# Technical Report

# NWU-EECS-11-06

Ranking continuous nearest neighbors for uncertain trajectories:
Full and Peer Reviewed Accepted Version


## Authors

Goce Trajcevski, Roberto Tamassia, Isabel F. Cruz, Peter Scheuermann,
David Hartglass and Christopher Zaimerowski



## Abstract

This article addresses the problem of performing Nearest Neighbor (NN) queries in a Moving Objects Database (MOD) when the trajectories of the objects are uncertain. The answer to an NN query for certain trajectories is time parameterized due to the continuous nature of the motion. However, for uncertain trajectories, even in a single time instant there may be several objects that have a non-zero probability of being a nearest neighbor to a given object. This fact affects the semantics of the answer to an NN query---an effect that is further amplified when the query spans over a time interval.

We capture the impact that the uncertainty of the trajectories has on the semantics of the answer to continuous NN queries and we propose a tree structure for representing the answers, along with efficient algorithms to compute them.

We also address the issue of performing NN queries when the motion of the objects is restricted to road networks. Finally, we formally define and show how to efficiently execute several variants of continuous NN queries. Our prototype implementation and experiments demonstrate that the proposed algorithms yield significant performance improvements when compared with the corresponding naive approaches.

*This is a full version of the article that was peer-reviewed and accepted for publication in the VLDB Journal, special issue on Data Management for Mobile Applications, 2011. It includes the final modifications based on the reviewers' comments accompanying the acceptance note. However, upon finalizing the camera-ready copy, we were told that the space limit is 25 pages, and we had to reduce some material, which we are making available here.*



**Keywords:** Nearest Neighbor, Uncertainty, Moving Objects Database

**VLDB Journal manuscript No.**
(will be inserted by the editor)

# Ranking Continuous Nearest Neighbors for Uncertain Trajectories: Full and Peer Reviewed Accepted Version

**Goce Trajcevski · Roberto Tamassia · Isabel F. Cruz · Peter Scheuermann · David Hartglass · Christopher Zamierowski**

Received: date / Accepted: date

**Abstract** This article addresses the problem of performing Nearest Neighbor (NN) queries in a Moving Objects Database (MOD) when the trajectories of the objects are *uncertain*. The answer to an NN query for certain trajectories is time parameterized due to the continuous nature of the motion. However, for uncertain trajectories, even in a single time instant there may be several objects that have a non-zero probability of being a nearest neighbor to a given object. This fact affects the semantics of the answer to an NN query—an effect that is further amplified when the query spans over a time interval. We capture the impact that the uncertainty of the trajectories has on the semantics of the answer to continuous NN queries and we propose a tree structure for representing the answers, along with efficient algorithms to compute them. We also address the issue of performing NN queries when the motion of the objects is restricted to road networks. Finally, we formally define and show how to efficiently execute several variants of continuous NN queries. Our prototype implementation and experiments demonstrate that the proposed algorithms yield significant performance improvements when compared with the corresponding naïve approaches.

**Keywords** Nearest Neighbor · Uncertainty · Moving Objects Database

Research supported in part by NSF awards CCF–0830149, CNS–0910952, IIS–0513553, and IIS-0812258, and by the Center for Geometric Computing at Brown University.

G. Trajcevski, P. Scheuermann, D. Hartglass, C. Zamierowski
Department of Electrical Engineering and Computer Science
Northwestern University
Phone: +1-847-491-7069
Fax: +1-847-491-4455
E-mail: goce,peters,david-h,chris-z@eecs.northwestern.edu

R. Tamassia
Department of Computer Science
Brown University
Providence, RI 02912
E-mail: rt@cs.brown.edu

I. F. Cruz
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60612
E-mail: ifc@cs.uic.edu

## 1 Introduction

A broad set of applications including traffic management, emergency response, disaster remediation, context-aware tourist information, and battlefield management, require some type of *location-based services* (*LBS*) [60]. Such services rely on the whereabouts in time and space of the participating mobile entities and require fundamental techniques in the field of *Moving Object Databases* (*MOD*) [23], which include efficient storage/retrieval of (*location*, *time*) information, along with the efficient processing of various spatio-temporal queries of interest, e.g., range, density, variants of nearest neighbor (reverse, surface, aggregate), and skyline [6, 14, 18, 22, 33, 35, 45, 46, 78, 80].

As previously noted in the literature [77], 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., [3, 50, 64, 65]), including spatial and spatio-temporal settings (e.g., [9, 53, 54, 70, 73]).

Consider the following application scenarios:

1. To balance the promptness of emergency response with the safety of the officers involved, the command center may want to orchestrate the traffic light sequence so that more than one patrol car can simultaneously arrive at the site of an incident. This objective is complicated by the fact that there are city areas where location determination cannot be performed accurately. Hence, the associated uncertainty needs to be taken into account when determining the routes and planning the traffic light sequence.
2. In environmental studies that investigate the correlation of migrations among species, tracking data that represent the motion of a species is typically generated by a GPS-enabled beacon device attached to only a subset of the animals. Therefore the study of the collective behavior of the motion of a herd or flock needs to incorporate imprecision in the "average trajectory" as estimated from that subset.
3. In studies related to climate and weather data, one may be interested in the spatio-temporal proximity among hurricanes. However, typically their tracking (via satellite) has a rather uncertain location-in-time information of the "eye" of a hurricane and of the overall size of the affected region.

Motivated by these applications, we focus on variations of *nearest neighbor queries*, for brevity referred to as *NN-queries*, which incorporate the *u*ncertainty of the moving objects locations.

Contrary to what happens in pure spatial settings [29,57], the answer to a *continuous* NN-query in a spatio-temporal setting is *time parameterized* [67,68] in the sense that the actual nearest neighbor of a given object need not be the same throughout the time interval of interest. However, when uncertainty is incorporated in the trajectories' model in MOD, the situation becomes significantly more complicated, as illustrated in the following motivational example.

*Example 1* (*Motivational Example*) Consider a MOD with the following four trajectory-segments
$\boldsymbol{Tr}_1 = \{(120, 60, 10), (220, 300, 20)\}$
$\boldsymbol{Tr}_2 = \{(310, 100, 10), (190, 260, 20)\}$
$\boldsymbol{Tr}_3 = \{(150, 100, 10), (30, 260, 20)\}$
$\boldsymbol{Tr}_4 = \{(370, 570, 10), (270, 330, 20)\}$
where the values for the $(x, y, t)$ coordinates are chosen in some reference 2D-spatial plus 1D-temporal coordinate system expressed for example in meters for the spatial coordinates and in seconds for the temporal coordinate. The following query is posed to the MOD:

*Q_NN*: *Retrieve the nearest neighbor of the trajectory* $\boldsymbol{Tr}_1$ *between* $t_1 = 10$ *and* $t_2 = 20$.

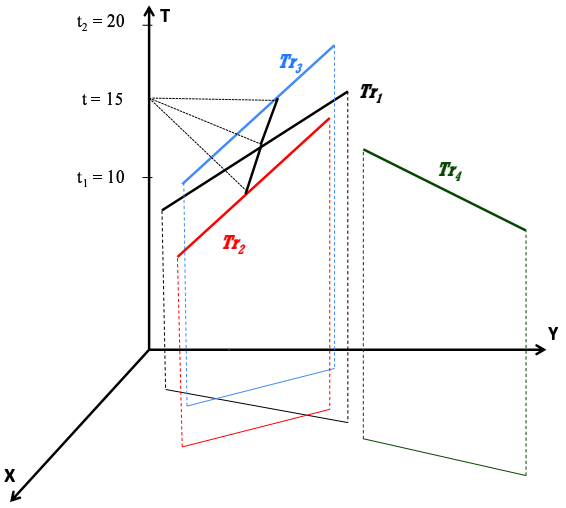


**Fig. 1** Continuous NN Query for Crisp Trajectories

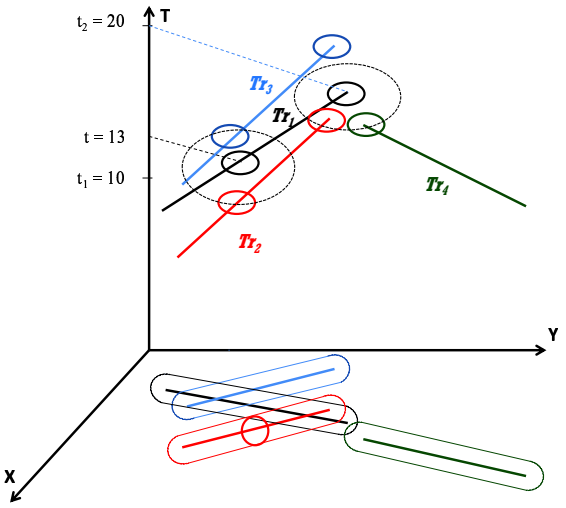


**Fig. 2** Uncertainty Impact for Continuous NN Queries

This scenario is depicted in Figure 1. Using algorithms available in the literature (e.g., [48, 67, 68]), the answer $\mathbf{A}_{Q\_NN}$ to the query is the set $\{[\boldsymbol{Tr}_3, (10, 15)], [\boldsymbol{Tr}_2, (15, 20)]\}$, meaning that during the first 5 seconds of the time interval of interest, i.e., between $t_1 = 10$ and $t = 15$, the nearest neighbor of $\boldsymbol{Tr}_1$ is the trajectory $\boldsymbol{Tr}_3$ and during the last 5 seconds, i.e., between $t = 15$ and $t_2 = 20$, the nearest neighbor of $\boldsymbol{Tr}_1$ is the trajectory $\boldsymbol{Tr}_2$.

Contrary to the settings in Figure 1 where the location of the objects at each time instant are crisp (i.e., there is no uncertainty), we now consider the scenario depicted in Figure 2, where the spatial locations of the objects along their trajectories are uncertain. Specifically, at a given time instant, an object can be anywhere within a disk with a 30 meter radius, centered at its expected location for that time instant. In this

setting, at time $t = 13(< 15)$, both $\boldsymbol{Tr}_3$ and $\boldsymbol{Tr}_2$ have a non-zero probability of being the NN-trajectory to $\boldsymbol{Tr}_1$. However, that is not the case for $\boldsymbol{Tr}_4$ which, at $t = 13$, cannot possibly be the nearest neighbor to $\boldsymbol{Tr}_1$. However, at $t_2 = 20$, $\boldsymbol{Tr}_4$—which was not part of the answer $\mathbf{A}_{Q\_NN}$ when there was no uncertainty in the objects' motion—has a non-zero probability of being the NN-trajectory to $\boldsymbol{Tr}_1$, albeit smaller than the one associated with $\boldsymbol{Tr}_2$.

The main objective of this work is to provide formalisms and methodologies for efficient processing of queries like:

SELECT $Tr_i$ FROM MOD
WHERE ProbabilityNN$(Tr_i, Tr_Q, T) > 0$ AND
($T$ BETWEEN $t1$ AND $t2$ )

In this spirit, the challenges brought by the consideration of uncertainty (as illustrated in Figure 2) prompt a more detailed investigation of the following two issues:

1. *Ranking:* we need to be able to distinguish the rank of a given trajectory's probability (e.g., highest or lowest) of being a nearest neighbor to a given querying object [64] at a particular time instant.
2. *Continuity:* we need to efficiently manage the changes to the continuous ranking of the objects that qualify to be nearest neighbors (with non-zero probability) along the corresponding portions of the time interval of interest for a given NN-query.

Combining ranking and continuity affects the structure of the answer that is returned to the users of a MOD with uncertain trajectories. However, to further ease the burden of factoring out the uncertainty from the answers to the queries, the users need to be presented with a suite of syntactic constructs that will enable them to express their interests when posing the queries.

Towards these goals, the main contribution of our work consists of the following results, which can be used in the refinement stage of the overall query processing workflow:

1. We identify mathematical properties that enable the generation of a ranking of the probabilities associated with objects that have a non-zero probability of being the nearest neighbor to a query trajectory; we demonstrate that these properties are applicable to a large class of probability density functions, or *pdf*s, representing the possible location of the moving objects.
2. We formalize the declarative semantics of (the structure of) the answers to continuous spatio-temporal NN-queries for uncertain trajectories and present a compact data structure to represent the answers.
3. We present efficient algorithms for constructing the geometric dual of the proposed data structure and show how it can be used to efficiently determine those trajectories that do not belong to the query answer.
4. We address the variant when motion is restricted to road networks and identify the main aspects of the interplay between uncertainty and inter-trajectory distance.
5. We systematically incorporate uncertainty in the syntax of NN-queries and consider its impact on the corresponding answers. We present efficient processing algorithms for the different syntactic variants.
6. We evaluate experimentally the proposed algorithms and show that for several NN-query variants our results are more efficient by orders of magnitude than those provided by naïve approaches.

A preliminary version of this article was presented in [72]. In addition to the stylistic and structural improvements from the conference version, we introduce the following new contributions:

1. We extend the problem setting to incorporate uncertain NN-query processing for objects moving on road networks.
2. We present a formal treatment of the queries and of the algorithms for processing them.
3. We expand the implementation the proposed query algorithms and we present an extensive experimental evaluation.

The rest of this article is structured as follows. In Section 2 we describe necessary background. Section 3 gives a first contribution of our work: the transformation that we apply to the collection of uncertain trajectories for processing NN-queries along with the demonstration of its applicability to a large class of *pdf*s. Based on these properties, in Section 4 we focus on the processing of NN-queries for uncertain trajectories; we present the structure of the answer, along with the corresponding algorithm that its generation. In Section 5, the previous context is changed so as to consider objects that move along an existing road network. Section 6 addresses different syntactic variants of NN-queries for uncertain trajectories. In Section 7 we present our experimental results. Section 8 discusses related work. We conclude with Section 9, where we outline directions for future work.

## 2 Preliminary Background

We now introduce background material. First, we overview the uncertainty models typically used in MOD

settings and formally define the one used in this work. Next, we discuss previous results on instantaneous NN-queries for uncertain objects for the special case when the query object is *crisp* (i.e., its location is exact, without any uncertainty) [9]. We conclude this section with a brief discussion on *completeness* issues for probabilistic NN queries.

### 2.1 Modeling Uncertainty of Motion

Selecting the model for the motion plan of the moving objects affects the choice of the representation [21, 25] of trajectories in a MOD and, consequently, the overall strategy of query processing, including indexing [1, 2, 18, 27, 39, 45, 51, 59, 69, 71, 81] and pruning/refinement [10, 15, 26, 34, 36, 47, 67, 79, 82]. In addition, the choice of the motion model affects the corresponding uncertainty model that can be associated with it. The three prevalent models for uncertainty in MOD settings can be described as follows:

**M1**: The location and time data of the moving objects is obtained by receiving periodic updates of the form $(x, y, t)$, as is the case for on-board GPS systems [17, 46, 47]. In this case, nothing is known about the location of the objects between consecutive updates. The typical assumption is that motion is bounded by some maximum speed $v_{max}$. Using ideas from time-geography [28] in a MOD context (based on the definition of the geometric set of 2D points), it has been shown that the projection of the uncertain locations on the $X-Y$ plane is bounded by an ellipse with foci in two consecutive update points [53]. A subsequent spatio-temporal version of the model [30] names the volume between two update points a *bead* (also called *space-time prism* [40])and presents the first formal analysis of the properties of the model in terms of query languages. An illustration of this uncertainty model is provided in Figure 3.a.

**M2**: In some applications, each object transmits its *expected* velocity along with its current sampled location [34]. Typically, velocity information is provided for the purpose of saving on bandwidth consumption at the expense of imprecision in the MOD [17, 77]. As long as the sampled location of a given object at a particular time instant does not deviate by more than a certain (pre-defined) threshold, $D_{max}$, from its *expected* location, the object need not transmit an update to the MOD server. This policy is known as *dead-reckoning* [77]. The possible locations of an object under consecutive MOD updates are illustrated in Figure 3.b.

**M3:** Another common model of motion is the one in which the trajectories are represented as sequences of

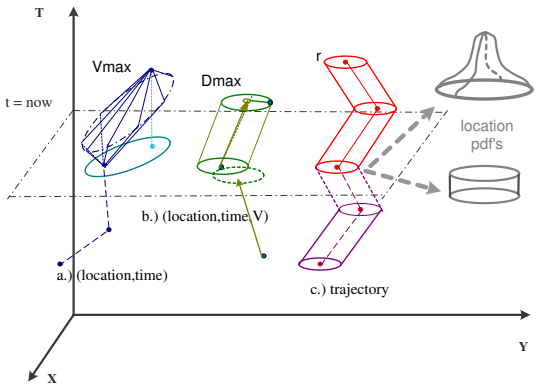


**Fig. 3** Motion models and uncertainty.

$(x, y, t)$ points and the locations between such points are obtained via linear interpolation [25, 27]. Although this model is typically used for past trajectories, it can also be used to represent future trajectories. In a typical scenario, users transmit to the MOD server: (1) the *beginning location*; (2) the *ending location*; (3) the *beginning time*; and (4) possibly a set of points to be visited. Based on the information available from electronic maps and traffic patterns, the MOD server will construct and transmit the *shortest travel time* or *shortest path trajectory* to the user. This model is applicable to the routing of commercial fleet vehicles (e.g., FedEx and UPS) as well as to web services for driving directions, where tens of millions of computations of shortest path trajectories are executed monthly by services such as MapQuest, Yahoo! Maps, and Google Maps [56]. Previous work considers an uncertainty model for processing spatio-temporal range queries that considers full trajectories and assumes that at each time instant the object's location is bounded [73]. An illustration of this model is given in Figure 3.c, which also shows how at given time instant, the *pdf* of the location of the object can be specified with different functions (e.g., uniform, bounded Gaussian).

In this work, we focus on the **M3** model and introduce the following definitions.

**Definition 1** A *trajectory* 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:

$$\boldsymbol{Tr}_i = (oid_i,\ (x_{i_1}, y_{i_1}, t_{i_1}),\ \ldots,\ (x_{i_k}, y_{i_k}, t_{i_k})),$$

where $t_{i_1} \leq t_{i_2} \leq \ldots \leq t_{i_k}$.

