Probabilistic k-Nearest Neighbor Monitoring of Moving Gaussians

Kostas Patroumpas
IMIS, Athena Research Center &
National Technical University of Athens, Hellas
kpatro@dblab.ece.ntua.gr

Christos Koutras
Department of Computer Science & Engineering
Hong Kong University of Science & Technology
ckoutras@cse.ust.hk

ABSTRACT

We consider a centralized server that receives streaming updates from numerous moving objects regarding their current whereabouts. However, each object always relays its location cloaked into a broader *uncertainty region* under a Bivariate Gaussian model of varying densities. We wish to monitor a large number of continuous queries, each seeking k objects nearest to its own focal point with likelihood above a given threshold, e.g., "which of my friends are currently the $k = 3$ closest to our preferred café with probability over 75%". Since an exhaustive evaluation would be prohibitive, we develop heuristics based on spatial and probabilistic properties of the uncertainty model, and promptly issue approximate, yet reliable answers with confidence margins. We conducted a comprehensive empirical study to assess the performance and response quality of the proposed methodology, confirming that it can efficiently cope with large numbers of moving Gaussian objects under fluctuating uncertainty conditions, while also offering timely response with tolerable error to multiple queries of varying specifications.

CCS CONCEPTS

•Information systems →Spatial-temporal systems; Location based services; Data streaming;

KEYWORDS

geostreaming, nearest neighbors, Bivariate Gaussians, uncertainty

ACM Reference format:
Kostas Patroumpas and Christos Koutras. 2017. Probabilistic k-Nearest Neighbor Monitoring of Moving Gaussians. In *Proceedings of SSDBM '17, Chicago, IL, USA, June 27-29, 2017,* 12 pages.
DOI: http://dx.doi.org/10.1145/3085504.3085525

1 INTRODUCTION

A wide range of modern applications deal with imprecise or uncertain data (sensor readings, text, locations, etc.), opening new perspectives for their modeling and management. Particularly regarding spatial information, uncertainty can be either locational or existential [2]. Under a *locational* model, an object always exists, but its location is uncertain and described by a probability distribution. This is totally different from an *existential* model, where each object has a precise location but it appears with a given probability.

We strictly focus on locational uncertainty of moving objects (e.g., people, sensors), each with an *uncertainty region* according to a probability distribution. If this distribution is *discrete*, then location is modeled by a *probability mass function* (pmf) as in [11], i.e., by a number of discrete samples each of a probability that the object may be located there. Otherwise, distribution is *continuous* [8, 19], such as uniform or Gaussian, defined by a *probability density function* (pdf) and employing random variables to express the varying probability of an object to be in a given location.

In this work, we consider a monitoring application that accepts streaming updates of such uncertain locations and provides timely response to continuous *k-nearest neighbor* (kNN) *queries*. Of course, kNN search has been studied a lot, mostly in spatial databases [10, 17] and in continuous monitoring over moving objects like [15, 21, 22]. Despite their real-world impact, such state-of-the-art kNN algorithms consider objects with *exact* coordinates. So, they cannot be applied over uncertain locations, which require specialized methods like [8, 11, 18, 23]. We take a different perspective and suggest a novel, *probabilistic* kNN monitoring, assuming that uncertain locations have *Bivariate Gaussian* pdf of varying characteristics. We propose a methodology that searches amongst moving Gaussian objects and returns the k nearest to a given *focal query point* with probability above a query-specified *threshold* θ. For instance, in a geosocial networking application with such a Gaussian model for user locations, a user may wish to get notified for her $k = 3$ friends who are probably closest (say, with probability over 75%) to a given point of interest, e.g., a bar or a sporting venue. Focal points may also be moving, yet they always have exact coordinates. In our setting, a centralized server should efficiently process incoming uncertainty regions and update its response to such continuous kNN queries. But Gaussian distribution can describe uncertainty in many more applications. In robotics, use of Gaussian pdf is common in modeling the location of observed objects. And in environmental monitoring, it can suitably model noise effects or CO_2 emissions from vehicles collected via on-board sensors.

Due to the inherent uncertainty in streaming positional updates relayed by numerous moving objects, a central processor can only provide *approximate* kNN results. Issuing exact answers would incur prohibitive cost, since multiple queries of different parameters need continuous re-evaluation against frequently changing uncertain locations; so eventually performance would slump. Besides, an approximation algorithm is more realistic, given the probabilistic nature of the underlying spatiotemporal data that cannot possibly eliminate quality flaws in the expensively paid "exact" answers.

Table 1: Notations

μ	Mean of Bivariate Gaussian pdf of object o, i.e., its coordinates (μ_x, μ_y)
σ	Standard deviation of Bivariate Gaussian pdf of object o per dimension x, y
Σ	Discrete uncertainty levels $\{\sigma_0, \sigma_1, \ldots, \sigma_{n-1}\}$ for locations
r_o	Truncated uncertainty region of object o at radius 3σ around its mean μ
V_σ	Discretized probabilistic verifier for uncertainty level $\sigma \in \Sigma$
δ	Side length of each square-shaped elementary box ε in every verifier
λ	Granularity (in elementary boxes per dimension) of a verifier
τ	Timestamp value in uncertain locations or query specifications
q	Focal point of interest to a $k\theta$NN query
k	Number of nearest neighbors to search for a given focal point
θ	Cutoff threshold for qualifying $k\theta$NN objects
Q	Result set of qualifying $k\theta$NN objects to query q
P_c	Circular coverage probability around focal point q
$\hat{P}_b$	Box coverage probability estimated from elementary boxes around q
d_k^θ	Largest (k^{th}) cutoff distance at threshold θ amongst objects in result Q
ξ	Least count of elementary boxes required to attain probability $\geqslant \theta$
G	Regular grid partitioning of the monitored area into square equi-sized cells
Θ	A set $\{\theta_1, \theta_2, \ldots, \theta_m\}$ of m typical threshold values
B	Lookup of precomputed ξ-bounds at various distances and thresholds in Θ
d_k^b	Most extreme box distance from q to the k^{th} object in result Q
$\mathcal{A}$	Auxiliary list of *guard* objects for query q, one per uncertainty level $\sigma \in \Sigma$

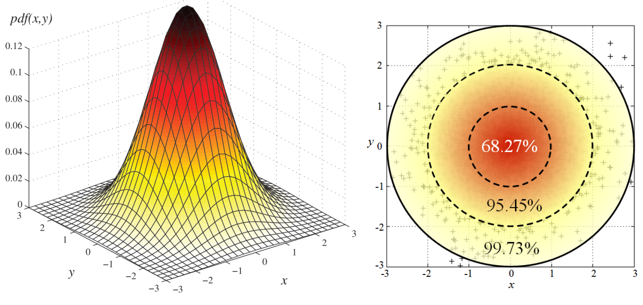
The Gaussian uncertainty of each object may be of varying density, essentially cloaking its position in a broader area and thus allowing varying *levels of uncertainty*. In general, the larger the spread of such a region, the more uncertain the location. In order to get the most probable kNN results to a given query, only those that qualify above a prespecified *threshold* θ are returned (e.g., $\theta = 0.75$). To offer answers of good quality in near real-time, we suggest a discretization scheme for uncertainty regions and employ several novel heuristics that can effectively prune the search space and choose candidates most likely to qualify for the final response.

To the best of our knowledge, this is the first work on continuous probabilistic kNN search over moving objects modeled specifically with a Bivariate Gaussian distribution. In particular, our contributions can be summarized as follows:

- We model object locations as a stream of moving Bivariate Gaussians with dynamically updated densities (Section 2).
- We introduce a specification for probabilistic k-nearest neighbor search over uncertain objects (called $k\theta$NN queries) employing thresholds to validate candidates (Section 3).
- We develop an approximation strategy for online $k\theta$NN monitoring, introducing pruning heuristics to effectively avoid examination of non-qualifying objects (Section 4).
- We empirically demonstrate that this methodology can provide timely results to multiple continuous $k\theta$NN queries against numerous objects of varying levels of uncertainty with tolerable concession to quality of results (Section 5).

2 UNCERTAINTY MODEL

We consider a large number N of uncertain objects moving on the 2-D Euclidean plane and communicating with a server. Messages relayed to the server either notify about updates in the cloaked positions of objects or modify specifications of spatial queries. All messages (object and query updates alike) are timestamped according to a global clock at distinct instants τ (e.g., every few seconds or minutes). Next, we formalize this application scenario in detail. Table 1 summarizes the notations used throughout the paper.



(a) Probability density (b) Truncated uncertainty region r_o

Figure 1: Standard Bivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{1})$

2.1 Object Locations as Bivariate Gaussians

Let $O = \{o_1, o_2, \ldots, o_N\}$ a set of N uniquely identified uncertain objects. For each monitored object $o \in O$, the server maintains an *uncertainty region* modeled by Bivariate Gaussian (a.k.a. Normal) random variables X, Y with mean $\mu = (\mu_x, \mu_y)$ and standard deviation (σ_x, σ_y) over axes x, y. Assuming that objects are moving freely, X and Y are independent, hence uncorrelated. Also, location coordinates may spread similarly along each axis, so $\sigma_x = \sigma_y = \sigma$. This results into a simplifed *joint probability density function*:

$$pdf(x, y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{(x-\mu_x)^2 + (y-\mu_y)^2}{2\sigma^2}} \qquad (1)$$

The resulting distribution is the well-known "bell-shaped" surface illustrated in Fig. 1a, signifying that density diminishes rapidly at increasing distances from its origin (mean μ). From an application point of view, μ should never coincide with the exact geographical location of any object (e.g., to protect privacy in geosocial networks). As depicted in the scatterplot in Fig. 1b for the *standard distribution* $\mathcal{N}(\mathbf{0}, \mathbf{1})$ having its mean at $\mu = (0, 0)$ and standard deviation $\sigma = 1$, a circle of radius $d = 3\sigma$ around the mean always contains almost 99.73% of the *cumulative probability distribution* (cdf). In practice, the probability that o can be found outside this circle is negligible.

