

Towards Managing Uncertain Spatial Information for Situational Awareness Applications

Yiming Ma Dmitri V. Kalashnikov Sharad Mehrotra
Information and Computer Science
University of California, Irvine, USA

This research was sponsored by NSF Award number 0331707 and 0331690 to the RESCUE project

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 article, 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 datasets demonstrates the effectiveness and efficiency of our approach.

I. INTRODUCTION

Recent events (Southeast Asia Tsunamis, Hurricane Katrina) have illustrated the need for accurate and timely situational awareness tools in emergency response. Developing effective situational awareness (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. In this article, 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 September 11th, 2001:

Example I.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, neither the reports 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 afore-mentioned 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 Situational Awareness (SA) applications abound in a variety of domains including homeland security, emergency response, command and control, process monitoring/automation, 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 datasets: (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 S.E. 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 article 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.

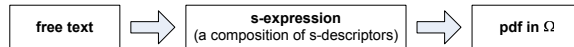
To support spatial analyses on free text reports, we need to project the spatial properties of the event described in the report onto the domain. In this article, we model uncertain event locations as random variables that have certain probability density functions (*pdfs*) associated with them. We develop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

Our approach is based on the assumption¹ that people report event locations based on certain *landmarks*. Let $\Omega \subset \mathbb{R}^2$ be a 2-dimensional physical space in which the events described in the

¹We 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 [16], [28].

<i>free text</i>	<i>s-expression</i>
'near WTC'	$\text{near}(WTC)$
'on West St., north of WTC'	$\text{on}(\text{West St.}) \wedge \text{north}(WTC)$

Fig. 1. Examples of s-expressions.

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

reports are immersed. Landmarks correspond to significant spatial objects such as buildings, streets, intersections, regions, cities, areas, etc. 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*, *infrontof*, etc) 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. Figure 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 Figure 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 article are:

- An approach to mapping and modeling uncertain location information from free text into the corresponding spatial expressions. (Section III).
- Identification of the query requirements of SA applications (Section IV).
- A novel **U-grid** indexing framework and algorithms for efficient spatial query processing (Sections VI and VII).
- Extensive empirical evaluation of the modeling (Section VIII) and indexing (Section IX) approaches using both synthetic and real data sets.

II. RELATED WORK

This article is based on our previously published work on uncertain spatial information modeling [20], and indexing [19]. However, this article has the following major differences

and improvements:

- This article 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 situational awareness applications.
- Section VII 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 article outperform those covered in [19], [20] by a significant margin.
- Section VIII 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 article provides an extensive empirical evaluation to justify our modeling and indexing strategies.

This article 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 (a) modeling spatial uncertainty in text, (b) representation, (c) indexing, and (d) query processing. In this section, we highlight only the most related research.

Modeling. We will use the probabilistic model for spatial uncertainty developed in [4]–[6], 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 probability density function (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 $\int_R f_\ell(x, y) dx dy$.

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 [10], [11], [18], [25]. Spatial relations are classified as topological relations (e.g., disjoint, overlap), direction relations (e.g., North, South), ordinal relations (e.g., inside, contain), and distance relations (e.g., far, near). Geospatial ontologies have been explored in [1],

[17]. Uncertain spatial information has been explored in the context of moving objects in [31]. In [23] authors proposed a probabilistic spatial data model that captures positional uncertainty arising due to imprecise data collection (e.g., such as GPS). In [32] 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. [33]. 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. [9], 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 III.

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 V). It is interesting to note that the existing solutions that also deal with probabilistic spatial queries [4]–[6] 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 V and VII we introduce novel indexing and query processing techniques. Indexing pdfs to support efficient query processing has been explored in previous literature as well [5], [6]. 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 $(x, y) \notin U_\ell$. Figure 5(a) in Section IV illustrates that concept by showing that locations of events a_1, a_2, a_3, 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 Figure 5(a) 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 [6], [29] for 1- 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 1-dimensional 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 Sec. IV) with range R and threshold p_τ overlaps the 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 2-dimensional 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 [29], authors formalize the U-PCR, which is a bounding rectangle formed based on x -bounds. Given a axis parallel range query, a new set of pruning and validation strategies have been introduced. U-PCR can be extended to handle the region query. In [30], authors have proposed to convert a region query to a range query. So the U-PCR bounding strategies can be applied. However, in our problem domain, region query 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 [6], [29] 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 [30] the authors have proposed, among other things, enhanced versions for some of the heuristics covered in [6], [29]. Both this article and [30] have done a comparison to R-tree and the results of our framework appears to be significantly better than those of [30]. But,

since the datasets 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 [30] is an interesting direction of future work.

General purpose uncertain/probabilistic database. There are also research efforts on developing Probabilistic Relational Algebra (PRA) [15]. PRA has been widely viewed as an extension to the relational algebra. Most of the extensions [3], [7], [27] 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 article 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 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 article. 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 article. Other concepts/techniques (e.g., histograms, quad-trees, indexing), which are related to our work as well, will be discussed in this paper when the need arises.

III. 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 $s = near(WTC)$ 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 III-A). S-expression s is a composition of s-descriptors $\mathcal{D}_1, \dots, \mathcal{D}_n$. S-descriptors are less complex than s-expressions, and can be mapped into the corresponding pdfs (Section III-B). 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 III-C).

A. 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 is *not* a focus of this article, 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 $\{\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_n\}$ 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, inside) [10], cardinal direction relations (e.g., north, west) [12], orientation relations (e.g., left of, right of) [14], and distance relations (e.g., near, around) [13]. Examples of landmarks and s-descriptors are provided in Tables I and II. Each s-descriptor is of the form $\mathcal{D}_i(\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_m)$: it takes as input $m \in \mathbb{N}$ landmarks, where m is determined by the type of s-descriptor. Figure 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 $\{s_1, s_2, \dots, s_n\}$. 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: $\{\text{near}(A), \text{near}(B)\}$.

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.

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 I

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

TABLE II

EXAMPLES OF LANDMARK OBJECTS

B. 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 (a) computing the projection (i.e., the pdf) of each individual s-descriptor in the s-expression; and (b) combining the projections, as illustrated in Figure 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 III-C, 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 Figure 4(a). 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 `near( $A$ )`. Notice that, in general, at run time the algorithm might need to compute the pdf for `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

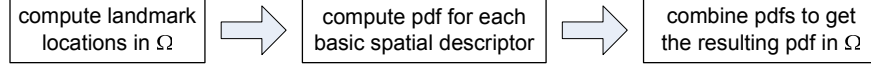
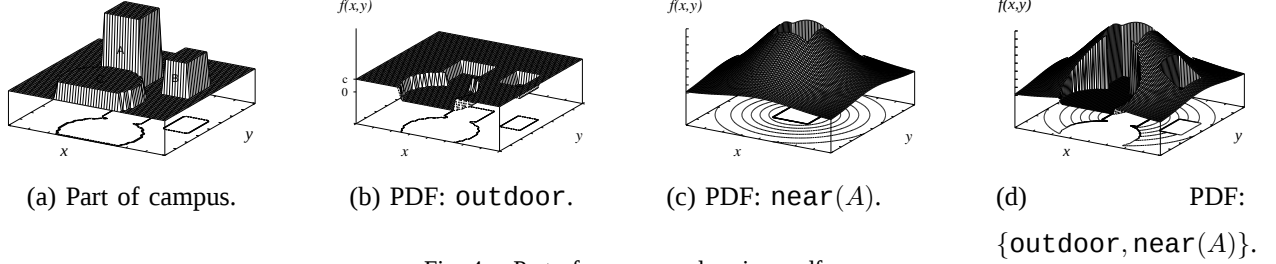
Fig. 3. Combination of s-descriptors $\mapsto$ pdf.

Fig. 4. Part of campus and various pdfs.

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 $\mathcal{D}(\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_m)$ 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 [8].

