using a space partitioning index, such as a quad-tree. First, we build a *complete* quad-tree $\mathcal{T}_\ell$ for H_ℓ. The algorithm we use is consistent with that for building quad-tree for images [31], where the goal is to index the pixels that belong to a particular image. In our case, a histogram plays the role of an image, and the nonzero cells the role of the pixels. The algorithm recursively partitions each node $\mathcal{N}$ of the quad-tree. The process stops either when $\mathcal{N}$ covers a single vcell or when all vcells $\mathcal{N}$ covers have the same probability values. In other words, it stops when the curvature under $\mathcal{N}$ is flat.

Each node $\mathcal{N}$ of the quad-tree $\mathcal{T}_\ell$ is adjusted to store certain aggregate information. Assume that the BR of $\mathcal{N}$ consists of n vcells, whose p^ℓ_{ij} values are $p_1, p_2, \ldots, p_n$. Then, if $\mathcal{N}$ is a leaf node, then by construction $p_1 = p_2 = \cdots = p_n$.



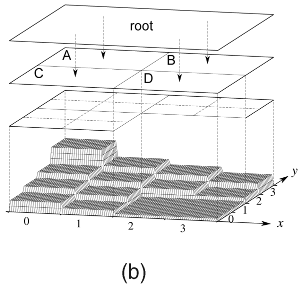


Fig. 6. Quad-tree representation of pdf. (a) Before compression. (b) After compression.

In that case, $\mathcal{N}$ stores a positive real value p_{sum} computed as $p_{sum} = \sum_{i=1}^{n} p_i = np$, where $p = p_1$. If $\mathcal{N}$ is an internal node, it stores two values: $p_{sum} = \sum_{i=1}^{n} p_i$ and $p_{max} = \max_{i=1,\dots,n} p_i$, which are used for pruning. Fig. 6a shows an example of a (three-level) complete quad-tree built on a 4×4 histogram.

The complete quad-tree pdf representation does address the issue of maintaining aggregate information at different levels of resolution. It still does not address the storage concern. We have designed a greedy compression algorithm to limit the amount of space utilized by a quad-tree. Our lossy compression technique compresses the nodes in a quad-tree recursively (bottom-up), each time compressing less costly node, until the quad-tree fits into the allotted space. The compression algorithm is covered in Section 4 in the electronic appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2008.49.

6 Indexing

Assume the goal is to evaluate a τ-RQ with some threshold. The quad-tree representation of pdfs might help to efficiently evaluate this query over each *individual* event location $\ell \sim f_\ell(x, y)$ stored in the database. However, if nothing else is done, answering this query will first require a *sequential scan* over all the event locations stored in the database, which is undesirable both in terms of disk I/O as well as CPU cost.

To solve this problem, we can create a *directory index* on top of U_ℓ (or, MBR of the histogram) for each location ℓ in the database, using an index of our choice, such as R*-tree [3]. When processing the τ-RQ with region R, we can effectively use R*-tree index to prune away all the U_ℓ which do not intersect with R. Similar techniques have been studied in [6] and [7]. This method is known to lead to improvement when compared to a sequential scan. However, this method essentially utilizes only the *spatial* part of a pdf (i.e., U_ℓ or the MBR) and disregards the fact that there is another "dimension" that can be used for pruning as well—the values of the pdf.[3] The challenge becomes to create an indexing solution that is capable of using the values of pdfs. We next discuss how to solve this challenge.

3. Let us note that "dimension" here is used figuratively, as it is not really a dimension, as defined in the traditional multidimensional indexing. Consequently, there is no straightforward way of applying the traditional approach of employing 3D index [7], [34].

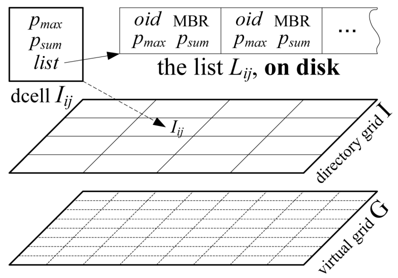


Fig. 7. Grid.

6.1 U-Grid

In this section, we propose a novel **U**ncertain **grid** (U-grid) indexing structure to effectively index uncertain event data. In addition to storing spatial information in the grid files [23], [24], [29], U-grid also stores probability summarizations, which can be effectively utilized by the query processor.

Unlike the virtual grid G, the directory U-grid I is materialized. The grid I is much coarser than G and can be viewed as a small 2D array of cells $I = \{I_{ij}\}$, residing in main memory (but it can also be stored on disk). As before, we will use I_{ij} to refer to both: the cell and the region it represents. We will refer to cells in the directory grid I as *dcells*. The structure of the grid I is shown in Fig. 7. Each cell I_{ij} in the grid I stores certain aggregate information about each event location $\ell \sim f_\ell(x, y)$ whose uncertainty region U_ℓ intersects with I_{ij}. Let us denote the set of those locations as the "*set*" L_{ij}.

The most important information in I_{ij} is a *pointer* to a **disk-resident** list, called the "*list*" L_{ij}. For each event location ℓ in the set L_{ij}, there is a list-element e_ℓ in the list L_{ij}, which stores aggregate information $\{oid, p_{max}, p_{sum}, MBR\}$ for ℓ, as illustrated in Fig. 7. The attribute $e_\ell.oid$ points to the location of the quad-tree for ℓ, on disk. The MBR of ℓ is stored in $e_\ell.MBR$. Let $n = vcellsin(U_\ell \cap I_{ij})$, and let $p_1, p_2, \dots, p_n$ be the values of the p_{ij}^{ℓ} probabilities of the n vcells in $U_\ell \cap I_{ij}$. Then, $e_\ell.p_{max}$ and $e_\ell.p_{sum}$ store the values: $e_\ell.p_{max} = \max_{i=1,\dots,n} p_i$ and $e_\ell.p_{sum} = \sum_{i=1}^{n} p_i$. The list is **sorted** in descending order on either p_{sum} or p_{max}.

Clustering. Observe that we also can retrieve $\{p_{max}, p_{sum}, MBR\}$ information directly from the root node of the quad-tree for ℓ, instead of the list-element e_ℓ for ℓ, so why do we store it in e_ℓ again? The reason is that, when we analyze elements of the list sequentially, if we access the quad-trees, we will incur an extra disk I/O per *each* location stored in the list. By storing all those attributes, we achieve *clustering* of locations. Since each e_ℓ stores information from the root (level zero) node of the quad-tree, we call that summarization technique $L0Sketch$.