Definition 2.1 (Bivariate Gaussian Object). An uncertain object $o(\mu, \sigma) \in O$ is modeled by a Bivariate Gaussian pdf with mean $\mu = (\mu_x, \mu_y)$ and standard deviation σ in each dimension. Its *truncated uncertainty region* r_o is derived by a circle of radius 3σ around μ and asymptotically contains all its cdf (by almost 99.73%).

How locations get cloaked is orthogonal to kNN search and actually depends on the application; e.g., a location anonymizer may be used as in [14] to protect user privacy or Gaussian noise models may be employed in environmental monitoring. We only prescribe that each moving object o must relay its current (μ, σ), both expressed in distance units (e.g., meters). Larger σ values indicate that an object's actual location can be hidden in a greater area around its mean μ. As object o is on the move, it relays updates, so the server must accept a *stream* of tuples $\langle o, \mu, \sigma, \tau \rangle$ from uncertain objects in O, always ordered by their timestamp values τ.

An object can dynamically adjust its degree of uncertainty, by setting its σ to a value taken from a set of n fixed *uncertainty levels*

$\Sigma = \{\sigma_0, \sigma_1, \ldots, \sigma_{n-1}\}$. In effect, σ controls the spread of the distribution: with larger σ values, the spread is wider, also lowering the peak of the bell (Fig. 1a). Instead, with smaller σ values, the spread is more constrained (i.e., the bell becomes narrower, but taller). Note that objects may report their updates in a non-synchronized fashion, e.g., upon significant changes in their whereabouts that invalidate their uncertainty region currently known to the server. Hence, at a given timestamp value τ, the server may refresh the truncated uncertainty regions for a varying number of objects in O. The server is never aware of the exact (x, y) coordinates of a given object o, but it can be certain by 99.73% that o is somewhere within its truncated uncertainty region r_o until further notice.

2.2 kNN Search over Bivariate Gaussians

In the server, a varying number M of continuous nearest neighbor queries may be registered at any given time τ. Each such query specifies its own (integer) number k of objects to be sought as currently closest to its actual *focal point* q. Note that the position of focal point q is always defined by its *exact* coordinates (q_x, q_y).

Query evaluation takes place periodically at *execution cycles*. Without loss of generality, each cycle starts at successive timestamp values τ, once the corresponding batch of updates from objects and queries have been received by the server. Since the specifications of both objects and queries may change at each execution cycle, the validity of previously emitted results is generally not preserved, so computation on the server must be repeated from scratch.

Suppose a query q that continuously searches for its kNNs among objects in O, as illustrated in Fig. 2. Also assume that there exists a measure $P(q, o)$, which for any given object $o \in O$ returns its likelihood of being one of the kNNs to q. Objects may specify differing $\sigma \in \Sigma$, hence their uncertainty regions will be varying in size (as depicted with the magnitude of the circles), and also of varying density spread around their mean (note the degraded shading in each circle). For each such query q over the current setting of objects in O, a subset of k objects is returned as $1^{st}, 2^{nd}, \ldots, k^{th}$ NN ranked by descending probability $P(q, o_i), i = 1, 2, \ldots, k$. Formally:

Definition 2.2 (kNN Query Over Gaussians). Let a kNN query at focal point q against a set O of Bivariate Gaussian objects. Given $P(q, o)$ as a measure of influence $\forall o \in O$ on q, the result set $Q = kNN(q, O, \tau)$ of qualifying objects at timestamp τ is:

$$(Q \subseteq O) \land (|Q| = k) \land (\forall a \in Q, \forall o \in O \setminus Q, P(q, a) > P(q, o)). \quad (2)$$

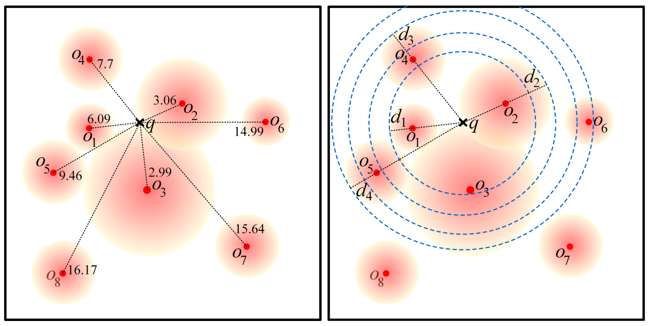
But one tough issue remains: how to quantify probability $P(q, o)$ of object o qualifying for query q? One possibility is *Mahalanobis distance*, used in statistics to measure the distance of point q from a distribution [3]. In our setting, this can be defined as follows:

Definition 2.3 (Mahalanobis Distance). The *Mahalanobis* distance of a Bivariate Gaussian $o(\mu, \sigma) \in O$ from focal point q is

$$Mahalanobis(q, o) = \sqrt{\frac{(q_x - \mu_x)^2}{\sigma_x^2} + \frac{(q_y - \mu_y)^2}{\sigma_y^2}}. \quad (3)$$

But given that uncertainty is modeled with mean $\mu = (\mu_x, \mu_y)$ and standard deviation $\sigma = \sigma_x = \sigma_y$, Eq. (3) can be simplified to:

$$Mahalanobis(q, o) = \frac{\sqrt{(q_x - \mu_x)^2 + (q_y - \mu_y)^2}}{\sigma} = \frac{L_2(q, \mu)}{\sigma} \quad (4)$$



(a) Mahalanobis distances (b) Maximal distances

Figure 2: Distance measures of Gaussians from focal point q

where L_2 denotes the Euclidean distance of q from mean μ of the distribution. Naturally, this value reflects the likelihood of o to be among the kNNs in terms of the proximity of its mean to q and inversely to its uncertainty level. Figure 2a depicts Mahalanobis distances of various Bivariate Gaussian objects from a focal point q computed according to Eq. (4). It can be easily observed that Mahalanobis distance favors objects close enough to q, but which are more spread, i.e., with higher variance and thus a greater degree of uncertainty. Note that objects o_1 and o_2 have their means at equal Euclidean distance from q, but o_2 has almost half the Mahalanobis distance compared to o_1. In that sense, it is like q being assigned as a "sample" to an uncertain object (viewed as a "cluster").

Yet, we can take a different approach. If we draw circles at increasing radii d that respectively enclose the truncated uncertainty regions of objects (Fig. 2b), we can easily infer that o_1 overall has a higher influence with respect to the rest, and thus may be ranked as the 1^{st} NN. In practice, such radii correspond to the *maximal influence* of every truncated uncertainty region from q. We define:

Definition 2.4 (Maximal Distance). For a Bivariate Gaussian object $o(\mu, \sigma) \in O$, the *maximal* distance of its truncated uncertainty region r_o from focal point q is

$$MAXDIST(q, o) = L_2(q, \mu) + 3\sigma. \quad (5)$$

Intuitively, this value quantifies the greatest possible Euclidean distance of any location within region r_o from q, as depicted with the straight lines in Fig. 2b. Based on such *MAXDIST* values, the *less uncertain* (i.e., a more restricted region r_o) and *closer* to q an object is, the *more probable* to be amongst its kNNs. Under this interpretation, Mahalanobis in Eq. (4) would not be an adequate measure. Indeed, in our case, the problem is neither to assign focal point q to its closest cluster nor to classify it. Instead, it is the other way round; we wish to identify which k objects are more likely to be relevant to q, so *MAXDIST* seems preferrable. Still, such distances actually represent the *maximal influence* of respective objects and practically take into account all their uncertainty.

Interestingly, if *all* Bivariate Gaussian objects in O have identical σ values, then standard kNN monitoring techniques like [15, 21, 22] can be employed by only considering Euclidean distances of their means from q and ignoring uncertainty altogether. Either via Mahalanobis or *MAXDIST* distances, it is trivial to prove this:

Lemma 2.5. *Suppose that* $\forall\, o_1(\mu_1, \sigma_1), o_2(\mu_2, \sigma_2) \in O, o_1 \neq o_2$, *it holds that* $\sigma_1 = \sigma_2 = \sigma$. *If* $L_2(q, \mu_1) < L_2(q, \mu_2)$, *then* o_1 dominates o_2 *and it is closer to focal point* q.

Unfortunately, for objects of *varying* uncertainty level (as most often occurs in our setting), no standard kNN methods can be applied as they only consider exact point locations.

3 SPECIFICATIONS FOR CONTINUOUS PROBABILISTIC kNN SEARCH

As we opt for obtaining approximate results in kNN monitoring over moving Bivariate Gaussians, we should *probabilistically* quantify their influence with respect to a known focal point q. The problem is that Gaussian distributions cannot be integrated analytically, so we would need to resort to costly numerical methods like Monte-Carlo simulations [19] to get a fair estimation of the kNN results. In this Section, we first investigate two such baseline approaches, showcasing their inherent flaws. Towards a more generic and configurable specification, we then introduce $k\theta$NN queries, i.e., continuous probabilistic kNN search with a cutoff threshold θ.

3.1 kNNs by Estimating Circular Coverages

A naïve solution to identify kNNs probabilistically may involve iterative searching at gradually increasing distances around focal point q. What we need to examine is probability $P_c(q, d, r_o)$ for a candidate object o to be contained within a circle $C(q, d)$ of varying radius d, but always centered at q. As illustrated in Fig. 3a, this is equivalent to slicing the truncated uncertainty region r_o by circle C and assessing the portion of the distribution (cdf) within C. In fact, this is the result of the *circular coverage function*, as it is known in statistics [9]. In our setting, this is expressed as follows:

Definition 3.1 (Circular Coverage Probability). For a Bivariate Gaussian object $o(\mu, \sigma) \in O$, its *circular coverage probability* at a given distance d from focal point $q(q_x, q_y)$ is

$$P_c(q, d, r_o) = P_c(q, d, \mu, \sigma) =$$
$$= \frac{1}{2\pi\sigma^2} \int_{q_x-d}^{q_x+d} \int_{q_y-\sqrt{d^2-(x-q_x)^2}}^{q_y+\sqrt{d^2-(x-q_x)^2}} e^{-\frac{(x-\mu_x)^2+(y-\mu_y)^2}{2\sigma^2}} dydx. \quad (6)$$

So, a baseline algorithm would start examination at a very small radius $d = d_0$ (even with $d = 0$), and would estimate with a Monte-Carlo simulation the cumulative probability of each object $o \in O$ according to Eq. (6). This algorithm would progressively increase search radius d by a small (ideally infinitesimal) amount dr and repeat calculations until k objects are found and all have their truncated uncertainty regions within this circle by more than 99.73% circular coverage probability, i.e., with negligible error. Ranking of kNNs may be based on their estimated coverage probabilities.