For instance, we can define the pdf $f(x, y|\text{outdoor})$ 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 Figure 4(b). That is $f(x, y|\text{outdoor}) = c$ for any point $(x, y) \in \Omega$ except when (x, y) is inside the footprint of a building, in which case $f(x, y|\text{outdoor}) = 0$. The real-valued constant c is determined from the constraint $\int_{\Omega} f(x, y|\text{outdoor}) dx dy = 1$.

Another example of an s-descriptor is `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 `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 Figure 4(c). Using the above procedure, we can construct pdfs for arbitrarily complex s-descriptors in an automated fashion.

C. 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

sub-expressions: $s = \{s_1, s_2\}$, as in $\{\text{near}(A), \text{near}(B)\}$. Our goal is to derive $f(x, y|s_1, s_2)$ from 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:

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

where $P(s_1, s_2|x, y)$ 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, $P(s_1, s_2|x, y) = P(s_1|x, y)P(s_2|x, y)$, but $P(s_1, s_2) \neq P(s_1)P(s_2)$. 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:

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

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

Thus

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

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 $U_{1 \wedge 2}$, Equation 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 $1 / \int_{U_{1 \wedge 2}} f(x, y|s_1)f(x, y|s_2)dx dy$. 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 L.A., California and London, England’.

Thus

$$\begin{cases} f(x, y|s_1, s_2) = f(x, y|s_1)f(x, y|s_2) \cdot \frac{1}{I} & \text{if } I \neq 0; \\ f(x, y|s_1, s_2) \text{ is undefined} & \text{if } I = 0, \end{cases} \quad (2)$$

$$\text{where } I = \int_{U_{1 \wedge 2}} f(x, y|s_1)f(x, y|s_2)dxdy.$$

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

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 $t = \text{‘home robbery’}$ 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 $\{\text{outdoor}, \text{near}(A)\}$ shown in Figure 4(d), in the context of the university campus scenario from Figure 4(a). 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 IV, take up the types of spatial queries that need to be supported by SA applications and then, in Section V, we discuss methods for representing pdfs.

IV. 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 a *region* query, such as “find all the events, the location of which can be inside a given region”.

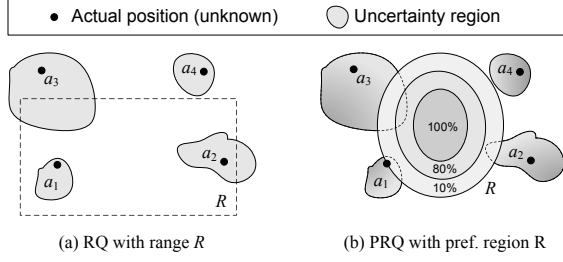
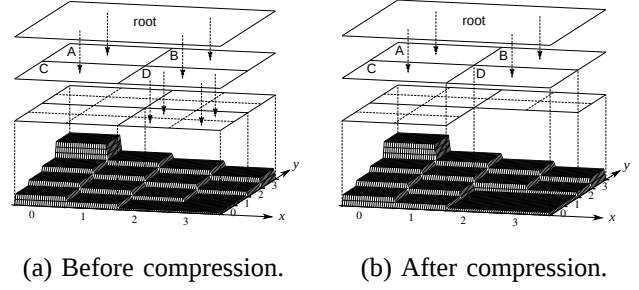
Fig. 5. Examples of RQ(R) and PRQ(q).

Fig. 6. Quad-tree representation of pdf.

Definition 1 Given a region R and a set of objects $A = \{a_1, a_2, \dots, a_n\}$, a basic **region query (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 $P(\ell \in R) = \int_R f_\ell(x, y) dx dy$.

Consider the example in Figure 5(a). 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:

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 $R(x, y) : \Omega \rightarrow \{0, 1\}$, such that $R(x, y) = 1$ when $(x, y) \in R$, and $R(x, y) = 0$ otherwise. The formula for $P(\ell \in R)$ can be written as

$$P(\ell \in R) = \int_{\Omega} f_{\ell}(x, y) R(x, y) dx dy. \quad (3)$$

The concept of a regular region generalizes to the concept of a *preference region* $R(x, y) : \Omega \rightarrow [0, 1]$, which maps a point to a preference value between zero and one. Similarly, the concept of a region query generalizes to the concept of a **preference region query (PRQ)**: everything is the same as for RQ, except $P(\ell \in R)$ is computed using Equation 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 Figure 5(b). 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.

V. 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 i -th row and j -th 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_{ij}^ℓ of ℓ to be inside this cell is computed as $p_{ij}^\ell = \int_{G_{ij}} f_\ell(x, y) dx dy$. For an event with location ℓ , we are naturally interested in the set of all the cells in which this event can be located: $U_\ell = \{G_{ij} : p_{ij}^\ell \neq 0\}$. We will call all cells for which $p_{ij}^\ell = 0$ the *zero-cells* (for ℓ). Since $f_\ell(x, y)$ is a pdf, $\sum_{G_{ij} \in U_\ell} p_{ij}^\ell = 1$. 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_{ij}^ℓ for each $G_{ij} \in U_\ell$ defines the histogram H_ℓ for $f_\ell(x, y)$: $H_\ell = \{U_\ell, \{p_{ij}^\ell : G_{ij} \in U_\ell\}\}$.

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

A. Quad-tree representation of pdfs

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

Firstly, 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.

²We 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 non-aligned 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.

Secondly, 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_{max}^ℓ among all the p_{ij}^ℓ probabilities in the histogram. Then, we might be able to answer the τ -RQ more efficiently. Specifically, if $np_{max}^\ell < p_\tau$, then since $P(\ell \in R) \leq np_{max}^\ell < p_\tau$, 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 [26], 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 non-zero 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 bounding rectangle of $\mathcal{N}$ consists of n vcells, whose p_{ij}^ℓ values are $p_1, p_2, \dots, p_n$. Then, if $\mathcal{N}$ is a leaf node, then by construction $p_1 = p_2 = \dots = p_n$. 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. Figure 6(a) shows an example of a (3-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 XIII in the electronic appendix.

VI. 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 [2]. 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 [5], [6]. 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.

A. U-grid

In this section, we propose a novel U-grid (**Uncertain grid**) indexing structure to effectively index uncertain event data. In addition to storing spatial information in the grid files [21], [22], [24], 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 2-dimensional 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 Figure 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} .

³Let us note that “dimension” here is used figuratively, as it is not really a dimension, as defined in the traditional multi-dimensional indexing. Consequently, there is no straightforward way of applying the traditional approach of employing 3-dimensional index [6], [29].

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 Figure 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 $n = vcells_{in}(U_\ell \cap I_{ij})$, 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 Bounding Rectangles (BR) 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 VII-B 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.

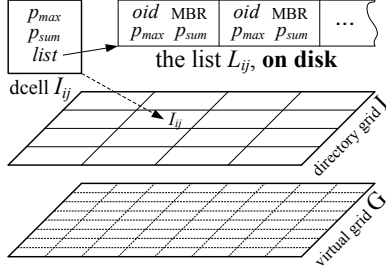
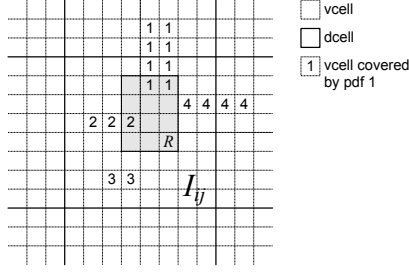
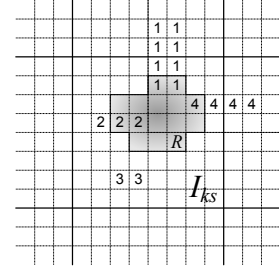


Fig. 7. Grid.

Fig. 8. τ -RQ in I_{ij} Fig. 9. PRQ in I_{ks}

VII. EFFICIENT QUERY PROCESSING

Processing of τ -RQs and τ -PRQs consists of two logical phases: the index (pruning) phase and the object (post-processing) 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 *post-processed* later, on the second phase. The way in which each individual event location $\ell \in L_{proc}$ is post-processed, 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.