Other aggregate information. In addition to storing $L0Sketch$, list elements can also store other aggregate information for the purpose of pruning. For instance, we can naturally extend the $L0Sketch$ technique by storing $\{p_{max}, p_{sum}\}$ information from more levels of the quad-tree. In particular, we shall see that using $L1Sketch$ produces the best results. $L1Sketch$ requires small additional overhead, compared to $L0Sketch$, since level-1 BRs can be derived from $e_\ell.MBR$, and thus, they do not need to be stored. The pruning power of $L1Sketch$ outweighs that of $L0Sketch$ and easily compensates for the storage overhead.

One can also consider storing x-bounds in e_ℓ as well, but that turned out not to work well in practice. The main reason is that the storage overhead caused by the multiple x-bounds.

Other attributes of I_{ij}. In addition to the pointer to the list L_{ij}, the dcell I_{ij} also has p_{max} and p_{sum} attributes, which store the maximum over all p_{max}s and over all p_{sum}s in the list L_{ij}, respectively.

Secondary grid lists for I_{ij}. Section 6.3 introduces another pruning technique. Besides keeping the primary list L_{ij} in I_{ij}, the new technique also keeps several secondary lists for better pruning.

In the next section, we discuss methods for efficient query processing using the grid.

Processing of τ-RQs and τ-PRQs consists of two logical phases: the index (pruning) phase and the object (postprocessing) phase. The first phase employs the directory-index to prune the locations that cannot satisfy the query. Two different factors are important in this process: the speed and the quality of pruning. The locations that cannot be pruned using the directory index are inserted in the special list L_{proc} to be *postprocessed* later, on the second phase. The way in which each individual event location $\ell \in L_{proc}$ is postprocessed, with respect to a given query, is independent from the choice of directory-index. The procedure that achieves that is rather straightforward. It simply traverses the quad-tree to compute the desired probability for a given query. It might stop earlier, without computing the total probability, e.g., when it becomes clear that the probability for given ℓ to satisfy the query either will or will not exceed the threshold p_τ. The details of that procedure are omitted.

6.2 Processing τ-RQs Using U-Grid

In this section, we present the algorithms for processing of a τ-RQ with a region R and a threshold p_τ using the directory U-grid I. The idea of the solution is to avoid the costly computation of the exact probability for an event location ℓ to be in R, that is, $P(\ell \in R) \overset{\text{def}}{=} \sum_{ij:G_{ij}\in U_\ell \cap R} p^\ell_{ij}$. We accomplish that by being able to find a good upper-bound λ for it, such that $P(\ell \in R) \le \lambda$. The value λ will be computed based on the aggregate information stored in the grid I. The pruning will utilize the fact that, if $\lambda < p_\tau$, then $P(\ell \in R) \le \lambda < p_\tau$, and thus, ℓ cannot satisfy the τ-RQ and can be pruned without computing the exact value of $P(\ell \in R)$.

For clarity of conveying the pruning ideas, we present the discussion in the context of the scenario where R is completely inside a single dcell I_{ij}, as illustrated in Fig. 8. This is a likely scenario when I is a coarse grid. The scenario where R intersects with multiple cells is sketched out in Section 1 in the electronic appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2008.49. When R is inside I_{ij}, the pruning phase of the τ-RQ algorithm consists of four levels of pruning:

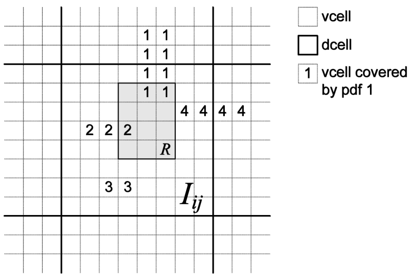


Fig. 8. τ-RQ in I_{ij}.

1. *The grid-level pruning.* Given R intersects with only one dcell I_{ij}, we know that the grid index I is constructed such that only the locations in L_{ij} can satisfy the τ-RQ. The other locations need not be considered, and thus, they are pruned.
2. *The dcell-level pruning.* Let us set
$$\lambda = \min(m \times I_{ij}.p_{max}, I_{ij}.p_{sum}),$$
where $m = vcellsin(R)$. Observe that if $\lambda < p_\tau$, then no object $\ell \in L_{ij}$ satisfies the τ-RQ, because $P(\ell \in R) \le \lambda < p_\tau$. Let us note that this pruning is carried out using only the content of the dcell I_{ij}, without performing any disk I/Os.
3. *The list-level pruning.* If the above condition does not hold, the algorithm starts to process the disk-resident list L_{ij} sequentially. In the example shown in Fig. 8, the list L_{ij} will consist of four elements, for the locations ℓ_1, ℓ_2, ℓ_3, and ℓ_4. Assume that the elements in L_{ij} are sorted in descending order of p_{sum}. Suppose that the algorithm currently observes a list-element e_ℓ that corresponds to a location ℓ. Let us choose $\lambda = e_\ell.p_{sum}$. Observe that if $\lambda < p_\tau$, then neither ℓ nor the *rest* of the location in the list will satisfy the τ-RQ, i.e., **all** of them can be pruned.
4. *The element-level pruning.* If the above pruning is not successful for the element e_ℓ, the algorithm might still be able to prune ℓ alone, but not the whole list. Let us set $\lambda = n \cdot e_\ell.p_{max}$, where $n = vcellsin(R \cap e_\ell.MBR)$. If the $L1Sketch$ strategy is employed, a possibly better upper-bound λ is computed by summing up the estimations from each quadrant: $\lambda = \sum_{i=1}^{4} n_i \cdot e_\ell.p_{max,i}$. If $\lambda < p_\tau$, then ℓ is pruned. Else, ℓ is not pruned, and ℓ is inserted in the list L_{proc} to be postprocessed. After that, the algorithm extracts the next element $e_\ell \in L_{ij}$ and goes to step 3 to apply the list-level pruning until all the element of L_{ij} are processed.

6.3 Processing τ-RQs Using Multiple Lists Per Grid Cell

In this section, we describe another powerful pruning strategy which is based on the notion of a BR τ-BR:

Definition 4. *For location ℓ and threshold p_τ, a* **bounding region** *(denoted τ-BR) is any range S such that* $P(\ell \in S) \ge p_\tau$.