The major problem with this naïve approach is that Monte-Carlo simulation generally incurs excessive CPU cost as it requires a sufficiently large number (at the order of 10^6 [19]) of samples per object. Even worse, such simulations must be performed repeatedly at multiple increasing radii for Eq. (6) and generally may concern many candidate objects per query. Given the mobility of objects and the mutability of queries over time, such a solution is utterly prohibitive for processing multiple kNN requests in online fashion over a large number of moving Gaussian objects.

3.2 kNN Search by Maximal Distance

The aforementioned technique could be substantially improved if all objects were represented by their truncated uncertainty regions, i.e., like circles as in Fig. 1b. The initial search radius d_0 should be the smallest *MAXDIST* among uncertainty regions from q. For a given q, this variant must calculate *MAXDIST* values for all candidates according to Eq. (5) and incrementally search only at those selected radii in ascending order. The method terminates after checking objects against circles for exactly k successive *MAXDIST* radii.

The crux of this *incremental radial search* method is that costly Monte-Carlo simulation is no longer necessary. Indeed, the first radius $d_1 = MAXDIST(q, o_1)$ covers the truncated uncertainty region of one object (ties between such distance values may be resolved arbitrarily). As illustrated in Fig. 2b, object o_1 has more than 99.73% chance to be within the inner circle of radius d_1, hence it immediately qualifies as 1^{st} NN. Then, searching expands, designating o_2 as 2^{nd} NN, o_4 as 3^{rd} NN, and so on. By construction, d_i is the i^{th} *MAXDIST* value in ascending order, and it designates a circular area around q that covers the truncated uncertainty region of one more object with respect to $d_i \geqslant d_{i-1}, i = 2, \ldots k$. Of course, this method requires many distance computations per query q and sorting them in ascending order. Taking the kNN results is trivial:

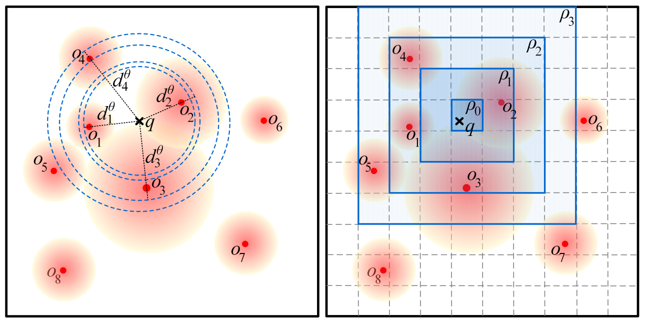
Lemma 3.2. *The order of* $MAXDIST$ *values from focal point* q *coincides with the ranking of the respective objects as kNNs to* q.

Yet, there is a subtle assumption behind this method. It considers that a given candidate object o may be one of the kNNs by examining practically *all* its uncertainty region r_o, i.e., based on its *maximal influence.* In effect, no probabilty estimations are involved in computing those *MAXDIST* values. From another point of view, the returned results may be sometimes considered unfair. Indeed, in the setting of Fig. 2b, the uncertainty region of object o_3 almost touches focal point q and its mean (centroid) is the third closest to q, much closer than the mean of objects o_4 and o_5. Still, o_3 does not qualify among the $k = 4$ NNs according to this incremental radial search. In contrast, results would differ if we considered only portions of uncertainty regions (Fig. 3a), e.g., that an object has at least $\theta = 75\%$ probability to be one of the kNNs. In that case, object o_3 would be ranked higher than o_4 and o_5, and thus qualify as the 3^{rd} NN. Intuitively, such an approach would pick objects that are not only close to the query point, but they are less uncertain than others and also qualify as kNNs with probability above a given threshold θ. According to this rationale, we next introduce continuous probabilistic kNN queries that employ a threshold when choosing objects as nearest neighbors and exclude from further consideration any candidates with insufficient probability.

3.3 kNN Search by Cutoff Distance

In the sequel, we assume that every kNN query also prescribes a real-valued *threshold* θ, with $0.5 \leqslant \theta < 1$, essentially expressing the lowest tolerable probability of an object to be ranked among the kNNs to focal point q. We dictate that θ can be no less than 50%, so that it is more probable than not for each qualifying object to be one of the kNNs to a given query.

In this paper, such requests are called *continuous probabilistic k-nearest neighbor queries* ($k\theta$NN), and return k objects as results.



(a) Cutoff distances at θ=0.75 (b) Search in successive frames of grid G

Figure 3: Probabilistic $k\theta$NN search over Bivariate Gaussians

During the lifetime of a $k\theta$NN query, its focal point q may be moving and its threshold θ may arbitrarily change. Thus, along with streaming updates from moving Gaussian objects, the server should also accept updated query specifications as $\langle q, k, \theta, \tau \rangle$. At each execution cycle τ, the server must issue the k most probable NN results for each $k\theta$NN query, and at descending probability above its threshold θ. Before giving a formal definition of such queries, we introduce the following notion:

Definition 3.3 (Cutoff Distance). For a Bivariate Gaussian object $o(\mu, \sigma) \in O$ of truncated uncertainty region r_o, its *cutoff distance* d_o^θ from focal point q w.r.t. a known threshold θ is the smallest radius of a circle $C(q, d_o^\theta)$ at which object o obtains a *circular coverage probability* $P_c(q, d_o^\theta, r_o) \geqslant \theta$.

Let object a be the i^{th} NN to q for a given θ. Its cutoff distance indicates the Euclidean distance from q at which object a attains enough probability ($\geqslant \theta$) to be among the $k\theta$NNs. So, picking objects with the top-k cutoff distances from q can provide a ranking amongst qualifying $k\theta$NNs. Figure 3a depicts the four circles and the corresponding cutoff distances based on circular coverage probabilities at $\theta = 0.75$ for objects o_1, o_2, o_3, o_4. Assuming that such a cutoff distance d_i^θ is known for every i^{th} NN, $i = 1, \ldots k$, we know from Eq. (6) that all $k\theta$NN objects are covered by a probability at least θ within the largest such radius $d_k^\theta = \max\{d_i^\theta\}$, which obviously corresponds to the k^{th} NN. For instance, object o_4 determines the largest cutoff distance d_4^θ amongst $k\theta$NNs; so, the outer circle in Fig. 3a covers all $k = 4$ θNNs by at least $\theta = 0.75$. Formally:

Definition 3.4 ($k\theta$NN Query over Gaussians). Let a query at focal point q with threshold θ for selecting its k-nearest neighbors from a set O of Bivariate Gaussian objects at timestamp τ. If the k^{th} smallest *cutoff distance* d_k^θ from q amongst objects in O is known, result set $Q = k\theta NN(q, O, \tau)$ of qualifying objects is:

$$(Q \subseteq O) \ \wedge \ (|Q| = k) \ \wedge (\forall a \in Q, P_c(q, d_k^\theta, r_a) \geqslant \theta) \ \wedge$$
$$\wedge \ (\forall a \in Q, \forall o \in O \setminus Q, P_c(q, d_k^\theta, r_a) > P_c(q, d_k^\theta, r_o)) \quad (7)$$

Ties in the ranking of $k\theta NN$ results may be resolved arbitrarily. In case of multiple objects equally qualifying as the k^{th} NN, i.e., all having exactly the same circular coverage probability, then one may be randomly chosen for inclusion in response Q. Alternatively, such equivalent objects may be all appended to answer Q as k^{th} NNs, relaxing the requirement that $|Q| = k$ in rule (7).

As the example in Fig. 3a illustrates, cutoff distance d_k^θ is a more aggressive measure, offering tighter circular coverages for any given θ, in contrast to the conservative *MAXDIST* values (Fig. 2b). It can be easily verified that with $\theta = 99.73\%$, $k\theta$NN results obtained from (7) are identical to kNNs issued by the incremental radial search that examines the entirety of truncated uncertainty regions (Section 3.2). Overall, the rules in (7) provide a generic query specification dynamically configurable by a user-defined θ. Unfortunately, it is hard to estimate cutoff distances for arbitrary thresholds and uncertainty characteristics. According to (7), the server may need to estimate circular coverage probabilities against potentially many objects in order to provide a valid response and each such estimation requires another costly Monte-Carlo simulation.

4 APPROXIMATION STRATEGY

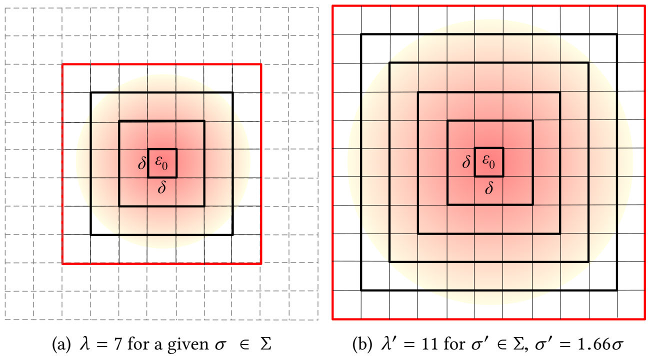
As numerical estimation of $k\theta$NNs is not affordable in real time, next we propose a relaxed specification and a method that yields approximate results by eagerly pruning non-qualifying objects.

4.1 Discretized Verifiers over Gaussians

To overcome the burden of estimating cutoff distances, we propose a *discretization scheme* for uncertainty regions that can assist in quick verification of their probability to qualify as $k\theta$NNs. In particular, we create one discretized *verifier* V_σ for each uncertainty level $\sigma \in \Sigma$; this yields n verifiers in total. As illustrated in Fig. 4, we specify the *elementary box* ε of each V_σ as a square of fixed side δ units, such that $3\sigma = \nu \cdot \delta$, $\nu \in \mathbb{N}^*$. The first such box ε_0 of any V_σ is placed with its center coinciding with the mean location μ of the respective Gaussian pdf. Then, we arrange a set of 8 extra elementary boxes like a square-shaped *frame* around ε_0. This process continues recursively enclosing the previous frames, as depicted with the thick squares in Fig. 4. It can be easily observed that the i^{th} successive frame around central box ε_0 appends $8i$ extra elementary boxes covering more of the uncertainty region.