When clear from the context, we will interchangeably use $Tr_i$ and $oid_i$. In between two consecutive points, the objects are assumed to move along straight line segments and with constant speed, calculated as:

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

Thus, the coordinates of an object $oid_i$ at time $t \in (t_{i_{(k-1)}}, t_{i_k})$ can be obtained by linear interpolation:

$$\begin{aligned} x_i(t) &= x_{i_{(k-1)}} + v_{i_k} \cdot (t - t_{i_{(k-1)}}) \\ y_i(t) &= y_{i_{(k-1)}} + v_{i_k} \cdot (t - t_{i_{(k-1)}}) \end{aligned} \quad (2)$$

**Definition 2** An *uncertain trajectory* $\boldsymbol{Tr}_i^{\mathbf{u}}$ 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. Therefore, we have:

$$\boldsymbol{Tr}_i^{\mathbf{u}} = (oid_i,\ r,\ pdf,\ (x_{i_1}, y_{i_1}, t_{i_1}), \ldots, (x_{i_k}, y_{i_k}, t_{i_k})).$$

The center of the uncertainty disk is referred to as the *expected location* of the object and we use $D_i(t)$ to denote the uncertainty disk of $\boldsymbol{Tr}_i^{\mathbf{u}}$ at time $t$. When it comes to future trajectories generated by trip planning services, this type of location uncertainty at a given time instance indicates: (1) The acceptable location error, with respect to the planned trajectory, as measured by the on-board device; (2) 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 that is perpendicular to the 2D vector of the direction of motion. Note that the latter actually makes the time discrepancy tolerance a function of the velocity at a given time instance [7].

A specific assumption used in this paper is that parameters $r$ and *pdf* are the same for all the trajectories in a given MOD. Another assumption, commonly used in the literature (e.g., [9,70]) is that the locations of the uncertain objects are *independent* random variables.

In our examples, we use *uniformly distributed* 2D random variables in the uncertainty zone, which implies that if the expected location of the object with $oid_k$ at time $t$ is $(x_k(t), y_k(t))$, then the *pdf* of the object is given by

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

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

## 2.2 Instantaneous NN-queries for Uncertain Objects with a Crisp Query Object

We assume that we are given a query object $\mathbf{o}_q$ whose location at a particular time instant is *crisp*, i.e., a 2D point $\mathbf{Q}$, with no uncertainty associated with it. We also assume that the possible locations of the other objects are disks with radius $r$ (cf. Definition 2). A thorough treatment of the problem of processing range and NN queries for such settings is presented elsewhere [9]. In what follows, we present a concise summary along with observations relevant to our work.

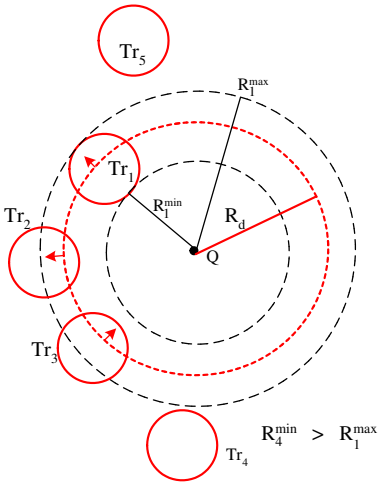


**Fig. 4** Uncertain NN-query (crisp $\mathbf{o}_q$).

**Observation O1**: Our first observation pertains to the bounds on the distances that possible nearest neighbors can have from $\mathbf{Q}$. As illustrated in Figure 4, the distance between $\mathbf{Q}$ and the *most distant point of the closest disk*, $R_{max}$, is the upper bound on the distance that any possible nearest neighbor of $\boldsymbol{Tr}_q$ can have. Consequently, any object $\mathbf{o}_i$ (a snapshot of a trajectory $\boldsymbol{Tr}_i$) whose closest possible distance to $\mathbf{Q}$, denoted with $R_i^{min}$, is larger than $R_{max}$) has a zero probability of being a nearest neighbor to $\boldsymbol{Tr}_q$ and can therefore be safely discarded from the evaluation of the NN-query. As can be seen from Figure 4, $R_4^{min} > R_1^{max}$ and similarly $R_5^{min} > R_1^{max}$, which means that $\boldsymbol{Tr}_4^{\mathbf{u}}$ and $\boldsymbol{Tr}_5^{\mathbf{u}}$ have zero probability of being a nearest neighbor of $Tr_q$. We use $R_{min}$ to denote the distance from $\mathbf{Q}$ to the closest point of the closest disk.

**Observation O2**: Our next observation pertains to the probability of the location along a given trajectory $\boldsymbol{Tr}_i^{\mathbf{u}}$ being *within distance* $R_d$ from $\mathbf{Q}$. This probability can be formulated as:

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

where $A$, the integration bound, denotes the area of the intersection of the disk with radius $R_d$ centered at $\mathbf{Q}$ and the uncertainty disk of $Tr_i$, with a corresponding $pdf_i(x, y)$.

*Example 2* [9] *When $pdf_i(x,y)$ is uniform, the probability $P^{WD}_{i,Q}(R_d)$ can be calculated as:*

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

*where $\Theta = \arccos\frac{d^2_{iQ}+r^2-R^2_d}{2d_{iQ}r}$, $\alpha = \arccos\frac{d^2_{iQ}+R^2_d-r^2}{2d_{iQ}R_d}$, and $d_{iQ}$ is the distance between $\mathbf{Q}$ and the expected location of $\boldsymbol{Tr}^{\boldsymbol{u}}_i$. We note that modifications are needed to Equation 4 when $\mathbf{Q}$ is located inside the uncertainty zone of $\boldsymbol{Tr}^{\boldsymbol{u}}_i$ [9]. Taking the derivative of $P^{WD}_{i,Q}$ yields $pdf^{WD}_{i,Q}(R_d)$ which, in the case of uniform distribution, will be non-zero only when $d_{iQ} - r \leq R_d \leq d_{iQ} + r$.*

**Observation O3**: When calculating the probability that a given object, $\boldsymbol{Tr}^{\mathbf{u}}_j$, is a nearest neighbor of the crisp querying object $\boldsymbol{Tr}_q$ at a given time instant we consider:

1. The probability of $\boldsymbol{Tr}_j$ being within distance $\leq R_d$ from $\boldsymbol{Tr}_q$; combined with:
2. The probability that *every other* object $Tr_i$ $(i \neq j)$ is at a distance greater than $R_d$ from the location $\mathbf{Q}$ of $\boldsymbol{Tr}_q$; and
3. The fact that the distributions of the objects are assumed to be independent from each other.

Using these observations, the generic formula for the nearest-neighbor probability is given by:

$$P^{NN}_{j,Q} = \int_0^\infty pdf^{WD}_{j,Q}(R_d) \cdot \prod_{i \neq j}(1 - P^{WD}_{i,Q}(R_d))\, dR_d \tag{5}$$

We note that the boundaries of the integration need not be 0 and $\infty$ because the effective boundary of the region for which an object *can qualify* to be a nearest neighbor of $\boldsymbol{Tr}_q$ is the ring centered at $\mathbf{Q}$ with radii $R_{min}$ and $R_{max}$. More specifically, $pdf^{WD}_{j,Q}(R_d)$ is 0 for any $R_d < R^{min}_j$ and $1 - P^{WD}_{i,Q}(R_d)$ is 1 for $R_d < R^{min}_i$.

By sorting the objects that have a non-zero probability of being nearest neighbors according to the minimal distances of their boundaries from $\mathbf{Q}$, one can break the evaluation of the integral from Equation 5 into subintervals corresponding to each $R_{min_i}$ and the computation of the $P^{NN}_{j,Q}$ can be performed in a more efficient manner, based on the sorted distances and the corresponding intervals [9]. This is especially important because the evaluation of the integrals like the one specified in Equation 5 may need to be computed numerically. When the *pdf* of the locations is uniform, the objects can be sorted according to the distances of their respective expected locations from $\mathbf{Q}$.

### 2.3 On the Completeness of NN-Probabilities

While the ideas presented in observations **O1**–**O3**, upon which we rely in subsequent sections, are intuitive and sound, there may be certain issues related to their *completeness*. Namely, the computation of the values $P^{NN}_i(Q)$ using Equation 5 will not yield a *probability space* [20]. In other words, it may be the case that adding all probabilities $\Sigma_{\forall i} P^{NN}_{i,Q}$ would yield a value that is less than 1. The reason for this is that the probability of a given object being the nearest neighbor to $Tr_q$ consists of two parts:

$$P^{NN}_{i,Q} = P^{NN\text{-}E}_{i,Q} + P^{NN\text{-}J}_{i,Q} \tag{6}$$

- $P^{NN\text{-}E}_{i,Q}$ denotes the *exclusive* probability that $\boldsymbol{Tr}^{\mathbf{u}}_i$ is the nearest neighbor of $Tr_q$ and is calculated in the spirit of Equation 5.
- $P^{NN\text{-}J}_{i,Q}$, represents a *joint* probability that corresponds to the case(s) in which $\boldsymbol{Tr}^{\mathbf{u}}_i$ is the nearest neighbor of $\boldsymbol{Tr}_q$ along with some other $\boldsymbol{Tr}^{\mathbf{u}}_j$. Therefore, it consists of several additional sums. Each sum can be represented as:

$$S_2 = \Sigma_j \int_0^\infty pdf^{WD}_{i,Q}(R_d) \cdot pdf^{WD}_{j,Q}(R_d) \cdot \prod_{k \neq i,j}(1 - P^{WD}_{k,Q}(R_d))\, dR_d \tag{7}$$

The right-hand side of Equation 7 captures the case(s) when $\boldsymbol{Tr}^{\mathbf{u}}_i$ is *paired* with another $\boldsymbol{Tr}^{\mathbf{u}}_j$ for being the nearest neighbor with respect to the location $\mathbf{Q}$, while all the other moving objects have a smaller probability of that occurring.

By analogy, we have:

$$S_3 = \Sigma_j \Sigma_k \int_0^\infty pdf^{WD}_{i,Q}(R_d) \cdot pdf^{WD}_{j,Q}(R_d) \cdot pdf^{WD}_{k,Q}(R_d) \cdot \prod_{l \neq i,j,k}(1 - P^{WD}_{l,Q}(R_d))\, dR_d \tag{8}$$

where the right-hand side of Equation 8 captures the contributions when all object triples (one of which is $\boldsymbol{Tr}^{\mathbf{u}}_i$) are simultaneously considered as nearest neighbors of $\boldsymbol{Tr}_q$ at $\mathbf{Q}$.

Assuming a collection of $N$ objects, $S_N$ is the value denoting the probability that all of them are simultaneously neighbors of $\boldsymbol{Tr}_q$ at $\mathbf{Q}$:

$$S_N = \int_0^\infty \prod_k pdf^{WD}_{k,Q}(R_d)\, dR_d \tag{9}$$

| Notation | Meaning |
|---|---|
| $\boldsymbol{Tr}_i$ | Trajectory of the $i$-th moving object |
| $\boldsymbol{Tr}_i^{\mathbf{u}}$ | Uncertain trajectory of the $i$-th moving object |
| $P_{i,Q}^{WD}(R_d)$ | Probability that the $i$-th object is within distance $R_d$ from querying object $Q$ |
| $P_{j,Q}^{NN}$ | Probability that $j$-th object is the nearest neighbor of the querying object $Q$ |
| $D_i$ | Uncertainty disk (whereabouts) of the object $i$ |
| $(D_q \oplus R_d)$ | Minkowski sum of $D_q$ and a disk with radius $R_d$ |
| $\overline{V}_i$ | 2D random variable of possible locations of $\boldsymbol{Tr}_i^u$ at time instant |
| $pdf(\overline{V}_i) \circ pdf(\overline{V}_j)$ | Convolution of the *pdf*s of the 2D random variables |
| $TR_{iq}$ | The *difference-trajectory* of $\boldsymbol{Tr}_i^{\mathbf{u}}$ and $\boldsymbol{Tr}_q^{\mathbf{u}}$ |
| $LE_{i,j}$ | Lower Envelope of $i$-th and $j$-th difference-trajectories (distances from querying trajectory) |
| $l(e_{sk})$ | Length of the edge between the vertices $s$ and $k$ in a road network graph |
| $v_{sk}^{max}$ | Maximum speed along the edge $e_{sk}$ in a road network |
| $\boldsymbol{RN\text{-}Tr}_i$ | Trajectory of the $i$-th object on a road network |
| $\boldsymbol{RN\text{-}Tr}_i^{\mathbf{u}}$ | Uncertain trajectory (of the $i$-th object) on a road network |

**Table 1** Notation used in this paper.

We observe that the importance of the properties identified above for the current work is as follows: when focusing on the **1**NN-query for uncertain trajectories with respect to a crisp query object, the values of $S_2$, $S_3$, $\ldots S_N$ will not be of relevance for the *ranking* the respective values of the probabilities in the answer. This fact will be used in our results in both Section 4 and Section 5. However, in the case of processing *kNN* variant of the query, where $k \geq 2$, the values of such $S_i$'s cannot be neglected.

We conclude this section by providing in Table 1 a summary of the notation used in this paper, some of which has been already introduced.

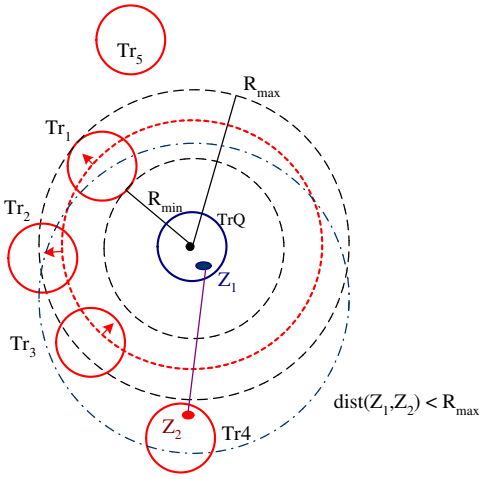


**Fig. 5** Uncertain NN-query (uncertain $Tr_q$).

## 3 Uncertain Querying Object and Convolutions

In this section, we present a first set of results of our work. First, we illustrate 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 2.2 are applicable with appropriate modifications.

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 $\boldsymbol{Tr}_q$ is located somewhere in the zone denoted by $Z_1$ inside of its own uncertainty disk and $\boldsymbol{Tr}_4$ is located somewhere in the zone denoted by $Z_2$, their distance is less than $R_{max}$ and, consequently, $\boldsymbol{Tr}_4$ has a non-zero probability of being a (possible) nearest neighbor of $\boldsymbol{Tr}_q$. 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 (cf. [9]). Strictly speaking, at the heart of the problem is the calculation of the probability that a given object $\boldsymbol{Tr}_i$ is *within distance* $R_d$ of $\boldsymbol{Tr}_q$.

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., [12]) of the uncertainty disk of $\boldsymbol{Tr}_q$ 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 – which 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.

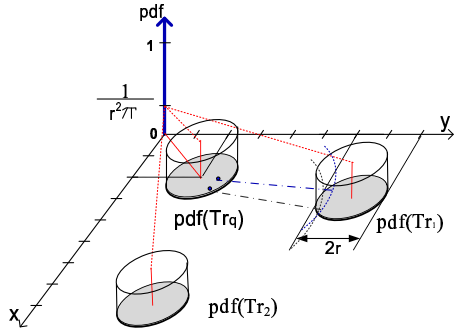


**Fig. 6** Evaluating within distance probability.

This method yields a quadruple integration in the corresponding version of Equation 3 used for evaluating $P^{WD}_{i,q}(R_d)$ and yields additional overhead in determining the $pdf^{WD}_{i,q}(R_d)$ (via differentiation), in order to be able to use Equation 5 for evaluating $P^{NN}_{i,q}$. 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 $\boldsymbol{Tr}_i$ is at location $l_1 \in D_i$, given that $\boldsymbol{Tr}_q$ 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 [9,70]). 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* $\boldsymbol{Tr}_q$ *illustrate part of the calculation of the probability of* $\boldsymbol{Tr}_1$ *being within distance* $\leq 4$ *from* $\boldsymbol{Tr}_q$ *(obviously, 0 for* $\boldsymbol{Tr}_2$*).*

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 $\boldsymbol{Tr}^u_i$ at a given time instant. Recall that the crux for evaluating a probabilistic NN-query is determining the expression for the probability of $\boldsymbol{Tr}^u_i$ being within a given distance $R_d$ from $\boldsymbol{Tr}^u_q$, which is, the value of $P^{WD}_i(R_d)$. 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, denote it $\overline{V}_{iq}$ which, in probability and signal/image processing is known as a *cross-correlation* of $\overline{V}_i$ and $\overline{V}_q$ [44,74]. 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 [9,70]), it is a well-known fact from the probability theory that the random variable $\overline{V}_{iq}$ has a $pdf_{iq}$ which is a *convolution* of the corresponding *pdf*'s of $\overline{V}_i$ and $-\overline{V}_q$) [74]. In other words:

$$pdf(\overline{V}_{iq}) = pdf(\overline{V}_i) \circ pdf(-\overline{V}_q) \quad (6)$$

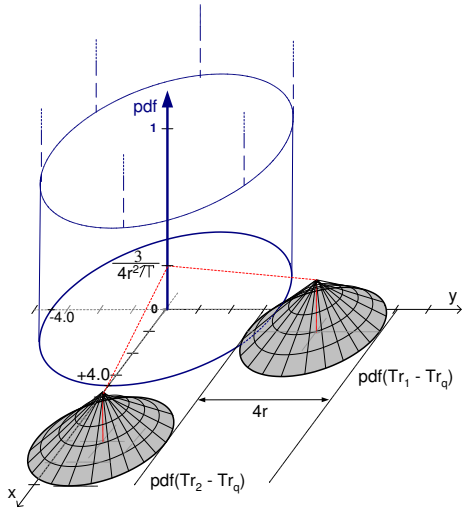


**Fig. 7** Within distance probability: convolution.

*Example 4 As one can readily verify (cf. [74]), 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* $\frac{3}{4r^2\pi}$. *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* $\boldsymbol{Tr}_q$ *(cf. 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, given the Equation (2), we have*

$$pdf(\overline{V}_{iq}(t)(X,Y)) = \quad (7)$$

$$\begin{cases} 0, & \sqrt{((x_i(t) - x_q(t)) - X)^2 + ((y_i(t) - y_q(t)) - Y)^2} > 2r \\ \frac{3}{4r^2\pi}\left(1 - \frac{\sqrt{((x_i(t) - x_q(t)) - X)^2 + ((y_i(t) - y_q(t)) - Y)^2}}{2r}\right) & \text{otherwise} \end{cases}$$

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* [58]. However, in many practical settings, the *pdf*'s of the objects' locations (e.g., uniform, Gaussian) are *Riemann-integrable* [58], 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 $\overline{V}_{iq}$.*

Before we proceed with the outline of the proof of Property 1, as well as the other claims, we briefly note that when a *translation*, e.g., $\overline{s} \mapsto \overline{s} + \overline{w}$ is applied as a transformation to a 2D variable (in the sense of variable substitution), as well as rotation around the center, e.g., $\overline{s} \mapsto \overline{w} (= \rho_{(0,0),\phi}(\overline{w}))$, the Jacobian determinant evaluates to "1".