A. 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, $P(\ell \in R) \stackrel{\text{def}}{=} \sum_{ij: G_{ij} \in U_\ell \cap R} p_{ij}^\ell$. We accomplish that by being able to find a good upper-bound λ for it, such that $P(\ell \in R) \leq \lambda$. 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 $P(\ell \in R) \leq \lambda < p_\tau$, 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 Figure 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 XII in the electronic appendix. When R is inside I_{ij} , the pruning phase of the τ -RQ algorithm consists of four levels of pruning:

(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 $P(\ell \in R) \leq \lambda < p_\tau$. 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 Figure 8, the list L_{ij} will consist of four elements, for the locations $\ell_1, \ell_2, \ell_3, \ell_4$. 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 post-processed. 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.

B. 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 bounding rectangle τ -BR:

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

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 $P(\ell \in R) < 1 - p_S \leq p_R$.

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 .

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 VII-A 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.

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 . Figure 10 demonstrates an example histogram for ℓ . Figure 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 V. 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.

0.1	0	0.02	0
0	0.2	0.06	0.02
0	0.2	0.2	0
0.1	0	0	0.1

Fig. 10. Example: Histogram of ℓ .

0.1		0.1	
0.1			0.1

Fig. 11. The Chosen τ -BRs.

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 $m = 8$ (one byte), levels from 0 to 255 can be encoded. Therefore, with $m = 8$ the algorithm can effectively represent the four τ -BRs in one integer, leading to small additional overhead to store them in e_ℓ .

C. Index Slicing for Processing τ -RQs

We can observe that many of the techniques described in Sections VII-A and VII-B.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 IX. 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 i -th R-tree would store only x-bounds for τ_i . To process a region query 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.

D. 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 Equation 3 transforms into $P(\ell \in R) \stackrel{\text{def}}{=} \sum_{ij: G_{ij} \in R \cap U_\ell} p_{ij}^\ell r_{ij}$. The goal is, to avoid the costly computation of the exact probability $P(\ell \in R)$ by being able to find a good upper-bound λ for it, such that $P(\ell \in R) \leq \lambda$. 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 Figure 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 non-zero r_{ij} 's of the preference region as 1's. Observe that $P(\ell \in R) = \sum_{ij: G_{ij} \in R \cap U_\ell} p_{ij}^\ell \times r_{ij} \leq \sum_{ij: G_{ij} \in R \cap U_\ell} p_{ij}^\ell \times 1 = P(\ell \in R')$. Thus, the bounding methods in the RQ pruning algorithm can be easily applied. The upper-bound for $P(\ell \in R')$ is also an upper-bound for $P(\ell \in R)$ as well, because $P(\ell \in R) \leq P(\ell \in R') \leq \lambda < p_\tau$.

We also employ Cauchy-Shwarz (CS) inequality:

$$x_1 y_1 + x_2 y_2 + \dots + x_n y_n \leq (x_1^2 + x_2^2 + \dots + x_n^2)^{\frac{1}{2}} (y_1^2 + y_2^2 + \dots + y_n^2)^{\frac{1}{2}}$$

to improve on the above approach by computing a better bound λ . Using CS inequality, we have: $P(\ell \in R) \stackrel{\text{def}}{=} \sum \dots \leq (\sum_{ij: G_{ij} \in I_{ks}} (p_{ij}^\ell)^2)^{\frac{1}{2}} \times (\sum_{ij: G_{ij} \in I_{ks}} (r_{ij})^2)^{\frac{1}{2}} = S_\ell S_R$, 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 $P(\ell \in R)$.

VIII. EXPERIMENTAL EVALUATION: MODELING

In this section, we experimentally evaluate the effectiveness of the proposed modeling approach (Section III). In the next section (Section IX), we will study the efficiency of the proposed

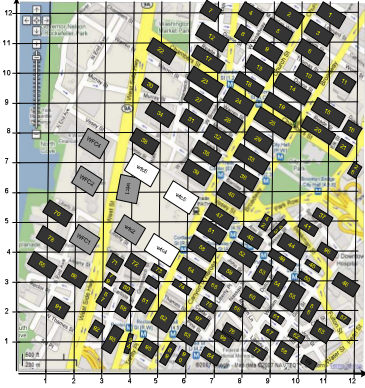


Fig. 12. Spatial Domain

Landmark Class	Template	Parameters	Descriptors
Building	Gaussian	σ_x, σ_y, ρ	near, around
Street	Uniform	margins	at/on, around
Street Intersection	Gaussian	σ_x, σ_y, ρ	near, around

Fig. 13. Model Templates and Parameters

indexing approach. We ran all the experiments on a P4-2GHz PC with 1GB RAM. All our implementations are in C++. We use the numerical integration functions and the histogram data structure from GNU Science Library (gsl) for computing and storing pdf values.

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 September 11th, 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 4 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. Due to the page limit, we put the query centric experiments into the electronic appendix.

A. Event Centric Experiment Setup

Spatial Information Extraction. We extract the uncertain location data from 164 reports filed by NYPD Officers after the events of September 11th, 2001. We focus our attention to the area around World Trade Center shown in Figure 12. To make sure that the landmark objects have up-to-date names and locations, we obtain them from Google map server, and store 69 building objects, 30 street objects, and 28 intersection objects as look-up dictionaries in our extractor. We also incorporate name aliases and short-forms for different landmarks. Using the dictionaries and simple pattern matching rules, our extraction tool extracts the landmarks (buildings, street

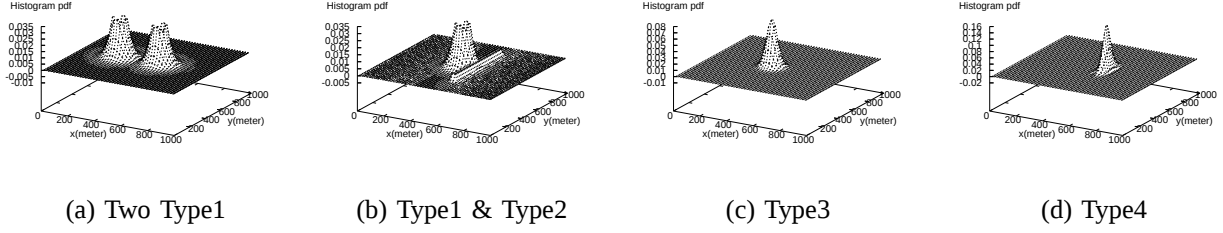


Fig. 14. Spatial expression types.

intersections, and streets in our case) automatically, and zooms into the corresponding text and highlights the s-descriptors. Our tool also provides an initial suggestion of combining related s-descriptors at sentence level to form an s-expression. The analyst can either accept or correct the suggestion; the latter offers an opportunity to the analyst to combine them differently. As a result, our extractor is able to correctly identify 90% of the spatial expressions by itself. The extraction errors are mainly from identifying the spatial relation names (e.g., `in_front_of`)⁴. After the extraction process, we identified 1097 spatial events that are within the region. We form our spatial expression based on these event descriptions. The spatial expressions are composed from 56 spatial descriptors; 44% of the spatial expressions have more than 1 descriptor⁵.

Modeling Spatial Expressions. For the modeling purpose, a 120×120 virtual grid is overlaid on top of this dataset, with 10×10 m² cells, covering 1.2×1.2 km² (Figure 12). Using the modeling templates illustrated in Figure 13, we modeled the 56 s-descriptors used in the s-expressions. The 1097 s-expressions are composed using the s-descriptors (see Section III). We summarize the s-expressions into the following four basic types:

- Type 1: $\{Near(B1)\}$, where B1 is a building or intersection
- Type 2: $\{On(St)\}$, where St is a street
- Type 3: $\{Near(B1), Near(B2)\}$
- Type 4: $\{Near(B1), On(St)\}$

S-expressions involving other topological, directional, and distance descriptors can be modeled based on these four types. In fact, these four types cover more than 80% of the spatial expressions. Given the four types of s-expressions, Table III shows the parameter settings we used for

⁴We believe the extraction performance can be further improved if we use more complicated extraction rules for identifying spatial relations.