Discretization is over once we get all the boxes required to cover the truncated uncertainty region r_o. Only them contain meaningful probabilities (the portion of cdf within each box); we can dispense with the rest as they attain negligible probability (Fig. 4). We represent each V_σ as a square matrix of $\lambda \times \lambda$ items (i.e., cdf values) corresponding to equi-sized boxes symmetrically around the mean. The total area (actually a square shown in red in Fig. 4) covered by $\lambda \times \lambda$ elementary boxes fully contains the respective truncated uncertainty region r_o. Thus, the total cumulative probability in each verifier is greater than 99.73%, exceeding the cdf contained in that r_o, and it is asymptotically tending to 100%. Of course, due to the differing magnitude of each distinct $\sigma \in \Sigma$, a diverse number of boxes are included in its respective verifier V_σ. Formally:

Definition 4.1 (Discretized Verifier). Given an elementary box of side length δ, verifier V_σ for uncertainty level $\sigma \in \Sigma$ is a matrix of $\lambda \times \lambda$ such boxes arranged symmetrically around the mean of the respective pdf. By construction, we pick δ such that $3\sigma = \nu \cdot \delta$, $\nu \in \mathbb{N}^*$, so granularity of V_σ is $\lambda = \lceil \frac{6\sigma}{\delta} \rceil + 1$. *Weight* $V_\sigma(\varepsilon)$ is the fraction of the cdf contained within a box $\varepsilon \in V_\sigma$.



(a) $\lambda = 7$ for a given $\sigma \in \Sigma$ (b) $\lambda' = 11$ for $\sigma' \in \Sigma$, $\sigma' = 1.66\sigma$

Figure 4: Verifiers of fixed box side length δ for any $\sigma \in \Sigma$

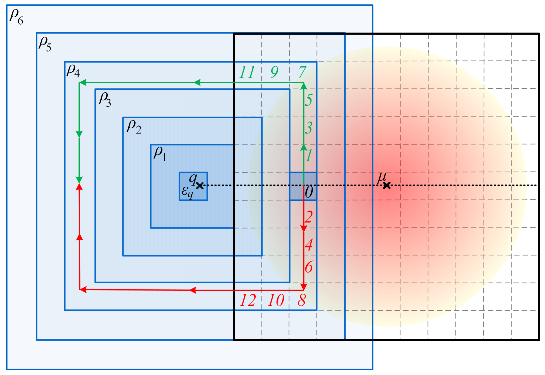


Figure 5: Symmetrical frame-based traversal of a verifier

We enumerate boxes with a pair of (positive or negative) integers (i, j) by their row and column in the matrix. Since λ is always an odd integer due to the symmetry of frames, central box ε_0 is always at position $(0, 0)$ in every matrix V_σ. Although the spatial area of any box ε is always δ^2, its weight $V_\sigma(\varepsilon)$ is not, because each box contains a varying portion of the cdf w.r.t. to its placement in the uncertainty region. A box near the mean contains much more probability than a more distant box, hence it weighs more in the verifier, as shown in Fig. 6 for $\lambda = 7$. Since each uncertainty level $\sigma \in \Sigma$ is known in advance, the probability contained in the boxes of its respective verifier V_σ can be estimated in a preprocessing step using Monte-Carlo simulation and then stored in a lookup matrix.

As each elementary box has fixed resolution δ in all verifiers no matter their uncertainty level, it can be trivially proven that:

LEMMA 4.2. *Let V a verifier of granularity λ pertinent to uncertainty level σ. Then, verifier V' at level $\sigma' = m \cdot \sigma, m \geqslant 1, m \in \mathbb{R}$ has granularity $\lceil m \cdot (\lambda - 1) \rceil + 1$ with equal box side length δ. For their weigths at any elementary box (i, j), it holds that $V'(i, j) \leqslant V(i, j)$.*

For instance, the uncertainty region of object o' in Fig. 4b has $\sigma' = 1.66\sigma$ compared to object o in Fig. 4a. So, if verifier V for o has granularity $\lambda = 7$, then verifier V' for o' will have granularity $\lambda' = \lceil 1.66 \cdot (7 - 1) \rceil + 1 = 11$ in either dimension.

This notion of probabilistic discretization differs from the one introduced in [16] for range monitoring over Gaussians. In that case, all verifiers circumscribed the truncated uncertainty regions and had a fixed granularity λ irrespective of the distribution density; so the weight in any position in the matrix was the same across verifiers for any σ values. As illustrated in Fig. 4, now λ varies in proportion to each σ (by definition); hence, boxes in diverse verifiers at the same distance from their central box ε_0 contain different probabilities in order to suitably distinguish candidate $k\theta NN$s with a brand new traversal strategy, as detailed next.

4.2 Symmetrical Frame Traversal of Verifiers

In order to probabilistically quantify proximity of a given object $o(\mu, \sigma) \in O$ to focal point q, what matters is not the absolute coordinates of its actual mean μ, but only (i) its distance $L_2(q, \mu)$ relative to q and (ii) its uncertainty level σ, as illustrated in Fig. 3a. Thanks to the symmetry in any Bivariate Gaussian pdf with $\sigma_x = \sigma_y = \sigma$ around its mean, we can apply the following transformation: for a candidate object $o(\mu, \sigma)$, its verifier V_σ is symmetrically placed at distance $L_2(q, \mu)$ over a horizontal axis originating from q. Then, we can approximately compare candidates on the basis of their verifiers across this single *verification axis* (the horizontal dotted line in Fig. 5). By probing verifier V_σ for a candidate o, we want to approximately estimate the probability contained in a subset S of its elementary boxes that suffices to exceed θ. More concretely:

Definition 4.3 (Box Coverage Probability). For a Bivariate Gaussian $o(\mu, \sigma) \in O$, its estimated *box coverage probability* $\hat{P}_b(q, d, V_\sigma)$ is the sum of weights from the least number ξ of elementary boxes in its verifier V_σ such that $\hat{P}_b(q, d, V_\sigma) \geqslant \theta$, once V_σ is centered at distance $d = L_2(q, \mu)$ from focal point q along the verification axis.

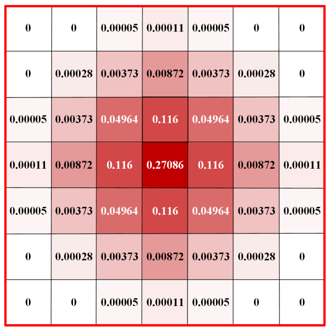
In identifying which elementary boxes to visit in V_σ, we should naturally start probing from box ε_q that contains focal point q. So:

(i) if $d = L_2(q, \mu) < 3\sigma + \frac{\delta}{2}$, then q falls inside some box ε_q of verifier V_σ, thus q is *internal* to V_σ (Fig. 7);
(ii) if $d = L_2(q, \mu) \geqslant 3\sigma + \frac{\delta}{2}$, then q does not fall within any elementary box of verifier V_σ, so q is *external* to V_σ (Fig. 5).

In either case, probing verifier V_σ should provide a collection S of elementary boxes, organized as a series of successive *frames* outwards from box ε_q, such that their cumulative probability is $\hat{P}_b(q, d, S) \geqslant \theta$. Note that q may not necessarily be at the center of box ε_q, yet always along the verification axis; as we opt for an approximate computation via discretization, the exact position of q in a box is ignored. As shown in Fig. 5, the respective box ε_q alone constitutes the first frame ρ_0 to probe in verifier V_σ, and initializes the estimated box coverage probability to $\hat{P}_b = V_\sigma(\varepsilon_q)$.

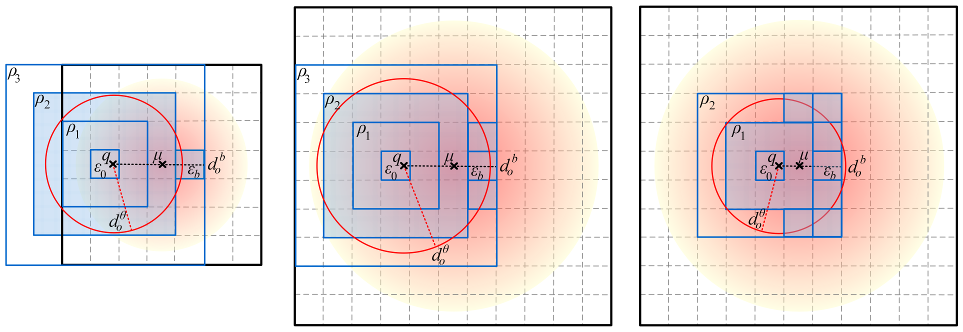
Each successive frame ρ_i outwards around ε_q in verifier V_σ accumulates more boxes in S, as depicted with the blue square-shaped strips in Fig. 5. When a box ε is visited, its weight $V_\sigma(\varepsilon)$ is added to box coverage probability $\hat{P}_b$. Visiting boxes progressively across each next frame ρ_i continues until we attain $\hat{P}_b \geqslant \theta$. Intuitively, picking boxes by rotating in spiroidal fashion over successive frames aims to provide a fair estimation of probability, as a discretized analogue of cutoff distance in circular coverage probabilities.

Instead of probing a given frame ρ_i (in a clockwise or counterclockwise fashion) starting from an arbitrary box, we propose a *symmetrical frame-based traversal* of boxes above and below the verification axis, which can yield more comparable estimates.



