Continuous Probabilistic Nearest-Neighbor Queries for Uncertain Trajectories

Goce Trajcevski
Department of EECS
Northwestern University
goce@eecs.northwestern.edu

Roberto Tamassia*
Department of CS
Brown University
rt@cs.brown.edu

Hui Ding
Department of EECS
Northwestern University
hdi117@eecs.northwestern.edu

Peter Scheuermann†
Department of EECS
Northwestern University
peters@eecs.northwestern.edu

Isabel F. Cruz‡
Department of CS
University of Illinois at Chicago
ifc@cs.uic.edu

ABSTRACT

This work addresses the problem of processing continuous *nearest neighbor* (*NN*) queries for moving objects trajectories when the exact position of a given object at a particular time instant is not known, but is bounded by an uncertainty region. As has already been observed in the literature, the answers to continuous NN-queries in spatio-temporal settings are time parameterized in the sense that the objects in the answer vary over time. Incorporating uncertainty in the model yields additional attributes that affect the semantics of the answer to this type of queries. In this work, we formalize the impact of uncertainty on the answers to the continuous probabilistic NN-queries, provide a compact structure for their representation and efficient algorithms for constructing that structure. We also identify syntactic constructs for several qualitative variants of continuous probabilistic NN-queries for uncertain trajectories and present efficient algorithms for their processing.

1. INTRODUCTION

Moving Objects Databases (MODs) [8] are a fundamental technology for a wide variety of applications that may require some type of Location Based Services (LBS) [26] for mobile entities. The main tasks associated with MODs are: (1) the efficient management of the location-in-time information associated with mobile entities; (2) the efficient processing of various queries of interest, such as range or nearest neighbor (NN) queries. However, as has already been observed in the literature [5, 22], due to the imprecision of positioning technologies (e.g., roadside sensors, GPS), it is not always possible to ascertain the exact location of a particular moving object. Hence, *uncertainty* must be taken into account in the *data models*, in the *linguistic constructs* of the queries, and in the *processing algorithms*. The impact of various sources of imprecision in the context of probabilistic and uncertain data management has received considerable attention recently (e.g., [21, 31]), including spatial and spatio-temporal settings (e.g., [5, 22, 35, 37]).

Contrary to what happens in pure spatial settings [10, 24], the answer to a *continuous* NN-query in a spatio-temporal setting is *time parameterized* [33, 34] in the sense that the actual nearest neighbor of a given object need not be the same throughout the time interval of interest. As an example, assume that we have a MOD that consists of a set of trajectories: $\mathcal{S} = \{Tr_1, Tr_2, \ldots, Tr_N\}$, and a query **Q_nn(q)**: *"Retrieve the nearest neighbor of the moving object whose trajectory is Tr_q between t_b and t_e"*. The answer to the query is represented as a sequence **A_nn(q)**: $[(Tr_{i1}, [t_b, t_1]), (Tr_{i2}, [t_1, t_2]), \ldots, (Tr_{im}, [t_{m-1}, t_e])]$, expressing the fact that $Tr_{i1} \in \mathcal{S}$ is the nearest neighbor of Tr_q initially and up to time t_1. However, the nearest neighbor of Tr_q during the time interval $[t_{k-1}, t_k] \subseteq [t_b, t_e]$ $(k > 1)$ is the trajectory $T_{ik} \in \mathcal{S}$.

At the heart of the motivation for this work is the observation that incorporating *uncertainty* in the representation of the trajectories must be properly reflected in the syntax of both NN-queries and of their respective answers. For example, consider a simple extension to **Q_nn(q)**, in a manner that includes some uncertainty awareness, **UQ_nn(q)**: *"Retrieve all the objects that have a non-zero probability of being a nearest neighbor to the moving object Tr_q, between t_b and t_e."* In this case, in addition to a trajectory, e.g., Tr_{i1} being the nearest neighbor of Tr_q during $[t_b, t_1]$, it may well be that some other objects have a non-zero probability of being a nearest neighbor of Tr_q in some sub-intervals of $[t_b, t_1]$.

EXAMPLE 1. *Consider the scenario depicted in Figure 1. It illustrates 4 trajectories: Tr_1, Tr_2, Tr_3, and Tr_q, shown as 3D line segments; and possible bounds of the uncertainties of their locations, shown as sheared cylinders. Ignoring the*

*Research supported in part by NSF Awards IIS–0324846 and CCF–0830149.

†Research supported in part by NSF Award IIS-0325144/003.

‡Research supported in part by NSF Awards ITR IIS-0326284, IIS-0513553, and IIS-0812258.

EDBT 2009, March 24–26, 2009, Saint Petersburg, Russia.

uncertainty, the nearest neighbor of Tr_q is Tr_1 in $[t_b, t_1]$, and Tr_2 in $[t_1, t_e]$. However, if location uncertainty is taken into consideration, we see that not only Tr_1, but also Tr_3 has a non-zero probability of being the nearest neighbor to Tr_q at $t = t_{b1}$. Similarly, at $t = t_{11}$ all three trajectories have non-zero probabilities.

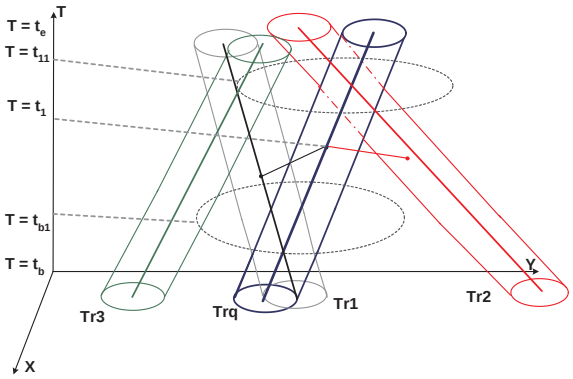

Figure 1: Continuous nearest neighbor for uncertain trajectories.

Clearly, we need to consider the entire duration $[t_b, t_e]$. However, it is even more important that we properly reflect the non-zero probability of being a nearest neighbor in all the sub-intervals at a level of granularity dictated by the problem setting. We postulate that the structure of the answer, **UA_nn(q)**, needs to be organized in a way that:

- It identifies the trajectories $Tr_{i1}, Tr_{i2}, \ldots$, which have the *highest probability* of being the nearest neighbor to Tr_q, and the corresponding time intervals $[t_b, t_1]$, $[t_1, t_2]$, ...
- It identifies *sub-intervals* within each $[t_{k-1}, t_k]$ during which a particular trajectory would have been ranked as the one with highest probability nearest neighbor of Tr_q, had it not been for Tr_{ik}.
- The structure is recursively refined for each sub-interval of time, until no lower granularity exists containing trajectories with non-zero probability of being a nearest neighbor to Tr_q.

Each component of the answer may be augmented by an extra *descriptor* of the properties of the probability values of the trajectory associated with the particular time interval. For instance, such descriptors may contain: coefficients/functions of an analytical expression (if possible), *min*/*max* values, plus a discrete sequence of values of the probability at time instants within the given interval, etc.

To represent the structure of **UA_nn(q)**, we propose an *interval tree* in which:

- The root consists of the parameters of the query (i.e., query trajectory Tr_q and the time interval $[t_b, t_e]$).
- The children of each internal node are the nodes that, with the exclusion of their parents, have the highest probability of being the nearest neighbors of Tr_q, within the time interval bounded by the parent.

The structure of each internal or leaf node consists of the following attributes:

1. time-interval $[t_i, t_{i+1}]$ of relevance;
2. unique *ID*, say, Tr_i, of the trajectory corresponding to the answer during the time-interval $[t_i, t_{i+1}]$;
3. *descriptor* D_i of the properties of the probability of Tr_i being the nearest neighbor to Tr_q within $[t_i, t_{i+1}]$;
4. pointers to the children-trajectories that have the next-highest probability of being the nearest neighbor within the disjoint sub-intervals of $[t_i, t_{i+1}]$.

Clearly, this type of tree need not be balanced in terms of the height and number of children for each internal node, but we note that the leaf nodes correspond to the trajectories that have the smallest probability of being an uncertain nearest neighbor of Tr_q within the corresponding time-intervals (i.e., no other trajectory has a smaller non-zero probability). We call this tree *IPAC-NN* (Interval-based Probabilistic Answer to a Continuous NN query) and we illustrate it in Figure 2. We note that if the root of the tree is removed, in effect we have a Directed Acyclic Graph (DAG), which represents the answer. Given this declarative description of the semantics of the answer to a continuous probabilistic NN-query, the focus of the rest of this work is on the procedural counterpart: constructing the *IPAC-NN* tree for a given query. We note that we do not address the issue of calculating the descriptors D_i of the individual nodes. Instead, we concentrate on *ranking* [30]. In addition to formalizing the semantics of the structure of the answers to continuous probabilistic NN-queries for uncertain trajectories, our main contributions can be summarized as follows:

- We identify a simple transformation of a view over the uncertain trajectories, which enables a construction of the *relative ranking* of the probabilistic values for instantaneous uncertain NN-queries.
- We demonstrate that our transformation is applicable to a large class of probability density functions (*pdf*s) that describe the uncertainty associated with the location.
- We develop efficient algorithms to construct a geometric dual of a IPAC-NN tree.
- We identify several syntactic variants for systematic incorporation of uncertainty in the statement of the continuous NN-queries; for each variant we present an efficient algorithm for its processing, based on the dual of the IPAC-NN tree.
- We present experimental observations demonstrating the benefits of our approach.