⁵All the s-expressions include the `Outdoor` descriptor by default.

Type	Parameters	Level 1 Target	Level 2 Target	Level 3 Target
Type1	$\sigma_x = EXT.x$ $\sigma_y = EXT.y$ $\rho = 0$	$\Delta\sigma = \pm[0, 20\%]\sigma$ $\rho = \pm[0, 0.1]$	$\Delta\sigma = \pm[20\%, 50\%]\sigma$ $\rho = \pm[0.1, 0.25]$	$\Delta\sigma = \pm[50\%, 100\%]\sigma$ $\rho = \pm[0.25, 0.5]$
Type2	$M1 = M2 = 10m$	$\Delta M = \pm[0, 20\%]M$	$\Delta M = \pm[20\%, 50\%]M$	$\Delta M = \pm[50\%, 100\%]M$
Type3	Same as Type1	Same as Type1	Same as Type1	Same as Type1
Type4	Same as Type1,2	Same as Type1,2	Same as Type1,2	Same as Type1,2

TABLE III

INITIAL MODEL DESCRIPTIONS AND TARGET PERTURBATION LEVELS

constructing the s-expression models. Based on our spatial domain, the parameters for a Type1 model has standard deviations along x and y directions that equal to the spatial extents of the landmark ⁶, and it has no correlations between x and y directions. A Type2 model has 10m margins to model the spatial extents along the street. Type3 and Type4 has the same settings as Type1 and Type2. Figure 14 demonstrates the example models for the different spatial expression types.

Target Spatial Expression Models. During the process of converting s-expressions to pdfs, it is possible that the default model parameter settings for some s-expressions will deviate from the optimal parameter settings, resulting in a certain level of modeling error. In our experiments, we study the impact of such modeling error to the retrieval quality. Specifically, we study how the retrieval quality changes when the optimal model parameters are perturbed at different levels. This simulates the situation when the parameters set by default by the system deviate from the optimal ones by different degrees, ranging from minor to major perturbations. Table III shows the perturbation levels for different spatial expression types used in our experiments. The perturbations are not fixed within a level, and bounded by a range (e.g., $\pm[0, 20\%]$ of the initial value). Some parameters like standard deviation can go both direction (i.e., increase/decrease), which is decided randomly.

Competing Modeling Strategies. Given the target expression models, we can evaluate different modeling strategies. In this article, we compare four different modeling approaches: two types of uniform modeling approach, a classic information retrieval approach (IR), and our probabilistic modeling approach (*Prob*). As common in related probabilistic approaches, in *Prob* we also employ bounded versions of all densities, e.g. bounded/truncated Gaussian. The first

⁶If an intersection is involved, we set the spatial extents to be 50m for x and y .

type of uniform model is called UBR (Uniform Bounding Rectangle), and the second type is called UGC (Uniform Grid Cells). A UBR model uses one single BR to bound the modeling region of an s-expression. The approach is widely supported by the database systems (e.g., DB2, Oracle), and many applications use it to model spatial uncertainties. Instead of one BR, one can also use multiple BRs or other shapes, to compare to these variations, a UGC model uniformly distributes the probability mass to the involved grid cells in the model region of an s-expression, and therefore fully preserves the shape of the uncertain region. In another word, UGC approach tries to represent the best uniform models that can be achieved. For the IR approach, we first construct a feature space that has all the landmark objects as its dimensions. Given an s-expression, its IR model consists of a binary vector Γ_{IR} that represents the landmarks appeared in the s-expression.⁷ The model is similar to the bag-of-word model used in the IR literature, for which a set of distinct words are used as feature dimensions.

Query Evaluation. We construct a set of 78 RQ queries around WTC. It consists of 3 query classes: 58 small (10×10 vcells), 16 medium (20×20 vcells), and 4 large (50×50 vcells) queries. To produce unbiased queries, for each query class, the queries are non-overlapping and uniformly placed in the spatial domain.

Given a RQ and a probabilistic s-expression model (*Prob*), we have described the evaluation method in Section IV; it computes the integral of the pdf in the RQ region. For a UBR or UGC model, we essentially compute the ratio of the region intersection between a uniform model and RQ region. To evaluate an IR model, we use the standard Cosine similarity. Similar to the process of constructing an IR model for an s-expression, we also construct a binary vector that represents the landmark objects in a RQ region (Γ_{RQ}). Given the landmark vector in the IR model (Γ_{IR}), we compute the similarity score using: $Score(IR, RQ) = \frac{\Gamma_{RQ} \cdot \Gamma_{IR}}{\sqrt{|\Gamma_{IR}| \cdot |\Gamma_{RQ}|}}$.

To evaluate a set of s-expressions given a RQ, we first generate target s-expression models based on the target perturbation strategy described earlier. At a fixed query threshold τ , we evaluate the RQ on the target s-expression models, and store the s-expressions that satisfy the RQ into a set S_T . We then evaluate the RQ on the s-expression models generated by a modeling approach ($\widehat{M}$), and define $S_{\widehat{M}}$ as a set of s-expressions that satisfy the RQ. We can now define

⁷Instead of binary vector, more complicated weighing scheme can also be used to attach weights to different landmark dimensions.

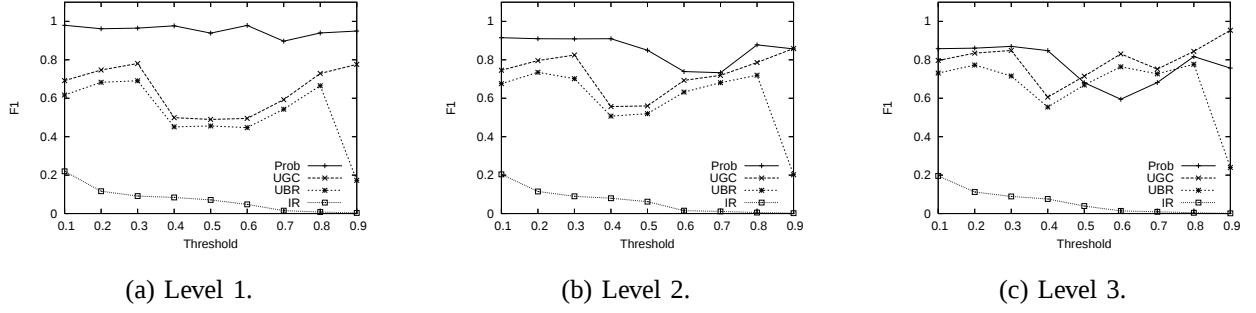


Fig. 15. F1 measures with different level of target perturbations

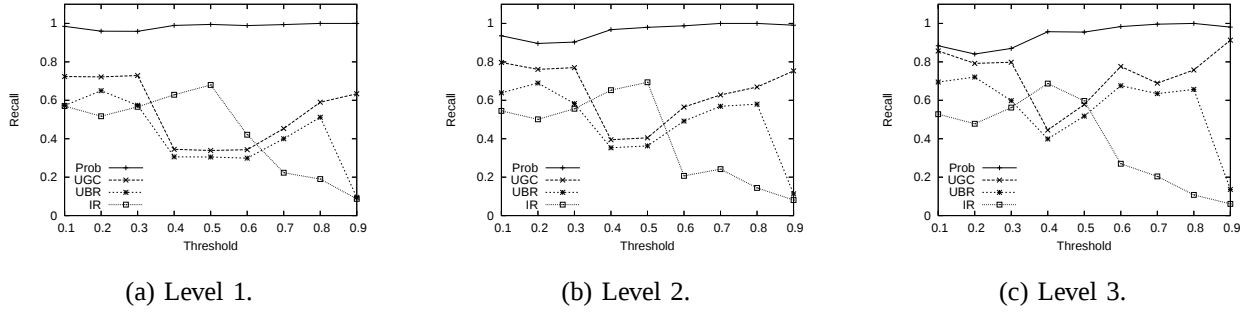


Fig. 16. Recall measures with different level of target perturbations

the retrieval quality metrics of a modeling approach at a particular query threshold as follows:

$$\bullet \text{ Recall}_\tau = \frac{S_{\hat{M}} \cap S_T}{|S_T|} \quad \bullet \text{ Precision}_\tau = \frac{S_{\hat{M}} \cap S_T}{|S_{\hat{M}}|} \quad \bullet F1_\tau = \frac{2 \times \text{Recall}_\tau \times \text{Precision}_\tau}{\text{Recall}_\tau + \text{Precision}_\tau}$$