0	0	0.00005	0.00011	0.00005	0	0
0	0.00028	0.00373	0.00872	0.00373	0.00028	0
0.00005	0.00373	0.04964	0.116	0.04964	0.00373	0.00005
0.00011	0.00872	0.116	0.27086	0.116	0.00872	0.00011
0.00005	0.00373	0.04964	0.116	0.04964	0.00373	0.00005
0	0.00028	0.00373	0.00872	0.00373	0.00028	0
0	0	0.00005	0.00011	0.00005	0	0

Figure 6: Weights in a verifier of 7×7 elementary boxes



(a) $L_2(q, \mu) = 1.75\delta$, λ=7, ξ=26 (b) $L_2(q, \mu) = 1.75\delta$, $\lambda = 11$, $\xi = 28$ (c) $L_2(q, \mu) = 0.75\delta$, $\lambda = 11$, $\xi = 16$

Figure 7: Probing verifiers by $\theta = 0.8$ for several placements of q and diverse uncertainty levels

More specifically, at a given frame ρ_i we first probe its box closest to μ across the verification axis (marked 0 in Fig. 5), and we append it to S. This bias on boxes towards the mean of the distribution intends to eagerly attain probability θ as early as possible. Next, we continue by *alternating* selection of boxes from above and below the verification axis. Boxes of ρ_i above the axis are traversed in a *counterclockwise* fashion, while those below the axis are visited *clockwise*, as shown by the numberings and arrows in Fig. 5. This offers the advantage that we first pick boxes with greater probability, and hence achieve more fairness in comparing objects.

It may occur that frame ρ_i exceeds verifier V_σ, like ρ_3 in Fig. 7a or it is even completely outside of V_σ as several frames in Fig. 5. Thanks to the symmetry in the discretization scheme, we can skip traversal of such boxes, although they will be added as *virtual* boxes in collection S, since they contain negligible probability.

For sparing frame traversals when q is external to V_σ, we exploit the fact that always $\theta \geqslant 0.5$. In a preprocessing step, we compute the total probability p contained in all columns of V_σ on the left of its central box ε_0, and we lookup this $p < 50\%$. At runtime, if q is external to V_σ, we initialize $\hat{P}_b = p$ and directly start probing V_σ from its frame ρ_i that includes its central elementary box ε_0.

Each time a box ε is added to collection S, we check whether accumulated probability $\hat{P}_b \geqslant \theta$. If not, there is room for more boxes to add in S, and we continue with the next box in that frame or with the successive frame outwards from the starting box ε_q. Once threshold θ is exceeded, we stop picking any more boxes and their count ξ is finalized. Note that only a subset of boxes in the last traversed frame ρ_i may need be probed, since condition $\hat{P}_b \geqslant \theta$ may become true before exhausting all boxes in ρ_i, as illustrated with the blue-shaded boxes in Fig. 7. Clearly, traversal is biased towards boxes of higher density, which naturally outweigh others more distant from μ. The distance from focal point q of the extreme side of box $\varepsilon_b \in S$ that is the remotest along the verification axis w.r.t. q is called *extreme box distance* d_o^b for candidate object o.

At the end of such a traversal, the *box count* of collection S is

$$\xi = \min(\{|S| : \theta \leqslant \sum_{\varepsilon \in S} V_\sigma(\varepsilon)\}), \tag{8}$$

and expresses the least number of elementary boxes from verifier V_σ required to attain a box coverage probability $\hat{P}_b \geqslant \theta$. This approach is based on the reasonable assumption that we prescribe a limited number n of discrete *uncertainty levels* $\Sigma = \{\sigma_0, \sigma_1, \ldots, \sigma_{n-1}\}$ for all objects, hence we can precompute their verifiers.

Given that all boxes are equi-sized but have varying weights according to uncertainty levels, the less the box count ξ corresponding to threshold θ for a given candidate, the less uncertain and closer to q this candidate can be. Indeed, a low count ξ_1 indicates that candidate o_1 is more dense (Fig. 7a), since it can get $\hat{P}_b \geqslant \theta$ after probing less frames as opposed to another candidate o_2 with $\xi_2 > \xi_1$ boxes (Fig. 7b). In addition, the fact that $\xi_1 < \xi_2$ could also imply that the mean of a given o_1 is nearer to q (Fig. 7c) compared to o_2 (Fig. 7b). In such cases, we say that object o_1 *dominates* object o_2 as a $k\theta$NN candidate to query q. Hence, box counts are used as a rough measure for comparing candidate $k\theta$NNs.

4.3 Bounds on Least Box Counts

Suppose that object $o(\mu, \sigma)$ should be probed for a $k\theta$NN query with a given threshold θ. Depending on the location of focal point q along the verification axis, traversal of its respective verifier V_σ provides a different count ξ of elementary boxes that collectively gain a probability $\hat{P}_b \geqslant \theta$. Let box $\varepsilon_q \in V_\sigma$ be the one containing q. The idea is that we can establish bounds on box counts ξ for several possible placements of ε_q w.r.t. to mean μ. Indeed, for a given combination of threshold θ and uncertainty σ, we can precompute bounds iterating over possible placements of ε at discretized *distance ranges* along the middle row of its pertinent verifier V_σ as follows:

- $0 \leqslant L_2(q, \mu) < \frac{\delta}{2}$. Initially, q is in the central box ε_0 of verifier V_σ. Starting a traversal of V_σ from ε_0 until we reach $\hat{P}_b \geqslant \theta$ establishes the least required box count ξ_0.
- $\frac{(2i-1)\delta}{2} \leqslant L_2(q, \mu) < \frac{(2i+1)\delta}{2}$. At iteration $i > 0$, q is assumed to be i boxes to the left of ε_0 across the verification axis. So, traversing V_σ from box ε_i towards a probability at least θ gives the respective box count ξ_i.
- $L_2(q, \mu) \geqslant d_U$. The last iteration occurs once the current box count ξ_U from V_σ (with virtual boxes ignored) is stabilized w.r.t. to the previous iteration. d_U is the upper bound in the distance from q; if an object has its mean μ farther than d_U from q, it can never get a box count $\xi < \xi_U$.

As a result, given a threshold θ and uncertainty level σ, we can obtain a list with box counts ξ, each corresponding to a *distance range*

$d_{\sqcup}$ discretized by odd multiples of $\frac{\delta}{2}$. In a preprocessing step, we can repeat this process against all uncertainty levels $\sigma \in \Sigma$ and for a small set of *indicative* threshold values $\Theta = \{\theta_B : 0.5 \leqslant \theta_B < 1\}$ typical in many requests. This yields a lookup table of *upper bounds* $B = \{\langle \sigma, \theta_B, d_{\sqcup}, \xi \rangle\}$ at typical values $\theta_B \in \Theta$ and discretized distances $d_{\sqcup}$ for every $\sigma \in \Sigma$. In particular, for a specific pair of uncertainty level σ and threshold θ_B, when the discretized distance of a given object from q falls in a certain range $d_{\sqcup}$, then we lookup in B to get the least count ξ_B of elementary boxes required to cover probability at least θ_B. This guarantees that, at this distance range, we would never get more than ξ boxes in order to get sufficient probability for a candidate $k\theta$NN. As will be discussed next, at runtime we make use of lookup B in order to quickly prune candidates without traversing their verifiers.

4.4 Approximate Validation of Candidates

Our key intuition behind the use of discretized verifiers is that they can provide an approximate, yet computationally affordable means of probabilistically quantifying the proximity of candidate objects to a given query point. Admittedly, given the arbitrary placements of objects and their diverse uncertainty levels, there may occur situations when verifiers may falsely include an invalid (false positive) or discard a valid answer (false negative).

In order to employ verifiers in query evaluation, we relax the $k\theta$NN specification in Section 3.3 to issue *approximate results*:

Definition 4.4 (Approximate kθNNs over Gaussians). Let a query at focal point q that specifies a threshold θ for selecting its k-nearest neighbors from a set O of Bivariate Gaussian objects at timestamp τ. Then, a result set $Q = k\theta NN(q, O, \tau)$ of qualifying objects can be *approximately* computed as:

$$(Q \subseteq O) \ \wedge \ (|Q| = k) \ \wedge (\forall a \in Q, \hat{P}_b(q, d_a^b, V_{\sigma_a}) \geqslant \theta) \ \wedge \\ \wedge \ (\forall a \in Q, \forall o \in O \setminus Q, \xi_a \leqslant \xi_o) \quad (9)$$

where d_a^b is the extreme box distance of object $a \in O$ from q, and ξ_a is the box count after probing its corresponding verifier V_{σ_a}.

Again, ties amongst qualifying objects of equal box counts may be resolved arbitrarily. Following this specification, the core idea of our evaluation strategy at each timestamp τ is to check candidates and keep only those k of them with the least box counts ξ. Two lists track objects relevant to a given query q_i with threshold θ:

- List Q_i maintains up to k *currently qualifying* $k\theta$NN objects sorted by their ξ counts. Once no more objects need be checked for q_i, items in Q_i are emitted as results at time τ.
- List $\mathcal{A}_i$ holds up to n objects designated as *guards*, one guard per uncertainty level $\sigma \in \Sigma$. For a given σ, guard $o^\sigma \in \mathcal{A}_i$ can be (i) either the object in Q_i having the *maximal* box count ξ_{max} amongst items in Q_i with the same uncertainty σ, or (ii) if no object of uncertainty σ belongs to Q_i, then guard o^σ is a past candidate that so far has obtained the *minimal* box count ξ_{min} for uncertainty level σ. In this latter case, guard $o^\sigma \notin Q_i$. Intuitively, guard o^σ is the first object that should be replaced in $\mathcal{A}_i$ (and in Q_i, if it also qualifies as a $k\theta$NN), once a better candidate of similar σ is met.

Thus, while a given query q_i is being evaluated, less than $k + n$ objects need be maintained to safeguard its final results.