The rest of this paper is structured as follows. In Section 2, we gather the necessary background. Section 3 presents the main contribution of our work in terms of the transformation of the uncertain trajectories and its implication towards algorithmic construction of the IPAC-NN tree, as well as identifying the class of (instantaneous) location *pdf*s for which the transformation is applicable. In Section 4, we present the different variants of the continuous probabilistic NN-queries and their processing. Section 5 presents our experimental observations and Section 6 positions our work with respect to the related literature. Finally,

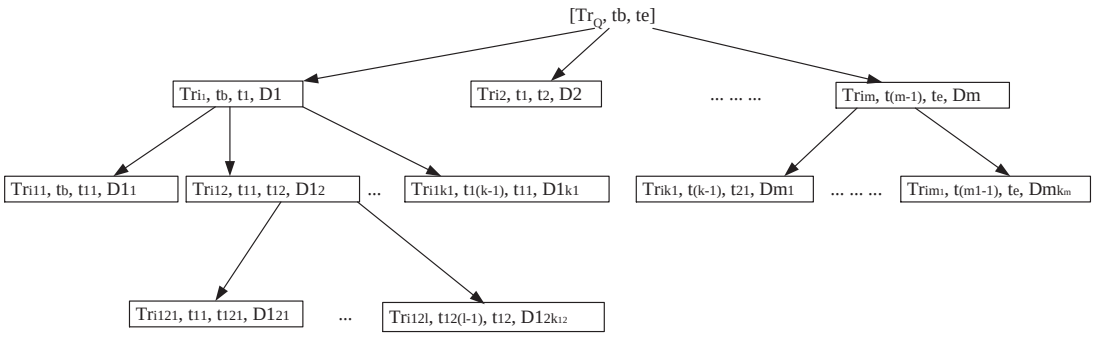

Figure 2: Interval tree of the answer to a probabilistic continuous NN-query.

in Section 7, we give some concluding remarks and outline directions for future work.

Due to space limitations, we have omitted several details and most proofs. For a full version of this paper, see [36].

2. PRELIMINARIES

In this section, we introduce the background necessary for the development of our main results. First, we define the model of uncertain trajectories used throughout this work. Subsequently, we recall some results pertaining to instantaneous NN-queries for uncertain objects for the special case when the querying object is *crisp* (i.e., its location is exact, without any uncertainty) [5].

2.1 Uncertainty of motion

Selecting the model for the motion plan of the moving objects affects not only the algorithms for processing fundamental spatio-temporal queries (e.g., range queries and NN queries) [8], but also the representation of uncertainty. For example, the moving object may transmit periodic updates of the form (x, y, t) reporting its (x, y) location at time t (obtained, e.g., by an on-board GPS system) [18]. Given an upper-bound on its maximum speed v_{max}, the location in between two updates is bounded by an ellipse (see Figure 3.a) [11, 22].

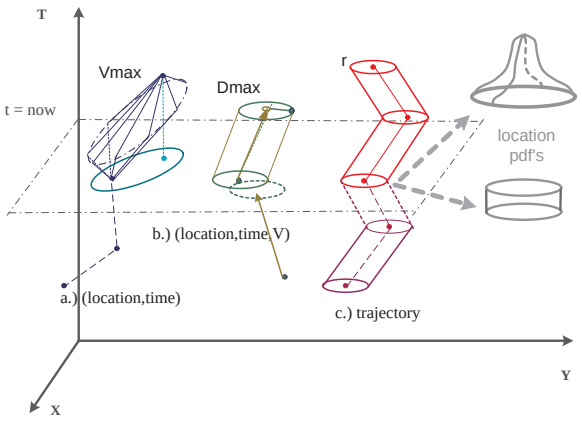

Figure 3: Motion models and uncertainty.

In case the object transmits updates of the form (x, y, t, v), where v is its *expected* velocity, it need not transmit another update for as long as its sampled location at any time $t_1 > t$ does not deviate more than a certain threshold, say D_{max}, from its expected location at t_1. This is called a *dead-reckoning* policy [38], and the possible whereabouts of an object are illustrated in Figure 3.b.

Our work assumes that each moving object has a *full trajectory* as its motion model, corresponding to the settings in which: (1) the *beginning location*; (2) the *ending location*; (3) the *beginning time*; and (4) possibly a set of points to be visited, are transmitted to the MOD server. Based on the information available at the electronic maps, along with the traffic patterns, the server constructs the *shortest travel time* or *shortest path* trajectory, and transmits it back the user, keeping a copy for query processing [9]. The uncertainty model with the full trajectory is often based on the assumption that at each time instant there is a bound on the object's possible whereabouts [37], as shown in Figure 3.c. This figure also illustrates that at a given time instance, the *pdf* of the location of the object within its boundaries may take different forms (e.g., uniform or bounded-Gaussian).

Formally, a *trajectory* Tr_i of a given object oid_i is a function $Time \rightarrow \mathcal{R}^2$, represented as a sequence of 3D (2D spatial plus Time) points, accompanied by a unique ID of the moving object: $Tr_i = \{oid_i, (x_{i_1}, y_{i_1}, t_{i_1}), (x_{i_2}, y_{i_2}, t_{i_2}), \ldots, (x_{i_k}, y_{i_k}, t_{i_k})\}$ where $t_{i_1} \leq t_{i_2} \leq \ldots \leq t_{i_k}$. In between two consecutive points, the location at time $t \in (t_{i_{(k-1)}}, t_{i_k})$ is obtained by linear interpolation, assuming that oid_i is moving along a straight line-segment and with a constant speed, calculated as:

$$v_{i_k} = \frac{\sqrt{(x_{i_k} - x_{i_{(k-1)}})^2 + (y_{i_k} - y_{i_{(k-1)}})^2}}{t_{i_k} - t_{i_{(k-1)}}} \qquad (1)$$

An *uncertain trajectory* Tr_i^u is a trajectory augmented with (1) the information about the *radius* of the circle bounding the *uncertainty zone*, i.e., the disk representing the 2D set of possible locations of the object at a given time instant; and (2) the *pdf* of the location within the uncertainty disk. Hence, we have: $Tr_i^u = \{r, pdf, (x_{i_1}, y_{i_1}, t_{i_1}), (x_{i_2}, y_{i_2}, t_{i_2}), \ldots, (x_{i_k}, y_{i_k}, t_{i_k})\}$. The location of the object in the center of the uncertainty disk is now called its *expected location* and we use $D_i(t)$ to denote the uncertainty disk of Tr_i at time t. When it comes to future trajectories generated by trip-planning services, this type of a location-uncertainty at a given time-instance indicates the following:

- The acceptable location-error, with respect to the planned trajectory, as measured by the on-board devices.
- The acceptable time-discrepancy for a given location (e.g., the moving object has arrived earlier or later than predicted), the boundaries of which can be obtained by intersecting the sheared cylinder at a given (X,Y) value, with a vertical plane, perpendicular to the 2D vector of the direction of motion.

Note that the time-discrepancy tolerance is a function of the velocity at a given time-instance [3].

Throughout this work, we assume the parameters r and *pdf* are the same for the trajectories in a given set. As commonly assumed in the literature (e.g., [5, 35]) we also assume that, viewed as random variables, the *pdf*s of the locations of the uncertain objects are *independent* from each other. We note that in the examples we use *uniformly distributed* 2D random variables in the uncertainty zone. Assuming that the expected location of the object with oid_k at time t is $(x_k(t), y_k(t))$, we have

$$pdf_k(t)(X,Y) = \begin{cases} 0, & \sqrt{(x_k(t)-X)^2+(y_k(t)-Y)^2} > r \\ \frac{1}{r^2\pi}, & \sqrt{(x_k(t)-X)^2+(y_k(t)-Y)^2} \leq r \end{cases} \quad (2)$$

However, as we will formally demonstrate in Section 3, our results are applicable to a much larger class of *pdf*s.

2.2 NN-Query for a crisp querying object

Assume that the location of the querying object Tr_q is a *crisp* 2D point Q, and the possible locations of the other trajectories are disks with radii r. A thorough treatment of this problem setting is presented in [5]. We summarize this previous work with the following observations that are immediately relevant to our paper.

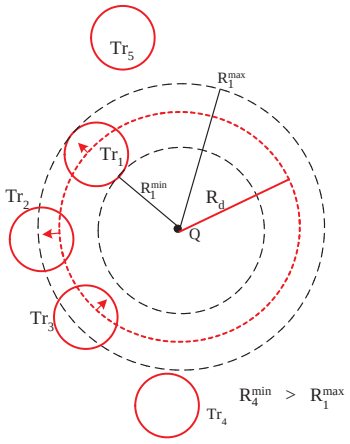

Figure 4: Uncertain NN-query (crisp Tr_q).