B. Event Centric Experiment Results

We execute the 78 RQs on all the s-expression models at 9 different threshold levels (0.1 to 0.9). We average the performance measures at different thresholds. In Figure 15, we show the F1 measures for different modeling approaches. At low target perturbation level (Level 1), our probabilistic modeling approach (*Prob* in the plots) clearly outperforms other methods. This is expected since *Prob* models are similar to their respective target models. It still performs well for the moderate perturbation (Level 2), but becomes less effective when target models are away from the *Prob* models (Level 3). However, when we study the recall measures (Figure 16), we notice *Prob* models have much better recall measures than the uniform and IR models at all 3 levels. Recall measure is important to many situational awareness applications. Poor recall measure may lead to the failures in retrieving relevant events. Overall, we observe that the IR modeling approach leads to the worst retrieval quality. This shows the advantages of mapping spatial event descriptions to the spatial models. We also observe that more accurate probabilistic

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

TABLE IV

PRUNING STRATEGIES

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

TABLE V

DEFAULT SYSTEM SETTINGS

models have advantages over simple uniform models (UBR and UGC).

IX. EXPERIMENTAL EVALUATION: INDEXING

In this section, we experimentally study 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.

A. Experimental Setup

Indexing methods. Table IV summarizes the basic probability summarization techniques that can be used by an indexing structure. The existing techniques are 1 and 2, see [29]. The new ones, proposed in this article, 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 IV 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 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 [29], (2) Our implementation of R*-tree with compressed x -bounds, also known as U-tree [29], and (3) Naïve 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), (7) Index slicing for RTree (RTree-Slicing).

Naïve 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 [29]. 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 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 (10K–100K) synthetic datasets, based on this 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 dataset, by mixing objects with different uncertainty levels. In our experiments, a dataset with the *medium* data uncertainty level has object uncertainty mixture ratio of ($low^{90\%} : med^{5\%} : high^{5\%}$), the low and high levels are defined as: $low = (98\% : 1\% : 1\%)$, and $high = (50\% : 30\% : 20\%)$. The *medium* data uncertainty has roughly the same mixture ratio as the 2359 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.

Methods	Novel	Overall I/O Cost						Phase I 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

TABLE VI

DISK I/O FOR VARIOUS INDEX STRATEGIES

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 V 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.

B. Experiments

Table VI 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 10K of data. All the techniques, except for linear scan, have two phases as discussed in the beginning of Section VII: the indexing (phase I) and post-processing (phase II).

Overall performance. Table VI 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

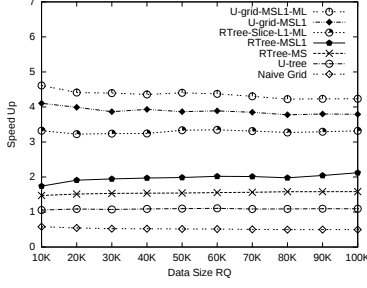


Fig. 17. Data Size

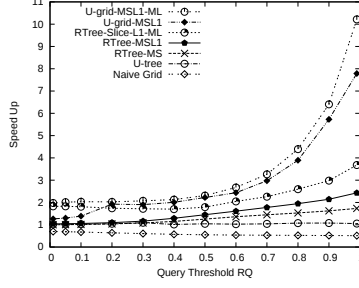


Fig. 18. Threshold

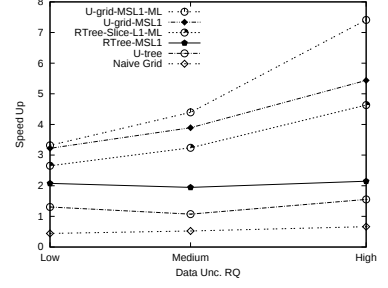


Fig. 19. Data Uncertainty

in R-tree index. However, the existing techniques based on the x -bounds (U-PCR and U-tree) do not perform well. This shows that *the x -bound idea does not work well in 2-d 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 a R-tree node. This reduces fan out by more than 8 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 4 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. Figure 17 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 4 times speedup than the baseline, and also over 4 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 3 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 VI also shows similar behavior when the query threshold p_τ for τ -RQs is varied. As we can observe from Figure 18, 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 (MS+L1+ML) can dismiss most of the false positive objects instead of pushing them to phase-II.

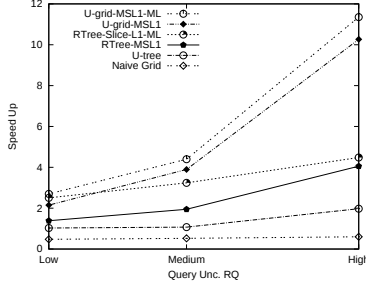


Fig. 20. Query Unc.

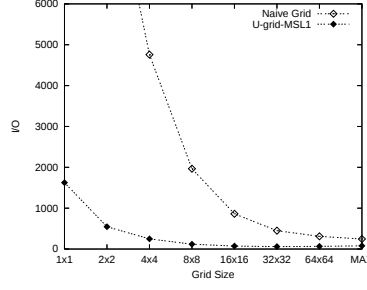


Fig. 21. Grid size

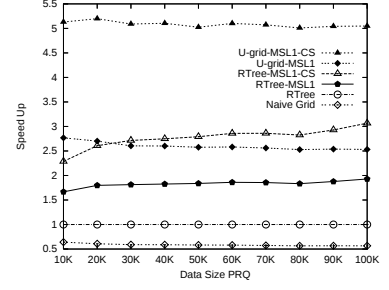


Fig. 22. PRQ

Misc. performance. In Figures 19 and 20 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 (MS+L1+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. Figure 21 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 post-processing. 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. Figure 22 studies the effect of the data size on the performance of τ -PRQs. The new technique, U-grid-MSL1-CS, achieves 5 time speedup over the baseline, and 4 time 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.

X. CONCLUSION AND FUTURE WORK

In this article 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.

There are several future research direction 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.

REFERENCES

- [1] I. Arpinar, A. Sheth, and C. Ramakrishnan. *Handbook of Geographic Information Science*. Blackwell Publish., 2004.
- [2] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *ACM SIGMOD*, 1990.
- [3] O. Benjelloun, A. D. Sarma, A. Halevy, and J. Widom. Uldbs: Databases with uncertainty and lineage. In *VLDB Conference*, 2004.
- [4] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD*, 2003.
- [5] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Querying imprecise data in moving object environments. *IEEE TKDE*, 16(9), Sept. 2004.
- [6] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and Vitter. Efficient indexing methods for probabilistic threshold queries over uncertain data. In *Proc. of VLDB*, 2004.
- [7] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB Journal*, 2006.
- [8] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley and Sons, 2001.
- [9] S. Dutta. Approximate spatial reasoning. *IEA/AIE*, 1, 88.
- [10] M. Egenhofer and J. Herring. A mathematical framework for the definitions of topological relationships. In *Int'l Symp. on Spatial Data Handling*, 1990.
- [11] A. Frank. Ontology for spatio-temporal databases. In *Spatio-Temporal Databases: The CHOROCHRONOS Approach*, 2003.
- [12] A. U. Frank. Qualitative spatial reasoning with cardinal directions. In *In Proceedings of the Austrian Conference on Artificial Intelligence*, 1991.
- [13] A. U. Frank. Qualitative spatial reasoning about distance and directions in geographic space. In *Journal of Visual Languages and Computing*, 1992.
- [14] C. Freksa. Using orientation information for qualitative spatial reasoning. In *In Proceeding of International Conference on Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, 1992.
- [15] N. Fuhr and T. Rolleke. A probabilistic relational algebra for the integration of information retrieval and database system. In *ACM Trans. Inf. Syst. (TOIS)*, 1997.
- [16] R. Golledge. *Wayfinding behaviour*. The Johns Hopkins University Press, 1999.
- [17] K. Hiramatsu and F. Reitsma. Georeferencing the semantic web: ontology based markup of geographically referenced information. In *EuroSDR/EuroGeographics*, 2004.