Observe that

> Consider any τ-BR for location ℓ with a range S and threshold p_S and any τ-RQ with region R and threshold p_R. If ℓ belongs to the answer set of such a τ-RQ and $p_R + p_S \geq 1$ then R **must intersect** with S of any such τ-BR.

This is because the chance that ℓ is outside S is less than $1 - p_S$, and thus, for any region R that does not intersect S, it follows that $\mathrm{P}(\ell \in R) < 1 - p_S \leq p_R$.

6.3.1 Multiple Lists Per Grid Cell

Given the above observation, we design a pruning strategy that, like the x-bound technique, trades the required amount of storage to gain efficiency. Initially, n threshold values $\tau_1, \tau_2, \dots, \tau_n$ are predefined and fixed for grid I. For instance, for $n = 4$, the values might be $\tau_1 = 1$, $\tau_2 = 0.75$, $\tau_3 = 0.5$, and $\tau_4 = 0.25$. Instead of having one list L_{ij} per cell I_{ij}, the new grid I stores n such lists per I_{ij}: $L_{ij}^{(1)}, L_{ij}^{(2)}, \dots, L_{ij}^{(n)}$. Each list $L_{ij}^{(k)}$ is associated with threshold τ_k. Each element $e_\ell \in L_{ij}^{(k)}$, in addition to the information stored in the original $e_\ell \in L_{ij}$, also compactly stores information about one, or several, τ-BRs for ℓ with threshold τ_k.

6.3.2 Processing of τ-RQs Using Multiple Lists

The idea of the processing of a τ-RQ with region R and threshold p_R using the new grid is as follows: When the original algorithm would need to process L_{ij}, the new algorithm first chooses the right list $L_{ij}^{(k)}$ out of n possible lists. Observe that any list $L_{ij}^{(k)}$ such that $\tau_k + p_R \geq 1$ can be chosen for processing. The algorithm specifically chooses the $L_{ij}^{(k)}$ that corresponds to the *minimum* τ_k for which $\tau_k + p_R \geq 1$ holds. The choice of the minimum should become apparent from the subsequent discussion. Since each $e_\ell \in L_{ij}^{(k)}$ stores the same information as e_ℓ in the original list L_{ij}, all of the pruning strategies described in Section 6.2 still apply. However, since $e_\ell \in L_{ij}^{(k)}$ also stores information on τ-BRs with threshold τ_k, additional pruning is achieved by checking that each of those τ-BRs must intersect with R, otherwise ℓ can be pruned away.

6.3.3 Choosing and Representing τ-BRs for Multiple Lists

Since there can be multiple τ-BRs with threshold τ_k for each location ℓ, there should be a strategy to select them to be included in e_ℓ. We choose τ-BRs by taking into account the multiple criteria summarized next:

- **Maximizing pruning power of τ-BRs**. Ultimately, τ-BRs should be chosen such that the pruning power in processing τ-RQs is maximized.
  - **The number of τ-BRs per** e_ℓ. We need to choose the number of τ-BRs to store in each $e_\ell \in L_{ij}^{(k)}$. Intuitively, the more τ-BRs are stored in an e_ℓ, the higher the likelihood that one of them will not intersect with the query region R, and ℓ can be pruned. However, the I/O cost needed to scan $L_{ij}^{(k)}$ will also increase since more information needs to be stored in each e_ℓ.
  - **Minimizing size of τ-BRs**. It is desirable that each of the τ-BRs has small and compact region S, so that S has a greater chance not to intersect with the query region R.
  - **Separation of τ-BRs for** e_ℓ. It is desirable that the τ-BRs stored in the same e_ℓ are far apart from each other spatially and have as little intersection as possible. That is, that they have *good separation.* So, if one τ-BR intersects with R, this would increase the chance of another τ-BR not intersecting with the query region R.
- **Compact representation for τ-BRs**. τ-BRs for ℓ should be chosen such that they can be compactly encoded in e_ℓ, since the increase of e_ℓ leads to the increase of the number of the I/Os required to scan $L_{ij}^{(k)}$.
- **Fast computation of τ-BRs**. To allow for fast storage and indexing of new locations that arrive in the system, it is desirable that τ-BRs are chosen such that they can be computed quickly.

Given the above criteria, we construct list $L_{ij}^{(1)}$ by observing that the MBR of ℓ is also its τ-BR for threshold $\tau_1 = 1$, and such chosen τ-BR meets the above criteria well. Since the original L_{ij} already stores desired τ-BRs, we simply choose $L_{ij}^{(1)} = L_{ij}$ without any modifications. Since we can utilize $L_{ij}^{(1)}$ alone to answer all queries, we will refer to it as the **primary** list, while referring to the rest of the lists as **secondary.** Secondary lists, while not crucial for processing a query, are needed to do the processing more efficiently.

For secondary lists $L_{ij}^{(k)}$, where $k = 2, 3, \dots, n$, we introduce a heuristic to keep four τ-BRs per each $e_\ell \in L_{ij}^{(k)}$, which are placed at the four corners of the MBR of location ℓ. Fig. 10 demonstrates an example histogram for ℓ. Fig. 11 shows the chosen four τ-BRs for $\tau_k = 0.1$ for that histogram.

To satisfy the fast computation and compactness criteria, the algorithm extracts the four τ-BRs for ℓ from the quad-tree node for ℓ, see Section 5. For instance, the first τ-BR is chosen as the smallest top- and left-most quadrant such that its p_{sum} is greater or equal to τ_k. The other three τ-BRs are chosen similarly. Since these four τ-BRs are located at the corners, they have good separations. Also, because they are chosen from the deepest level of the quad-tree, their spatial footprints are small.

The above scheme allows for compact representation. Each τ-BR can be stored in m bits, where m is a fixed predefined parameter for grid I. Since MBR of ℓ is stored in e_ℓ, to represent a τ-BR the algorithm only needs to store the level of the quad-tree at which the quadrant for the τ-BR is chosen. If that level exceeds $2^m - 1$, it is chosen to be $2^m - 1$ in order to fit into m bits. For instance, when $m = 8$ (1 byte), levels from 0 to 255 can be encoded. Therefore, with $m = 8$, the algorithm can effectively represent the four τ-BRs in one

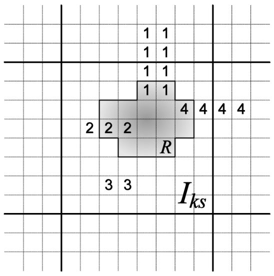


Fig. 9. PRQ in I_{ks}.