Observation I.

The distance from Q to the most distant point of the closest disk, R_{max}, is the upper bound on the distance that any possible nearest neighbor of Tr_q can have. Consequently, any Tr_i whose closest possible distance to Q, denoted by R_i^{min}, is larger than R_{max}, has a 0 probability of being a nearest neighbor to Tr_q and can therefore be safely pruned (i.e., ignored from any computation). As illustrated in Figure 4, $R_4^{min} > R_1^{max}$, and similarly $R_5^{min} > R_1^{max}$, which means that Tr_4 and Tr_5 cannot have a non-zero probability of being a nearest neighbor to Tr_q. We use R_{min} to denote the distance from Q to the closest point of the closest disk.

In general, the probability that (the location along the trajectory at a given time of) a given object Tr_i is *within distance* R_d from $Q(= Tr_q)$ can be specified as:

$$P_{i,Q}^{WD}(R_d) = \int_A \int pdf_i(x,y)\, dx\, dy \quad (3)$$

where A is the area of the intersection of the disk with radius R_d centered at Q and the uncertainty disk of Tr_i and $pdf_i(x,y)$ is the corresponding *pdf* of Tr_i.

EXAMPLE 2. *[5] When $pdf_i(x,y)$ is uniform, the probability $P_{i,Q}^{WD}(R_d)$ can be calculated as:*

$$P_{i,Q}^{WD}(R_d) \begin{cases} 0 & if(R_d < r_{min_i}) \\ \frac{1}{R_d^2\pi}(\Theta - \frac{1}{2}\sin 2\Theta) + \frac{1}{\pi}(\alpha - \frac{1}{2}\sin 2\alpha) & \\ & if(d_{iQ} - r \leq R_d \leq d_{iQ} + r) \\ 1 & if(d_{iQ} + r < R_d) \end{cases} \quad (4)$$

where $\Theta = \arccos\frac{d_{iQ}^2 + r^2 - R_d^2}{2 d_{iQ} r}$ *and* $\alpha = \arccos\frac{d_{iQ}^2 + R_d^2 - r^2}{2 d_{iQ} R_d}$, *and d_{iQ} is the distance between Q and the expected location*[1] *of Tr_i. Taking the derivative of $P_{i,Q}^{WD}$, yields $pdf_{i,Q}^{WD}(R_d)$ which, in the case of uniform distribution, will be non-zero only when $d_{iQ} - r \leq R_d \leq d_{iQ} + r$.*

Observation II.

The probability $P_{j,Q}^{NN}$ of a given object, say Tr_j, being a nearest neighbor of the crisp querying object Tr_q is as follows:

$$P_{j,Q}^{NN} = \int_0^\infty pdf_{j,Q}^{WD}(R_d) \cdot \prod_{i \neq j}(1 - P_{i,Q}^{WD}(R_d))\, dR_d \quad (5)$$

We note that the boundaries of the integration need not be 0 and ∞ because the effective boundary of the region for which an object can be a nearest neighbor of Q is the ring centered at Q with radii R_{min} and R_{max}. In addition, $pdf_{j,Q}^{WD}(R_d)$ is 0 for any $R_d < R_j^{min}$, and $1 - P_{i,Q}^{WD}(R_d)$ is 1 for $R_d < R_i^{min}$. By sorting the objects that have a non-zero probability of being nearest neighbors according to the minimal distances of their boundaries from Q, one can break the evaluation of (5) into several intervals (one for each R_{min_i}), and the computation of the $P_{j,Q}^{NN}$ can be performed in an efficient manner, based on the sorted distances and the corresponding intervals [5]. Such efficiency is especially important because the actual evaluation of the integrals like those in Equation (5), may often rely on numerical computations. In a uniform distribution this is equivalent to sorting the objects according to the distances of their respective expected locations from Q.

Observation III.

We note that the ideas above, although intuitive, have a slight problem in the context of *soundness* vs. *completeness*. Namely, the evaluations of $P_i^{NN}(Q)$ as defined by Equation (5), do not constitute a *probability space* [7] or, in terms of classical probability, $\Sigma_{\forall i} P_{i,Q}^{NN} < 1$. The reason is that, strictly speaking, the probability of a given object being the nearest neighbor to Tr_q consists of two parts:

[1]Appropriate modifications are needed when Q is located inside the uncertainty zone of Tr_i [5].

$$P_{i,Q}^{NN} = P_{i,Q}^{NN\_E} + P_{i,Q}^{NN\_J} \quad (6)$$

The first part, $P_{i,Q}^{NN\_E}$, denotes the *exclusive* probability that Tr_i is the nearest neighbor of Tr_q and is calculated in the spirit of (5). The second part, $P_{i,Q}^{NN\_J}$, represents the *joint* probability and corresponds to the case(s) in which Tr_i is the nearest neighbor of Tr_q along with some other Tr_j's. It consists of the following sums:

- $\Sigma_j \int_0^\infty pdf_{i,Q}^{WD}(R_d) \cdot pdf_{j,Q}^{WD}(R_d) \cdot \prod_{k\neq i,j}(1 - P_{k,Q}^{WD}(R_d))\, dR_d$
  corresponding to the cases when Tr_i is a *paired-NN* with other Tr_j's;
- $\Sigma_k \int_0^\infty pdf_{i,Q}^{WD}(R_d) \cdot pdf_{j,Q}^{WD}(R_d) \cdot pdf_{k,Q}^{WD}(R_d) \cdot \prod_{l\neq i,j,k}(1 - P_{l,Q}^{WD}(R_d))\, dR_d$
  capturing all the cases of triplets of trajectories being the nearest neighbor to Tr_q;
- ...
- $\int_0^\infty \prod_i pdf_{k,Q}^{WD}(R_d)\, dR_d$
  calculating the probability that all trajectories can simultaneously be nearest neighbors.

3. MOVING CONVOLUTIONS

In this section, we present a first set of results of our work. We start by illustrating the problems that arise when the query object has an uncertainty associated with its location. Next, by using a simple transformation, we show that for a large class of *pdf*s, we can reduce this case to one in which the ideas presented in Section 2.1 can be applied almost unmodified. We subsequently present a methodology for constructing the geometric dual of the IPAC-NN tree.

3.1 Within distance: uncertain query object

For the time being, let us still consider a "snapshot" query in which the location of the querying object Tr_q is also uncertain, and can be anywhere within the disk of radius r centered at the expected location Q.

The first observation is that we can no longer prune the objects whose uncertainty disk is further than R_{max} from Q. An illustration is provided in Figure 5. Namely, when Tr_q is located somewhere in the zone denoted by Z_1 inside of its own uncertainty disk and Tr_4 is located somewhere in the zone denoted by Z_2, their distance is less than R_{max} and, consequently, Tr_4 has a non-zero probability of being a (possible) nearest neighbor of Tr_q. This fact complicates the main benefits in terms of compactness of the representation and the efficiency of processing probabilistic NN-queries with respect to using the formulas from Section 2.2 [5]. Strictly speaking, at the heart of the problem is the calculation of the probability that a given object Tr_i is *within distance* R_d of Tr_q.

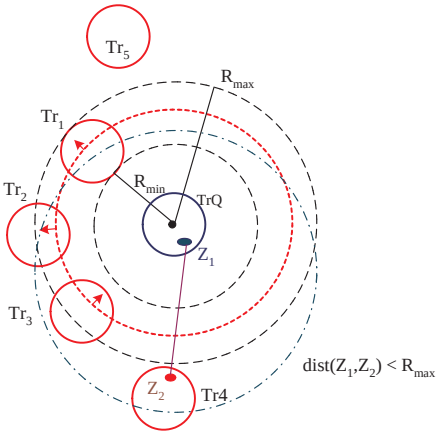

Figure 5: Uncertain NN-query (uncertain Tr_q).

Since the distributions of the objects within their spatial boundaries are independent, one can obtain the probability of two objects being within distance $\leq R_d$ from each other as follows:

1. Find the set of all the possible locations in the uncertainty disk D_i that are at distance R_d from *some* point in the disk D_q. This set is actually the intersection $D_i \cap (D_q \oplus R_d)$, where $(D_q \oplus R_d)$ denotes the *Minkowski sum* (see, e.g., [6]) of the uncertainty disk of Tr_q with a disk of diameter R_d.
2. For each point $P(=(x_p, y_p)) \in D_i \cap (D_q \oplus R_d)$ and a point $Q \in D_q$, evaluate $P_{q,P}^{WD}(R_d)$ using, e.g., Equation (3), and "add" the uncountably-many such results—that is, integrate over the area $D_i \cap (D_q \oplus R_d)$, with dx_p and dy_p as the extra variables of differentiation.