Algorithm 1: *kθNN* Monitoring (timestamp τ)

```
1  Input: Specifications $\langle q_i, k_i, \theta_i, \tau \rangle$, $i \in \{1, \ldots, M\}$ of $k\theta$NN queries
2  Input: Updates $\langle o_j, \mu_j, \sigma_j, \tau \rangle$, $j \in \{1, \ldots, N\}$ of Gaussian objects
3  Preprocessing: Verifiers $\{V_\sigma, \forall \sigma \in \Sigma\}$ of elementary box side $\delta$
4  Preprocessing: Lookup table of bounds $B = \{\langle \sigma, \theta_B, d_{\sqcup}, \xi \rangle\}$
5  State: Grid partitioning $G$ of 2-$D$ Euclidean plane in $g \times g$ cells
6  Output: $\bigcup_i \{\langle q_i, Q_i \rangle$: set $Q_i$ of $k\theta$NN objects per query $i = 1 \ldots M\}$
7  for each object $o_j$ updated at timestamp $\tau$ do
8      $c' \leftarrow$ cell where $o_j$ was indexed until now;
9      $c \leftarrow$hash$(G, \mu_j)$ ; // Cell where $o_j$ has currently its mean $\mu_j$
10     if $c \neq c'$ then
11         $c'.objList \leftarrow c'.objList \setminus \{o_j\}$;
12         $c.objList \leftarrow c.objList \cup \{o_j\}$;
13 for each focal query point $q_i$ do
14     $Q_i \leftarrow \emptyset$; $\mathcal{A}_i \leftarrow \emptyset$ ;          // Initialize lists of objects
15     $c_i \leftarrow$hash$(G, q_i)$ ;  // Cell where focal point is now located
16     $cList \leftarrow \{c_i\}$ ;          // List of cells to search w.r.t. $q_i$
17     $\rho \leftarrow 0$ ;           // Initial frame is the cell containing $q_i$
18     $d_k^b \leftarrow \infty$ ;  // No qualifying objects yet; distance undefined
19     repeat
20         for each cell $c \in cList$ do
21             CheckObjectsInCell($c, q_i, k_i, \theta_i, Q_i, \mathcal{A}_i$);
22         if $Q_i \neq \emptyset$ then
23             $d_k^b \leftarrow \max\{d_j^b, o_j \in Q_i\}$ ;    // Extreme box distance
24         $\rho + +$ ;   // Next frame in grid $G$ around focal point $q_i$
25         $cList \leftarrow$ FetchCells$(G, \rho, q_i, d_k^b)$ ; // Cells at next frame
26         $stop \leftarrow (|Q_i| \geqslant k)$ ;     // Are there enough candidates?
27         for each cell $c \in cList$ do
28             $stop \leftarrow stop \ \wedge (d_k^b \leq MINDIST(q_i, c))$;
29     until $cList = \emptyset$ or $stop$;
30     Report $\langle q_i, Q_i \rangle$ with $k\theta$NN objects in $Q_i$ sorted by $\xi$ counts
```

Algorithm 1 summarizes the monitoring process per timestamp τ. Typically for kNN monitoring [15, 21, 22], we apply a *grid partitioning* G that subdivides the entire monitored area into $g \times g$ equi-sized cells. We stress that grid cells are used for spatially indexing the *mean locations* of Gaussian objects; they should not be confused with elementary boxes in verifiers that discretize the cdf of objects at various uncertainty levels. Once fresh object updates are received at time τ, grid G gets updated in order to list which mean locations are contained in each cell $c \in G$ (Lines 7-12).

Each query is evaluated separately against the current uncertainty characteristics of objects. For each focal point q_i, we start examining objects with mean locations in grid cell c_i where q_i is found, and we continue with cells in successive *frames* outwards from c_i (Fig. 3b). This policy for cell-based inspection of potentially qualifying objects is similar to the one introduced in [15] for kNN monitoring over exact point locations. But, in our case we search for *uncertain* objects; even if such an object has not its mean in a cell c_i, its uncertainty region may overlap with c_i. As uncertainty levels vary among objects, we may need to check cells in extra frames outwards from c_i. So, in a priority queue *cList*, we keep grid cells pending for inspection. Objects in each such cell will be checked as detailed in Section 4.5. Initially (Lines 14-18), *cList* includes only cell c_i where q_i is now located; this is the initial frame (ρ_0) in the grid as shown in Fig. 3b. At each successive frame ρ outwards from q_i, a cell c from ρ is added to priority queue *cList* only if potential candidate objects are indexed in c, determined according to the following pruning rule:

Lemma 4.5. *Let grid cell $c \in G$ be the next one to visit at frame ρ outwards from focal point q. Given that $\theta \geqslant 0.5$, no object indexed in c can be a $k\theta$NN to q if $MINDIST(q,c) \geqslant d^b_k$, where d^b_k is the most extreme box distance amongst currently qualifying objects in Q.*

Proof. Most extreme box distance d^b_k depends on the last (k^{th}) object in Q (Lines 22-23), since items in Q are sorted by their box counts; the more the elementary boxes required for a candidate to reach θ, the larger its extreme box distance from q. Now, let an object $o(\mu,\sigma)$ indexed in cell c. Clearly, its mean μ cannot get any closer to q but at distance $L_2(q,\mu) = MINDIST(q,c)$, where $MINDIST$ is the Euclidean distance from q to the closest point in the perimeter of rectangle c, exactly as in [17]. Then, the circular coverage probability of object o at distance $d^b_k \leqslant MINDIST(q,c)$ is always $P_c(q, d^b_k, \mu, \sigma) \leqslant 0.5 \leqslant \theta$, whatever its uncertainty level σ. According to (7), there is no chance that o could replace any existing object in Q; the same holds for any other object $o'(\mu',\sigma')$ also indexed in cell c, because $L_2(q,\mu') \geqslant MINDIST(q,c)$. □

This rule is important because we can skip grid cells in a frame, but also because it provides a *termination* condition for the algorithm. Indeed, evaluation for query q_i must stop as soon as k qualifying objects have been collected in Q_i after visiting a number of grid frames and examining more frames outwards from q cannot alter Q_i. So, if we reach a frame ρ in grid G and Lemma 4.5 holds for all its cells, then no object indexed in or beyond this frame can qualify as an answer to q (Lines 26-28). Of course, once *cList* is exhausted, evaluation for q_i terminates (Line 29). Finally, the k objects contained in list Q_i are returned as the approximate $k\theta$NN results ranked in ascending order by their box counts ξ (Line 30).

4.5 Checking Objects Indexed in a Grid Cell

If a grid cell c cannot be skipped, then each object $o(\mu,\sigma)$ having its mean μ indexed in c should be checked (Algorithm 2). To avoid costly traversal of its respective verifier V_σ in cases that o cannot possibly qualify, we develop two pruning rules. In particular:

Pruning with Guards. For the uncertainty level σ of candidate o, we can promptly identify its respective guard $o^\sigma \in \mathcal{A}$. If there is one object at uncertainty level σ contained in $Q \cup \mathcal{A}$ that may be dropped, this is no other than o^σ. But, according to Lemma 2.5, if candidate o is farther from q than guard o^σ, then it cannot possibly alter either Q or $\mathcal{A}$, provided that guard $o^\sigma \notin Q$. Indeed, in case that o^σ is amongst the $k\theta$NNs in Q but at a rank $< k$, then candidate o (after probing its verifier) could perhaps qualify for some lower rank in Q and evict an object of different uncertainty. Otherwise, if o^σ is just a guard and (compared to o) it is closer to q, there is no possibility that candidate o can replace it (Lines 2, 5-6).

Pruning with Bounds on Least Box Counts. We make use of the precomputed bounds in lookup B (Section 4.3) so as to prune candidate o without probing. Indeed, we can find at which discrete

Algorithm 2: CheckObjectsInCell (grid cell c, focal point q, integer k, threshold θ, list Q of candidate objects, list $\mathcal{A}$ of guard objects)

```
1  for each object $o(\mu,\sigma) \in c.objList$ do
2      $o^\sigma \leftarrow$GetGuard($\mathcal{A}$, $\sigma$) ;    // for uncertainty level $\sigma$ of $o$
3      $o' \leftarrow Q$.back() ;   // Least probable object in current $k\theta$NNs
4      $\xi_B \leftarrow B(\sigma, \theta_{max}, d_{\sqcup})$ at $\theta_{max} = \max(\{\theta_B \in \Theta : \theta_B \leqslant \theta\})$ ;
5      if $((L_2(q, o.\mu) > L_2(q, o^\sigma.\mu)) \wedge (o^\sigma \notin Q))$ then
6          continue ;     // $o$ can neither be a $k\theta$NN nor a guard
7      else if $(o'.\xi \leqslant \xi_B)$ then
8          continue ;  // Even for a lower $\theta_{max}$, $o$ cannot be $k\theta$NN
9      else
10         $\langle \hat{P}_b, \xi \rangle \leftarrow$Probe($V_\sigma$, $q$, $\mu$, $\theta$) ; // #boxes to attain $\hat{P}_b \geqslant \theta$
11         if $((o^\sigma \notin Q) \wedge (o^\sigma.\xi > \xi))$ then
12             SetGuard($\mathcal{A}$, $\sigma$, $o$) ;     // $o$ replaces $o^\sigma$ at level $\sigma$
13         if $(o'.\xi > \xi)$ then
14             $Q$.insert($\langle o, \xi, \hat{P}_b \rangle$) ;        // $o$ qualifies as $k\theta$NN
15             if $(|Q| > k)$ then
16                 $Q$.pop() ;     // Evict $o'$ from $Q$, but $o'$ may...
17                 CheckGuard($\mathcal{A}$, $o'$) ; // become guard if $o'.\sigma \neq \sigma$
```

range $d_{\sqcup}$ its distance $L_2(q,\mu)$ falls in. We can also identify the greatest typical threshold $\theta_{max} = \max(\{\theta_B \in \Theta : \theta_B \leqslant \theta\})$ listed in B. So, we can readily lookup for item $B(\sigma, \theta_{max}, d_{\sqcup})$ and get the box count ξ_B required to achieve probability at least $\theta_{max} \leqslant \theta$ (Line 4). Thanks to discretization of verifiers, this guarantees that a candidate of such uncertainty characteristics would never traverse more than ξ_B boxes in order to qualify for $k\theta$NN. So, if o' is the currently qualifying k^{th} item in Q and its box count is $o'.\xi \leqslant \xi_B$, then we can safely discard candidate o; it cannot possibly replace existing $k\theta$NNs since it would need to traverse more elementary boxes, even for a lower threshold θ_{max} (Lines 7-8).