- [18] W. Kainz, M. Egenhofer, and I. Greasley. Modeling spatial relations and operations with partially ordered sets. In *Int'l J. of GISs*, 7(3):215-229, 1993.
- [19] D. Kalashnikov, Y. Ma, S. Mehrotra, and R. Hariharan. Index for fast retrieval of uncertain spatial point data. In *Proc. of Int'l Symposium on Advances in Geographic Information Systems (ACM GIS)*, Nov 10–11 2006.
- [20] D. Kalashnikov, Y. Ma, S. Mehrotra, R. Hariharan, and B. Butts. Modeling and querying uncertain spatial information for situational awareness applications. In *Proc. of Int'l Symposium on Advances in Geographic Information Systems (ACM GIS)*, Nov 10–11 2006.
- [21] D. V. Kalashnikov, S. Prabhakar, and S. Hambrusch. Main memory evaluation of monitoring queries over moving objects. *Distributed and Parallel Databases*, 15(2):117–135, Mar. 2004.
- [22] D. V. Kalashnikov, S. Prabhakar, S. Hambrusch, and W. Aref. Efficient evaluation of continuous range queries on moving objects. In *Proc. DEXA Conference*, September 2–6 2002.
- [23] J. Ni, C. Ravishankar, and B. Bhanu. Probabilistic spatial database operations. In *SSTD*, 2003.
- [24] J. Nievergelt, H. Hinterberger, and Sevcik. The grid file: An adaptable, symmetric multi-key file structure. In *ECI'81*.
- [25] D. Papadias and T. Sellis. Qualitative representation of spatial knowledge in two-dimensional space. In *VLDB J*, 94.
- [26] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1990.
- [27] A. D. Sarma, O. Benjelloun, A. Halevy, and J. Widom. Working models for uncertain data. In *IEEE ICDE Conference*, 2006.
- [28] M. Sorrows and S. Hirtle. The nature of landmarks for real and electronic spaces. *Spatial Information Theory*, 1661, 99.
- [29] Y. Tao, R. Cheng, X. Xiao, W. Ngai, and et al. Indexing multi-dimensional uncertain data with arbitrary probability density functions. In *VLDB*, 2005.
- [30] Y. Tao and X. Xiao. Range search on multidimensional uncertain data. In *TODS*, 2007.
- [31] G. Trajcevski and O. Wolfson. Managing uncertainty in moving objects databases. In *ACM TODS*, 29(3), 2004.
- [32] T. Windholz, K. Beard, and M. Goodchild. Data quality: A model for resolvable objects. In *Advances in Spatial Data Quality*, Taylor-Francis, 2001.
- [33] A. Woodruff and C. Plaunt. *GIPSY: Georeferenced Information Processing SYstem*. Journal of the American Society for Information Science, 1994.

ELECTRONIC APPENDIX

XI. QUERY CENTRIC EXPERIMENTS

In Section VIII-A we illustrated the overall flow of the spatial retrieval process from the event-centric perspective. We focused on the spatial events and showed the advantages of the probabilistic modeling. In this section we study the proposed solution from the query-centric perspective. We focus on the four basic types of s-expressions introduced in Section VIII-A. We analytically study the retrieval quality under a reasonable large set of RQs.

A. Query Centric Experiment Setup

Evaluation Framework. We define the concept of relevant query set (Definition 5), and use it to derive appropriate performance metrics that can be used objectively.

Definition 5 (Relevant Range Query Set (RRQ)) *Given a spatial expression model M and a query threshold τ , the relevant query set S_M contains all the range queries that satisfy the threshold τ when evaluated on M .*

Given a spatial expression, let us assume the true expression model to be M . A modeling approach generates an approximated model $\widehat{M}$. At a fixed query threshold, based on Definition 5, we can define two query sets, Q_M and $Q_{\widehat{M}}$, of RQs that are relevant to M and $\widehat{M}$. Then, precision and the recall of the modeling approach at a particular query threshold can be defined as:

$$\bullet \text{ Recall}_\tau = \frac{|Q_{\widehat{M}} \cap Q_M|}{|Q_M|} \quad \bullet \text{ Precision}_\tau = \frac{|Q_{\widehat{M}} \cap Q_M|}{|Q_{\widehat{M}}|}$$

If the modeling approach performs well, its $Q_{\widehat{M}}$ should be almost identical to Q_M . Since we consider all the relevant RQs, we effectively cancel away the subjectiveness involved in choosing the RQs. Based on the recall and precision measures, we can also compute the well-known F1 measure as the harmonic mean of the precision and recall.

Definition 6 (Candidate Range Query Set (CRQ)) *Given a spatial expression model M represented as a pmf in a grid G , and its associated MBR (Minimum Bounding Rectangle), a candidate query set (CRQ) contains all the range queries in G , such that each RQ intersects with MBR.*

To form the RRQ (relevant range query set), we first construct a CRQ (Candidate Range Query Set) defined in Definition 6. Obviously, CRQ can contain infinite number of RQs in a

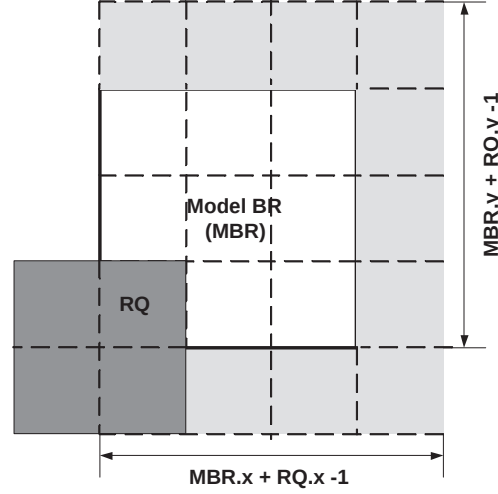


Fig. 23. Cand. RQ Set (CRQ)

continuous spatial domain. However, in a grid setting, when the spatial extents (i.e., the number of cells on x and y dimensions) of the RQs are fixed, the CRQ cardinality is equal to $(MBR.x + RQ.x - 1) \times (MBR.y + RQ.y - 1)$, which is bounded by $O(MBR.x \times MBR.y)$. For example, Figure 23 shows a 3×3 MBR, if the spatial extents for RQs are 2×2 , we will have 16 (4×4) RQs in CRQ. We can now evaluate CRQ queries on model M to get the RRQ.

For each basic query type, we construct *one* typical s-expression model. For Type1 s-expression, we specify the spatial extents of a building or an intersection landmark to be $70m \times 70m$. For Type2 s-expression, we set the street segment to be $700m$ long. The rest of the settings are in Table III. We focus on expression Type1, Type3, and Type4 models since Type2 model is a single uniform model, which can be reasonably approximated using UBR or UGC methods. Based on the same target model generation strategy in the event centric experiments, we also use 3 level perturbations defined in Table III. At each perturbation level and for each s-expression model type, we generate 10 target models. Since we have 3 perturbation levels and 3 expression models, in total, we have 90 ($3 \times 3 \times 10$) target and initial s-expression model pairs.