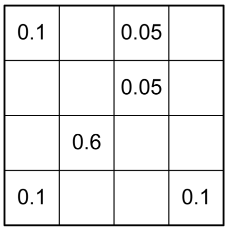


Fig. 10. Example: Histogram of ℓ.

integer, leading to small additional overhead to store them in e_ℓ.

Fig. 12 demonstrates another way to choose four τ-BRs. There, the left τ-BR for ℓ is the rectangle formed by considering the area of the MBR for ℓ that is to the left of the ℓ's x-bound, where $x = \tau$. The right, top, and bottom τ-BRs are constructed in a similar fashion. This specific way of storing τ-BRs is interesting because it is straightforward to prove that 1) all four τ-BRs require the same amount of space to store in index as one τPCR[4] and 2) the pruning power of the PCR containment heuristic from [34] is equivalent to the pruning power of using these four τ-BRs. The τ-BRs heuristic has several advantages, compared to the PCR containment strategy. First, the technique applies to a wider class of region queries, whereas the PCR heuristic from [34] does not apply to regions of arbitrary shapes, as it has been designed for range queries. Second, the τ-BR technique naturally extends to using more than four τ-BRs to potentially increase the pruning power further. The PCR technique does not extend that way, since it utilizes a PCR, and an object has only PCR.

6.4 Index Slicing for Processing τ-RQs

We can observe that many of the techniques described in Sections 6.2 and 6.3.3 also can be implemented for R-tree family of indexes by modifying leaf-level nodes of R-tree. We will describe various modifications in Section 7. We have also designed a modification of R-tree which works at the level of internal nodes of R-tree. This modification applies to R-trees that stores x-bounds. Storing multiple x-bounds increases the pruning power but, at the same time, it increases the height of R-tree since more information needs to be stored in each node. The latter, however, is not necessary if the new technique we call *index slicing* is employed.

Advanced indexes are often created by enlarging an existing index, such as R-tree, by storing extra information in nodes of the index. However, in certain situations, an alternative solution might be to create multiple indexes instead of one, each intended for a separate piece of information. For instance, x-bound index stores all k x-bounds in a node per each threshold level $\tau_1, \tau_2, \dots, \tau_k$. Instead, the idea is to create k R-trees, one per each threshold level. The ith R-tree would store only x-bounds for τ_i. To process an RQ Q with region R and threshold p_τ, the x-bound algorithm will decide which x-bound (for which τ_k) to use in each node for p_τ. In the new algorithm, the task is simply to pick the right R-tree (the one that is associated with the same τ_k) for p_τ at the beginning of processing Q. This technique keeps the pruning power of x-bounds without having the downside of increasing the height of the R-tree.

4. The 1) follows because if you know these four τ-BRs, then you can derive the τPCR and vice versa. See [34] for the definition of a PCR.

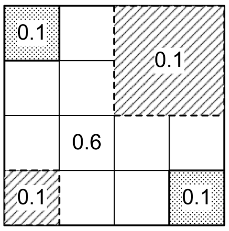


Fig. 11. The chosen τ-BRs.

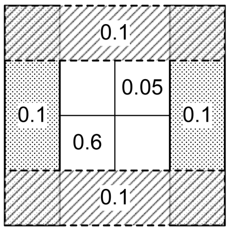


Fig. 12. Another τ-BRs.

6.5 Processing τ-PRQs Using U-Grid

Assume a τ-PRQ has a threshold p_τ and a preference region $R(x, y)$. Let us denote the average value of $R(x, y)$ in the cell G_{ij} as

$$r_{ij} : r_{ij} = \frac{1}{\delta^2} \int_{G_{ij}} R(x, y) dx dy.$$

Then, preference region $R(x, y)$ can be viewed as the set of tuples: $\{(G_{ij}, r_{ij}) : r_{ij} \neq 0\}$ and (3) transforms into $\mathrm{P}(\ell \in R) \overset{\text{def}}{=} \sum_{ij:G_{ij} \in R \cap U_\ell} p^\ell_{ij} r_{ij}$. The goal is to avoid the costly computation of the exact probability $\mathrm{P}(\ell \in R)$ by being able to find a good upper-bound λ for it, such that $\mathrm{P}(\ell \in R) \leq \lambda$. The value of λ will be derived from the aggregate information stored in the directory grid I. The pruning will be based on the observation that, if $\lambda < p_\tau$, then we can prune ℓ since it cannot be in the answer set of the τ-PRQ.

We study only the scenario illustrated in Fig. 9, where R intersects with a single dcell I_{ks}. The technique generalizes to any number of cells. First, observe that any preference region $R(x, y)$ induces a regular region $R'(x, y) = \{(x, y) : R(x, y) \neq 0\}$. That regular region can be obtained by treating all nonzero r_{ij}s of the preference region as 1s. Observe that

$$\mathrm{P}(\ell \in R) = \sum_{ij:G_{ij}\in R\cap U_\ell} p^\ell_{ij} \times r_{ij} \leq \sum_{ij:G_{ij}\in R\cap U_\ell} p^\ell_{ij} \times 1 = \mathrm{P}(\ell \in R').$$

Thus, the bounding methods in the RQ pruning algorithm can be easily applied. The upper-bound for $\mathrm{P}(\ell \in R')$ is also an upper-bound for $\mathrm{P}(\ell \in R)$ as well, because $\mathrm{P}(\ell \in R) \leq \mathrm{P}(\ell \in R') \leq \lambda < p_\tau$.

We also employ Cauchy-Shwarz (CS) inequality:

$$\begin{aligned} &x_1y_1 + x_2y_2 + \cdots + x_ny_n \\ &\leq (x_1^2 + x_2^2 + \cdots + x_n^2)^{\frac{1}{2}}(y_1^2 + y_2^2 + \cdots + y_n^2)^{\frac{1}{2}}, \end{aligned}$$

to improve on the above approach by computing a better bound λ. Using CS inequality, we have: $\mathrm{P}(\ell \in R) \stackrel{\mathrm{def}}{=} \sum \cdots \leq (\sum_{ij:G_{ij}\in I_{ks}} (p^\ell_{ij})^2)^{\frac{1}{2}} \times (\sum_{ij:G_{ij}\in I_{ks}} (r_{ij})^2)^{\frac{1}{2}} = S_\ell S_R$, where S_ℓ and S_R are the two $(\cdot)^{\frac{1}{2}}$ terms. The value $S_R \in \mathbb{R}$ is computed once per query R and cell I_{ks}. The value $S_\ell \in \mathbb{R}$ is a priori stored in each element e_ℓ of the list L_{ks} along with other aggregate information. So, when processing a list-element $e_\ell \in L_{ij}$, we can compute $\lambda = S_R \times e_\ell.S_\ell$ as one of the upper-bounds for $\mathrm{P}(\ell \in R)$.