In case neither pruning condition is met, we have to probe verifier V_σ to retrieve a box count ξ, as well as its equivalent box coverage probability $\hat{P}_b$ (Line 10). This latter is useful, because if o eventually qualifies amongst the $k\theta$NNs, then $\hat{P}_b - \theta$ can be reported as an indicative *error margin* of its ranking. From box count ξ we can determine whether candidate o should become a guard or even qualify as a $k\theta$NN. Indeed, if existing guard o^σ has a greater box count, then o will replace it as the new guard in $\mathcal{A}$ for uncertainty level σ (Lines 11-12). Besides, let o' be the currently k^{th} object in Q having box count greater than ξ. Then, candidate o should replace o' and qualify itself among the $k\theta$NNs in Q. Note that disqualified object o' may still serve as guard in $\mathcal{A}$, if it refers to a diverse uncertainty level and there is no other object of equivalent uncertainty left in Q (Lines 13-17). Objects discarded from Q cannot be reinserted, even though they can still serve as guards. This happens because the box count of the k^{th} item in Q diminishes monotonically with the insertion of more items. So, if object o' disqualifies, it cannot belong to the final answer at timestamp τ.

5 EMPIRICAL STUDY

In this Section, we empirically validate our $k\theta$NN monitoring method against frequently updated moving objects of Gaussian uncertainty.

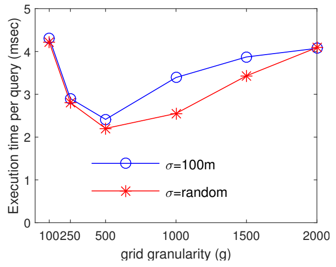


Figure 8: Grid granularity

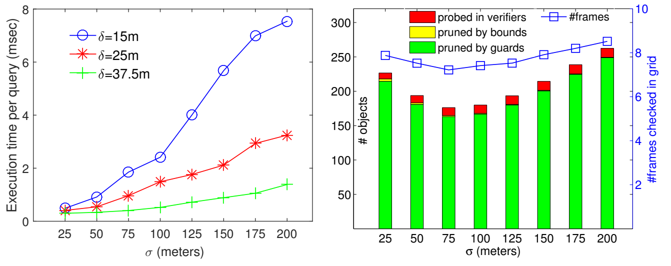


Figure 9: Effect from various uncertainty levels

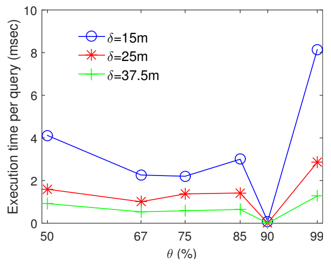


Figure 10: Varying threshold θ

5.1 Experimental Setup

We generated synthetic datasets for objects and queries moving at diverse speeds in an area of 25km×25km. We consider $N = 100\,000$ uncertain objects, each relaying updates of its (μ, σ) characteristics regularly at every timestamp $\tau = 1, \ldots, 200$. Similarly, we generated the initial focal points for $M = 10\,000$ queries at $\tau = 0$. Afterwards, movement *agility* of queries is set to 0.1; so, at each timestamp, a random 10% of focal points are chosen and they get randomly displaced to another location. This does not affect correctness of results, since we evaluate each query from scratch at every τ (so, we next report average execution times per query over 200 timestamps), but may occasionally increase execution cost depending on the actual density of uncertain objects nearby.

The algorithm was implemented in GNU C++ and all experiments were conducted on an Intel(R) Xeon(R) CPU E5-2660 at 2.20 GHz CPU and 96GB RAM running Ubuntu Linux. Performance measures are averages over all timestamps, whereas qualitative results are reported indicatively at $\tau = 100$. Table 2 lists the parameters and their range of values tested in the experiments; default values used in most simulations are shown in bold. Unless otherwise specified, at each timestamp any object can change randomly its level of uncertainty to a value from Σ.

5.2 Performance Results

The first set of experiments concerns selection of a suitable granularity g for the spatial grid. In Fig. 8, we plot average processing cost per query at each execution cycle τ, i.e., for each fresh batch of object updates. One test involved objects at a fixed uncertainty level $\sigma = 100$m, whereas a second test allowed objects to choose their uncertainty level randomly from Σ at each τ. Clearly, execution cost soars for either too coarse or too fine partitioning per dimension. In coarser subdivisions, each grid cell generally indexes a larger number of objects that need inspection, incurring significant overhead. But too fine partitionings also result to poor execution time and more expensive maintenance cost, as each uncertainty region covers a larger number of cells; so, the termination condition in Algorithm 1 is delayed. Hence, in the sequel, we fix $g = 500$ for this application setting. This yields the best performance at any uncertainty level, offering a good trade-off between efficiency in searching and reduced cost for grid maintenance.

Table 2: Experiment parameters

Grid granularity g per axis	100, 250, **500**, 1000, 1500, 2000
Uncertainty levels Σ (meters)	{ 25, 50, 75, **100**, 125, 150, 175, 200 }
Elementary box side δ (meters)	**15**, 25, 37.5
Number k of nearest neighbors	1, 2, 3, 4, **5**, 10, 20
Cutoff threshold θ	0.5, 0.67, **0.75**, 0.85, 0.9, 0.99
Thresholds Θ used in bounds	{ 0.6, 0.7, 0.8, 0.9 }

Next, we study the effect of uncertainty levels in performance. In each test, we fix σ at a specific value, so all moving objects have identical uncertainty. As Fig. 9a illustrates, the larger the σ, the more it takes to evaluate a query; indeed, more objects need inspection since their uncertainty regions have wider spread and thus cover more grid cells. Besides, higher uncertainty of objects also increases the cost of probing the respective verifiers, since they contain many more elementary boxes. Execution time for $\delta = 25$m is almost halved compared with $\delta = 15$m, since elementary boxes get bigger and thus probing of verifiers takes less time.

Figure 9b displays the total amount of objects in visited cells also for various σ. Of these candidates, the vast majority gets pruned thanks to the guards employed at each uncertainty level. Pruning via precomputed bounds regarding least box counts by certain distance ranges is rather weak, as all objects have the same σ, thus the pruning rule loses its utility. In particular, list Q will contain objects that are close to the focal point and have uncertainty regions of the same size, yielding the same box counts. Few objects can be pruned by this rule, as the box count in their bound most of the times cannot exceed that of the currently qualifying $k\theta$NN in Q. Objects that remain after pruning are those requiring probing against their respective verifiers. So, it is important that only a very small percentage really entails that costly operation; this test shows that pruning works adequately (especially with the guards), and costly traversals are avoided when possible. Also note that the number of frames checked in the grid is limited; thus, a relatively small number of grid cells will be visited. That explains why this number (the line plot) fluctuates similarly with the total number of objects being checked (the bar plot). Note that local minima are observed at $\sigma = 75$m. A possible reason might be that uncertainty regions of this magnitude fit better to the chosen grid granularity, so the termination condition (involving *MINDIST* distances from cells) is reached just in time without need to visit extra grid frames.

We also conducted tests with varying thresholds θ (Fig. 10). Generally, when θ is high, query execution time increases. This is because validation of each object asking for its verifier to be probed becomes more expensive, as more boxes must be traversed. However, note that execution also takes slightly longer for thresholds less than 60%; this is because there is no precomputed bound for

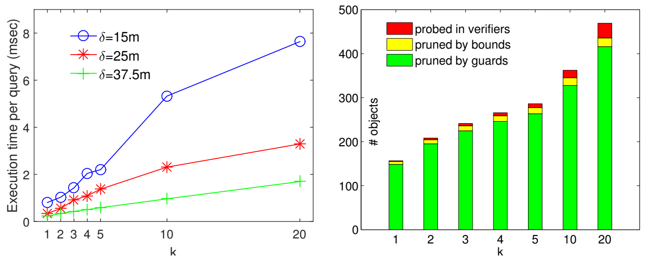
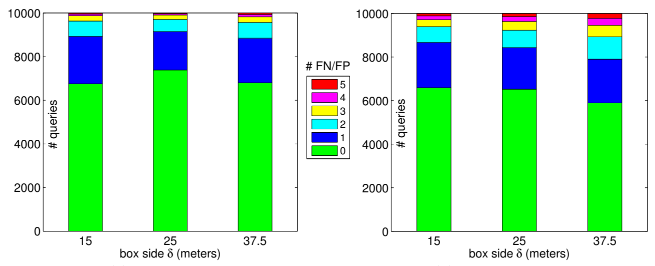


Figure 11: Effect of varying k



(a) False negatives (b) False positives

Figure 12: Qualitative results for $k = 5$

such θ that can be used in pruning (Section 4.5). Furthermore, when query threshold θ is very close to a threshold θ_B in a bound (e.g., $\theta = 75\%, \theta_B = 70\%$), then we expect such pruning to be more successful, as their box counts will be similar due to discretization. In contrast, when θ deviates noticeably from its respective θ_B (e.g. $\theta = 99\%, \theta_B = 90\%$), bounds are of little help in pruning. In case that query threshold θ coincides exactly to a θ_B (e.g., $\theta_B = 90\%$), execution time drops to less than 1msec per query, as practically the bound suffices to directly provide all results. Again, cost diminishes with coarser resolutions in elementary boxes (e.g., at $\delta = 37.5$m), as they get bigger in size and by inspecting fewer of them the method can fast surpass threshold θ in cumulative probability.

Of course, the number of k objects being searched as NNs also plays an important role in performance as Fig. 11a testifies. As expected, query execution takes longer with increasing k, as more grid cells may be reached around the focal point, so more objects need be checked and occasionally probed in the verifiers. Note that this increase is smooth for elementary boxes of greater size (i.e., larger box side δ), and the algorithm scales better as probing verifiers is much cheaper, since these contain less boxes. Traversing those verifiers is fast, hence execution time decreases.

Concerning the number of objects examined per query (Fig. 11b), again we observe that very few objects need probing in verifiers, thanks to pruning. As already noted, guards are particularly effective in discarding irrelevant candidates. It is no wonder that the total count of objects increases sublinearly with k. This happens because a similar number of grid cells may be visited for similar k values, and all objects indexed therein need examination.