**Proof:** *(of* **Property 1***) Firstly, observe that $E(\overline{V}_{iq}) = E(\overline{V}_i) + E(-\overline{V}_q)$ simply because $\overline{V}_{iq}$ is the sum of $\overline{V}_i$ and $-\overline{V}_q$. By definition, the centroid of $f$ can be calculated as: $\overline{C}_c = (\int \overline{x} f(\overline{x}) d\overline{x}) / (\int f(\overline{x}) d\overline{x})$. Let us observe separately the:*
*(1) Denominator: by the definition of the convolution, we have: $\int f(\overline{x}) d\overline{x} = \int [\int g(\overline{u}) \cdot h(\overline{x} - \overline{u}) d\overline{u})] d\overline{x}$ = ...substitute variables $\overline{x} = \overline{x} + \overline{u}$, noting that $d\overline{x}$ remains the same and the Jacobian is "1" (translation)... $= \int g(\overline{u}) d\overline{u} \cdot \int h(\overline{x}) d\overline{x}$ = ...since $h$ and $u$ are pdfs, each integral evaluates to "1" ... $= 1$.*
*(2) Numerator: Similarly, $\int \overline{x} f(\overline{x}) d\overline{x} = \int \overline{x} [\int g(\overline{u}) \cdot h(\overline{x} - \overline{u}) d\overline{u})] d\overline{x}$ = ...applying the same substitution: $\overline{x} = \overline{x} + \overline{u}$ ... $= \int (\overline{x} + \overline{u}) [\int g(\overline{u}) \cdot h(\overline{x}) d\overline{u})] d\overline{x} = ((\int \overline{x} h(\overline{x}) d\overline{x}) \int g(\overline{u}) d\overline{u}) + ((\int \overline{u} g(\overline{u}) d\overline{u}) \int h(\overline{x}) d\overline{x})$.*
*Observing once again that $\int h(\overline{x}) d\overline{x} = \overline{C}_1$ and $\int g(\overline{u}) d\overline{u} = \overline{C}_2$, the claim follows.* □.

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 [74]. 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 [44]. 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 $\overline{C}_c$.*

**Proof:** *(of* **Property 2***) Assume $\overline{P}$ and $\overline{Q}$ are points from the domain of $f$ such that $\|\overline{PC}_c = \overline{QC}_c\|$. Then, there exists a rotation $\rho$ with a center at $\overline{C}_c$ and an angle $\phi$, such that $\rho_{C_c,\phi}(P) = Q$. This can also be viewed as a composition of: (1) translation of $\overline{C}_c$ to the origin; (2) rotation for angle $\phi$; (3) (de)translation back to $\overline{C}_c$).*
*Observe $f(\overline{P} - \overline{C}_c)$ = ...by Property 1 ... $= f(\overline{P} - (\overline{C}_1 + \overline{C}_2))$. By definition, this is equal to $\int g(\overline{u}) \cdot h(\overline{P} - \overline{C}_1 - \overline{C}_2 - \overline{u}) d\overline{u}$ = ...substituting $\overline{u}$ with $\overline{u} - \overline{C}_1$, $d\overline{u}$ remains, and the Jacobijan is "1" ... $= \int g(\overline{u} - \overline{C}_1) \cdot h(\overline{P} - \overline{C}_2 - \overline{u}) d\overline{u}$ = ... by the assumed rotational symmetry of $h$, if $Q$ is a point such that $\|\overline{PC}_2 = \overline{QC}_2\|$ ... $= \int g(\overline{u} - \overline{C}_1) \cdot h(\overline{Q} - \overline{C}_2 - \overline{u}) d\overline{u}$ = ...substituting $\overline{u}$ with $\overline{u} - \overline{C}_1$ ... $= \int g(\overline{u}) \cdot h(\overline{Q} - \overline{C}_1 - \overline{C}_2 - \overline{u}) d\overline{u} = f(\overline{Q} - (\overline{C}_1 + \overline{C}_2))$. Since the convolution is translation (shift) invariant [44], the claim follows.* □

Assume that $\boldsymbol{Tr}_1^u$ and $\boldsymbol{Tr}_2^u$ 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 1:** *Let $Q$ denote a 2D point. If $\|\overline{QC}_1\| < \|\overline{QC}_2\|$, then $P_1^{NN}(Q) > P_2^{NN}(Q)$.*

**Proof:** *(of* **Lemma 1***) It suffices to prove the claim for the exclusive NN probabilities (i.e. $P_{1,Q}^{NN\text{-}E} > P_{2,Q}^{NN\text{-}E}$, cf. Section 2.2), because the joint NN probability will appear equally in each of $P_{1,Q}^{NN}$ and $P_{2,Q}^{NN}$. Due to the assumption(s), we have that $R_{min}^1 < R_{min}^2$ and $R_{max}^1 < R_{max}^1$. Appropriately modifying Equation (5), we have: (I): $P_1^{NN}(Q) = \int_0^\infty pdf_1^{WD}(R_d) \cdot (1 - P_2^{WD}(R_d)) dR_d = \int_{R_1^{min}}^{R_{max^1}} pdf_1^{WD}(R_d) \cdot (1 - P_2^{WD}(R_d)) dR_d$ and, similarly: (II): $P_2^{NN}(Q) = \int_0^\infty pdf_2^{WD}(R_d) \cdot (1 - P_2^{WD}(R_d)) dR_d = \int_{R_2^{min}}^{R_1^{max}} pdf_2^{WD}(R_d) \cdot (1 - P_1^{WD}(R_d)) dR_d$.*
*The claim follows from the observations that for every $\nu$, when evaluating $pdf_2^{WD}(R_2^{min} + \nu)$ in (II), there exists an equivalent $pdf_1^{WD}(R_1^{min} + \nu)$ which, however, is multiplied by a larger value of $(1 - P_2^{WD}(R_d))$ in (I).* □

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 $\boldsymbol{Tr}_q$ denote the (uncertain) querying trajectory. The main result of this section can be 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 $\boldsymbol{Tr}_q^u$ 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 $\boldsymbol{Tr}_q^u$.*

**Proof:** Theorem 1 follows from the properties of the convolution for independent random variables with rotational symmetry and Lemma 1.

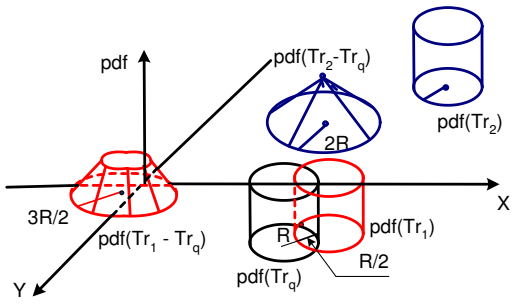


**Fig. 8** Convolution of intersecting *pdf*s.

As an illustration, recall Figure 7. Since the centroid of $\boldsymbol{Tr}_1^u - Tr_q^u$ is closer to the coordinate-center than the centroid of $\boldsymbol{Tr}_2^u - \boldsymbol{Tr}_q^u$, we have that $P_1^{NN}(Q) > P_2^{NN}(Q)$.

We conclude this section with an observation regarding the mutual-whereabouts of the uncertainty disks. In the examples so far, we assumed that the uncertainty disks of the respective trajectories did 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. Essentially, the radius of the disk in the basis of the convolution cone will be smaller than $2r$, and the cone itself will be reduced to a frustum – i.e., bounded by a smaller disk at its top, instead of a vertex. However, it can be readily demonstrated that Theorem 1 is still valid. Specifically, at the time-instant illustrated in the scenario in Figure 8, $Tr_2$ has zero probability of being a nearest neighbor.

## 4 Answering Uncertain NN-query – Semantics and Processing

In this section, we describe the structure of the answer to a continuous NN query for uncertain trajectories and we present an algorithm for constructing this structure.

### 4.1 Recursive Time-Parameterizing

Recall that in Example 1 (motivational example) of Section 1, the answer to a continuous NN-query for crisp trajectories was presented in a compact manner as a sequence:

$\mathbf{A}_{Q\_NN}$ : $[\boldsymbol{Tr}_3, (10, 15)]$, $[\boldsymbol{Tr}_2, (15, 20)]$.

Our goal is to devise something similar for the case of an NN-query for uncertain trajectories. However, relying on a simple sequence in a manner of $\mathbf{A}_{Q\_NN}$ would, at best, provide an opportunity to capture one trajectory within a particular time interval. As we discussed in Section 1, this need not be the case when the trajectories are associated with an uncertainty of the location at a given time-instant. As a consequence, throughout a particular time-interval, there may be more than one object that could have a chance of being the nearest neighbor to the query trajectory.

Within a give time instant some objects will have higher probability of being the nearest neighbor to a given $\boldsymbol{Tr}_q^{\mathbf{u}}$ than the others. We would like to extend this observation throughout the entire time-interval of interest for the query by identifying the critical time-instants at which the relative ranking of the probabilities changes—equivalently, by identifying the time-intervals throughout which portions of the relative ranking do not change.

We identify the following objectives for the representation of the answer to an NN-query for uncertain trajectories:

1. The time-interval of interest $[t_b, t_e]$ should be split into sub-intervals $[t_b, t_1]$, $[t_1, t_2], \ldots,$ $[t_{n-1}, t_e]$ so that the trajectory that has the highest probability of being the nearest neighbor of $\boldsymbol{Tr}_q^{\mathbf{u}}$ in each sub-interval is unique. For example, $\boldsymbol{Tr}_{i-1}^{\mathbf{u}}$ is the uncertain trajectory with the highest probability of being the nearest neighbor of $\boldsymbol{Tr}_q^{\mathbf{u}}$ all throughout $[t_{i-1}, t_i]$.
2. Each such sub-interval, in turn, is further split into its own sub-intervals – e.g., $[t_{i-1}, t_i]$ is split into $[t_{i-1}, t_{(i-1),1}]$, $[t_{(i-1),1}, t_{(i-1),2}], \ldots,$ $[t_{(i-1),(k-1)}, t_i]$. To each of this sub-intervals, again a unique trajectory is matched – representing the trajectory which would have been the actual highest-probability nearest neighbor of $\boldsymbol{Tr}_q^{\mathbf{u}}$, if the MOD did not contain $\boldsymbol{Tr}_{i-1}^{\mathbf{u}}$.
3. The process is recursively repeated for each sub-interval, terminating when no further split is possible which would contain an uncertain trajectory with non-zero probability of being nearest neighbor to $\boldsymbol{Tr}_q^{\mathbf{u}}$.

According to the above goals, the answer to the NN-query from the motivational scenario in Section 1, assuming that every trajectory has an uncertainty radius of 30 meters, would be formulated as:

$\mathbf{A}_{Q\_NN}$:
{ [$\boldsymbol{Tr}_3^{\mathbf{u}}$, (10,15) [$\boldsymbol{Tr}_2^{\mathbf{u}}$, (12,15)]],
[$\boldsymbol{Tr}_2^{\mathbf{u}}$, (15,20) [$\boldsymbol{Tr}_3^{\mathbf{u}}$, (15,18)],[$\boldsymbol{Tr}_4^{\mathbf{u}}$, (19,20)]] }

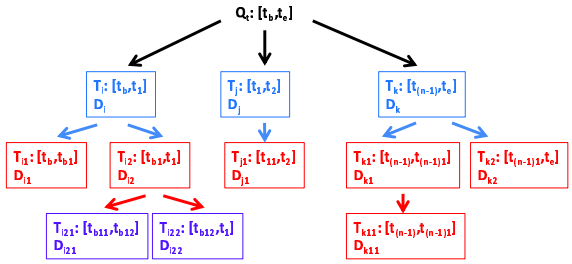


**Fig. 9** Structure of the answer to an NN-query: the IPAC-NN tree

Specifically, the answer $\mathbf{A}_{Q\_NN}$ indicates that during the time-interval $(10, 15)$, the uncertain trajectory $\boldsymbol{Tr}_3^{\mathbf{u}}$ always has the highest probability of being the nearest neighbor of $\boldsymbol{Tr}_q^{\mathbf{u}}$. However, during the (sub)interval $(12, 15)$ $\boldsymbol{Tr}_2^{\mathbf{u}}$ also has a non-zero probability of being $\boldsymbol{Tr}_q^{\mathbf{u}}$'s nearest neighbor, except at every time-instant that probability is lower than the corresponding probability of $\boldsymbol{Tr}_3^{\mathbf{u}}$. Similarly, in the rest of the time-interval for the query, $(15, 20)$, $\boldsymbol{Tr}_2^{\mathbf{u}}$ has the highest probability of being the nearest neighbor of $\boldsymbol{Tr}_q^{\mathbf{u}}$. However, there exist time-(sub)intervals during which other objects have nonzero probability (though lower than that of $\boldsymbol{Tr}_2^{\mathbf{u}}$) of being the nearest neighbor of $\boldsymbol{Tr}_q^{\mathbf{u}}$ (e.g., $\boldsymbol{Tr}_4^{\mathbf{u}}$ for $t \in (19, 20)$).

As a general way of representing the answer of a continuous NN query for uncertain trajectories, we propose a tree-structure with the following properties:

- The root of the tree is node labeled with the description of the parameters of interest for the specification of the query, e.g., $\boldsymbol{Tr}_q^{\mathbf{u}}$, along with $[t_b, t_e]$.
- The root has one child for each sub-interval of $[t_b, t_e]$, throughout which there is a unique uncertain trajectory $\boldsymbol{Tr}_i^{\mathbf{u}}$ having the highest probability of being the nearest neighbor to $\boldsymbol{Tr}_q^{\mathbf{u}}$. Each child of the root is labelled with the corresponding trajectory (e.g., $\boldsymbol{Tr}_i^{\mathbf{u}}$) and the time-interval of its validity as the highest probability nearest neighbor (e.g., $t_{i-1}, t_i$).
- After obtaining the respective labels from the respective parent-node, each child-node checks whether if it is removed from the MOD, there could still be some object with nonzero probability of being the nearest neighbor of $\boldsymbol{Tr}_q^{\mathbf{u}}$ in the time sub-interval of its label.
  - If so, then it is an internal node, and each internal node follows the principle of splitting its own (sub)interval like it has been done in the root, and uses the same labelling for its children.
  - If not, then that node is a leaf-node.

We call this structure *IPAC-NN* tree (Intervals of Possible Answers to Continuous-NN), and an illustration of its definition is provided in Figure 9. Note that the nodes in Figure 9 also contain another component of their labelling, $D$, which is an application-dependent *Descriptor* of that node. For example, this descriptor can be the maximum value of the probability of being the nearest neighbor in the respective time-interval. We will present a specific example of using the *Descriptor* attribute in Section 5.

Observe that the removal of the root of the tree yields a DAG (Directed Acyclic Graph) as a structure to represent $\mathbf{A}_{Q\_NN}$. In the rest of this section, we focus on developing methodologies for constructing the IPAC-NN tree for a given set of uncertain trajectories, with respect to a particular query trajectory.

### 4.2 Constructing the IPAC-NN Tree

The basic observation that the difference of two trajectories can be expressed as a single random variable, along with Theorem 1, forms the foundation for constructing the IPAC-NN tree introduced in Section 4.1.

To simplify the presentation, and without loss of generality, throughout most of this section we assume that each trajectory consists of a single segment during the time-interval of interest for a given query $\mathbf{UQ\_nn}(\boldsymbol{Tr}_q^{\mathbf{u}})$, $[t_b, t_e]$. In other words, the expected location of each trajectory during $[t_b, t_e]$ consists of one straight line segment. We analyze the impact of the general case (i.e., the removal of this assumption) on the complexity of the algorithms at the end of Section 4.

Let $(x_{bi}, y_{bi})$ denote the expected location of the uncertain trajectory $\boldsymbol{Tr}_i^{\mathbf{u}}$ at $t_b$ and, similarly, let $(x_{ei}, y_{ei})$ denote the expected location of $\boldsymbol{Tr}_i^{\mathbf{u}}$ at time $t_e$. The expected motion of $Tr_i^u$ during time interval $[t_b, t_e]$ is characterized by a velocity vector whose $X$ and $Y$ components are given by:

$$v_{xi} = (x_{ei} - x_{bi})/(t_e - t_b); \text{ and}$$
$$v_{yi} = (y_{ei} - y_{bi})/(t_e - t_b).$$

Hence, the expected location at a time instant $t \in [t_b, t_e]$ will have coordinates:

$$x_i(t) = x_{bi} + v_{xi}(t - t_b); \text{ and}$$
$$y_i(t) = y_{bi} + v_{yi}(t - t_b),$$

which are the coordinates of the center of the uncertainty disk at $t$.