7 Experimental Evaluation

In this paper, we experimentally evaluate the effectiveness and efficiency of the proposed indexing approach (this section) and modeling approach (electronic appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2008.49). We ran all the experiments on a P4-2 GHz PC with 1-Gbyte RAM. All our implementations are in C++. For computing and storing pdf values, we use the numerical integration functions and histogram data structure from GNU Science Library (gsl).

We have conducted two sets of modeling experiments. In the first set of experiments, we focus on the spatial events (*event centric*). In the second set of experiments, we focus on the spatial queries (*query centric*). The data set used in the event centric experiments is derived from 164 reports filed by NYPD Officers after the events of 11 September 2001. We demonstrate the advantages of our spatial modeling approach by comparing with several popular alternatives including an IR approach. For the query centric experiments, we isolate the most frequently used s-expressions into four different types. We generate one typical s-expression model for each type. We evaluate all the reasonable queries on the fixed s-expression models, and analytically compare the retrieval performance for different modeling approaches. The results indicate that our modeling approach has significant advantages over the baseline approaches. Due to the page limit, these results are covered in the electronic appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2008.49.

In the remainder of this section, we focus on the efficiency of the proposed indexing approach. The results will demonstrate the significant speedup achieved by the proposed solution, compared to the current state-of-the-art methods.

TABLE 3
Pruning Strategies

Strategies	Novel	R*-tree		U-grid		
		internal	leaf	dcell	list	element
(1) Spatial-bound		Y	Y	Y	Y	Y
(2) x-bound		Y	Y			Y
(3) Slicing	Y	Y				
(4) Max-Sum (MS)	Y	Y	Y	Y	Y	Y
(5) L1 Sketch (L1)	Y		Y			Y
(6) Mult. Lists (ML)	Y	Y	Y			Y
(7) CS	Y		Y	Y	Y	Y

7.1 Experimental Setup

Indexing methods. Table 3 summarizes the basic probability summarization techniques that can be used by an indexing structure. The existing techniques are 1 and 2, see [34]. The new ones, proposed in this paper, are 3, 4, 5, 6, and 7. While the new techniques have been described in the context of U-grid, one can notice that some of them are also applicable to R*-tree and vice versa. Thus, to have a complete study, we also incorporate our techniques in R*-tree and evaluate their results as well.

Table 3 illustrated to which part of various indexing structures (R*-tree and U-grid) a given summarization technique is applicable. For instance, it shows that L1 technique can be applied to the element level of U-grid and leaf level of R*-tree. ***Max-Sum (MS)*** refers to using (p_{max}, p_{sum}) summary to prune objects. It can be applied to R*-tree and U-grid at all levels. ***CS*** is the Cauchy-Shwarz technique employed by PRQ queries. Therefore, we have many individual index variations and we will only test the most prominent solutions.

To evaluate these techniques effectively, we group them under two schemes: R*-tree and U-grid. We use R*-tree with spatial-bound pruning as our comparison baseline, to evaluate the following existing and novel techniques:

- **Existing:** 1) Our implementation of R*-tree with x-bounds, also known as U-PCR [34], 2) our implementation of R*-tree with compressed x-bounds, also known as U-tree [34], and 3) naive grid.
- **Novel:**

  1. U-grid-MS with *MS* pruning,
  2. U-grid with x-bounds (U-grid-x) and with compressed x-bounds (U-grid-x$^+$),
  3. U-grid-MSL1 for *MS* pruning and *L1Sketch*,
  4. RTree-MS,
  5. RTree-MSL1,
  6. Multiple lists for U-grid cell (U-grid-ML), and
  7. Index slicing for RTree (RTree-Slicing).

Naive grid is a standard grid, which does not use probabilities for pruning. Note that to reduce the indexing overhead, x-bounds can also be compressed using linear approximation method [34]. U-grid-x and U-PCR store nine x-bounds for values from 0.1 to 0.9 at the interval of 0.1; the

TABLE 4
Default System Settings

Parameter	**Value**
Data Size	30,000
p_τ	0.8
Grid Size	16×16
Data Uncertainty	Medium
Query Uncertainty	Medium
Page Size	1024

compressed versions—U-grid-x$^+$ and U-tree—are built on top of these nine x-bounds.

Domain. We propose U-grid as the best solution for city-level domains. We use a larger spatial domain than the modeling experiments; a 400×400 virtual grid is overlaid on top of the Manhattan Area of New York.

Uncertain location data for testing the efficiency. The number of the spatial expression models used in the modeling experiments is too small for the scalability tests. Hence, we have generated several larger (10,000-100,000) synthetic data sets, based on these real models, as follows: We first partition the real events into three uncertainty categories, based on the size of their spatial uncertainty regions: the low (the pdf covers less than 10×10 vcells), medium (less than 20×20 vcells), and high (less than 100×100 vcells). To generate a synthetic location, we randomly choose a real uncertain location, and generate a new one with a similar pdf on top of a landmark (building, street intersection) in the domain. When storing the new pdf as a quad-tree and paginating it to the disk, an average-sized pdf occupies 5 to 10 disk pages. Using the lossy compression techniques, we constrain the size of any pdf to 100 pages maximum.

We can now control the *data uncertainty level* of a synthetic data set, by mixing objects with different uncertainty levels. In our experiments, a data set with the *medium* data uncertainty level has object uncertainty mixture ratio of $(low^{90\text{ percent}} : med^{5\text{ percent}} : high^{5\text{ percent}})$, the low and high levels are defined as:

$$low = (98\text{ percent} : 1\text{ percent} : 1\text{ percent}),$$

and $high = (50\text{ percent} : 30\text{ percent} : 20\text{ percent})$. The *medium* data uncertainty has roughly the same mixture ratio as the 2,359 real events.

Queries. Similar to the concept of object uncertainty levels, spatial queries have the low, medium, and high *coverage* levels to characterize query size: 10×10, 20×20, and 100×100 vcells. They also have *query uncertainty levels* to characterize mixes of queries of various coverage levels: *low/med/high* are the same as data uncertainty levels. All queries were aligned to vcells.