For every target and an initial model pair, we generate all the possible range queries in CRQ (Candidate Range Query set) with the size proportioned to 10% to 100% of the MBR of the target and initial models. We evaluate every queries in CRQ against the target model on 9 query threshold levels (10% to 90%). For each threshold level, we obtain a RRQ (Relevant Range Query set). Similarly, we also evaluate queries on a competing strategies. Note that IR approach is not applicable in the query centric experiments. Therefore, we have 2 sets of RRQ for each

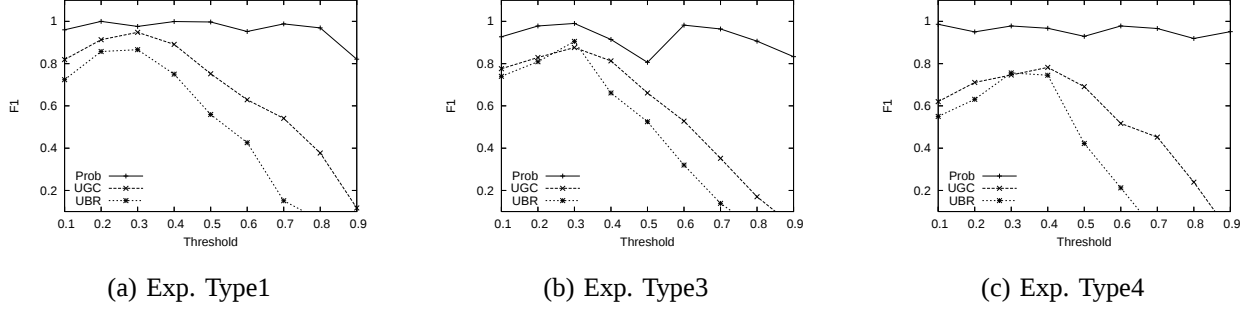


Fig. 24. F1 measure with Level 1 target perturbations

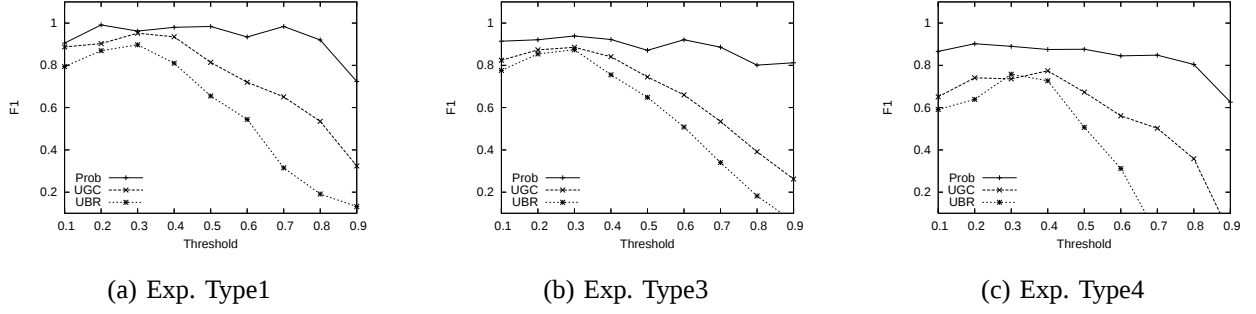


Fig. 25. F1 measure with Level 2 target perturbations

threshold – one represents a true set of queries that will retrieve the spatial expression, and the other one represents an estimated set of queries from the competing modeling strategy. Based on the differences of the two sets, we can compute the precision, recall and F1 measures of the modeling strategy.

B. Query Centric Experiment Results

The set of experiments studies the effects of the target perturbation levels and expression model types to different modeling strategies. Perturbation level corresponds to the distance between target and initial models, and hence corresponds to the modeling difficulties. Expression model types correspond to the model complexity. Obviously, expression Type3 and Type4 is more complex than Type1 since more parameters are involved. Conceptually, for each target and initial model pair, we can show the F1 plots similar to the ones in Figure 24, but it will be 90 of them. Since we generate 10 target models for each initial model, we average the performance of the target models related to the same initial model.

Figure 24, Figure 25 and Figure 26 show the 9 performance plots. Clearly, at the low level of modeling difficulties (Figure 24), the initial probabilistic model (*Prob*) performs well on all

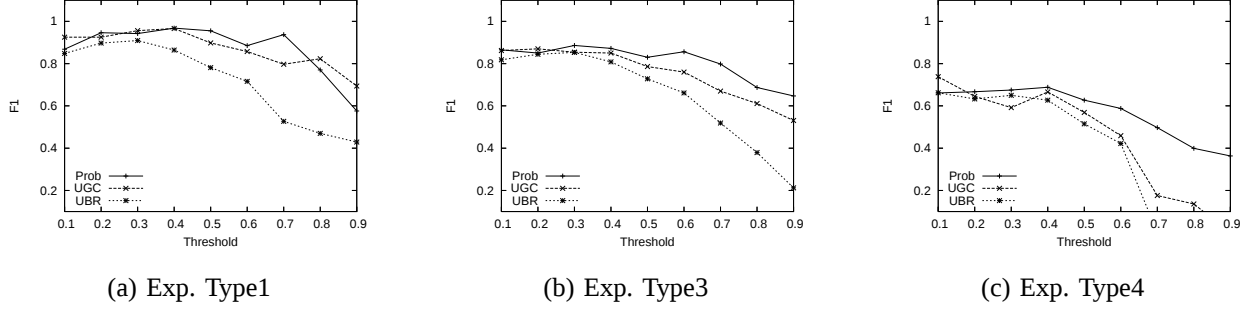


Fig. 26. F1 measure with Level 3 target perturbations

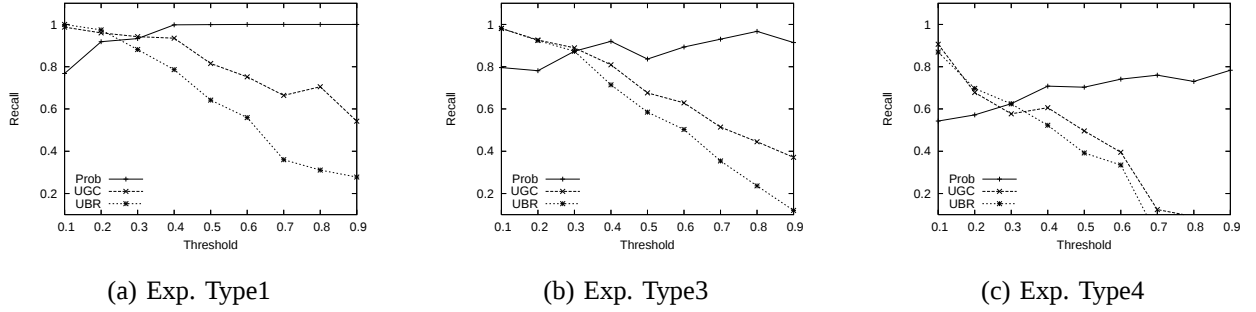


Fig. 27. Recall measure with Level 3 target perturbations

expression models. We also notice that even with the low level of perturbations, the uniform models (UBR, UGC) fail to achieve good results especially at high query threshold level (e.g., 0.7), and the situations getting worse when perturbation increases. In Figure 27, we show the recall measure at target perturbation level 3. We notice that the uniform models failed to recall many relevant queries at reasonable threshold level (e.g., 0.7). This means if uniform modeling methods are used, many queries fail to retrieve the spatial expressions, whereas the spatial expressions should have been retrieved if the target models were used. In terms of real-life SA applications, this leads to the failures of retrieving relevant events. This conclusion is similar to the event centric experiments. We also notice that as the modeling complexity increases (e.g., Type4) and target model is away from the initial model (i.e., level 3), the performance of the initial probabilistic models (*Prob*) degrades, e.g., It performs worse in Figure 26(c) than in Figure 26(a). This also applies to other competing strategies. However, we notice even for the Type4 s-expression and level 3 target perturbations, *Prob* models still have very good recall measure as in Figure 27(c).