For a given trajectory $\boldsymbol{Tr}_i^{\mathbf{u}}$ that is not the query trajectory (i.e., $i \neq q$), let $TR_{iq}$ denote the *difference-trajectory* $\boldsymbol{Tr}_i^{\mathbf{u}} - \boldsymbol{Tr}_q^{\mathbf{u}}$. In other words, at each time instant $t$, the expected location of the object moving along $TR_{iq}(t)$ is the vector-difference of the expected locations of the corresponding points along $\boldsymbol{Tr}_i^{\mathbf{u}}$ and $\boldsymbol{Tr}_q^{\mathbf{u}}$. Trajectory $TR_{iq}(t)$ captures the spirit of Section 3, in

the sense that the 2D distance between the expected locations of the objects moving along $\boldsymbol{Tr}_i^{\mathbf{u}}$ and $\boldsymbol{Tr}_q^{\mathbf{u}}$ at time $t$ (see, e.g., [5, 55]) now becomes the distance at time $t$ that an object moving along $TR_{iq}$ has from the origin $(0, 0)$. Let $V_{xiq}$ and $V_{yiq}$ denote the components of the velocity of the object whose expected trajectory is $TR_{iq}$. We have

$$V_{xiq} = v_{xi} - v_{xq},$$
$$V_{yiq} = v_{yi} - v_{yq}.$$

Also, let $X_{biq}$ and $Y_{biq}$ denote the coordinates of the expected location at $t_b$. We have

$$X_{biq} = x_{bi} - x_{bq},$$
$$Y_{biq} = y_{bi} - y_{bq}.$$

The distance of $TR_{iq}$ from the origin, as a function of time, is given by (cf. [5, 55]):

$$d_{iq}(t) = \sqrt{At^2 + Bt + C},$$

where:

$A = V_{xiq}^2 + V_{yiq}^2$,
$B = -2(V_{x_{iq}}^2 t_b + V_{x_{iq}} X_{b_{iq}} + V_{y_{iq}}^2 t_b + V_{y_{iq}} Y_{b_{iq}})$, 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, 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, that is, 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.[1] Consequently, they can have zero, one or two 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 10 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 two 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 the existence of an function, denoted $Env2(TR_{iq}, TR_{jq}, t_1, t_2)$, which takes two difference-trajectories as input, and returns their lower envelope as output, along with the critical times, between times $t_1$ and $t_2$. Essentially, $Env2(TR_{iq}, TR_{jq}, t_1, t_2)$ does exactly what we described in the previous paragraph:

- Solves the quadratic equation in which the Left-hand Side is the respective quadratic function $TR_{iq}$ and the Right-hand Side is the corresponding one of $TR_{jq}$;
- Checks which ones of the solutions are inside $[t_1, t_2]$, defining the critical time-points;
- For each interval in between critical time points, including the boundaries $t_1$ and $t_2$, determines which one of the $TR_{iq}$ and $TR_{jq}$ defines the lower envelope;
- Returns the result.

Clearly, algorithm $Env2(TR_{iq}, TR_{jq}, t_1, t_2)$ runs in $O(1)$ time, since the difference-trajectories can intersect in at most two points.

*Example 5* In the example of Figure 10.a, algorithm $Env2(TR_1, TR_2, t_b, t_e)$ outputs 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 10.b, algorithm $Env2(TR_3, TR_4, t_b, t_e)$ 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 [12, 63]. For our settings, we have implemented a divide-and-conquer method in the spirit of *MergeSort*, which has been also been used in our experiments. 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

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

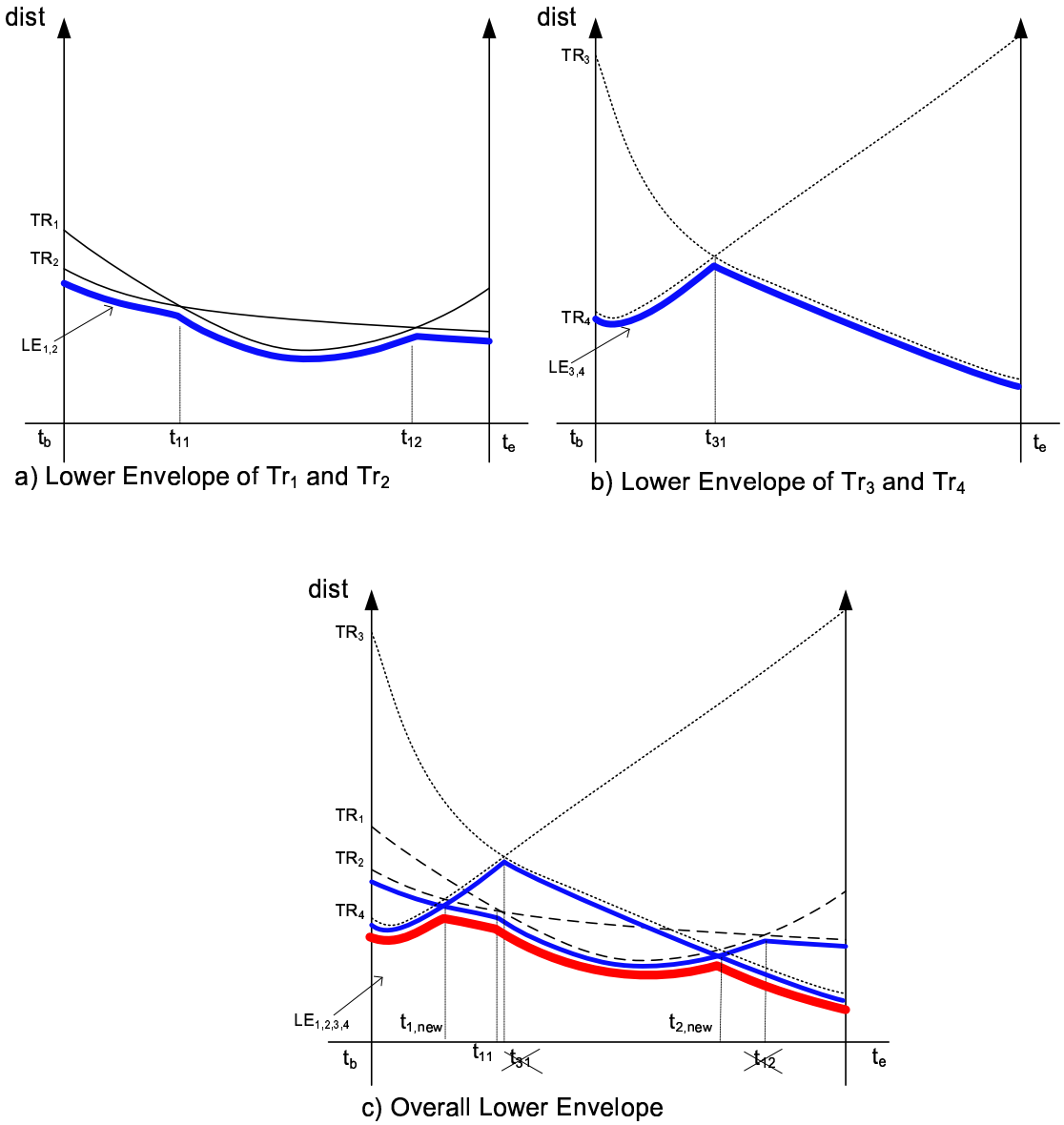


**Fig. 10** Constructing the lower envelope.

**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,$t_b, t_e$));*

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 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 $[(TR_j, [t_{j1}, t_{j3}])]$.

For completeness, we show the details of method *Merge_LE* in Algorithm 2.

**Algorithm 2** Merging two lower envelopes.

**Merge_LE**$(LE_1, LE_2)$
**Input:** *Two lower-envelopes with their critical time-points*

$LE_1 = [(TR_{1i_1}, [t_b, t_{11}]), (TR_{1i_2}, [t_{11}, t_{12}]), \ldots, (TR_{1i_m}, [t_{1(m-1)}, t_{1m}])]$
$LE_2 = [(TR_{2i_1}, [t_b, t_{21}]), (TR_{2i_2}, [t_{21}, t_{22}]), \ldots, (TR_{2i_n}, [t_{1(n-1)}, t_{1n}])]$

**Output:** *The combined lower-envelope* $LE_{1,2} = LE_1 \uplus LE_2$

*Let* $LE_{1,2} = \emptyset$;
$k = p = 0$;
**while**$((k < m) \vee (p < n))$
{ $t_1^{cl} = t_{1k}$; $t_2^{cl} = t_{2p}$;
$t_1^{cu} = t_{1(k+1)}$; $t_2^{cu} = t_{2(p+1)}$; // assume $t_{10} = t_{20} = t_b$
$t^{cl} = max(t_1^{cl}, t_2^{cl})$; // current lower bound
$t^{cu} = min(t_1^{cu}, t_2^{cu})$; // current upper bound
// of the sweeping time-interval
$LE_{1,2} = LE_{1,2} \odot Env2(TR_1 i_k, TR_{2j_p}, t^{cl}, t^{cu})$
// concatenate ($\odot$) the currently obtained
// envelope to the existing one.
**if** $(t_1^{cu} < t_2^{cu})$ $k$++;
**else_If** $(t_2^{cu} < t_1^{cu})$ $p$++;
**else** // $(t_2^{cu} = t_1^{cu})$
{ $p$++; $k$++; } // advance }

As a consequence of the properties of the Davenport-Schinzel sequences [63], 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. The time complexity of Algorithm 2 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 with the following example.

*Example 6* Observe Figure 10, and assume that the envelopes in Part a.) and b.) represent the inputs to the Merge_LE. Initially, the *current lower bound* $t^{cl}$ is $t_b$ (since $t_1^{cl} = t_2^{cl} = t_b$), whereas the *current upper bound* is $t^{cu} = min(t_{11}, t_{31}) = t_{11}$. Hence, $Env2(TR_2, TR_4, t_b, t_{11})$ is applied in the first iteration, obtaining a new critical time-point ($t_{1,new}$) and generating an envelope with two portions ($TR_4, [t_b, t_{1,new}]$) and ($TR_2, [t_{1,new}, t_{11}]$). Since $t_{11} < t_{31}$, we increment $k$ at the end of the loop which, in turn, means that $t_1^{cl} = t_{11}$ and $t_1^{cu} = t_{12}$. Consequently, throughout the second iteration of the while-loop we have $t^{cl} = max(t_1^{cl}(= t_{11}), t_2^{cl}(= t_b)) = t_{11}$ and $t^{cu} = min(t_1^{cu}(= t_{12}), t_2^{cu}(= t_{31})) = t_{31}$. $Env2(TR_1, TR_4, t_{11}, t_{31})$, yields the next part of the overall envelope $[(TR_1, [t_{11}, t_{31}])]$. Since $t_{31} < t_{12}$, this time we increment $p$ before we enter the next iteration. Subsequent iterations will consecutively invoke the following operation:

- $Env2(TR_1, TR_3, t_{31}, t_{12})$, generating a new critical time-point ($t_{2,new}$ in Figure 10.c) and removing $t_{31}$ from the list of critical time-points because $TR_1$ continues to be the lower envelope at it (cf. $\odot$-concatenation). After this iteration, $LE_{1,2,3,4}$ consists of $[(TR_4, [t_b, t_{1,new}])$, $(TR_2, [t_{1,new}, t_{11}])$, $(TR_1, [t_{11}, t_{2,new}])$, and $(TR_3, [t_{2,new}, t_{12}])]$;
- $Env2(TR_2, TR_3, t_{12}, t_e)$, generating $[(TR_3, [t_{12}, t_e])]$, which "absorbs" $t_{12}$ as a critical time-point when appended to the existing $LE_{1,2,3,4}$.

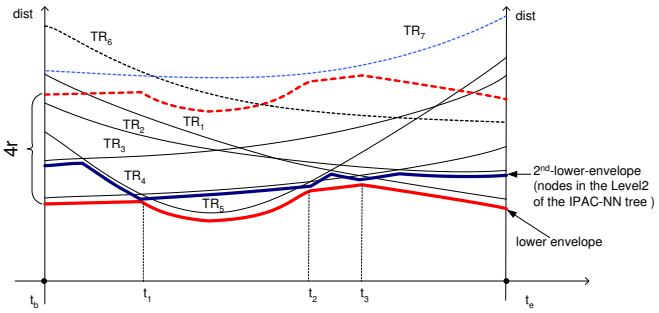


**Fig. 11** Envelopes and IPAC-NN tree.

One of the benefits of constructing the lower envelope is that it provides a *continuous pruning* criteria. Namely, in the *(distance, time)* space, any trajectory whose distance function does not intersect the region bounded by the lower envelope and its vertically-translated (i.e., along the *distance* axis), copy for a vector of length $4r$, 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 11, $TR_7$ can be safely pruned from any consideration, because its distance from the lower envelope at any time instant is greater than $4r$.

Algorithm 3 constructs the IPAC-NN tree, which can be used for answering queries based on the continuous ranking of the uncertain trajectories which have a non-zero probability of being nearest neighbors to a given querying trajectory.

The combinatorial complexity of the lower envelope is $O(N)$ and its construction takes $O(N \log N)$ time.

**Algorithm 3** 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.*

*Construct the lower envelope using Algorithm 1. The lower envelope corresponds to the nodes in Level_1 of the IPAC-NN tree;*
*Prune all the objects that cannot have a non-zero probability of being a nearest neighbor;*
**for** *each level L*
  **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*
    *Remove from consideration $TR_i^{L-1}$ defining the envelope at level $L-1$ in $(t_i, t_{i+1})$;*
    *Construct the portion of the lower-envelope at level L applying Algorithm 1;*
  **end_for**
**end_for**

Since each of the $(N)$ distance functions will need to be compared against the each of the $O(N)$ segments of the lower envelope, the completion of the pruning phase has $O(N^2)$ time complexity. Assuming that after the pruning there are $\lceil N/K \rceil$ $(K \leq N)$ objects left for consideration, the running time for constructing the 2nd-lower-envelope (equivalently, the Level_2 nodes of the IPAC-NN tree) is $O(\lceil N/K \rceil \log \lceil N/K \rceil)$. Since two distance functions (hyperbolae) can intersect at most twice, we observe that the total number of intersection points within the zone bounded by the lower envelope and its translation for $4r$ along the vertical (*distance*) axis 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 11 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 finalize this section with a discussion on the complexity of the algorithms presented here, removing the assumption from the beginning of this section – which is, all the trajectories consist of single line-segments. Clearly, this need not be the case in practice, as each trajectory may have a number of segments (cf. Definition 1 in Section 2). Assume that a particular trajectory $\boldsymbol{Tr}_j^{\mathbf{u}}$ has $m_j$ segments throughout the time-interval of interest for the query, while all the rest of the trajectories still consist of one segment. Clearly, this will incur an additional factor of $m_j$ multiplying every complexity result presented in this section, for the simple reason that we will need to repeat the calculations for each of the $m_j$ segments of $\boldsymbol{Tr}_j^{\mathbf{u}}$. Generalizing this observation, if every trajectory $\boldsymbol{Tr}_i^{\mathbf{u}}$ $(i \in \{1, 2, ..., M\})$ has $m_i$ segments, then each of the corresponding complexity results will need to be multiplied by the factor $\sum_{i=1}^{M} m_i$.

## 5 Uncertain NN-query on Road Networks

We now consider the processing of uncertain NN-queries in the settings in which the moving objects are restricted to move along a road network. First, we study the impact of the road network constraint on the distance function. Subsequently, we address the uncertainty model and its ramifications on the semantics and processing of the uncertain NN-queries in these settings.

### 5.1 Trajectories' Distance in Road Networks

Many aspects of the problem of modeling and querying of spatio-temporal objects in road networks have been investigated the literature [11,13,22,48,61,80], and one of the typical assumptions is that the network is represented as a graph $G(V, E)$, where:

- $V$ denotes the set of nodes/vertices $\{n_1, n_2, \ldots, n_w\}$, where each vertex is associated with its coordinates in the reference coordinate system, e.g., $n_i(x_{ni}, y_{ni})$.
- $E$ denotes the set of edges $(E \subseteq V \times V)$ where, in addition to its own labeling, a given edge is often represented as a pair $e_{ks} = (n_k, n_s)$ of adjacent vertices.

A commonly accepted interpretation is that the vertices represent intersections and edges represent road segments in-between intersections. We assume an *undirected graph*, which means that each edge can be traversed in both directions. Finally, we assume that each edge $e_{sk}$ has the following two attributes:

1. the *length* of $e_{sk}$, denoted $l(e_{sk})$; and
2. the *maximum speed* of $e_{sk}$, denoted $v_{sk}^{max}$, which is the upper bound on how fast an object can move along edge $e_{sk}$.

**Definition 3** Given a road network graph $G(V, E)$, a *road network trajectory* consists of the ID of a moving object and a sequence of 3D points (2D spatial coordinates plus time), $\boldsymbol{RN\text{-}Tr}_i = \{oid_i, (x_{i_1}, y_{i_1}, t_{i_1}),\ (x_{i_2}, y_{i_2}, t_{i_2}),\ \ldots, (x_{i_k}, y_{i_k}, t_{i_k})\}$ where:

- $t_{i_1} \leq t_{i_2} \leq \ldots \leq t_{i_k}$
- Every two consecutive points, except, possibly $(x_{i_1}, y_{i_1})$ and $(x_{i_k}, y_{i_k})$, coincide with two adjacent vertices in $V$, and in between two points, the object is assumed to travel along the edge incident to the corresponding vertices
- The speed of the object along a given edge $e_{ks}$ is assumed to be *constant* and less than or equal to $v_{sk}^{max}$.