This yields a quadruple integration in the corresponding version of Equation (3) used for evaluating $P_{i,q}^{WD}(R_d)$ and yields additional overhead in determining the $pdf_{i,q}^{WD}(R_d)$ (via differentiation), in order to be able to use Equation (5) for evaluating $P_{i,q}^{NN}$. Most often, the procedure outlined above will rely on a numerical evaluation, which approximates the outer-integrals by a sum of the products of the probabilities that Tr_i is at location $l_1 \in D_i$, given that Tr_q is at location l_2, and $\|l_1 l_2\| \leq R_d$ (over all such locations l_1 and l_2, and after discretizing the corresponding location-*pdf*s [5, 35]). Since the locations of the individual objects are assumed to be independent, the conditional probability $Pr(Tr_i = l_1 \mid Tr_q = l_2)$ is simply $Pr(Tr_i = l_1)$.

EXAMPLE 3. *Figure 6 shows the locations of 3 uncertain objects with uniform pdf*s*. Each of them has the uncertainty radius of 1 and their respective expected locations are* $E_{loc}(Tr_q) = (2,2)$, $E_{loc}(Tr_1) = (7,3)$ *and* $E_{loc}(Tr_2) = (3,8)$. *The two dashed segments of circles, centered at two locations inside the uncertainty disk of* Tr_q *illustrate part of the calculation of the probability of* Tr_1 *being within distance* ≤ 4 *from* Tr_q *(obviously, 0 for* Tr_2*).*

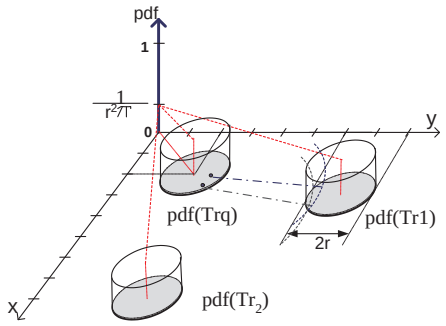

Figure 6: Evaluation of within-distance probability.

We are now in the position to explain the theoretical foundation of our main results. Let $\overline{V}_i$ denote the 2D random variable representing the possible locations of the uncertain trajectory Tr_i^u at a given time instant. Recall that the crux for evaluating a probabilistic NN-query is determining the expression for the probability of Tr_i^u being within a given distance R_d from Tr_q^u, which is the value of $P_i^{WD}(R_d)$. An equivalent interpretation is that we need to evaluate $P(\|\overline{V}_i - \overline{V}_q\| \leq R_d)$. Now, the key observation is that $\overline{V}_i - \overline{V}_q$ is another random variable, which we denote with $\overline{V}_{iq}$. In probability and signal/image processing, $\overline{V}_{iq}$ is known as the *cross-correlation* of $\overline{V}_i$ and $\overline{V}_q$ [17, 20]. Another interpretation is that $\overline{V}_{iq}$ can be viewed as a sum $\overline{V}_i + (-\overline{V}_q)$. Since $\overline{V}_i$ and $\overline{V}_q$ (consequently, $-\overline{V}_q$) are independent variables [5, 35], it is a well-known fact from the probability theory that the random variable $\overline{V}_{iq}$ has a pdf_{iq} that is a *convolution* of the corresponding *pdf*s of $\overline{V}_i$ and $-\overline{V}_q$) [20]. In other words, we have

$$pdf(\overline{V}_{iq}) = pdf(\overline{V}_i) \circ pdf(-\overline{V}_q)$$

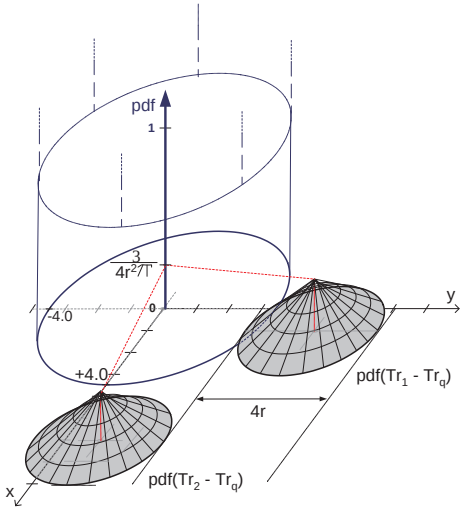

Figure 7: Within-distance probability: convolution.

Example 4. *The convolution of two cylinders with heights $\frac{1}{r^2\pi}$ is a cone whose base is a circle with radius $2r$ and height $\frac{3}{4r^2\pi}$ [20]). As illustrated in Figure 7, instead of performing uncountably many additions (e.g. adding an extra outer-integration) in the context of Example 2, for the various circles of radius 4 centered somewhere in the uncertainty disk of Tr_q (see Figure 6), we can now use a simpler calculation: evaluate the volume of the intersection of the cone centered at $(5,1)$ $(= (7,3) - (2,2))$, with the cylinder with radius 4 centered at the origin (0,0). Specifically, for uncertain trajectories with uniform location-*pdf*s, by Equation (2), we have*

$$pdf(\overline{V}_{iq}(t)(X,Y)) =$$
$$0 \text{ if } \sqrt{((x_i(t)-x_q(t))-X)^2+((y_i(t)-y_q(t))Y)^2} > 2r$$
$$\frac{3}{4r^2\pi}\left(1-\frac{\sqrt{((x_i(t)-x_q(t))-X)^2+((y_i(t)--y_q(t))Y)^2}}{2r}\right) \text{ otherwise}$$

We note that, in order for a convolution of two functions to exist (i.e., two functions to be *convolutable*) it is sufficient that each of them is *Lebesgue-integrable* [25]. However, in many practical settings, the *pdf*s of the objects' locations (e.g., uniform, Gaussian) are *Riemann-integrable* [25], which is a weaker condition. Before presenting the main result, we prove some properties which demonstrate that our proposed methodology is applicable to a wide range of *pdf*s for objects' locations. For brevity, we will use f to denote $pdf(\overline{V}_{iq})$, g to denote $pdf(\overline{V}_i)$, and h to denote the $pdf(-\overline{V}_q)$.

Property 1. *Assume that g (resp. h) has a centroid $\overline{C}_1$ (resp. $\overline{C}_2$), which coincides with its expected value $E(\overline{V}_i)$, resp. $E(-\overline{V}_q)$. Then their convolution $f = g \circ h$ has a centroid $\overline{C}_c = \overline{C}_1 + \overline{C}_2$, and $\overline{C}_c$ is the expected value of the variable $\overline{V}_{iq}$.*

As specific examples, the expected value of the convolution of two Gaussian distributions with means $\overline{\mu}_1$ and $\overline{\mu}_2$, is exactly $\overline{\mu}_{12} = \overline{\mu}_1 + \overline{\mu}_2$, and we note that the *pdf* of the convolution is also Gaussian [20]. Similarly for the expected value of two uniform distributions, however, as we saw in Example 3, the resulting *pdf* is no longer uniform.

Property 1 provides a basis for defining the categories of *pdf*s for which our main results are applicable, and towards that end, we need to define the concept of a *rotational* (a.k.a *cylindrical*) symmetry [17]. A given 2D function, say h, is said to be rotationally symmetric with respect to a point $\overline{C}$ in its domain and the vertical (Z) axis if, for all other points P and Q in its domain, $\|\overline{PC}\| = \|\overline{QC}\| \Rightarrow h(\overline{P}) = h(\overline{(}Q))$. Now we have:

Property 2. *Assume that g and h have a rotational symmetry around their respective centers, $\overline{C}_1$ and $\overline{C}_2$, with respect to the vertical ($Z = pdf$) axis. Then, their convolution $f = g \circ h$ also has a rotational symmetry around the vertical axis and with respect to its centroid $\overline{C}_c$.*

Assume that Tr_1^u and Tr_2^u denote two uncertain trajectories with centers (expected locations) C_1 and C_2 at some time-instant t. In addition, assume that they have same (modulo translation) corresponding location *pdf*s at t, which are rotationally symmetric. The last claim that is needed before we state our main result for this section is summarized in the following lemma:

Lemma 1. *Let Q denote a 2D point. If $\|\overline{QC}_1\| < \|\overline{QC}_2\|$, then $P_1^{NN}(Q) > P_2^{NN}(Q)$.*

Assume that we are given a collection of moving objects with equal *pdf*s (modulo translation with respect to their centers), which are rotationally symmetric. Let Tr_q denote the (uncertain) querying trajectory. The main result of this section is summarized in the following theorem.

Theorem 1. *The permutation of the oids representing the ranking of the probabilities of individual objects being nearest neighbor to Tr_q^u at a given time-instance is exactly the same as the permutation representing the ranking of the distances of their centers (expected locations) from the center (expected location) of Tr_q^u.*

Proof: This result follows from the properties of the convolution for independent random variables with rotational symmetry and from Lemma 1. □

As an illustration, recall Figure 7. Since the centroid of $Tr_1^u - Tr_q^u$ is closer to the coordinate-center than the centroid of $Tr_2^u - Tr_q^u$, we have that $P_1^{NN}(Q) > P_2^{NN}(Q)$.

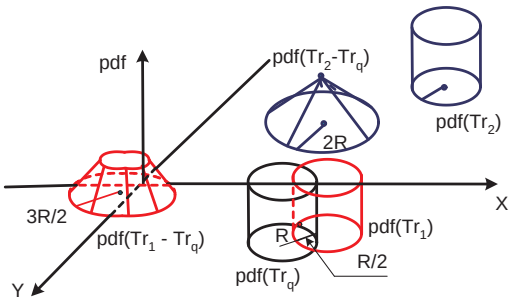

Figure 8: Convolution of intersecting *pdf*s.