Table 4 summarizes the default experimental settings. We vary those settings to analyze the performance of the different indexing strategies. For each setting, we execute a large number of spatial queries and then report the *average* disk I/O per one query.

7.2 Experiments

Table 5 studies the impact of different database sizes and query thresholds on the I/O cost of τ-RQs for the different indexing techniques. All tested techniques are several orders of magnitude better than linear scan (not shown). In terms of the overall execution time, linear scan takes more than 2 minutes to complete even for 10,000 of data. All the techniques, except for linear scan, have two phases as discussed in the beginning of Section 6.1: the indexing (phase I) and postprocessing (phase II).

Overall performance. Table 5 shows the phase I and total I/O costs separately. U-grid-MS has the best phase I costs, and U-grid-MSL1 has the best overall costs. In all experiments, Rtree-MS has better performance than R-tree (baseline). This proves MS pruning is effective in R-tree index. However, the existing techniques based on the x-bounds (U-PCR and U-tree) do not perform well. This

TABLE 5
Disk I/O for Various Index Strategies

Methods	**Novel**	*Overall I/O Cost*						*PhaseI I/O Cost*					
		Size			τ			**Size-I**			τ**-I**		
		10k	**50k**	**100k**	**0.1**	**0.5**	**0.8**	**10k**	**50k**	**100k**	**0.1**	**0.5**	**0.8**
R*-tree		88	385	740	437	259	235	24	65	98	44	44	44
U-PCR		109	457	853	553	306	238	60	209	359	160	136	136
U-tree		83	351	677	475	241	219	34	104	183	82	72	72
Rtree-MS	**Y**	60	250	467	432	207	154	22	60	95	40	39	37
U-tree-MS	**Y**	68	279	520	474	221	180	34	102	168	87	76	76
Rtree-MSL1	**Y**	51	194	349	414	178	121	36	116	194	73	68	60
Rtree-Slice-L1-ML	**Y**	**27**	**115**	**223**	**197**	**121**	**73**	**10**	**33**	**56**	**25**	**22**	**21**
Naïve Grid		152	746	1489	650	472	449	6	23	43	14	14	14
U-grid-MS	**Y**	32	152	300	384	157	91	**5***	**16***	**29***	**14***	**12***	**10***
U-grid-x$^+$-MS	**Y**	37	174	343	407	162	104	11	43	83	38	31	27
U-grid-MSL1	**Y**	22	100	195	315	116	60	10	39	74	34	27	25
U-grid-MSL1-ML	**Y**	**19***	**88***	**175***	**214***	**112***	**54***	7	26	51	23	20	16

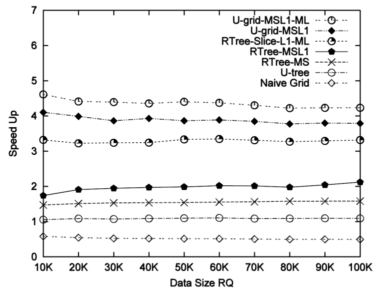


Fig. 13. Data size.

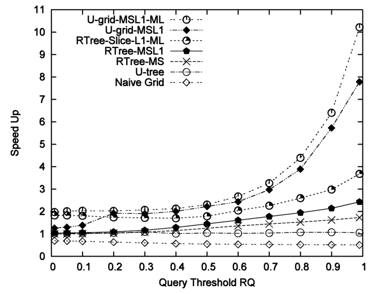


Fig. 14. Threshold.

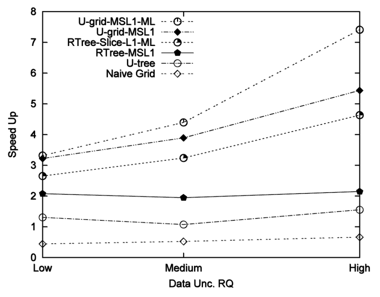


Fig. 15. Data uncertainty.

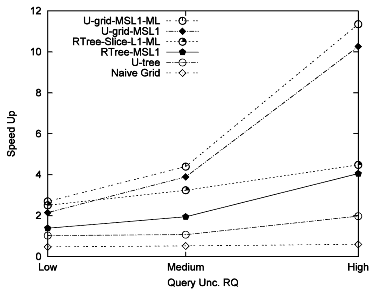


Fig. 16. Query unc.

shows that *the x-bound idea does not work well in 2D case*; the overhead in storing the x-bounds and compressed x-bounds clearly outweighs the gains from probability pruning. For a U-PCR indexing node, it needs to store additional 36 real numbers (i.e., representing various bounds) than an R-tree node. This reduces fan out by more than eight times. Even with the compression techniques, to store the parameters used in the linear approximations, U-tree still needs to reduce the fan out by more than four times. The U-tree's performance is only slightly better than that of the baseline, and U-PCR has the worst overall performance.

Data size. Fig. 13 shows the *improvement ratio (speedup)* of various indexing techniques, compared to the baseline (R-tree). The best technique from Grid group is U-grid-MSL1-ML. It shows over four times speedup than the baseline, and also over four times improvement over the best existing solution, U-tree. The best technique from R-tree group is Rtree-Slice-L1-ML. It shows more than three times better than the baseline, but it is still worse than the U-grid-MSL1-ML. The main reason for this difference is that for U-grid, the regions that correspond to lists are dcells, hence they cannot overlap; but, for RTree-List, the regions can (and do) overlap. Hence, the probabilistic pruning techniques become less effective. The experimental results confirm that $L1Sketch$ is a very effective technique for both, R-tree and U-grid. The results also show the advantages of building secondary index for both the u-grid and the Rtree-based solutions.

Query threshold. Table 5 also shows similar behavior when the query threshold p_τ for τ-RQs is varied. As we can observe from Fig. 14, the gain ratio for U-grid-MSL1-ML gets even better (up to the factor of 10) when the threshold increases. It shows the combined pruning strategy $(\mathrm{MS} + \mathrm{L1} + \mathrm{ML})$ can dismiss most of the false-positive objects instead of pushing them to phase II.

Misc. performance. In Figs. 15 and 16, we vary the data and query uncertainty levels. As data and query become more uncertain, pruning in phase I becomes very important since there are more objects intersecting with the query region. The results show that our combined pruning strategy $(\mathrm{MS} + \mathrm{L1} + \mathrm{ML})$ works very well for U-grid, but less effectively for R-tree, since the overlapping MBRs in R-tree degrade its performance.