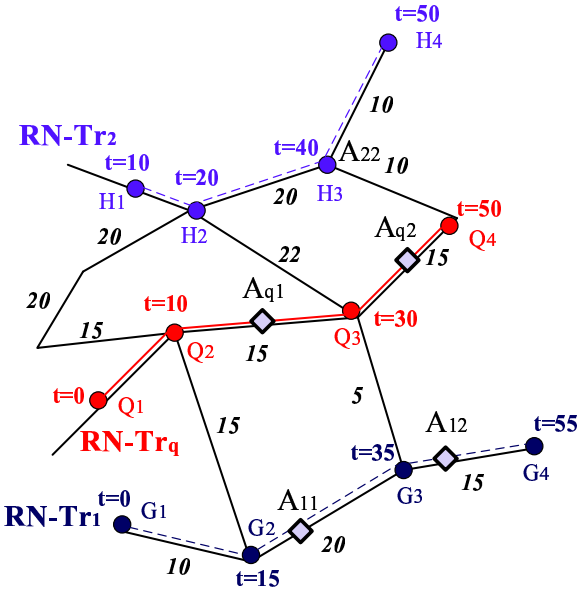


**Fig. 12** Trajectories on Road Networks

The concepts introduced in Definition 3 are illustrated in Figure 12, which shows three road network trajectories: $\boldsymbol{RN\text{-}Tr}_q$, $\boldsymbol{RN\text{-}Tr}_1$ and $\boldsymbol{RN\text{-}Tr}_2$. The labels along edges indicate the *minimum travel-time* needed to traverse a given edge $e_{sk}$, which can be readily calculated as $l(e_{sk})/v_{sk}^{max}$. The time labels next to the vertices indicate the time when a particular moving object is at a given vertex.

We note that $Q_1$, the first point of trajectory $\boldsymbol{RN\text{-}Tr}_q$ does not coincide with a vertex (intersection) of a graph. Indeed, a vehicle may start a trip from a parking spot on a street in between two intersections. Also, we observe that the motion along the road network trajectories may be slower than the maximum speed. For example, the last leg $(G_3, G_4)$ of $\boldsymbol{RN\text{-}Tr}_1$ is an edge that can be traversed in 15 time units, however, the trip along that edge for $\boldsymbol{RN\text{-}Tr}_1$ takes $\Delta = 55 - 35 = 20$ time units.

The main consequence of the model of road network trajectories in the context of our work is that the distance between two moving objects can *no longer* be measured using the 2D Euclidian distance ($L_2$-norm) since the objects are constrained to move along the edges of the road network. Instead we need to rely on the *shortest network distance* which, in turn, may have a two-fold interpretation ( [32, 48, 61, 80]):

1. shortest path distance, or
2. shortest travel-time distance.

The first interpretation captures the scenarios in which the length of the edges in the graph representing the road network are used to calculate the distance between two adjacent vertices. This corresponds to the cases where the trips are planned in a manner in which the goal is to minimize the total mileage travelled.
The second interpretation captures the fact that traveling along longer road segments but with higher maximum speed may yield a shorter overall duration of a given trip. In more dynamic settings, the travel-time based distance is also used to reflect different durations of a trip due to fluctuations in traffic density [13].

In the example scenario depicted in Figure 12, we have used shortest travel-time distance for labelling the edges, but the actual travel-time values that can be used to determine the distance(s) can be obtained from the labels of the individual vertices.

Assuming an undirected road network graph, the distance between two trajectories at any *given time-instant* $t$ will consist of the following three components [32]:

1. The time taken by the object to get to the first vertex along the shortest path.
2. The time taken by the object to get to the vertex nearest to the location of the object along the second trajectory at $t$.
3. The time taken by the object to get from that vertex to the actual location at $t$.

For the purpose of evaluating an NN-query, the following two key assumptions (cf. [32]) are used in the calculation of the travel-time distance between two objects:

1. If a particular edge belongs to the trajectory, then we use the travel-time distance information from the *trajectory data* itself, although it may be longer than the one obtained using the maximum speed and the edge length.

2. If a particular edge does *not* belong to the trajectory, we use the minimum travel-time (i.e., corresponding to the maximum speed motion along that edge) value.

In the example of Figure 12, the shortest path between trajectories $\boldsymbol{RN\text{-}Tr}_q$ and $\boldsymbol{RN\text{-}Tr}_1$ will consist of their individual corresponding portions, along with the edge $(Q_2, G_2)$ for as long as:

- $\boldsymbol{RN\text{-}Tr}_q$ is anywhere along the segments $(Q_1, Q_2)$ and $(Q2, A_{Q1})$
- $\boldsymbol{RN\text{-}Tr}_1$ is anywhere along the corresponding segments (for those time-values) $(G_1, G_2)$ and $(G2, A_{11})$

Let $o_Q$ denote the object moving along $\boldsymbol{RN\text{-}Tr}_q$, $o_1$ denote the object traveling along $\boldsymbol{RN\text{-}Tr}_1$, and $o_2$ denote the object traveling along $\boldsymbol{RN\text{-}Tr}_2$. As soon as $o_Q$ is located at $A_{q1}$ and $o_1$ is located at $A_{11}$, their shortest path distance will be obtained via the edge $(Q_3, G_3)$. A straightforward calculation yields that $o_Q$ will be at $A_{q1}$ and $o_1$ will be at $A_{11}$ at time $t = 20$.

Similarly, as soon as $o_Q$ is located at $A_{q2}$ and $o_1$ is located at $A_{12}$, the role of $o_1$'s nearest neighbor is taken by $o_2$—from $t = 40$ until the end of the trip.

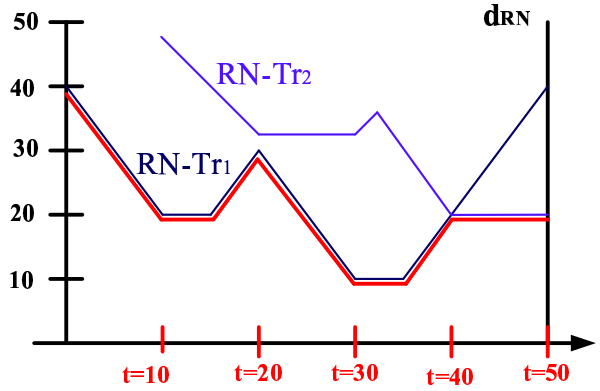


**Fig. 13** Continuous travel-time distance of trajectories on Road Network

An important consequence of the model of motion along a road network is that, in contrast to the case of free motion in 2D Euclidian space, the distance function between two trajectories is *piece-wise linear* (cf. [32]). In addition to the intuitively-expected changes in the instances in which the motion of a particular object changes from one edge to another at a given intersection, the cusps (non-differentiable points) of the distance function may occur in a time-instant in which an object is somewhere along a single edge. The reason for this phenomenon is that the edges that constitute the shortest path have changed. In the example of Figure 12, this happens to the distance between $o_Q$ and $o_1$ when $o_Q$ reaches location $A_{q1}$ along $\boldsymbol{RN\text{-}Tr}_q$, at $t = 20$.

Figure 13 shows the network distances of $\boldsymbol{RN\text{-}Tr}_1$ and $\boldsymbol{RN\text{-}Tr}_2$ from $\boldsymbol{RN\text{-}Tr}_q$ as a function of time, along with their lower envelope, throughout the period between $t = 0$ and $t = 50$. Although we do not have hyperbolae (cf. Section 4), Algorithm 1 and Algorithm 2 can be used on the line-segments verbatim. Just as importantly, since two line segments can intersect in at most one point, we can apply the theory of Davenport-Schinzel sequences [63] to obtain a bound of $\lambda_2(N) = 2N - 1 = O(N)$ on the combinatorial complexity of the lower envelope within a time-interval in which each distance function is continuous and monotonic (thereby, differentiable).

However, a bit of extra caution is needed to justify the upper bound on the time-complexity for constructing the lower envelope. Assume that a given trajectory $\boldsymbol{RN\text{-}Tr}_i$ has $m_i$ segments and recall that, according to Definition 3, all except the first and the last segment coincide with the edges of the graph representing the road network. As we noted, each of those trajectories can have an "extra cusp" (i.e., a point at which the distance function to the respective query-trajectory is non-differentiable). Hence, in addition to the vertices from the trajectories, we have to account that there may be an extra $O(m_i)$ critical time instants for the distance function between $\boldsymbol{RN\text{-}Tr}_i$ and the querying trajectory $\boldsymbol{RN\text{-}Tr}_q$ when calculating the lower envelope. However, adding a factor of 2 retains the upper bound $O((\sum m_i) \cdot N \log N)$ of the time complexity of constructing the lower envelope of the distance functions in the road network settings.

### 5.2 Uncertainty and Road Network Trajectories

An important aspect of uncertainty in the road networks settings is the *coupling* of its physical nature with the physical nature of the distance function used. Consider the following two cases:

1. The network distance between two trajectories is based on the length of the edges representing the road segments and the uncertainty of each object's location at each time instant is bounded by a fixed length.
2. The network distance between two trajectories is based on the travel times along the edges representing the road segments and the uncertainty of the object's location at each time instant is bounded by a fixed duration.

In each case, for as long as the instantaneous *pdf* satisfies the appropriately modified requirement from Section 3, i.e., the *pdf* is *axially symmetrical around the perpendicular to the expected location*, it can be readily demonstrated that the main results from Sections 3 and 4 hold in the road network settings. In other words, we can use the IPAC-NN tree to represent the answer to the NN-query and, in order to construct it we can again rely on Algorithm 3 (along with Theorem 2). Moreover, the pruning power of the IPC-NN tree is retained. As discussed in Section 5.1, the only modification is the appropriate definition of the distance function in Algorithm 1 and Algorithm 2.

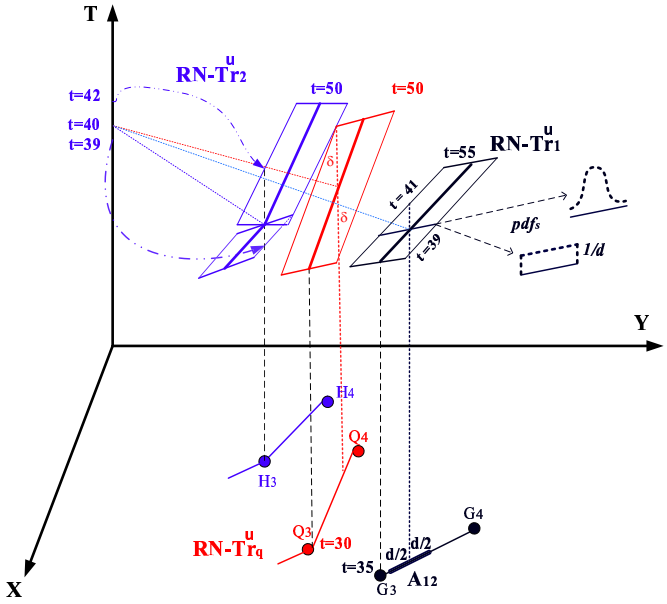


**Fig. 14** Uncertain trajectories in road networks

An especially interesting scenario occurs when the nature of the distance function used is different from the nature of the uncertainty parameter. As an illustration, assume that the travel-time distance is being used, and consider the following definition of uncertain road network trajectory.

**Definition 4** An *uncertain road network trajectory* $\boldsymbol{RN\text{-}Tr}_i^{\mathbf{u}}$ is a trajectory of an object moving along a road network augmented with: (1) information about the *distance* bounding the *possible locations of the object along the edges of the graph* a given time instant, denoted with $d$; and (2) the probability distribution function (*pdf*) of the location of the object within the uncertainty region.

From the previous definition, an uncertain road network is given by $\boldsymbol{RN\text{-}Tr}_i^{\mathbf{u}} = \{oid_i, d, pdf, (x_{i_1}, y_{i_1}, t_{i_1}), (x_{i_2}, y_{i_2}, t_{i_2}), \ldots, (x_{i_k}, y_{i_k}, t_{i_k})\}$.

An illustrating scenario for Definition 4 is provided in Figure 14, which shows a portion of uncertain variants of each of the trajectories $\boldsymbol{RN\text{-}Tr}_q$, $\boldsymbol{RN\text{-}Tr}_1$ and $\boldsymbol{RN\text{-}Tr}_2$ used in Figure 12. Looking at $\boldsymbol{RN\text{-}Tr}_1^{\mathbf{u}}$, for the *expected location* $A_{12}$, the object can be anywhere within the line segment centered at $A_{12}$ and stretching for $d/2$ toward vertices $G_3$ and $G_4$. As shown in Figure 4, the object $o_1$ can have different *pdf*'s along that line segment. We make the following observations:

1. Object $o_1$ can be at location $A_{12}$ at any time instant between $t = 39$ and $t = 41$. Since the dependency between the distance and time domains is linear, with constant factor $d/v_{max}$, it follows that the *pdf* of the variable describing the time bounds for a given location will have the same shape as the *pdf* of the location uncertainty (cf. Definition 4), with possibly different parameter values. In other words, if the *pdf* of the location at a given time instant is uniform (respectively, Gaussian), the *pdf* of the temporal random variable describing the possible time values for which the object can be at a given location will also be uniform (respectively, Gaussian) [74].
2. Although the physical length is the same for both $\boldsymbol{RN\text{-}Tr}_q^{\mathbf{u}}$ and $\boldsymbol{RN\text{-}Tr}_1^{\mathbf{u}}$, when looking at the location along the edge $(Q_3, Q_4)$ where $o_Q$ is expected to be at $t = 40$, the temporal interval $[40-\delta, 40+\delta]$ happens to be much larger than the corresponding temporal interval $[39, 41]$ of $o_1$. Looking at the slopes of the corresponding trajectories' segments, this follows from the fact that $o_Q$, whose trajectory is $\boldsymbol{RN\text{-}Tr}_q^{\mathbf{u}}$, is moving at a slower speed than $o_1$, whose trajectory is $\boldsymbol{RN\text{-}Tr}_1^{\mathbf{u}}$.
3. Observe that near the vertices of the graph, since the incident trajectory segments can have different speeds, it may be the case that the temporal error bounds may be different "before" and "after" a particular time-instant, as is illustrated in the vicinity of the point $H_3$ in Figure 14.

In general, the main consequence of the above observations is that the temporal uncertainty will have variable bounds along different trajectory segments. To analyze the impact on the calculation of the ranking of the answers to an uncertain NN-query, we proceed as follows. Let $\delta_0$ denote the (half) bound on the temporal uncertainty of the querying trajectory $\boldsymbol{RN\text{-}Tr}_q^{\mathbf{u}}$ at a particular time-instant, and let $\delta_i$ denote the corresponding bounds for the trajectory $\boldsymbol{RN\text{-}Tr}_i^{\mathbf{u}}$. Depending on the (relative) values of $\delta_0$ and $\delta_i$ (for $i = 1, 2, \ldots, N$), a situation may arise in which although the expected location of a given trajectory $\boldsymbol{RN\text{-}Tr}_j^{\mathbf{u}}$ is *not* the nearest neighbor of the querying trajectory, the

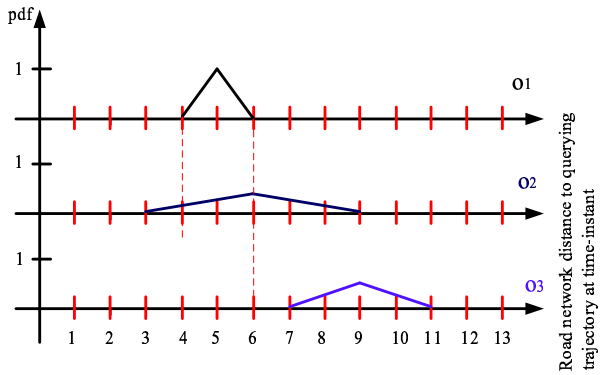


**Fig. 15** Travel-time distance and within-distance *pdf*s

corresponding object $o_j$ may still have a nonzero probability of being the nearest neighbor of $o_Q$.

*Example 7* Consider the scenario illustrated in Figure 15, which shows a snapshot of the distance of three objects, $o_1$, $o_2$ and $o_3$, to the query object $o_Q$ at some time instant $\tau$. Assume that the uncertainty in the temporal domain for each of the objects, including $o_Q$, is uniform, with the following values: $\delta_0 = 1/2$; $\delta_1 = 1/2$, $\delta_2 = 5/2$, and $\delta_3 = 3/2$. Figure 15 shows the status after computing (the *pdf*s of) the convolution of each object $o_i$ ($i \in \{1, 2, 3\}$). Since the closest possible point in time-distance of $o_3$, which is at distance $9 - (\delta_3 + \delta_0)$ $= 9 - 2 = 7$, is further away than the furthest possible point of $o_1$ from $o_Q$, which is $5 + (\delta_1 + \delta_0) = 5 + 1 = 6$, object $o_3$ has zero probability of being a nearest neighbor to $o_Q$ at $\tau$. However, although the expected value of the distance of $o_2$ from $o_Q$ is 6, given that: $6 - (\delta_0 + \delta_2)$ $= 6 - 3 = 3 < 5 - (\delta_0 + \delta_1) = 5 - 1 = 4$ – not only does $o_2$ have a non-zero probability of being $o_Q$'s nearest neighbor at $\tau$, but it also has a chance of being the actual nearest neighbor of $o_Q$, depending on the mutual whereabouts of the distances of $o_2$ and $o_1$ to $o_Q$, it may be the case that $o_2$ has a higher probability of being the nearest neighbor.

The scenario presented in Example 7 is actually a straightforward consequence of adapting the Equations 3 and 5 from Section 2 to the current settings of motion along a road network. Since the travel-time distance is one dimensional, Equation 3 needs to be modified as follows:

$$P^{WD}_{o_i,o_Q}(T_d) = \int_{l_{exp}-(\delta_0+\delta_i)}^{l_{exp}+(\delta_0+\delta_i)} pdf_i^{conv}(t_d)\, dt_d \qquad (10)$$

where the variable $T_d$ (respectively, $t_d$) denotes the the travel-time distance (respectively, its differentiation variable) between $o_i$ and $o_Q$, and $pdf_i^{conv}$ denotes the *pdf* of the convolution of the respective random variables corresponding to the uncertainties of $o_i$ and $o_Q$.