We conclude this section by observing that in the examples so far, we have assumed that the uncertainty disks of the respective trajectories do not intersect. However, in practice, this need not be the case. For instance, Figure 8 shows the impact on the (*pdf* of the) resulting convolution when a given trajectory intersects the querying trajectory. Nevertheless, it can be demonstrated that the main results presented in this section are still valid.

3.2 Continuous uncertain NN-queries

The basic observation that the difference of two trajectories can be expressed as a single random variable, along with Theorem 3.1, forms the foundation for constructing the IPAC-NN tree introduced in Section 1, which is what we focus upon now. Without loss of generality, we assume that throughout the duration of the time-interval of interest for a given query **UQ_nn(q)**, $[t_b, t_e]$, each trajectory consists of a single segment (i.e., each object's expected location is along a 2D line segment).

Let (x_{bi}, y_{bi}) denote the expected location of the uncertain trajectory Tr_i^u at t_b and, similarly, (x_{ei}, y_{ei}) denote the expected location of Tr_i^u at t_e. The expected motion of Tr_i^u during $[t_b, t_e]$ will be characterized by a velocity vector whose corresponding X and Y components are:
$v_{xi} = (x_{ei} - x_{bi})/(t_e - t_b)$ and $v_{yi} = (y_{ei} - y_{bi})/(t_e - t_b)$. Hence, the expected location at some time instant $t \in [t_b, t_e]$ will have coordinates:
$x_i(t) = x_{bi} + v_{xi}(t - t_b)$ and $y_i(t) = y_{bi} + v_{yi}(t - t_b)$, which are the coordinates of the center of the uncertainty disk at t.

For a given trajectory Tr_i^u that is not the query trajectory (i.e., $i \neq q$), let TR_{iq} denote the *difference-trajectory* $Tr_i^u - Tr_q^u$. In other words, at each time instant t, the expected location of the object moving along $TR_{iq}(t)$ is a vector-difference of the expected locations of the corresponding points along $Tr_i^u(t)$ and $Tr_1^u(t)$. $TR_{iq}(t)$ captures the spirit of Section 3.1, in the sense that the 2D distance between the expected locations of the objects moving along $Tr_i^u(t)$ and $Tr_1^u(t)$ (at time t) (see, e.g., [2, 23]) now becomes the distance at that same time t that an object moving along TR_{iq} has from the origin (0,0). Let $V_{x_{iq}} = v_{xi} - v_{xq}$, $V_{y_{iq}} = v_{yi} - v_{yq}$ denote the components of the velocity of the object whose expected trajectory is TR_{iq} and $X_{b_{iq}} = x_{bi} - x_{bq}$ and $Y_{b_{iq}} = y_{bi} - y_{bq}$ denote the coordinates of the expected location at t_b. Then, the distance of TR_{iq} from the origin, as a function of the time is $d_{iq}(t) = \sqrt{At^2 + Bt + C}$, where: $A = V_{x_{iq}}^2 + V_{y_{iq}}^2$,
$B = -2(V_{x_{iq}}^2 t_b + V_{x_{iq}} X_{b_{iq}} + V_{y_{iq}}^2 t_b + V_{y_{iq}} Y_{b_{iq}})$ and
$C = 2X_{b_{iq}} V_{x_{iq}} t_b + V_{x_{iq}}^2 t_b^2 + X_{b_{iq}}^2 + 2Y_{b_{iq}} V_{y_{iq}} t_b + V_{y_{iq}}^2 t_b^2 + Y_{b_{iq}}^2$.
Since $A \geq 0$, the function $d_{iq}(t)$ is a *hyperbola* and, based on the underlying parabola (under the square root), it attains a *minimum* at $t_m = -B/2A$ (if $t_m \notin [t_b, t_e]$, the hyperbola is strictly monotonic).

Given a collection of such distance functions (one for each moving object, except the querying one), based on the observations in Section 3.1, we know that at any time instant t, the ranking of the probabilities of a given object Tr_j^u being a nearest neighbor to Tr_q^u is the same as the ranking of the distance functions $d_{iq}(t)$. Hence, the problem of constructing the IPAC-NN tree—determining the member-nodes of each level along with their respective time-intervals—can be reduced to the problem of finding the *collection of (ranked) lower envelopes* for the set of distance functions $\mathcal{S}_{DF} = \{d_{1q}(t), d_{2q}(t), \ldots, d_{Nq}(t)\}$ between t_b and t_e. We now focus on describing how to construct the lower envelope of $\mathcal{S}_{DF}$.

We observe that two different distance functions, e.g., $d_{iq}(t)$ and $d_{jq}(t)$, in general, can intersect in *at most* two points[2]—consequently, they can have 0, 1 or 2 intersections throughout $[t_b, t_e]$. Their intersections (if any) can be obtained by setting $d_{iq}(t) = d_{jq}(t)$ which, after squaring both sides, amounts to solving a quadratic equation and checking whether each of the solutions (if any) is in $[t_b, t_e]$. Parts (a) and (b) in Figure 9 illustrate two cases in which pairs of distance functions (corresponding to pairs of *TR*-like trajectories) intersect in two points and one point, respectively. We call such intersection points *critical time-points*. To determine how each of the input-hyperbolae contributes to the lower envelope, it suffices to compare the corresponding distance functions in a single time value t_{in} anywhere in between two consecutive critical time-points. In the sequel, without loss of generality, we assume an existence of a function called $Env2(TR_{iq}, TR_{jq}, t_1, t_2)$ that takes two difference-trajectories as input and returns their lower envelope as output, along with the critical times, between times t_1 and t_2. Clearly, $Env2(TR_{iq}, TR_{jq}, t_1, t_2)$ runs in $O(1)$ time.

EXAMPLE 5. *In the example of Figure 9.a, the output of function* $Env2(TR_1, TR_2, t_b, t_e)$ *is the lower envelope:*
$LE_{1,2} = [(TR_2, [t_b, t_{11}]), (TR_1, [t_{11}, t_{12}]), (TR_2, [t_{12}, t_e])]$.
On the other hand, in the settings of Figure 9.b, function $Env2(TR_3, TR_4, t_b, t_e)$ *yields:*
$LE_{3,4} = [(TR_4, [t_b, t_{31}]), (TR_3, [t_{31}, t_e])]$.

Now, the main question is how to efficiently construct the lower envelope of the whole collection of distance-trajectories (i.e., the set $\mathcal{S}_{DF}$ of their distance functions to Tr_q). The problem of efficiently constructing a lower envelope has already been addressed in the literature [6, 29]. For our settings, we have implemented a divide-and-conquer method in the spirit of *MergeSort* and we have used it in our experiments (see [36] for details). Algorithm 1 constructs the lower envelope for a set of distance-trajectories $\mathcal{S}_{TR} = \{TR_1, TR_2, \ldots, TR_N\}$ (i.e., their distance functions $\mathcal{S}_{DF} = \{d_{1q}(t), d_{2q}(t), \ldots, d_{Nq}(t)\}$), assuming an additional base case specifying that the output of LE_Alg($\mathcal{S}_{TR}$,i,i,t_b,t_e)) is $[(TR_i, [t_b, t_e])]$.

Algorithm 1 uses method *Merge_LE* to merge two lower envelopes with a technique similar to the traditional merge

[2]In their intervals of strict monotonicity, they can have at most one intersection.

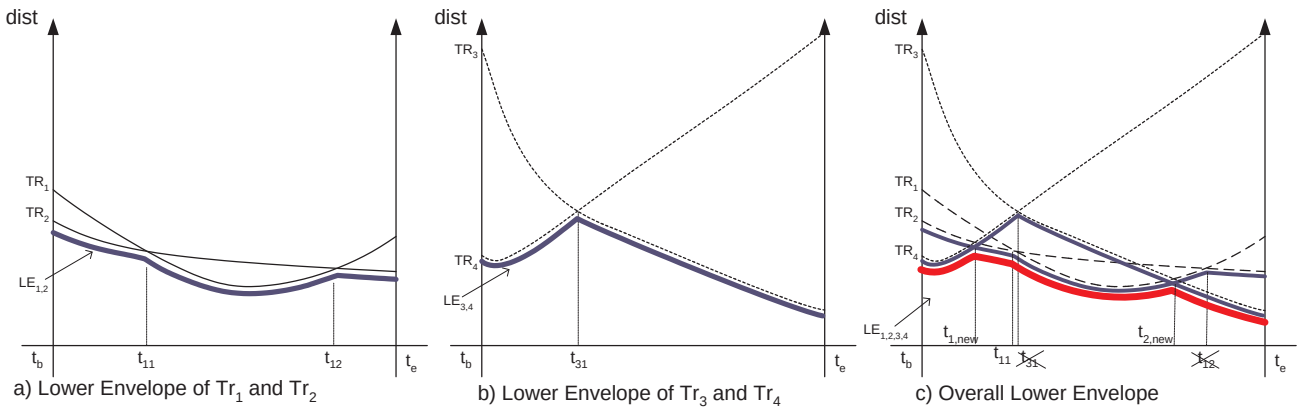

Figure 9: Constructing the lower envelope.