5.3 Quality of Answers

In order to assess the quality of returned $k\theta$NN results for all queries, we have conducted exhaustive Monte-Carlo simulations, but only at indicative timestamps due to their excessive cost. Then, we compared their answers with those offered by our approximation method, allowing more than $k = 5$ results in case of ties. As mentioned in Section 4.4, false negatives (*FN*) and false positives (*FP*) may occur due to the biased estimations by discretized verifiers and their fixed box resolution. In Fig. 12, we plot the number of queries that received up to 5 FNs or FPs at timestamp $\tau = 100$. In this breakdown, with green bar we indicate queries with fully matching results, i.e., the approximate $k\theta$NN answer is identical to that from Monte-Carlo, so there are zero FNs and FPs. Of course, discretization of verifiers influences the amount of erroneous results. With boxes of $\delta = 25$m, it appears that slightly more correct answers are returned; this is simply because there are less cases with multiple objects ranked as the 5^{th}NN. Ties in the resulting ranks also explain the small fluctuations in the number of correct answers. In general, more than 60% of the queries have no wrong or missing answers at all; and if we tolerate at most one FP or FN, then at least 80% of the queries can get a fair response. Concerning ranking of the k returned answers (plots not shown in the interest of space), we observed that about 40% of results are emitted with exactly the same rank as indicated by Monte-Carlo. Overall, given the wide variety in the uncertainty of objects, this method seems capable of providing results of tolerable quality; considering the prohibitive cost of numerical methods, such a concession in the quality of results can be a reasonable trade-off for real-time monitoring. In the future, we plan to improve this framework in order to provide quality guarantees for a given box resolution in verifiers.

6 RELATED WORK

Spatial queries over objects of existential or locational uncertainty [2] issue each answer with a presumed probability. Regarding *locational uncertainty*, there are probabilistic variants for range search [16], similarity search [5], moving range kNN queries [12], reverse kNN queries [4, 7], Voronoi-based NN search [23], and more.

Specifically for a *probabilistic kNN query*, results are not just the k objects likely to be closest to the query point, but also the probability of each one to qualify as NN. For continuous pdf, [5] introduced a pruning filter estimating the probability of dominance between objects; such approximation can be further improved with filter and refinement. This conservative rule was tested for several spatial query types, including probabilistic threshold kNN search in databases. Such kNN queries were also specified in [8] for searching in databases that store uncertain objects of some continuous pdf. The answer set consists of subsets of k objects each, and each subset has probability above the given threshold. At a filtering stage, the search space is pruned by spatial and probabilistic criteria. Next, at refinement, results are verified by upper and lower bounds of joint probability of the candidate subsets, excluding those that certainly lie beyond the threshold. In another approach also in spatial databases [11], a kNN query was applied against uncertain objects, each represented by a set of discrete Monte-Carlo samples. To improve performance, samples were clustered and then indexed in an R-tree. Our work differs because our approximate answers return only a set of k most probable NNs. Also, we allow frequent and arbitrary updates in either objects or queries, and no such index can be effectively maintained over mutable pdf of objects.

The kNN search algorithms in [18] consider offline and online queries on objects whose historical trajectories are known, but each location is uncertain and modeled by a uniform circular region of fixed radius per object. Such queries return the set of all possible kNNs at each time interval. Regarding uncertain trajectories stored in a database, the geometric approach for range and NN search in [20] applies 3-D cylinders enclosing each segment. In contrast, our focus is on streaming uncertain locations and not trajectories; and unlike their simplified uniform model, online treatment of numerous moving Gaussians is computationally far more demanding.

The approach in [1, 2] is theoretical and provides approximation algorithms and bounds specifically for 1-NN search over objects with arbitrary pdf. In [1] the goal is to find the object that minimizes the expected distance from a query point under different distance measures. In [2] a randomized, output-sensitive algorithm employs Voronoi diagrams for computing all objects that are NNs of a query point with nonzero probability. Probability of an object to be NN is estimated with error guarantees, essentially approximating a continuous pdf by a discrete one. But, neither is it straightforward how such methods can be practically applied over streaming uncertainty updates nor their extensions to search for $k > 1$ NNs.

With respect to static multi-dimensional objects, U-tree [19] is a generic index for arbitrary pdf using probabilistically constrained regions to prune or validate an object. The tree index in [13] employs adaptive, piecewise-linear approximations of arbitrary pdf and can answer range as well as a variant of kNN queries without thresholds. The Gauss-tree proposed in [6] is an extension of the R-tree specifically over Gaussians. Instead of spatial coordinates, it models means and variances for disk-based data and can return the k most probable objects inside a given query range. In a streaming context with dynamic updates from Gaussian objects, the framework in [16] makes use of probabilistic verifiers when evaluating range queries with a cutoff threshold so as to provide approximate answers with quality guarantees as early as possible. Although our discretization of uncertainty regions follows a similar pattern, in our case the elementary boxes have always the same size in order to provide a combined probabilistic and distance measure about the proximity of objects to a query point. Our work is also distinct from privacy-aware query processing like [14], where user locations are cloaked into rectilinear areas, but without any notion of probability distribution therein. To the best of our knowledge, ours is the first work on probabilistic kNN search over moving Bivariate Gaussians with frequent, dynamic updates in their uncertainty characteristics.

7 CONCLUSIONS

In this work, we considered probabilistic k-nearest neighbor queries over numerous uncertain moving objects with Bivariate Gaussian locations. Due to frequent updates and varying degrees of uncertainty amongst objects, we introduced a discretization scheme to quickly estimate proximity of a Gaussian pdf to the query point, offering a probability measure for their comparison. We devised pruning criteria to eagerly discard irrelevant candidates in order to get the qualifying ones fast. Experiments over synthetic data under diverse query specifications confirm that this technique can provide approximate results of good quality in a timely fashion.

ACKNOWLEDGMENTS

Christos Koutras was supported by GRF grant #617713 from Hong Kong RGC.

REFERENCES

[1] P.K. Agarwal, A. Efrat, S. Sankararaman, and W. Zhang. Nearest Neighbor Searching Under Uncertainty. In *ACM PODS*, pp. 225-236, 2012.
[2] P.K. Agarwal, B. Aronov, S. Har-Peled, J.M. Phillips, K. Yi, W. Zhang. Nearest Neighbor Searching Under Uncertainty II. *TALG*, 13(1): 3, 2016.
[3] C.C. Aggarwal. *Data Mining – the Textbook*. Springer, 2015.
[4] T. Bernecker, T. Emrich, H.-P. Kriegel, M. Renz, S. Zankl, and A. Züfle. Efficient Probabilistic Reverse Nearest Neighbor Query Processing on Uncertain Data. *PVLDB*, 4(10): 669-680, 2011.
[5] T. Bernecker, T. Emrich, H.-P. Kriegel, N. Mamoulis, M. Renz, and A. Züfle. A Novel Probabilistic Pruning Approach to Speed up Similarity Queries in Uncertain Databases. In *ICDE*, pp. 339-350, 2011.
[6] C. Böhm, A. Pryakhin, and M. Schubert. Probabilistic Ranking Queries on Gaussians. In *SSDBM*, pp. 169-178, 2006.
[7] M.A. Cheema, X. Lin, W. Wang, W. Zhang, and J. Pei. Probabilistic Reverse Nearest Neighbor Queries on Uncertain Data. *IEEE TKDE*, 22(4): 550-564, 2010.
[8] R. Cheng, L. Chen, J. Chen, and X. Xie. Evaluating Probability Threshold k-Nearest-Neighbor Queries over Uncertain Data. In *EDBT*, 2009.
[9] A.R. di Donato and M.P. Jarnagin. A Method for computing the Circular Coverage Function. *Mathematics of Computation*, 16: 347-355, 1962.
[10] G.R. Hjaltason and H. Samet. Distance Browsing in Spatial Databases. *ACM TODS*, 24(2): 265-318, 1999.
[11] H.-P. Kriegel, P. Kunath, and M. Renz. Probabilistic Nearest-Neighbor Query on Uncertain Objects. In *DASFAA*, pp. 337-348, 2007.
[12] E.-Y. Lee, H.-J. Cho, T.-S. Chung, and K.-Y. Ryu. Moving Range k Nearest Neighbor Queries with Quality Guarantee over Uncertain Moving Objects. *Information Sciences*, 325: 324-341, 2015.
[13] V. Ljosa and A.K. Singh. APLA: Indexing Arbitrary Probability Distributions. In *ICDE*, pp. 946-955, 2007.
[14] C.-Y. Chow, M.F. Mokbel, and W.G. Aref. Casper*: Query Processing for Location Services without Compromising Privacy. *ACM TODS*, 34(4): 24, 2009.
[15] K. Mouratidis, M. Hadjieleftheriou, and D. Papadias. Conceptual Partitioning: An Efficient Method for Continuous Nearest Neighbor Monitoring. In *ACM SIGMOD*, pp. 634-645, 2005.
[16] K. Patroumpas, M. Papamichalis, and T. Sellis. Probabilistic Range Monitoring of Streaming Uncertain Positions in GeoSocial Networks. In *SSDBM*, pp. 20-37, 2012.
[17] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest Neighbor Queries. In *ACM SIGMOD*, pp. 71-79, 1995.
[18] A. Sistla, O. Wolfson, and B. Xu. Continuous Nearest-Neighbor Queries with Location Uncertainty. *VLDB Journal*, 24(1):25-50, 2015.
[19] Y. Tao, R. Cheng, X. Xiao, W. Ngai, B. Kao, and S. Prabhakar. Indexing Multi-Dimensional Uncertain Data with Arbitrary Probability Density Functions. In *VLDB*, pp. 922-933, 2005.
[20] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. Managing Moving Objects Databases with Uncertainty. *ACM TODS*, 29(3): 463-507, 2004.
[21] X. Xiong, M.F. Mokbel, and W.G. Aref. SEA-CNN: Scalable Processing of Continuous k-Nearest Neighbor Queries in Spatio-temporal Databases. In *ICDE*, pp. 643-654, 2005.
[22] X. Yu, K. Q. Pu, and N. Koudas. Monitoring k-Nearest Neighbor Queries Over Moving Objects. In *ICDE*, pp. 631-642, 2005.
[23] P. Zhang, R. Cheng, N. Mamoulis, M. Renz, A. Züfle, Y. Tangs, and T. Emrich. Voronoi-based Nearest Neighbor Search for Multi-Dimensional Uncertain Databases. In *ICDE*, pp. 158-169, 2013.