In the context of Example 7, if we want to know for which travel-time distance values objects $o_1$ and $o_2$ have the same probability of being the nearest neighbor of $o_Q$ at $\tau$, we need to solve the following equation:

$$\int_0^\infty pdf^{WD}_{o_1,o_Q}(T_d) \cdot (1 - P^{WD}_{o_2,o_Q}(T_d))\, dT_d = \int_0^\infty pdf^{WD}_{o_2,o_Q}(T_d) \cdot (1 - P^{WD}_{o_1,o_Q}(T_d))\, dT_d \qquad (11)$$

Equation 11 essentially presents a generic form of stating that a value of the (road network) distance needs to be determined in which the probability of $o_1$ being within certain distance from $o_Q$ together with the probability of $o_2$ *not* being within that distance, is same as the probability of $o_2$ being within certain distance from $o_Q$ together with the probability of $o_1$ *not* being within that distance. Given the convolutions in the Example 7, this amounts to finding the value such that the areas of the subsets of the respective triangles representing the *pdf*s of $o_1$ and $o_2$ to the left of that value, are equal. For the specific settings at hand (cf. Figure 15), this yields a quadratic equation with solutions 15/4 (which is out of bounds) and 9/2. Thus, when the value of the travel-time distance from $o_Q$ is greater then 9/2, the highest-probability nearest neighbor of $o_Q$ is $o_1$. As we mentioned in Section 2.3, the main reason that we are justified to use Equation 11 is that we are considering strictly the **1**NN case. For the $k$-NN cases where $k \geq 2$, we would need to augment each side of the equation with the respective terms capturing the possibility of the distance functions of pairs of objects having the same within-distance probabilities.

### 5.3 Semantics of the Answers of Uncertain NN-Query on Road Networks

Based on the observations from Section 5.2, we now analyze the impact of the road network settings on the structure of the answer to the NN-query for uncertain trajectories, along with its computation. First, we present the modifications to the construction and interpretation of the lower envelope for uncertain trajectories and, subsequently, we discuss the modifications of the corresponding IPAC-NN tree.

Consider the scenario depicted in Figure 16, which shows the expected distances and uncertainty regions for four trajectories $\boldsymbol{RN\text{-}Tr}^{\mathbf{u}}_1$, $\boldsymbol{RN\text{-}Tr}^{\mathbf{u}}_2$, $\boldsymbol{RN\text{-}Tr}^{\mathbf{u}}_3$ and $\boldsymbol{RN\text{-}Tr}^{\mathbf{u}}_4$ from the querying trajectory $\boldsymbol{RN\text{-}Tr}^{\mathbf{u}}_q$ throughout the time-interval $[t_1, t_2]$ during which each distance is monotonic and continuous. As shown, the lower envelope consists of $\boldsymbol{RN\text{-}Tr}_1$ between $t_1$ and

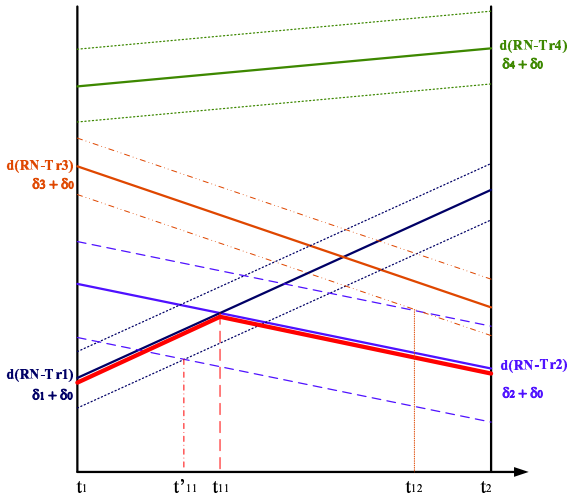


**Fig. 16** Lower envelope and nearest neighbor in road network settings

$t_{11}$, followed by $\boldsymbol{RN\text{-}Tr}_2$ between $t_{11}$ and $t_2$. Observe that the relative difference of the distance values between $\boldsymbol{RN\text{-}Tr}_4$ and $\boldsymbol{RN\text{-}Tr}_1$ is greater than $(\delta_0 + \delta_1) + (\delta_4 + \delta_0)$ throughout $[t_1, t_{11}]$. Similarly, the relative difference between $\boldsymbol{RN\text{-}Tr}_4$ and $\boldsymbol{RN\text{-}Tr}_2$ is greater than $(\delta_0 + \delta_2) + (\delta_4 + \delta_0)$ throughout $[t_{11}, t_2]$. Hence, we can readily conclude that $\boldsymbol{RN\text{-}Tr}_4$ has zero probability of being a nearest neighbor to $\boldsymbol{RN\text{-}Tr}_q$ throughout $[t_1, t_2]$. Complementary to this, we observe that there exist a time-instant $t_{12} \in [t_{11}, t_2]$, starting at which the value of $(\delta_0 + \delta_2) + (\delta_3 + \delta_)$ is larger than the value of the difference between the respective distance functions of $\boldsymbol{RN\text{-}Tr}_3$ and $\boldsymbol{RN\text{-}Tr}_2$. This, in turn, implies that $\boldsymbol{RN\text{-}Tr}_3$ *does* have a non-zero probability of being a nearest neighbor to $\boldsymbol{RN\text{-}Tr}_q$ throughout $[t_1, t_2]$.

To generalize and compare these observations with the corresponding results in Section 4, let $\boldsymbol{RN\text{-}Tr}_i$ denote the trajectory whose segment is defining the lower envelope of the distances to $\boldsymbol{RN\text{-}Tr}_q$ throughout a time-interval $[t_{i1}, t_{i2}]$. Also, let $d(\boldsymbol{RN\text{-}Tr}_i, \boldsymbol{RN\text{-}Tr}_j)$ denote the difference of respective distances of $\boldsymbol{RN\text{-}Tr}_i$ and $\boldsymbol{RN\text{-}Tr}_j$ from $\boldsymbol{RN\text{-}Tr}_q$. If $\boldsymbol{RN\text{-}Tr}_j$ satisfies:

$$(\delta_j + \delta_0) + (\delta_i + \delta_0) < d(\boldsymbol{RN\text{-}Tr}_i, \boldsymbol{RN\text{-}Tr}_j) \qquad (12)$$

then it has zero probability of being a nearest neighbor of $\boldsymbol{RN\text{-}Tr}_q$.

The impact of the road network settings on the pruning aspects of the lower envelope, when compared to the corresponding results in Section 4, is that each monotonic and continuous interval of the lower envelope may have a *different* upper-bound on the pruning value. To capture this, we can readily use the *Descriptor* field in each of the nodes in the IPAC-NN tree and, as far as its geometric dual – the lower envelope – is concerned, we can augment the data structure used to represent each segment of the distance function with a real-valued attribute that will store the corresponding $\delta$'s.

Recall the observation in Example 7 regarding the impact of the combination of the relative positions of the expected values and the values of the convolutions of the distance functions for the respective object. A continuous version of it is illustrated throughout the interval $[t'_{11}, t_{11}]$ in Figure 16. Namely, although $\boldsymbol{RN\text{-}Tr}_2$ begins to define the lower envelope of the distance from $\boldsymbol{RN\text{-}Tr}_q$ at $t_{11}$, due to the large enough value of $\delta_0 + \delta_2$, it has a chance of being the nearest neighbor as early as $t'_{11}$. This, in general, implies that additional specifications are needed in order to properly describe which object is the actual NN-answer. Once again, we rely on the *Descriptor* field of the IPAC-NN tree. More specifically, in the context of Figure 16, the leftmost node of the first level of the IPAC-NN tree:

$[\boldsymbol{URN\text{-}Tr}_1, (t_1, t_{11})]$

will be augmented with the following two fields:

- **Condition:**

$$pdf^{WD}_{o_1,o_Q}(T_d) \cdot (1 - P^{WD}_{o_2,o_Q}(T_d))\, dT_d < pdf^{WD}_{o_2,o_Q}(T_d) \cdot (1 - P^{WD}_{o_1,o_Q}(T_d))\, dT_d \qquad (13)$$

- **Duration:**

$[t'_{11}, t_{11}]$

to indicate that throughout $[t'_{11}, t_{11}]$, although the distance of the expected location along $\boldsymbol{RN\text{-}Tr}_2$ is still further than the corresponding value of $\boldsymbol{RN\text{-}Tr}_1$ from $o_Q$, it may be the nearest neighbor with the highest probability value, provided that **Condition** attribute (Equation 13) holds. We note that $t'_{11}$ can be obtained as the value of the intersection of the segment specifying the expected location of $o_1$ translated by $\delta_0 + \delta_1$, with the segment specifying the expected location of $o_2$ translated by $\delta_0 + \delta_2$. However, solving the **Condition** attribute – although reducible to quadratic equation at a given time-instant for uniform *pdf*s, in general may require numerical methods (e.g., if *pdf*s are Gaussian).

The last topic that we address in this section is the computation of the entire IPAC-NN tree for the case of uncertain trajectories on road networks. As it can be readily verified, a property similar to Theorem 2 holds for the case of trajectories moving on road networks. This, in turn enables us to use the collection of lower envelopes of linear segments, with the following adaptations:

– Firstly, we trivially observe that a function *RN-Env2*$(RN\text{-}Tr_i, RN\text{-}Tr_j, t_1, t_2)$, which calculates

the lower envelope of two line segments between the times of $t_1$ and $t_2$ will complete in $O(1)$.
- Secondly, as a consequence of the above, Algorithm 1 can be used verbatim, whereas Algorithm 2 can be used almost verbatim. Namely, we need to appropriately change the line:
$LE_{1,2} = LE_{1,2} \odot Env2(TR_{1i_k}, TR_{2j_r}, t^{cl}, t^{cu})$
of Algorithm 2 into:

$LE_{1,2} = LE_{1,2} \odot$ *RN-Env2*$(RN\text{-}Tr_{1i_k}, RN\text{-}Tr_{2j_r}, t^{cl}, t^{cu})$

When it comes to the construction of the overall collections of lower envelopes that can be used to generate the complete answer to the NN-query throughout the time interval of interest, the main complexity results from Section 4 are same in road network settings. As we discussed, the main difference will be in the values used for pruning throughout the sub-intervals consisting of a single monotone and continuous segment. We re-iterate that a peculiarity of the current settings is that the adapted versions of the Algorithms from Section 4 will need to augment the data structures used, in order to cater for the values in the *Descriptor* field of the geometric dual of the lower-envelope, the IPAC-NN tree.

## 6 Variants of the Uncertain NN-Query

We now address the issue of increasing MOD capabilities for continuous NN queries for uncertain trajectories, building upon the results from the previous sections and, towards that end, we present a suite of new predicates, along with the algorithms for their processing. Essentially, we are exploring different variations of the SQL query presented in Section 1, reflected through the corresponding argument-signatures of the predicates. In the sequel, we firstly discuss the syntactic variants and processing algorithms in the context of a specific trajectory, and we follow with a discussion for the entire MOD.

Deviating slightly from the standard terminology of logic programming and deductive databases [75], in the sequel we will assume that symbols that start with a lower_case, along with the symbols which have subscript/index, denote constants from the respective domains. We will use symbols starting with an Upper_case letter and without subscripts to denote variables. In addition, for brevity, we will use $LE_B$ to denote the " Belt" of the lower envelope – which is, the zone bounded by it and its $4r$-translated copy in the *(time, distance)* space.

Given a querying trajectory $\boldsymbol{Tr}_q^{\mathbf{u}}$, we may be interested in the *possibility* of a particular trajectory $\boldsymbol{Tr}_i^{\mathbf{u}}$ being its nearest neighbor throughout a time-interval of interest, $[t_b, t_e]$. However, the term "possibility" is not precisely defined – namely, it can pertain to a time-instant or a sub-interval, and it may also concern the relative ranking of $\boldsymbol{Tr}_i^{\mathbf{u}}$ among all the other possible nearest neighbors of $\boldsymbol{Tr}_q^{\mathbf{u}}$.

The first category of predicates address the nearest neighbor status of a particular uncertain trajectory with respect to the querying trajectory *at a given time-instant*:

- $\mathbf{C}_{11}$: $\mathsf{PossibleNN}(\mathbf{Tr}_i^{\mathbf{u}}, \mathbf{Tr}_q^{\mathbf{u}}, t_b, t_e, t)$
this predicate verifies whether $\boldsymbol{Tr}_i^{\mathbf{u}}$ has a non-zero probability of being the nearest neighbor of $\boldsymbol{Tr}_q^{\mathbf{u}}$ at the time-instant $t$.
- $\mathbf{C}_{12}$: $\mathsf{PossibleNN}(\mathbf{Tr}_i^{\mathbf{u}}, \mathbf{Tr}_q^{\mathbf{u}}, t_b, t_e, t, k)$
this predicate verifies whether $\boldsymbol{Tr}_i^{\mathbf{u}}$ has the $k$-th highest probability of being the nearest neighbor of $\boldsymbol{Tr}_q^{\mathbf{u}}$ at the time-instant $t$.

The algorithms for processing the $\mathbf{C}_{11}$ and $\mathbf{C}_{12}$ predicates are relatively straightforward:

1. For the $\mathbf{C}_{11}$ $\mathsf{PossibleNN}(...)$ predicate, we need to check whether the expected location of the trajectory $\boldsymbol{Tr}_i^{\mathbf{u}}$ at $t$, is within distance $\leq R_{max} + 4r$ from the expected location of $\boldsymbol{Tr}_q^{\mathbf{u}}$ at that same time-instant $t$. This is equivalent to checking whether the value of the distance between the two expected location is inside the zone $LE_B$ at $t$. For a collection of $N$ trajectories, this will clearly require a running time bounded by $O(N \log N)$. Note however, that this cost will be amortized when other trajectories are checked against the same $\boldsymbol{Tr}_i^{\mathbf{u}}$, since for each of the rest of them, the verification of the $\mathbf{C}_{11}$ $\mathsf{PossibleNN}(...)$ can be achieved in $O(1)$ time.
2. For the $\mathbf{C}_{12}$ $\mathsf{PossibleNN}(...)$ predicate, in addition to verifying that (at time $t$), the expected location of $\boldsymbol{Tr}_i^{\mathbf{u}}$ is inside the permissible zone, bounded by the lower envelope and its $4r$-translated copy, we also need to verify that it is exactly the $k$-th one in the distance from the lower envelope. In other words, we need to verify that the node of the *IPAC-NN* tree which has an entry labelled $\boldsymbol{Tr}_i^{\mathbf{u}}$ at time $t$, is at the depth $k$. This amounts to traversing a single-path (the time-interval containing $t$) along the *IPAC-NN* tree. Assuming a fan-out factor of $f$ for the internal nodes, and based on Theorem 2, this yields a time-complexity of $O(\log_f(\lceil N/K \rceil^2))$.

The next two predicates pertain to the properties of a particular uncertain trajectory with respect to a querying trajectory *throughout a portion of the time-interval*. To express these predicates concisely, we introduce the concept of a *$\phi$-portion* of an interval $[t_b, t_e]$ for a given $0 \leq \phi \leq 1$, which denotes a finite sequence of disjoint subintervals of $[t_b, t_e]$ whose total length is $\phi(t_e - t_b)$.

– $\mathbf{C}_{21}$: PossibleNN-Int($\mathbf{Tr}_i^{\mathbf{u}}, \mathbf{Tr}_q^{\mathbf{u}}, t_b, t_e, \phi$)
this predicates verifies whether $\boldsymbol{Tr}_i^{\mathbf{u}}$ has a non-zero probability of being the nearest neighbor of $\boldsymbol{Tr}_q^{\mathbf{u}}$, for at least a $\phi$-portion of the interval $[t_b, t_e]$.
– $\mathbf{C}_{22}$: PossibleNN-Int($\mathbf{Tr}_i^{\mathbf{u}}, \mathbf{Tr}_q^{\mathbf{u}}, t_b, t_e, \phi, \ell$)
this predicate verifies whether $\boldsymbol{Tr}_i^{\mathbf{u}}$ has the $\ell$-th highest probability of being the nearest neighbor of $\boldsymbol{Tr}_q^{\mathbf{u}}$, for at least a $\phi$-portion of the interval $[t_b, t_e]$.

The algorithm for evaluating the $\mathbf{C}_{21}$: PossibleNN-Int($\mathbf{Tr}_i^{\mathbf{u}}, \mathbf{Tr}_q^{\mathbf{u}}, t_b, t_e, \phi$) predicate is as follows:

**Algorithm 4** $\mathbf{C}_{21}$ – Evaluating PossibleNN-Int

PossibleNN-Int ($\boldsymbol{Tr}_i^{\mathbf{u}}, \boldsymbol{Tr}_q^{\mathbf{u}}, t_b, t_e, \phi$)
**Input:** *Uncertain trajectory $\boldsymbol{Tr}_i^u$, Uncertain querying trajectory $\boldsymbol{Tr}_q^u$, time-interval of interest $[t_b, t_e]$, temporal fraction of interest $\phi$*
**Output: True/False**

*Let $\{t_1, t_2, \ldots t_s\}$ denote all the intersection times of the distance function between $\boldsymbol{Tr}_i$ and $\boldsymbol{Tr}_q$ with the boundaries of the zone $LE_B$;*
*Let count = 1, Total = 0;*
**If** *($\overline{\boldsymbol{Tr}_i \boldsymbol{Tr}_q}$ inside $LE_B$ at $t_b$)*
  *then count = 0;*
**while** *(count $\leq s$)*
  { *Total = Total + $t_{(count+1)} - t_{count}$;*
    *count = count + 2;* }
**If** *(Total $\geq \phi \cdot (t_e - t_b)$)*
  *return* **True**;
*return* **False**;

Essentially, Algorithm 4 needs to consider the intersection-times of the distance function between $\boldsymbol{Tr}_i$ and $\boldsymbol{Tr}_q$ with the upper bound ($4r$ translated copy of the lower envelope) of $LE_B$. By the monotonicity properties of the distance function (hyperbola), it follows that there can be at most two such intersections with a given segment of the boundary. Since the combinatorial complexity of the boundaries of $LE_B$ is $O(N)$, after their initial construction ($O(N \log N)$), we obtain that the time-complexity of Algorithm 4 is $O(N)$.