Algorithm 1 Construction of the lower envelope for a set of distance-trajectories

LE_Alg($\mathcal{S}_{TR}$,1,N,t_b, t_e)
Input: set $\mathcal{S}_{TR} = \{TR_1, TR_2, \ldots, TR_N\}$ of distance-trajectories and query interval $[t_b, t_e]$
Output: lower envelope of $\mathcal{S}_{TR}$

Let $C = \lceil N/2 \rceil$
Merge_LE((LE_Alg($\mathcal{S}_{TR}$,1,C,t_b, t_e),LE_Alg($\mathcal{S}_{TR}$,C,N,t_b, t_e))

sort algorithm. Method *Merge_LE incrementally sweeps* over the critical time-points of each input lower envelope maintaining the output lower envelope E computed so far, along with the values of the *current lower bound* and *current upper bound* from among the critical times of the inputs. Function *Env2* is used to compute the next fragment F to append to E. Note that we cannot simply concatenate E and F. Instead, if the first fragment of F is defined by the same TR_j that terminates E, two consecutive time intervals have to be merged into one. In other words, appending $[(TR_j, [t_{j2}, t_{j3}])]$ to $[(TR_j, [t_{j1}, t_{j2}])]$ yields $[(TR_j, [t_{j1}, t_{j3}])]$.

Due to the properties of the Davenport-Schinzel sequences, the combinatorial complexity of the lower envelope is $\lambda_2(N) = 2N - 1 = O(N)$ since two hyperbolae can intersect in at most two points [29]. The time complexity of merging two lower envelopes is linear in the size of the sum of its inputs which, in turn, implies that the time complexity of Algorithm 1 is specified by the recurrence: $T(2N) = 2T(N) + 2N$. Hence, the complexity of constructing the lower envelope is $O(N \log N)$. We illustrate the above concepts in Figure 9.

One of the benefits of constructing the lower envelope is that it provides a *continuous-pruning* criteria. Namely, the trajectories whose distance functions do not intersect the region bounded by the lower envelope and its vertically-translated copy for a vector of length $4r$ in the *(distance, time)* space, can never have a non-zero probability of being a nearest neighbor to Tr_q^u. The reason for this is that at any time instant, in order for any (after convolution) object to have a non-zero probability of being a nearest neighbor to (0,0), its nearest location (which is $2r$ closer than the centroid of its convolution) must be no further than $2r$ from the ring centered at the nearest neighbor to (0,0) at that time, and with width $2r$. As an example, in Figure 10, TR_7 can be safely pruned from any consideration, because its distance from the lower envelope at any time instant is greater than $4r$.

Algorithm 2 constructs the IPAC-NN tree, which can be used for answering ranking-based continuous probabilistic NN queries for uncertain trajectories.

Algorithm 2 Construction of the IPAC-NN tree

Tree_IPAC-NN($\mathcal{T}$, Tr_q, $[t_b, t_e]$)
Input: *A collection of trajectories $\mathcal{T}$; a querying trajectory $Tr_q \in \mathcal{T}$, and a time-interval $[t_b, t_e]$*
Output: *The IPAC-NN tree for the continuous probabilistic NN-query.*

1. Construct the lower envelope using Algorithm 1. The lower envelope corresponds to the nodes in Level_1 of the IPAC-NN tree;
2. Prune all the objects that cannot have a non-zero probability of being a nearest neighbor;
3. **for** *each level L*
4. **for** *each time-interval bounded by a pair of consecutive critical time-points t_i and t_{i+1} on the level $L-1$ envelope*
5. Remove from consideration TR_i^{L-1} defining the envelope at level $L-1$ in (t_i, t_{i+1})
6. Construct the portion of the lower-envelope at level L applying Algorithm 1.
7. **end_for**
8. **end_for**

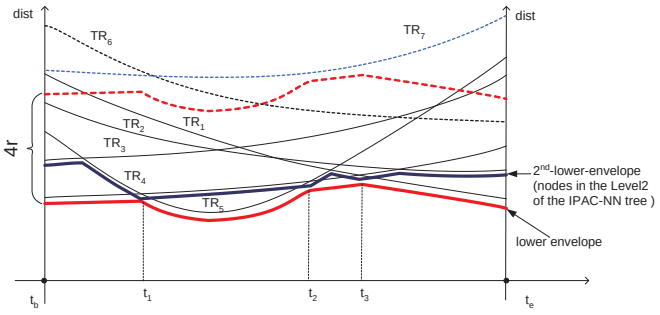

Figure 10: Envelopes and IPAC-NN tree.

The combinatorial complexity of the lower envelope is $O(N)$ and its construction takes $O(N \log N)$ time. The pruning phase has $O(N^2)$ time complexity. Assuming that after the pruning there are $\lceil N/K \rceil$ objects left for consideration, the running time for constructing the 2nd-lower-envelope (equivalently, the Level_2 nodes of the IPAC-NN tree) is $O(N\lceil N/K \rceil \log \lceil N/K \rceil)$. Since two distance function (hyperbolae) can intersect at most twice, we observe that the total number of intersection points within the zone bounded by the lower envelope and it translation for $4r$ in the *(distance,time)* space is $O(\lceil N/K \rceil^2)$, which is the upper bound on the complexity of (i.e., the number of nodes in) the IPAC-NN tree. Figure 10 illustrates the first two levels of lower envelopes for a given set of (distance functions of) uncertain trajectories. We summarize the results of this section with the following theorem:

THEOREM 2. *The graph of all the envelopes in the (distance, time) space that intersect the zone bounded by the lower envelope and its copy vertically translated by $4r$ between times t_b and t_e is the dual of the DAG obtained by removing the root of the IPAC-NN tree corresponding to a given continuous probabilistic NN-query between t_b and t_e. The combinatorial complexity of this graph is $O(\lceil N/K \rceil^2)$, which is the combinatorial complexity of the IPAC-NN tree.*

We conclude this section with an observation regarding our complexity results: the derivations the we presented assume that all the trajectories have one single segment. However, in the case where each trajectory has m segments throughout the time-interval of interest for the query, each of the bounds needs to be multiplied by a corresponding factor of m.

4. VARIATIONS OF THE NN-QUERY

One of the benefits of the IPAC-NN tree structure is that it provides a foundation for extending the capabilities of MOD in terms of processing continuous probabilistic NN queries. For example, one can define predicates that will enable users to pose a query like the following one:

SELECT T FROM MOD
WHERE ProbabilityNN$(T, Tr_Q, Time) > 0$ AND $Time$ IN $[t1, t2]$

In the rest of this section, we identify four categories of syntactic variants of probabilistic continuous NN-queries that can be answered using an IPAC-NN tree and we outline the algorithms for their processing. Due to space limitations, we do not provide a formal description of the set of predicates expressing them, however, we note that similar formalizations have been presented [37], albeit for range queries for uncertain trajectories. The corollaries in this section express the complexity results for the queries and follow directly from the results in Section 3.

4.1 Category 1 queries

The following queries pertain to verifying the properties of a single trajectory.

- $UQ_{11}(\exists t)$: *"Does Tr_i^u have a non-zero probability of being a NN to Tr_q^u at some time during $[t_b, t_e]$?"*
- $UQ_{12}(\forall t)$: *"Does Tr_i^u have a non-zero probability of being a NN to Tr_q^u all throughout $[t_b, t_e]$?"*
- $UQ_{13}(X\%$ of $[t_b, t_e])$: *"Does Tr_i^u have a non-zero probability of being a NN to Tr_q^u, at least X% of the time in $[t_b, t_e]$?"*

Answering UQ_{11} amounts to checking whether (the distance function of) TR_i is inside or intersects the boundaries of the zone between the lower envelope (Level_1 of the IPAC-NN tree) and its $4r$-translated copy at some $t \in [t_b, t_e]$. The processing of UQ_{12} requires checking that: (1) TR_i is inside the pruning-zone at t_b; and (2) it stays inside it until t_e, i.e., it does *not* intersect the envelope and its $4r$-translated copy, determining the boundaries of the pruning zone. In addition to checking for all the intersections, UQ_{13} needs an additional "accumulator" variable to sum up the time-intervals during which TR_i is inside the pruning zone.

COROLLARY 1. *Processing a Category 1 query takes $O(N)$ time after $O(N \log N)$ pre-processing time.*

4.2 Category 2 queries

These queries extend Category 1 queries with another parameter, k, for the purpose of *ranking* a particular trajectory.