U-grid size. Fig. 17 plots the impact of the grid size on the performance of τ-RQs for the U-grid-MSL1 technique. When the U-grid has only small number of cells, e.g., 1×1 or 2×2 cells, the overall I/O cost is high, since many objects are pushed to phase II for postprocessing. As the number of cells increases, the I/O cost **stabilizes**: having finer than 16×16 grid does not lead to significant improvement. Therefore, keeping the directory grid I in main memory, except for its disk-resident L_{ij} lists, is feasible.

PRQs. Fig. 18 studies the effect of the data size on the performance of τ-PRQs. The new technique, U-grid-MSL1-CS, achieves five times speedup over the baseline, and four times speedup over the best existing solution, U-tree.

Index overhead. The basic R-tree and Grid implementations have the smallest index sizes in their respective categories. Index slicing for an R-tree essentially builds multiple R-trees. Therefore, its size is in the order of the basic R-tree size multiplied by the number of slices. The index insertion/update time is also multiplied by the number of slices. Depending on the level of caching (L1 or L2), the size of U-grid can be twice to three times larger than that of the basic Grid index. The use of multiple grid lists multiplies the size of the index by the number of lists.

8 Conclusion and Future Work

In this paper, we presented our approach for building spatial awareness from textual input. We considered practical aspects of building such an end-to-end system such as modeling, representation, indexing, and query design and processing. We have conducted an extensive set of experiments to demonstrate the effectiveness of our solution.

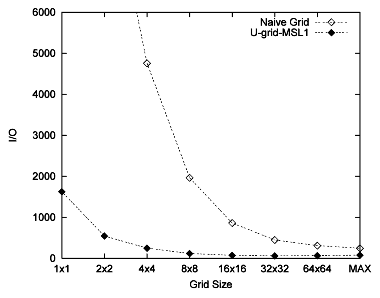


Fig. 17. Grid size.

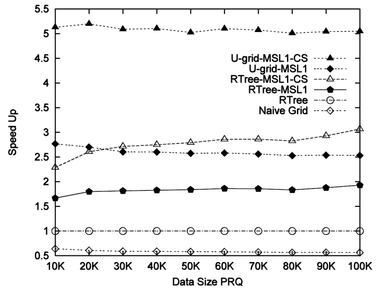


Fig. 18. PRQ.

There are several future research directions we would like to pursue. First, we would like to examine whether the existing framework can be applied to handle temporal uncertainties in addition to spatial uncertainty. Second, we intend to significantly enhance index slicing solution further, by modifying the index structure and exploring more advanced query processing algorithms. Third, we plan to examine other approaches for mapping text into probabilistic representation. Specifically, we plan to analyze the feasibility of a likelihood-based approach.

ACKNOWLEDGMENTS

This research was sponsored by US National Science Foundation Award numbers 0331707 and 0331690 to the RESCUE project.

REFERENCES

[1] I. Arpinar, A. Sheth, and C. Ramakrishnan, *Handbook of Geographic Information Science.* Blackwell Publishing, 2004.

[2] N. Ashish, D.V. Kalashnikov, S. Mehrotra, N. Venkatasubramanian, R. Eguchi, R. Hegde, and P. Smyth, "Situational Awareness Technologies for Disaster Response," *Terrorism Informatics: Knowledge Management and Data Mining for Homeland Security,* H. Chen, E. Reid, J. Sinai, A. Silke, and B. Ganoz, eds., Springer, Dec. 2007.

[3] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles," *Proc. ACM SIGMOD,* 1990.

[4] O. Benjelloun, A.D. Sarma, A. Halevy, and J. Widom, "Uldbs: Databases with Uncertainty and Lineage," *Proc. 30th Int'l Conf. Very Large Data Bases (VLDB),* 2004.

[5] R. Cheng, D.V. Kalashnikov, and S. Prabhakar, "Evaluating Probabilistic Queries over Imprecise Data," *Proc. ACM SIGMOD '03,* June 2003.

[6] R. Cheng, D.V. Kalashnikov, and S. Prabhakar, "Querying Imprecise Data in Moving Object Environments," *IEEE Trans. Knowledge and Data Eng.,* vol. 16, no. 9, pp. 1112-1127, Sept. 2004.

[7] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and Vitter, "Efficient Indexing Methods for Probabilistic Threshold Queries over Uncertain Data," *Proc. 30th Int'l Conf. Very Large Data Bases (VLDB),* 2004.

[8] N. Dalvi and D. Suciu, "Efficient Query Evaluation on Probabilistic Databases," *The VLDB J.,* 2006.

[9] R. Duda, P. Hart, and D. Stork, *Pattern Classification.* John Wiley & Sons, 2001.

[10] S. Dutta, "Approximate Spatial Reasoning," *Proc. First Int'l Conf. Industrial and Eng. Applications of Artificial Intelligence and Expert Systems (IES/AIE),* vol. 1, 1988.

[11] M. Egenhofer and J. Herring, "A Mathematical Framework for the Definitions of Topological Relationships," *Proc. Fourth Int'l Symp. Spatial Data Handling (SSDH),* 1990.

[12] A. Frank, "Ontology for Spatio-Temporal Databases," *Spatio-Temporal Databases: The CHOROCHRONOS Approach,* 2003.

[13] A.U. Frank, "Qualitative Spatial Reasoning with Cardinal Directions," *Proc. Austrian Conf. Artificial Intelligence (ÖGAI),* 1991.

[14] A.U. Frank, "Qualitative Spatial Reasoning About Distance and Directions in Geographic Space," *J. Visual Languages and Computing,* 1992.

[15] C. Freksa, "Using Orientation Information for Qualitative Spatial Reasoning," *Proc. Int'l Conf. Theories and Methods of Spatio-Temporal Reasoning in Geographic Space,* 1992.

[16] N. Fuhr and T. Rolleke, "A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database System," *ACM Trans. Information Systems,* 1997.

[17] R. Golledge, *Wayfinding Behaviour.* The Johns Hopkins Univ. Press, 1999.

[18] K. Hiramatsu and F. Reitsma, "Georeferencing the Semantic Web: Ontology Based Markup of Geographically Referenced Information," *Proc. Joint EuroSDR/EuroGeographics Workshop Ontologies and Schema Translation Services,* 2004.