The evaluation of the $\mathbf{C}_{22}$: PossibleNN-Int($\mathbf{Tr}_i^{\mathbf{u}}, \mathbf{Tr}_q^{\mathbf{u}}, t_b, t_e, \phi, \ell$) predicate is specified in Algorithm 5.

Based on Theorem 2, an equivalent specification of the Algorithm 5 could have been given relying on the envelopes inside the $LE_B$ zone. As for its running time, the worst-case complexity is bound by $O(f^\ell + \ell)$, where $\ell \leq \log_f O(\lceil N/K \rceil^2)$. We note however, that further improvements are possible in the sense of pruning parts of the *IPC-NN* tree from the search – namely, if a node with a label $\boldsymbol{Tr}_i^{\mathbf{u}}$ is encountered at depth $l < \ell$, then the entire subtree rooted at that node can be eliminated.

**Algorithm 5** $\mathbf{C}_{22}$ – Evaluating PossibleNN-Int

PossibleNN-Int ($\boldsymbol{Tr}_i^{\mathbf{u}}, \boldsymbol{Tr}_q^{\mathbf{u}}, t_b, t_e, \phi, \ell$)
**Input:** *Uncertain trajectory $\boldsymbol{Tr}_i^u$, Uncertain querying trajectory $\boldsymbol{Tr}_q^u$, time-interval of interest $[t_b, t_e]$, temporal fraction of interest $\phi$, rank $\ell$*
**Output: True/False**

*Traverse the nodes at depth $\ell$ in the IPAC-NN tree;*
*Let $IN_1, IN_2, \ldots, IN_s$ denote all the nodes at the depth $\ell$ in the IPAC-NN tree that are labelled with $\boldsymbol{Tr}_i^u$;*
*Let $[t_{i1}^{IN}, t_{i2}^{IN}]$ denote the time-interval in the label of the i-th such node;*
*Let j = 1, Total = 0;*
**while** *($j \leq s$)*
  { *Total = Total + $t_{j2}^{IN}$ - $t_{j1}^{IN}$;*
    *count++;* }
**If** *(Total $\geq \phi \cdot (t_e - t_b)$)*
  *return* **True**;
*return* **False**;

Regarding objects with uncertain trajectories moving along a road network, we have a corresponding variant for each of the predicates discussed above, namely:

– $\mathbf{C}_{11}^{RN}$: PossibleNN($(\mathbf{Tr}_i^{\mathbf{u}}, \mathbf{Tr}_q^{\mathbf{u}}, t_b, t_e, t)$, RoadNet)
this predicate verifies whether $\boldsymbol{RN\text{-}Tr}_i^{\mathbf{u}}$ has a non-zero probability of being the nearest neighbor of $\boldsymbol{RN\text{-}Tr}_q^{\mathbf{u}}$ at the time-instant $t$, when the motion of both objects is constrained by a given road-network RoadNet.
– $\mathbf{C}_{12}^{RN}$: PossibleNN($(\mathbf{Tr}_i^{\mathbf{u}}, \mathbf{Tr}_q^{\mathbf{u}}, t_b, t_e, t, \ell)$, RoadNet)
this predicate verifies whether $\boldsymbol{RN\text{-}Tr}_i^{\mathbf{u}}$ has the $\ell$-th highest probability of being the nearest neighbor of $\boldsymbol{RN\text{-}Tr}_q^{\mathbf{u}}$ at the time-instant $t$, when the motion of both objects is constrained by a given road-network RoadNet.
– $\mathbf{C}_{21}^{RN}$: PossibleNN-Int($(\mathbf{Tr}_i^{\mathbf{u}}, \mathbf{Tr}_q^{\mathbf{u}}, t_b, t_e, \phi)$, RoadNet)
this predicates verifies whether $\boldsymbol{RN\text{-}Tr}_i^{\mathbf{u}}$ has a non-zero probability of being the nearest neighbor of $\boldsymbol{RN\text{-}Tr}_q^{\mathbf{u}}$, for at least a $\phi$-portion of the interval $[t_b, t_e]$, where the motion of both objects is constrained by a given road-network RoadNet.
– $\mathbf{C}_{22}^{RN}$: PossibleNN-Int($(\mathbf{Tr}_i^{\mathbf{u}}, \mathbf{Tr}_q^{\mathbf{u}}, t_b, t_e, \phi, \ell)$, RoadNet)
this predicate verifies whether $\boldsymbol{RN\text{-}Tr}_i^{\mathbf{u}}$ has the $\ell$-th highest probability of being the nearest neighbor of $\boldsymbol{RN\text{-}Tr}_q^{\mathbf{u}}$, for at least a $\phi$-portion of the interval $[t_b, t_e]$, where the motion of both objects is constrained by a given road network RoadNet.

The individual steps of the algorithms used for processing/verifying $\mathbf{C}_{11}^{RN}$, $\mathbf{C}_{12}^{RN}$, $\mathbf{C}_{21}^{RN}$ and $\mathbf{C}_{22}^{RN}$ are exactly the same as the corresponding ones used for the processing/verifying of $\mathbf{C}_{11}$, $\mathbf{C}_{12}$, $\mathbf{C}_{21}$ and $\mathbf{C}_{22}$. However, there is "semantic" difference that stems from the nature of $LE_B$ in road network settings. Namely, as we discussed in Section 4, in the case of free 2D motion, $LE_B$ has a constant width ($4r$) and both of its boundaries (the lower-envelope and its $4r$-translated copy)

are continuous but not differentiable, i.e., each has the cusps at time-instants at which a change occurs in the trajectory that defines the lower envelope. In the case of motion along road-networks, while the lower-envelope of the distance itself is continuous (once again, not differentiable at cusps), the upper-bound is not. The main reason is that the width of the pruning-region can vary when different trajectories are defining different segments of the lower-envelope. As discussed in Section 5, this is a consequence of intermixing the travel-time distance with a spatially-defined uncertainty of the whereabouts of the objects. Thus, the respective comparisons in each of the algorithms will need to take this fact into account.

The predicates described above, together with the concepts introduced in Section 4, can be used as basic building blocks for answering queries related to the nearest neighbor property for uncertain trajectories in MOD settings. We note that the comprehensive (optimization of the) query processing requires a combination of *filtering* and *refinement* stages [23] – however, the issues related to indexing and optimization are beyond the scope of this article. In the rest of this section, we present several variants of nearest neighbor queries for uncertain trajectories and we outline the processing of their respective post-filtering part, i.e., after the relevant subsets of the trajectories have been brought from secondary storage.

**Q1**: *Does a particular moving $oid_i$ object have a chance of being a nearest neighbor of $\boldsymbol{Tr}_q^{\boldsymbol{u}}$ at any time between $t_b$ and $t_e$*

This existential (in the temporal domain) variant can be specified as:

SELECT $*$
FROM MOD
WHERE PossibleNN($\mathbf{Tr}_i^{\mathbf{u}}, \mathbf{Tr}_q^{\mathbf{u}}, t_b, t_e, T$)
AND ($T$ BETWEEN $t_b$ AND $t_e$)

Since the query **Q1** is interested in *any* time-instant, and the predicate $\mathbf{C}_{11}$: PossibleNN($\mathbf{Tr}_i^{\mathbf{u}}, \mathbf{Tr}_q^{\mathbf{u}}, t_b, t_e, t$) has a specific value of $t$ in its argument signature, we use the conjunction: AND ($T$ BETWEEN $t_b$ AND $t_e$ ). This bounds the possible values of the variable $T$ to the time-values of interest for the query.

The processing of **Q1** amounts to checking whether the distance function of $\boldsymbol{Tr}_i^{\mathbf{u}}$ and $\boldsymbol{Tr}_q^{\mathbf{u}}$ has any intersections with $LE_B$ or its boundaries, throughout $[t_b, t_e]$, the complexity of which is $O(N)$ due to the combinatorial complexity of the $LE_B$. This, however, is in addition to the $O(N \log N)$ needed to construct $LE_B$.

An example of the universal-variant in the temporal domain is:

**Q1'**: *Does a particular moving $oid_i$ object have a chance of being a nearest neighbor of $\boldsymbol{Tr}_1^{\boldsymbol{u}}$ throughout the entire interval $[t_b, t_e]$*
can be specified as:
SELECT $*$ FROM MOD
WHERE PossibleNN-Int($\mathbf{Tr}_i^{\mathbf{u}}, \mathbf{Tr}_q^{\mathbf{u}}, t_b, t_e, 1$)

When processing **Q1'**, in contrast to processing the **Q1** query, one needs to check the conjunction of:

1. Is the value of the distance function of $\boldsymbol{Tr}_i^{\mathbf{u}}$ and $\boldsymbol{Tr}_q^{\mathbf{u}}$ inside $LE_B$ or on its boundaries at $t_b$.
2. Does the distance function between $\boldsymbol{Tr}_i^{\mathbf{u}}$ and $\boldsymbol{Tr}_q^{\mathbf{u}}$ have no other intersection with the boundaries of $LE_B$ throughout $(t_b, t_e)$.

We have the same time-complexity for processing **Q1'** as the one for processing **Q1** – $O(N)$, with the overhead of $O(N \log N)$ for constructing $LE_B$ .

The predicates and the queries that we discussed so far all pertained to a single trajectory in the MOD, with respect to a given querying trajectory. However, in practice, one is likely to be interested in detecting all the data items that satisfy a particular property. In our settings, this yields the following queries:
**Q2**: *Select all the moving objects that have a chance of being a nearest neighbor of $\boldsymbol{Tr}_q^{\boldsymbol{u}}$ at any time between $t_b$ and $t_e$*
SELECT **Tr**
FROM MOD
WHERE PossibleNN($\mathbf{Tr}, \mathbf{Tr}_q^{\mathbf{u}}, t_b, t_e, T$)
AND ($T$ BETWEEN $t_b$ AND $t_e$)

where, once again, use the conjunction: AND ($T$ BETWEEN $t_b$ AND $t_e$ ) to bound the possible values of the variable $T$.
**Q2'**: *Select all the moving objects that have a chance of being a nearest neighbor of $\boldsymbol{Tr}_q^{\boldsymbol{u}}$ throughout the entire interval $[t_b, t_e]$.*
SELECT **Tr**
FROM MOD
WHERE PossibleNN-Int($\mathbf{Tr}, \mathbf{Tr}_q^{\mathbf{u}}, t_b, t_e, \phi$)

Both of the respective running times for processing **Q2** and **Q2'** are upper-bounded by $O(\lceil N/K \rceil^2)$.

The most general form of a nearest neighbor query for a MOD with uncertain trajectories is:
**Q3**: *Select all the moving objects that have at least $\ell$-th highest probability of being a nearest neighbor of $\boldsymbol{Tr}_q^{\boldsymbol{u}}$, throughout at least $\phi$-fraction of the interval $[t_b, t_e]$.*
SELECT **Tr**
FROM MOD
WHERE PossibleNN-Int($\mathbf{Tr}, \mathbf{Tr}_q^{\mathbf{u}}, t_b, t_e, \Phi$, *Level*)
AND ($\Phi \geq \phi$ ) AND (*Level* $\leq \ell$)

With the exception of reporting the resulting answer-set, the time-complexity of processing the query **Q3** can

be retained at the level of processing the variant of the query that would verify the property for a single trajectory. However, there is the overhead of $O(\lceil N/K \rceil)$ space requirements to update the variables that maintain the total time for each individual trajectory as the *IPAC-NN* tree is being traversed at the depth $k$ and lower, which will impact the reporting of the result.

We note that in the case of a MOD in which the objects' motion is constrained by a given road network described as a graph, in each of the three types of queries above – **Q1**, **Q2** and **Q3** – we need to use the properly modified predicates in the WHERE clause, which will include the RoadNet parameter.

To conclude this section, we reiterate that at the heart of the efficiency gains for the processing of all the above queries is the pruning enabled by the lower envelope of the distance functions and its boundaries. As our experiments will demonstrates, these gains are significant when compared to the corresponding brute-force approaches.

## 7 Experiments

We now present our experimental evaluation of the benefits of our proposed methodology. We have developed a prototype implementation of our algorithms in Java, and we have performed experiments on an Intel Core-2 3.0GHz machine with RedHat Enterprise Linux. The source code and the data sets used for our experiments are available at http://www.eecs.northwestern.edu/∼goce/UncertainNN.

We considered datasets ranging from 1,000 to 12,000 moving objects. Their trajectories were generated using a modified version of the random waypoint model, where:

1. Each object starts its motion at a random $(x, y)$ location at a given time;
2. Each object randomly picks a direction and a speed between $15mph$ and $60mph$;
3. All the objects change their velocity vectors simultaneously every 3 minutes, in a manner that satisfies the following constraints:
   (a) The new direction of motion is within +/- 60° from the previous one;
   (b) The new speed is again in the range $[15, 60]mph$.
4. The total duration of the trips for the trajectories is 1 hour and the geographic area for the motion is a square region with a 40-mile side.

We re-iterate that the goal of this work is not to present efficient strategies for the overall NN query processing, which would involve indexing and data structures focusing on the effectiveness of the pruning at large (cf. [16, 24, 32]).

In the sequel, we present the results of the seven groups of experiments that we conducted. Since our algorithms outperform the corresponding naïve approaches by several orders of magnitude, in the following charts, the time axis is represented using a logarithmic scale.

The first group of experiments investigates the efficiency of computing the lower envelope of the distance functions using our algorithms presented in Section 4 when compared to the naïve approach, which, in principle, works as follows:

1. Find the intersections of all the distance functions;
2. Sort them along the temporal dimension;
3. Scan the sorted sequence, comparing the lowest values in-between intersections.

Since there are $O(N^2)$ such intersections, applying merge-sort in Step 2 above yields an upper bound of $O(N^2 \log N)$ on the running time. As expected from the upper-bounds on the complexities, our approach is significantly faster and, as shown in Figure 17, the improvements are larger as the number of trajectories in the dataset increases.

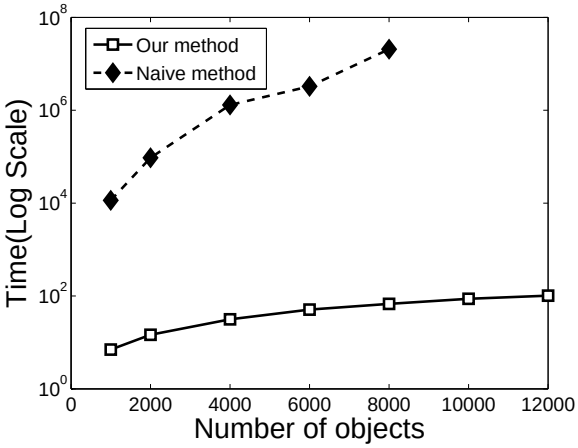


**Fig. 17** Construction of the Lower Envelope

The second group of experiments illustrates the pruning power of our approach in terms of the number of objects that can be eliminated from consideration in the refinement part of our algorithms for processing the query variants presented in Section 6. In Figure 18, we show the effect of pruning for two datasets, one with 2,000 trajectories and the other with10,000 trajectories. Here, the the Y-axis indicates the percentage of the trajectories that will need further processing during the refinement step.

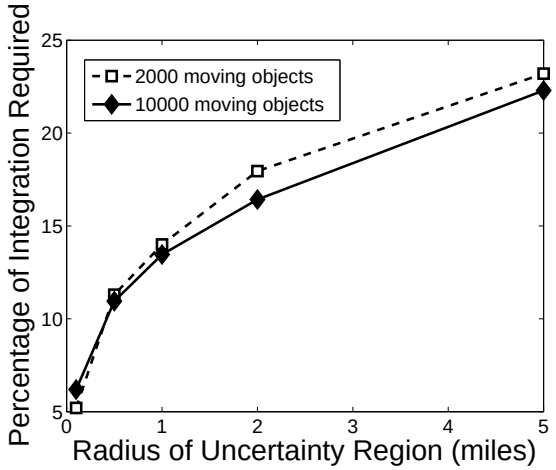


**Fig. 18** Pruning effects

As illustrated in the figure, for smaller values of the radius of the uncertainty disks, the pruning benefits are rather significant. Note that even for very large values of the radius, although the number of trajectories that will be part of $LE_B$ increases, one can still expect to prune over 75% of the trajectories before refinement.

Next, we compare the efficiency of our algorithms for processing the queries presented in Section 6 against the corresponding "brute-force" approaches. For each chart shown below, we ran 100 different executions, randomly choosing one trajectory in the dataset to serve as the query trajectory, $\boldsymbol{Tr}_q^{\mathbf{u}}$, and we averaged the results over all the 100 runs.

The naïve approach for processing query **Q1** compares the distance between $\boldsymbol{Tr}_q^{\mathbf{u}}$ and $\boldsymbol{Tr}_i^{\mathbf{u}}$ at all intersection points between two distance functions throughout the time-interval. Clearly, the brute-force approach incurs a major overhead caused by unnecessary computations, as illustrated in Figure 19.

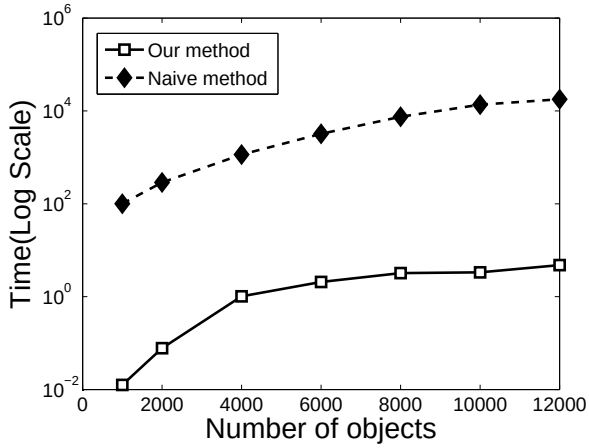


**Fig. 19** Processing Existential Query **Q1**

Our next set of experiments considers the evaluation of an existential variant of query **Q3**, which, as explained in Section 6, asks to verify whether there exists a time-instant at which a trajectory has a chance of being a $k$-th highest-probability nearest neighbor of the query trajectory, $\boldsymbol{Tr}_q^{\mathbf{u}}$.