- $UQ_{21}([(\exists t), k])$: *"Does Tr_i^u have a non-zero probability of being a k^{th} highest-probability NN of Tr_q^u at any time in $[t_b, t_e]$?"*
- $UQ_{22}([(\forall t), k])$: *"Does Tr_i^u have a non-zero probability of being a k^{th} highest-probability NN to Tr_q^u all throughout $[t_b, t_e]$?"*
- $UQ_{23}(X\%$ of $[t_b, t_e]$,k): *"Does Tr_i^u have a non-zero probability of being a k^{th} highest-probability NN to Tr_q^u, at least X% of the time in $[t_b, t_e]$?"*

To answer UQ_{21}, we check whether the IPAC-NN tree has a node at at Level_i $i \leq k$ that has Tr_i^u as its label-attribute. Equivalently, we check whether TR_i intersects the Level_i ($i \leq k$) lower envelope. UQ_{22} can be processed by checking whether TR_i is at the Level_i ($i \leq k$) lower envelope at t_b and maintains that property until t_e. Similarly to UQ_{13}, the processing of UQ_{23}, in addition to checking whether TR_i is initially at the Level_i ($i \leq k$) lower envelope, uses an "accumulator" variable to sum up the time-intervals during which TR_i maintains the desired property. Since at every Level_j the total combinatorial complexity of the lower envelope is bounded by $O(N)$, we have:

COROLLARY 2. *Processing a Category 2 query takes $O(kN)$ time after $O(\lceil N/K \rceil^2)$ pre-processing time.*

The next two categories of continuous probabilistic NN queries are extensions of Category 1 and Category 2 over the whole space of the uncertain trajectories. In practice, one would do an initial filtering based on some index-structure.

4.3 Category 3 queries

- $UQ_{31}(\exists t)$: *"Retrieve all the trajectories that have a non-zero probability of being NN to Tr_q^u some time during $[t_b, t_e]$."*
- $Q_{32}(\forall t)$: *"Retrieve all the trajectories that have a non-zero probability of being NN to Tr_q^u throughout the entire $[t_b, t_e]$."*
- $UQ_{33}(X\%$ of $[t_b, t_e])$: *"Retrieve all the trajectories that have a non-zero probability of being NN to Tr_q^u at least X% of the entire $[t_b, t_e]$.".*

Answering UQ_{31} essentially amounts to constructing the IPAC-NN tree. In addition to constructing the IPAC-NN tree (i.e., the collection of lower envelopes), the processing of UQ_{32} requires checking which TR_i intersects the $4r$-translation of the lowest (Level_1) lower envelope—an overhead of $O(N)$ time. Finally, UQ_{33} uses "accumulator" variables that measure the portion of time that each trajectory intersecting the $4r$-translation of the lowest (Level_1) lower envelope has spent outside of it.

COROLLARY 3. *Processing a Category 3 query takes* $O(\lceil N/K \rceil^2)$ *time.*

4.4 Category 4 queries

The last category of queries that we consider extends Category 3 in the same way as queries from Category 2 extend queries from Category 1 by adding the value of k as a ranking parameter in terms of the k-th highest NN probability. Due to space limitations, we do not formally present these queries here. However, we note that the complexity of their processing introduces an additional factor of k in the $O(\lceil N/K \rceil^2)$ complexity (Corollary 3).

We conclude this section with the observation that another variant of UQ_{11} and UQ_{21} and UQ_{31} considers a *fixed* time instant (i.e., $t = t_f$) and evaluates the properties at *that* time. The corresponding running times are the same as the ones expressed in the above corollaries.

5. EXPERIMENTS

In this section, we describe preliminary experiments on our proposed methodology. Our experiments are implemented in C++ on a Pentium IV 3.60GHZ, 1G MB memory, and Windows XP platform. For our experiments, we considered a geographic area of size 40×40 miles^2. The moving objects were generated using a modified version of the random way-point model, and each object starts at a randomly selected position in the region of interest. Subsequently, the object picks a random direction and moves at a speed randomly distributed between 15 mph and 60 mph. For simplicity, we assumed that all the objects change their velocity vectors synchronously, once every 3 minutes, in a manner that: (a) the new direction of the motion is within a randomly chosen value within +/- 60° from the current one; (b) the new speed is, once again, randomly selected from the $[15, 60]$ mph interval. The total duration of the motion is fixed to 60 min. In all the figures, the running time (seconds) is shown in a logarithmic scale.

In the first group of experiments, we investigate the efficiency of computing the lower envelope of the distance functions, by comparing our approach (Algorithm 1) against the naive approach, which finds the intersection of all the distance functions, sorts them in time, then sweeps in time comparing the lowest values between intersections ($O(N^2 \log N)$ time since there are $O(N^2)$ such intersections). We varied the number of moving objects from 1000 to 12000 and measured the running time of each approach. The results are plotted in Figure 11. As expected, based on the theoretical analysis, our approach is much faster, with orders of magnitude speed-up.

Next, we evaluated the efficiency of using our computed lower envelope to answer UQ_{11} and UQ_{13} (Section 4), where we set the value of $X = 50\%$ for UQ_{13}. We compared our approach with the naive approach, which checks all pairwise intersection times of the distance functions. Again, the total number of objects was between 1000 and 12000 and we randomly selected an object for the evaluation. The averaged results of 100 such selections are illustrated in Figure 12, which shows that the lower envelope yields significant speedup(s).

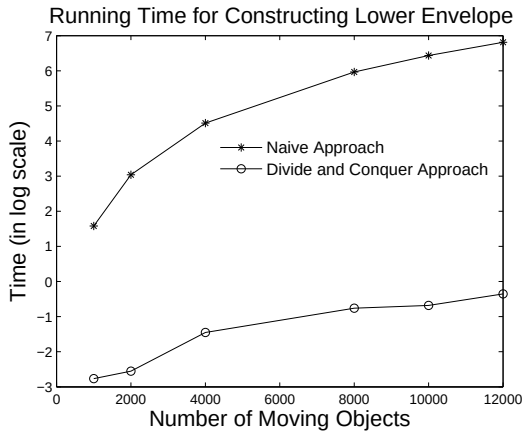

Figure 11: Construction of the lower envelope.

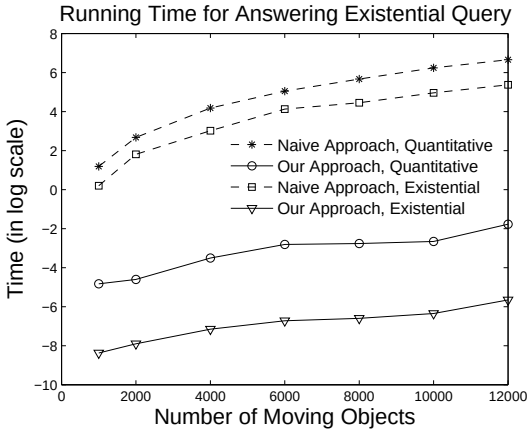

Figure 12: Existential queries.

Finally, we evaluated the pruning power of the lower envelope as a function of the uncertainty radius. We varied the radius of uncertainty for the moving objects from 0.1 mile to 2 miles, and measured the ratio when fixing the total number of moving objects to $2,000$ and 10,000, respectively. The result is shown in Figure 13. It can be observed that when the moving objects have an uncertainty radius of 0.5 mile, over 90% of the objects can be pruned from any consideration, based on the lower envelope. When the radius increased to 1 mile, about 85% of the objects can be pruned. An implication of this observation is that when the actual evaluation of the probabilities is needed, only about 15% of the objects will contribute to an uncertainty radius of 1 mile.

6. RELATED WORK

Nearest neighbor queries are essential operations in a wide variety of application domains, from machine learning and

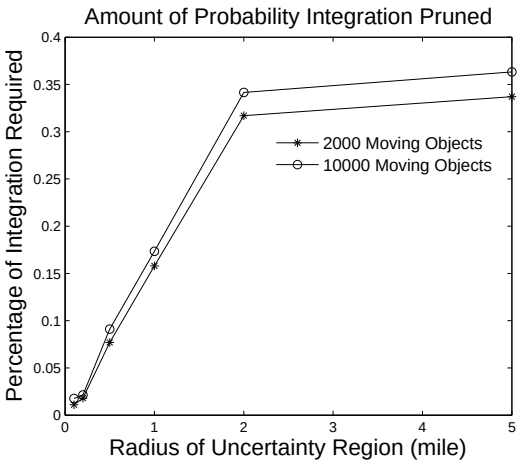

Figure 13: Pruning power of the lower envelope.

computer vision [28] to classification and clustering in data mining [32]. Voronoi diagrams, extensively studied in computational geometry [1, 6], provide a tool for finding the nearest neighbor of a query point among N static points in $O(\log N)$ query time for 2D. For spatial databases, the problem of efficient scalable processing of (k)NN-queries has been addressed in [24] with a branch-and-bound approach and in [10] with an incremental technique, both relying on R-trees for indexing.

In recent years, there have been many interesting results on (k)NN-queries in spatio-temporal settings. In [15], a dual transformation (points to lines) is explored for developing efficient algorithms when the objects are moving in one dimension. Generic methodologies for processing spatio-temporal queries for trajectories, based on a rich algebra of types, are presented in [16]. The generation of the time-parameterized answer to the continuous variant of the NN-queries, along with other traditional spatial queries, in spatio-temporal settings, and the efficient scalable processing of such queries based on TPR-trees was presented in [33, 34].