[19] W. Kainz, M. Egenhofer, and I. Greasley, "Modeling Spatial Relations and Operations with Partially Ordered Sets," *Int'l J. Geographical Information Science,* vol. 7, no. 3, pp. 215-229, 1993.

[20] D.V. Kalashnikov, Y. Ma, S. Mehrotra, and R. Hariharan, "Index for Fast Retrieval of Uncertain Spatial Point Data," *Proc. 14th ACM Int'l Symp. Advances in Geographic Information Systems (ACM-GIS '06),* Nov. 2006.

[21] D.V. Kalashnikov, Y. Ma, S. Mehrotra, R. Hariharan, and C. Butts, "Modeling and Querying Uncertain Spatial Information for Situational Awareness Applications," *Proc. 14th ACM Int'l Symp. Advances in Geographic Information Systems (ACM-GIS '06),* Nov. 2006.

[22] D.V. Kalashnikov, Y. Ma, S. Mehrotra, R. Hariharan, N. Venkatasubramanian, and N. Ashish, "SAT: Spatial Awareness from Textual Input," *Proc. Int'l Conf. Extending Database Technology (EDBT '06),* demo publication, Mar. 2006.

[23] D.V. Kalashnikov, S. Prabhakar, and S. Hambrusch, "Main Memory Evaluation of Monitoring Queries over Moving Objects," *Int'l J. Distributed and Parallel Databases,* vol. 15, no. 2, pp. 117-135, Mar. 2004.

[24] D.V. Kalashnikov, S. Prabhakar, S. Hambrusch, and W. Aref, "Efficient Evaluation of Continuous Range Queries on Moving Objects," *Proc. 13th Int'l Conf. Database and Expert Systems Applications (DEXA '02),* Sept. 2002.

[25] Y. Ma, D.V. Kalashnikov, S. Mehrotra, N. Venkatasubramanian, R. Hariharan, N. Ashish, and J. Lickfett, "On-Demand Information Portals for Disaster Situations," *Proc. IEEE Int'l Conf. Intelligence and Security Informatics (IEEE ISI '07),* short publication, May 2007.

[26] S. Mehrotra, C. Butts, D.V. Kalashnikov, N. Venkatasubramanian, K. Altintas, H. Lee, A. Meyers, J. Wickramasuriya, R. Hariharan, Y. Ma, R. Eguchi, and C. Huyck, "CAMAS: A Citizen Awareness System for Crisis Mitigation," *Proc. ACM SIGMOD '04,* demo publication, June 2004.

[27] S. Mehrotra, C. Butts, D.V. Kalashnikov, N. Venkatasubramanian, R. Rao, G. Chockalingam, R. Eguchi, B. Adams, and C. Huyck, "Project RESCUE: Challenges in Responding to the Unexpected," *Proc. SPIE on Technologies and Systems for Defense and Security (SPIE '04),* vol. 5304, pp. 179-192, Jan. 2004.

[28] J. Ni, C. Ravishankar, and B. Bhanu, "Probabilistic Spatial Database Operations," *Proc. Eighth Int'l Symp. Spatial and Temporal Databases (SSTD),* 2003.

[29] J. Nievergelt, H. Hinterberger, and Sevcik, "The Grid File: An Adaptable, Symmetric Multi-Key File Structure," *Proc. Third Conf. of the European Cooperation in Informatics: Trends in Information Processing Systems (ECI),* 1981.

[30] D. Papadias and T. Sellis, "Qualitative Representation of Spatial Knowledge in Two-Dimensional Space," *The VLDB J.,* 1994.

[31] H. Samet, *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1990.

[32] A.D. Sarma, O. Benjelloun, A. Halevy, and J. Widom, "Working Models for Uncertain Data," *Proc. 22nd Int'l Conf. Data Eng. (ICDE),* 2006.

[33] M. Sorrows and S. Hirtle, "The Nature of Landmarks for Real and Electronic Spaces," *Spatial Information Theory,* vol. 1661, 1999.

[34] Y. Tao et al., "Indexing Multi-Dimensional Uncertain Data with Arbitrary Probability Density Functions," *Proc. 31st Int'l Conf. Very Large Data Bases (VLDB),* 2005.

[35] Y. Tao and X. Xiao, "Range Search on Multidimensional Uncertain Data," *ACM Trans. Database Systems,* 2007.

[36] G. Trajcevski and O. Wolfson, "Managing Uncertainty in Moving Objects Databases," *ACM Trans. Database Systems,* vol. 29, no. 3, 2004.

[37] T. Windholz, K. Beard, and M. Goodchild, "Data Quality: A Model for Resolvable Objects," *Advances in Spatial Data Quality.* Taylor-Francis, 2001.

[38] A. Woodruff and C. Plaunt, "GIPSY: Georeferenced Information Processing SYstem," *J. Am. Soc. for Information Science,* 1994.

Yiming Ma received the BSc (Hon.) degree in computer science from National University of Singapore (NUS), in 1998 and the PhD degree in computer science from the University of California, Irvine, in 2007. He is currently a research scientist at Nokia Research Center, Palo Alto (NRC-PA), California. His research interests include uncertain spatial information management, mobile contextual information integration, and data mining techniques. He is a member of the ACM.

Dmitri V. Kalashnikov received the degree (summa cum laude) in applied mathematics and computer science from Moscow State University, Moscow, in 1999 and the PhD degree in computer science from Purdue University, in 2003. He is currently a researcher at the University of California, Irvine. His current research interests include the areas of entity resolution and disambiguation, web people search, spatial situational awareness, moving-object databases, spatial databases, and GIS. He has received several scholarships, awards, and honors, including an Intel Fellowship and Intel Scholarship.

Sharad Mehrotra received the PhD degree in computer science from the University of Texas, Austin, in 1993. He is currently a professor in the Department of Computer Science, University of California, Irvine (UCI). He is also the principal investigator of the ITR-RESCUE project, which is supported by the US National Science Foundation (NSF). Previously, he was a professor at the University of Illinois, Urbana-Champaign (UIUC). His research interests include data mining, OLAP, event-oriented systems, data cleaning, multimedia systems, spatio-temporal analysis, uncertainty, privacy, service-oriented architectures, sensors, mobility, localization, and pervasive computing. He has received numerous awards, and honors, including SIGMOD best paper award 2001, DASFAA best paper award 2004, and CAREER Award 1998 from NSF.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.