Therefore, when we focus on all the reasonable queries on a few fixed and commonly used s-expression models, we can clearly see the modeling advantage of our approach over other

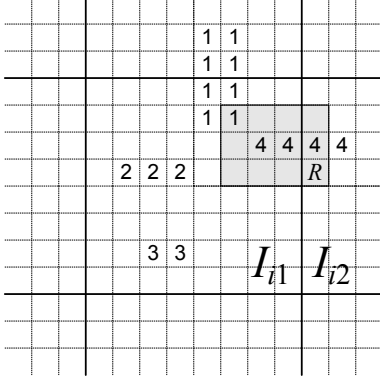
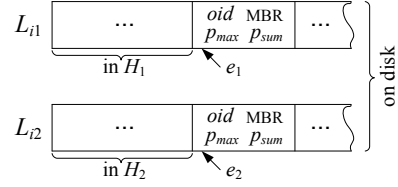
Fig. 28. τ -RQ intersects two dcells.

Fig. 29. List processing.

competing strategies.

XII. PROCESSING A τ -RQ WHEN R INTERSECTS MULTIPLE DCELLS

Let us now consider the situation shown in Figure 28, where R can intersect with more than one dcell. We study only two-cell scenario for illustration purpose; the technique generalizes to any number of dcells.

(1) *The grid-level pruning.* Assume that R intersects with only two dcells: I_{i1} and I_{i2} . We know that the index grid I is constructed such that only the locations in L_{i1} and L_{i2} can satisfy the τ -RQ, the rest of the locations are pruned.

(2) *The dcell-level pruning.* Let $m_1 = \text{vcellsin}(R \cap I_{i1})$, $m_2 = \text{vcellsin}(R \cap I_{i2})$, and $\lambda = \min(m_1 \times I_{i1} \cdot p_{\max}, I_{i1} \cdot p_{\text{sum}}) + \min(m_2 \times I_{i2} \cdot p_{\max}, I_{i2} \cdot p_{\text{sum}})$. Then, if $\lambda < p_\tau$, then no objects in L_{ij} satisfies the τ -RQ, hence all of them are pruned.

(3) *The list-level pruning.* Otherwise, the algorithm scans two lists L_{i1} and L_{i2} at the same time in a sequentially manner, as illustrated in Figure 29. Assume that the elements in those lists are sorted in descending order of p_{sum} . Suppose that the algorithm currently observes elements $e_1 \in L_{i1}$ and $e_2 \in L_{i2}$, which correspond to locations ℓ_1 and ℓ_2 . Let us note that ℓ_1 and ℓ_2 can be the same object. All the previously observed but not-yet-pruned elements from L_{i1} and L_{i2} are kept in two sets $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively.

We need to upper-bound the probability $P(\ell \in R)$ for each location ℓ stored in the rest of the lists L_{i1} and L_{i2} , such that ℓ is not yet marked for post-processing (i.e., $\ell \notin H_1, H_2$). This

probability is upper-bounded by $\lambda = e_1 \cdot p_{sum} + e_2 \cdot p_{sum}$. Thus, if $\lambda < p_\tau$, we do not need to load the rest of the lists from disk, and **all** the locations stored there can be pruned.

(4) *The element-level pruning.* Otherwise, we will not be able to prune *all* the locations, but we shall try to prune ℓ_1 and ℓ_2 *individually*. Let us consider how to prune ℓ_1 , the case for ℓ_2 is similar. First, we compute two values: λ_1 and λ_2 to bound $P(\ell \in R \cap I_{i1})$ and $P(\ell \in R \cap I_{i2})$ respectively, so that $P(\ell \in R) \leq \lambda_1 + \lambda_2$. We compute $\lambda_1 = \min(n_1 \times e_1 \cdot p_{max}, e_1 \cdot p_{sum})$, where $n_1 = vcellsin(R \cap I_{i1} \cap e_1 \cdot MBR)$. The value of λ_2 depends on whether or not we have already observed ℓ_1 on the previous steps. Namely, if there is already an element e_{ℓ_1} in $\mathcal{H}_2$ for ℓ_1 , then $\lambda_2 = \min(n_2 \times e_{\ell_1} \cdot p_{max}, e_{\ell_1} \cdot p_{sum})$, where $n_2 = vcellsin(R \cap I_{i2} \cap e_{\ell_1} \cdot MBR)$. Else $\lambda_2 = e_2 \cdot p_{sum}$. Now we can compute $\lambda = \lambda_1 + \lambda_2$ and if $\lambda < p_\tau$, then we can prune ℓ_1 . In the latter case, ℓ_1 is removed from $\mathcal{H}_2$ and L_{proc} – if it is already there. If the above methods cannot prune ℓ_1 , then ℓ_1 is inserted into L_{proc} and e_1 into $\mathcal{H}_1$.

After ℓ_2 is processed in a similar manner, the algorithm extracts the next two elements e_1 and e_2 from L_{i1} and L_{i2} and repeats the pruning procedure from step 3, until all the elements in the lists are processed. Similar to the one-cell scenario, *L1Sketch* can be used to further reduce the upper bound.

XIII. QUAD-TREE COMPRESSION

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 first concern, where the analyst can specify that any representation of a given pdf should fit into a certain amount of space $s_\tau \in \mathbb{N}$. The quad-tree (lossy) compression algorithm resolves this issue. It operates by maintaining the ‘compressible nodes’ in a priority queue. A node is compressible if all of its children are leaf nodes. In Figure 6(a), the compressible nodes are A, B, C, D . The algorithm extracts from the priority queue the ‘best’ node to compress next $\mathcal{N}$ and performs a *node compression* operation on $\mathcal{N}$. This operation consists of removing all the children of $\mathcal{N}$ and converting $\mathcal{N}$ into a leaf node. Semantically, the effect of this operation corresponds to replacing p_{ij}^ℓ ’s of all the cells, covered by the BR of $\mathcal{N}$, with the same value p , equal to the average of those p_{ij}^ℓ ’s. After each node compression operation, the histogram is replaced with another, more ‘crude’, histogram of the original pdf. In the resulting histograms the new p_{ij}^ℓ ’s still sum up to 1. A node compression can make the parent of $\mathcal{N}$ to become a compressible node, in which case

it is inserted in the priority queue. The algorithm keeps compressing nodes until the quad-tree fits into the specified amount of space s_τ .

The key for the priority queue can be chosen in a number of ways. Recall that for any histogram H there is pdf $f_H(x, y)$ that corresponds to it. Intuitively, we want the shape of $f_H(x, y)$ for the resulting histogram H to resemble/approximate that of the original pdf as closely as possible. The algorithm above tries to achieve that approximation goal in a greedy fashion, by repeatedly collapsing ‘best’ compressible nodes. Thus, we need to decide what constitutes the best node to compress next. Graphically, the effect of compressing a node $\mathcal{N}$ with bounding rectangle B corresponds to ‘flattening’ of the curvature of the histogram for the cells covered by B . Thus, we want to choose those nodes for flattening whose corresponding histogram curvature looks the flattest so that we minimally alter the shape.

Consider a compressible node $\mathcal{N}$ with bounding rectangle B which covers n cells. Suppose the probabilities in those cells are: $p_1, p_2, \dots, p_n$. By compressing this node, the probabilities will become $p_1 = \dots = p_n = \mu$, where $\mu = \frac{1}{n} \sum_{i=1}^n p_i$. So, one solution is to choose $\xi = \max_{i=1, \dots, n} p_i - \min_{i=1, \dots, n} p_i$. Solutions that are even more expensive can use, as the key, measures like the KL distance or squared-error, computed over B between the original pdf and the pdf for the histogram that would result in after compressing $\mathcal{N}$.

Figure 6(a) shows an example where four nodes are compressible: A, B, C, D , where node A covers cells $[0, 1] \times [2, 4]$, node D covers cells $[2, 3] \times [0, 1]$, etc. The cells covered by D clearly have the least variance, so D is picked compression. Figure 6(b) shows the result of compressing D . Notice the flattening effect in the cells covered by D .