When the motion of the objects is represented as a stream of (location, time) updates, the main issue is how to efficiently monitor and update the answer to (k)NN queries, for which scalable techniques have been proposed in [39, 40]. On the other hand, when the motion of the object is expressed by (location, time, velocity) updates, an incremental approach for processing (k)NN-queries is presented in [13]. In addition, some papers have focused on efficient processing of such queries on road networks [19, 27].

Two works that are very similar in spirit to ours are [2, 23]. Both of them consider the collection of hyperbolae representing the distance functions from a querying object. However, [23] focuses on processing a (k)NN-query but, unlike our approach, does not use the construction of the lower envelope for the purpose of pruning objects that have zero probability of being nearest neighbor to the querying object within a given time interval. The main goal of [2], on the other hand, is scalable processing of regular and reverse NN-queries, focusing on efficient management of modifications (insertions/deletions) and, once again, the uncertainty is not formally addressed.

Various models of uncertainty in spatio-temporal settings have been considered in the literature. As we mentioned, [22] considers the uncertainty for the *(location, time)* updates model and demonstrates that, under constraint maximal velocity, the spatial zone of the object's whereabouts is an ellipse. The 3D interpretation of that same model ("beads") was presented in [11]. However, the processing of continuous NN-queries under the uncertainty model was not considered. The uncertainty model that we consider in this work has been used for processing range queries in MOD settings [37], where various semantic categories of the (answers to the) queries were presented and geometric concepts were used for their efficient processing. In this paper, we rely on the results in [5] for processing instantaneous NN-queries in uncertain environments and, in a sense, this work provides a continuous extension of it, due to the properties of the convolution for the sum of independent variables.

A recent work addressing a problem similar to the one tackled in this paper is [12], where the goal is to present efficient algorithms for processing continuous kNN-query, for objects moving on road network with uncertain velocity. Inversely to our results, the work in [12] focuses on finding the upper envelope of the set of distance functions, guaranteeing that a certain object may be one of the k nearest neighbors. However, although there is no formal analysis of the complexity presented, it appears that the construction of the upper envelope takes quadratic time.

7. CONCLUSIONS AND FUTURE WORK

We have addressed the problem of continuous NN queries for uncertain trajectories of moving objects, where the uncertainty at any time instant is bounded by a circle with a fixed radius. We have demonstrated that our approach is applicable to a large class of location *pdf*s—those that are rotationally symmetric. For these settings, we have provided a compact structure, the *IPAC-NN* tree, to represent the answer to such queries and we have given algorithmic solution for constructing the geometric dual of this structure. In addition, we have identified several syntactic variants for the continuous probabilistic NN-queries and demonstrated how they can be efficiently answered.

There are several challenges that we plan to address in the future. One of them is to identify the basic properties of the *descriptors* of the probability values in the *IPAC-NN* trees which, in turn, will enable processing of *continuous threshold* NN-queries (e.g., retrieve the objects that have more than 65% probability of being a nearest neighbor within 50% of the time) [4]. Another interesting problem is to design data structures that provide for scalable processing of such uncertain queries, in a spirit similar to that of U-trees [35]. In addition, we are planning to address other variants of continuous probabilistic NN queries (e.g., all pairs, reverse) and compare the semantics of traditional *Top-k* NN queries for crisp trajectories with that for uncertain trajectories (cf. [2, 23, 30]). Finally, we plan to allow for different uncertainty zones of the object locations (i.e., circles with different radii), for which a promising foundation is the Voronoi diagram of moving disks [14].

8. ACKNOWLEDGMENTS

The authors wish to thank Abraham Haddad, Ajit Tamhane, and the anonymous reviewers for their constructive comments.

9. REFERENCES

[1] F. Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3), 1991.

[2] R. Benetis, C. Jensen, G. Karciauskas, and S. Saltenis. Nearest and reverse nearest neighbor queries for moving objects. *VLDB Journal*, 15(3):229–249, 2006.

[3] H. Cao, O. Wolfson, and G. Trajcevski. Spatio-temporal data reduction with deterministic error bounds. *VLDB Journal*, 15(3), 2006.

[4] R. Cheng, J. Chen, M. F. Mokbel, and C.-Y. Chow. Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data. In *ICDE*, 2008.

[5] R. Cheng, D. Kalashnikov, and S. Prabhakar. Querying imprecise data in moving objects environments. *IEEE Transactions on Knowledge and Data Engineering*, 16(9), 2003.

[6] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, 2001.

[7] B. V. Gnedenko. *Course of Probability Theory*. Nauka, 1988.

[8] R. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann, 2005.

[9] M. Hadjieleftheriou, G. Kollios, V. J. Tsotras, and D. Gunopulos. Efficient indexing of spatiotemporal objects. In *EDBT*, 2002.

[10] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM Transactions on Database Systems*, 24(2):265–318, 1999.

[11] K. Hornsby and M. Egenhofer. Modeling moving objects over multiple granularities. *Ann. Math. Artif. Intell.*, 36(1-2):177–194, 2002.

[12] Y.-K. Huang, C.-C. Chen, and C. Lee. Continuous k-nearest neighbor query for moving objects with uncertain velocity. *GeoInformatica*, 2007. DOI 10.1007/s10707-007-0041-0.

[13] G. Iwerks, H. Samet, and K. Smith. Maintenance of K-nn and spatial join queries on continuously moving points. *ACM Transactions on Database Systems*, 31(2), 2006.

[14] M. Karavelas. Voronoi diagrams for moving disks and applications. In *WADS*, pages 62–74, 2001.

[15] G. Kollios, D. Gunopulos, and V. Tsotras. Nearest neighbor queries in a mobile environment. In *Spatio-Temporal Database Management*, pages 119–134, 1999.

[16] J. Lema, L. Forlizzi, R. Güting, E. Nardelli, and M. Schneider. Algorithms for moving objects databases. *Computing Journal*, 46(6), 2003.

[17] J. S. Lim. *Two-Dimensional Signal and Image Processing*. Prentice Hall, 1990.

[18] M. Mokbel, X. Xiong, and W. Aref. SINA: Scalable incremental processing of continuous queries in spatio-temporal databases. In *ACM SIGMOD*, 2004.

[19] K. Mouratidis, M. Yiu, D. Papadias, and N. Mamoulis. Continuous nearest neighbor monitoring in road networks. In *VLDB*, pages 43–54, 2006.

[20] P. Olofsson. *Probability, Statistics and Stochastic Processes*. Wiley-Interscience, 2005.

[21] J. Pei, M. Hua, Y. Tao, and X. Lin. Query answering techniques on uncertain and probabilistic data: tutorial summary. In *ACM SIGMOD*, 2008.

[22] D. Pfoser and C. Jensen. Capturing the uncertainty of moving objects representation. In *SSD*, 1999.

[23] K. Raptopoulou, A. Papadopoulos, and Y. Manolopoulos. Fast nearest-neighbor query processing in moving-object databases. *GeoInformatica*, 7(2):113–137, 2003.

[24] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *ACM SIGMOD*, pages 71–79, 1995.

[25] H. Royden. *Real Analysis*. Macmillan Co., 1963.

[26] J. Schiller and A. Voisard. *Location-based Services*. Morgan Kaufmann Publishers, 2004.

[27] C. Shahabi, M. Kolahdouzan, and M. Sharifzadeh. A road network embedding technique for k-nearest neighbor search in moving object databases. *GeoInformatica*, 7(3):255–273, 2003.

[28] G. Shakharovich, T. Darrel, and P. Indyk, editors. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. MIT Press, 2006.

[29] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, 1995.

[30] M. Soliman, I. Ilyas, and K.-C. Chang. Top-k query processing in uncertain databases. In *ICDE*, 2007.

[31] D. Suciu and N. Dalvi. Foundations of probabilistic answers to queries. In *ACM SIGMOD*, 2005. Tutorial.

[32] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[33] Y. Tao and D. Papadias. Spatial queries in dynamic environments. *ACM Transactions on Database Systems*, 28(2), 2003.

[34] Y. Tao, D. Papadias, and Q. Shen. Continuous nearest neighbor search. In *VLDB*, 2002.

[35] Y. Tao, X. Xiao, and R. Cheng. Range search on multidimensional uncertain data. *ACM Transactions on Database Systems*, 32(3), 2007.

[36] G. Trajcevski, R. Tamassia, H. Ding, P. Scheuermann, and I. F. Cruz. Moving convolutions and continuous probabilistic nearest-neighbor queries for uncertain trajectories. Technical Report NWU-EECS-08-12, Northwestern University, Department of EECS, 2008. http://www.eecs.northwestern.edu/docs/techreports/.

[37] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. Managing uncertainty in moving objects databases. *ACM Transactions on Database Systems*, 29(3), 2004.

[38] O. Wolfson, A. P. Sistla, S. Chamberlain, and Y. Yesha. Updating and querying databases that track mobile units. *Distributed and Parallel Databases*, 7, 1999.

[39] X. Xiong, M. Mokbel, and W. Aref. SEA-CNN: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In *ICDE*, pages 643–654, 2005.

[40] X. Yu, K. Pu, and N. Koudas. Monitoring k-nearest neighbor queries over moving objects. In *ICDE*, pages 631–642, 2005.