When compared with the naïve approach for processing **Q1**, the corresponding naïve approach for processing **Q3** has an additional overhead of comparing the rank of the particular trajectory relative to the others, which is nonnegligible effect. Figure 20 show how our algorithm is much faster than the brute-force one for $k = 60$.

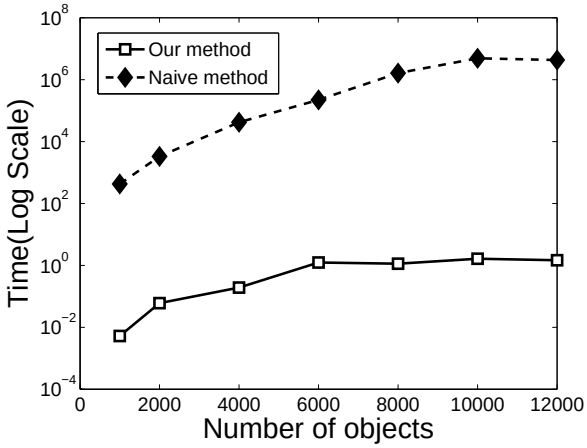


**Fig. 20** Processing Existential Query with Ranking **Q3**

We also investigate the speed-up offered by our approach as a function of $k$. The results for two datasets, of sizes 2,000 and 8,000 trajectories, respectively, are shown in Figure 21. We observe an increase in the running time as $k$ grows. However, since we are using a logarithmic scale for the Y-axis (time), the fluctuations are not obvious.

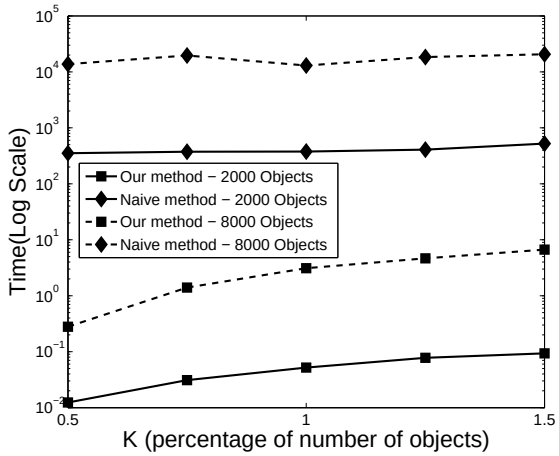


**Fig. 21** Impact of the ranking parameter on Existential Query **Q3**

Recall that query **Q2**, presented in Section 6, is essentially a version of query **Q1** applied to the entire collection of the trajectories. In the worst case, our algorithm for processing **Q2** traverses the entire *IPAC-NN* tree. Instead, the naïve strategy processes the entire MOD without any pruning. Once again, our algorithm is much faster than the naïve one, as illustrated in Figure 22.

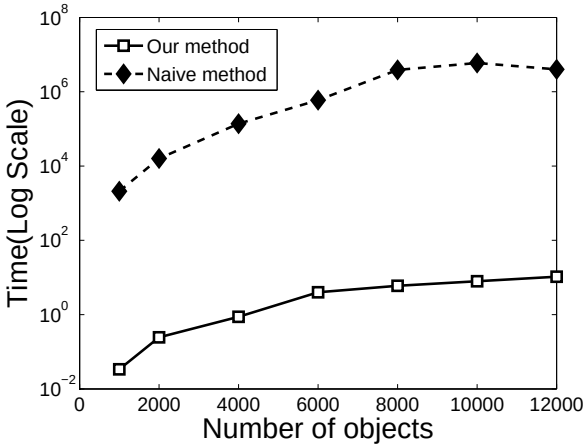


**Fig. 22** Processing of Existential Query **Q2**

Our final set of experiments evaluates the efficiency advantage of our techniques when processing a variant of query **Q3**. Specifically, we are not interested in the actual ranking of the probability of any particular NN-candidate, for as long as the object is among the possible nearest neighbors. However, we are interested in the objects that have a non-zero probability of being nearest neighbor for at least a given fraction, $\phi$, of the time-interval of interest for a given query. In Figure 23, we present the results of the comparison for two sets of trajectories, with cardinalities 2,000 and 8,000, respectively, where the X-axis represents $\phi$ as a percentage.

We note that there is not much of a fluctuation in the graphs for each method/dataset, which is to be expected. Namely, in our algorithm, one may end up traversing the entire *IPAC-NN* tree, whereas in the naïve method, the entire MOD.

## 8 Related Work

Efficient techniques for processing nearest neighbor queries are useful in a wide variety of application scenarios and in different areas, ranging from machine learning and computer vision [62] to clustering and data mining [66].

Voronoi diagrams, which are extensively studied in computational geometry [4,12], can be used to find the

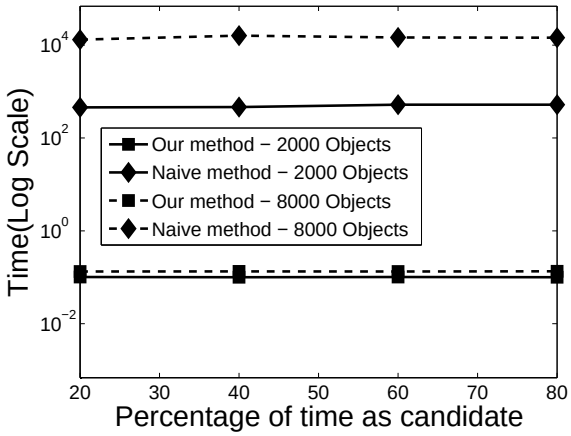


**Fig. 23** Processing of $\phi$-based Query (**Q3**)

nearest neighbor of a query point among N static 2D-points in $O(\log N)$ query time, after $O(N \log N)$ preprocessing time to construct the data structure. The problem of scalable and efficient processing of $k$-NN queries in the context of spatial databases has been addressed in [57] with a branch-and-bound approach and in [29] with an incremental technique, both of which use R-trees as an index.

Recent years have witnessed tremendous growth in MOD [23] research, where one of the main goals is the efficient processing of various categories of spatio-temporal queries. As part of the *filter + prune + refine* paradigm, a large number of spatio-temporal indexing structures have been proposed [1,2,18,27,39,45,51,59, 71,81].

Previous work has specifically targeted the problem of efficient processing of $k$-NN queries in spatio-temporal settings. In [38], 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 [42]. The generation of time-parameterized answers to the continuous variant of NN queries and their efficient scalable processing (based on TPR-trees) was presented in [67,68]. A specific setting occurs when a road network can be assumed [11, 13, 19, 76]. Techniques for efficient processing of NN queries in such setting have been presented in [48,61,80].

Closely related to our current work, results on the efficient processing of continuous $k$-NN queries on trajectories are presented in [16] and in [24]. Work on $k$-NN query processing on road networks is presented in [32]. Whereas the broader goal of our work is to represent the systematic treatment of uncertainty for continuous $k$-NN queries, the results of this article focus on the

refinement part. For example, we do not address indexing and data structures to prune the trajectories that cannot be in the answer set to a given query. However, we believe that the formal treatment presented in this article can serve as a basis for adapting the existing structures in [16, 24, 32] to the problem of efficient query processing of continuous NN queries on uncertain trajectories.

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. Scalable techniques have been proposed for this problem in [79,82]. A similar objective has been addressed in a different context: the motion of the object is expressed using (location, time, velocity) updates in [34], where an incremental approach for processing $k$-NN queries is presented.

Two works that are very similar in spirit to ours are [5, 55]. Both of them consider collection of hyperbolae representing the distance functions from a querying object. In particular, [55] 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 the nearest neighbor to the querying object within a given time interval. The main objective of [5] is the scalable processing of regular and reverse NN queries; its focus is on the efficient management of updates (insertions/deletions), however uncertainty is not formally addressed.

Various models of uncertainty in spatio-temporal settings have been considered in the literature, and a recent categorization and a generic model, along with a proposed query-interface is presented in [41]. The uncertainty associated with the model that uses (*location*, *time*) updates demonstrates that, under constrained maximal velocity, the spatial zone of the object's whereabouts is an ellipse [53]. The 3D interpretation of that same model ("beads") was discussed in [30]. Only recently have the data modeling/management issues been considered in those settings in [40], where a thorough mathematical characterization of the beads (also called "space-time prisms") is presented. However, the processing of continuous NN queries under that uncertainty model has not been considered.

The uncertainty model adopted in our work is used for processing range queries in MOD settings [73], where various semantic categories of the answers to the queries are presented and geometric concepts are used for their efficient processing. We rely on the results in [9] for processing instantaneous NN queries in uncertain environments and provide a two-fold extension: (1) the convolution property, which enables us to handle uncertainty in the locations of the query objects; and (2) the algorithms based on convolution for constructing the answers to the continuous variations of the NN query.

The problem of efficiently processing continuous $k$-NN queries for objects moving with uncertain velocity along a road network was recently studied in [31], focusing on finding the upper and lower envelopes of the set of distance functions, guaranteeing that a certain object may be one of the $k$-nearest neighbors. However, although no formal complexity analysis is given, it appears that the construction of the upper envelope takes quadratic time. More recently [43] has addressed the continuous variant of the problem of efficiently maintaining the information related to the probability of nearest neighbors for uncertain objects moving in road networks with uncertain speeds. Based on observations regarding the possible changes to the shortest distance between a given moving object and a query object inbetween updates, efficient methods are proposed that combine pruning (based on maximal and minimal distance functions), refinement and probability evaluation. The main difference from our work is that [31, 43] do not explicitly consider the consequences of mixing in the travel-time distance with the geographic uncertainty of the objects' whereabouts in a given time instant.

## 9 Conclusions and Future Work

We addressed the problem of processing efficiently the ranking of the answers to continuous nearest neighbor queries for uncertain trajectories, where the uncertainty at any time instant is bounded by a circle with a fixed radius, for objects travelling in 2D space. In addition, we have studied this problem in the context of motion restricted to road networks.

We demonstrated by using convolution that it is possible to transform the original problem into the problem where the query trajectory becomes crisp, at the expense of changing the *pdf*s of the rest of the trajectories. An important property that is retained after the transformation is that the relative NN-based ranking of the original trajectories (with respect to the querying one) is preserved. We showed that this feature is preserved for a large class of location uncertainty *pdf*s—namely, the ones that exhibit rotational symmetry—and that similar properties are retained for trajectories on road networks when the *pdf*s along the edges exhibit a symmetry with respect to the expected locations.

Based on these results, we were able to derive efficient algorithms for constructing a compact structure that represents the complete answer to a continuous NN query for uncertain trajectories: the *IPAC-NN* tree. An

important property of this tree is that it can be used to efficiently eliminate from consideration the uncertain trajectories that have no possibility of being a nearest neighbor to a given query trajectory. In addition, we presented a set of predicates and we have shown how to use them to extend the available categories of queries that take into consideration uncertainty when specifying continuous NN queries. Based on the *IPAC-NN* tree, as well as its geometric dual $LE_B$, we have also given efficient algorithms for the refinement stage of query processing, yielding a significant speed up when compared to the corresponding naïve approaches.

Given the importance of NN-like queries in different application domains, there are several challenging problems that we plan to investigate in the future:
– A first extension of our work will be to explore the applicability of our findings toward processing other variants of NN queries (e.g., all pairs and reverse [5, 78]) and compare the semantics of traditional *Top-k* NN queries [64] for crisp trajectories with that for uncertain trajectories. In a similar spirit, while in this work we have focused on the qualitative ranking of the answers, a practically important extension is to address *threshold-based* probabilistic queries such as *retrieve the objects that have more than 65% probability of being a nearest neighbor within 50% of the time* [8].
– As in this work we have only addressed the efficiency of the processing algorithm for the refinement stage, the development of an indexing structure for scalable processing of uncertain NN queries, is an open challenge. Such development will not only guarantee the absence of false negatives, but it will also decrease the number of false positives. We plan to devise a spatio-temporal variant of the U-tree structure presented in [70] for free 2D motion and extend the structures for NN queries in road networks [32] to handle uncertainty. Another problem, which is related to scalability, is how to allow for multiple query trajectories. The scalability of the pruning process is another challenge. We also plan to investigate the applicability of the Voronoi diagram of moving disks [37] towards that end.
– Given recent trends in MOD research, we plan to investigate how our results can be applied to spatio-temporal data warehousing and mining [49, 52].

## Acknowledgments

We are grateful to the anonymous reviewers for their helpful and constructive suggestions. We also thank Abraham Haddad and Ajit Tamhane for useful comments and Hui Ding for contributing to the implementation of some of the algorithms presented in the earlier version of this article [72].

## References

1. Pankaj K. Agarwal, Lars Arge, and Jeff Erickson. Indexing moving points. In *ACM PODS*, 2000.
2. Charu C. Aggarwal and Dakshi Agarwal. On nearest neighbor indexing of nonlinear trajectories. In *ACM PODS*, 2003.
3. Charu C. Aggarwal and Philip S. Yu. A survey of uncertain data algorithms and applications. *IEEE Trans. Knowl. Data Eng.*, 21(5):609–623, 2009.
4. Franz Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3), 1991.
5. Rimantas Benetis, Christian S. Jensen, Gytis Karciauskas, and Simonas Saltenis. Nearest and reverse nearest neighbor queries for moving objects. *VLDB Journal*, 15(3):229–249, 2006.
6. Christian Böhm, Beng Chin Ooi, Claudia Plant, and Ying Yan. Efficiently processing continuous k-NN queries on data streams. In *ICDE*, 2007.
7. Hu Cao, Ouri Wolfson, and Goce Trajcevski. Spatio-temporal data reduction with deterministic error bounds. *VLDB Journal*, 15(3), 2006.
8. Reynold Cheng, Jinchuan Chen, Mohamed F. Mokbel, and Chi-Yin Chow. Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data. In *ICDE*, 2008.
9. Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Querying imprecise data in moving objects environments. *IEEE-Trans. Knowl. Data Eng.*, 16(9), 2003.
10. Hae Don Chon, Divyakant Agrawal, and Amr El Abbadi. Range and $k$NN query processing for moving objects in grid model. *Mobile Networks and Applications*, 8, 2003.
11. Alminas Civilis, Christian S. Jensen, and Stardas Pakalnis. Techniques for efficient road-network-based tracking of moving objects. *IEEE Trans. Knowl. Data Eng.*, 17(5), 2005.
12. Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational geometry: algorithms and applications*. Springer-Verlag New York, Inc., 2001.
13. Ugur Demiryurek, Bei Pan, Farnoush Banaei Kashani, and Cyrus Shahabi. Towards modeling the traffic data on road networks. In *GIS-IWCTS*, 2009.
14. Ke Deng, Xiaofang Zhou, Heng Tao Shen, Kai Xu, and Xuemin Lin. Surface $k$-NN query processing. In *ICDE*, 2006.
15. Zhimin Ding and Ralf H. Güting. Managing moving objects on dynamic transportation networks. In *SSDBM*, 2004.
16. Yunjun Gao, Chun Li, Gencai Chen, Ling Chen, Xianta Jiang, and Chun Chen. Efficient $k$ -nearest-neighbor search algorithms for historical moving object trajectories. *J. Comput. Sci. Technol.*, 22(2):232–244, 2007.
17. Bugra Gedik and Ling Liu. MobiEyes: A distributed location monitoring service using moving location queries. *IEEE Transactions on Mobile Computing*, 5(10), 2006.
18. Bugra Gedik, Kun-Lung Wu, Philip S. Yu, and Ling Liu. Processing moving queries over moving objects using motion-adaptive indexes. *IEEE Trans. Knowl. Data Eng.*, 18(5):651–668, 2006.
19. Betsy George, Sangho Kim, and Shashi Shekhar. Spatio-temporal network databases and routing algorithms: A summary of results. In *SSTD*, 2007.
20. Boris V. Gnedenko. *Course of Probability Theory*. Nauka, 1988.

21. Ralf H. Güting, Michael H. Böhlen, Martin Erwig, Christian S. Jensen, Nikos Lorentzos, Enrico Nardelli, Markus Schneider, and J. R. R. Viqueira. Spatio-temporal models and languages: An approach based on data types. In *Spatio-Temporal Databases – the CHOROCHRONOS Approach*. 2003.
22. Ralf H. Güting, Victor T. de Almeida, and Zhiming Ding. Modeling and querying moving objects in networks. *VLDB Journal*, 15(2), 2006.
23. Ralf H. Güting and Markus Schneider. *Moving Objects Databases*. Morgan Kaufmann, 2005.
24. Ralf Hartmut Güting, Thomas Behr, and Jianqiu Xu. Efficient $k$-nearest neighbor search on moving object trajectories. *VLDB Journal*, 19(5):687–714, 2010.
25. Ralf Hartmut Güting, Michael H. Böhlen, Martin Erwig, Christian S. Jensen, Nikos A. Lorentzos, Markus Schneider, and Michalis Vazirgiannis. A foundation for representing and querying moving objects. *ACM Trans. Database Syst.*, 25(1), 2000.
26. Marios Hadjieleftheriou, George Kollios, Petko Bakalov, and Vassilis Tsotras. Complex spatio-temporal pattern queries. In *VLDB*, 2005.
27. Marios Hadjieleftheriou, George Kollios, Vassilis J. Tsotras, and Dimitrios Gunopulos. Efficient indexing of spatiotemporal objects. In *EDBT*, 2002.
28. Torsten Hägerstrand. What about people in regional science? *Papers of the Regional Science Association*, 24:7–21, 1970.
29. Gisli R. Hjaltason and Hanan Samet. Distance browsing in spatial databases. *ACM Trans. Database Syst.*, 24(2):265–318, 1999.
30. Kathleen Hornsby and Max J. Egenhofer. Modeling moving objects over multiple granularities. *Ann. Math. Artif. Intell.*, 36(1-2):177–194, 2002.
31. Yuan-Ko Huang, Chao-Chun Chen, and Chiang Lee. Continuous k -nearest neighbor query for moving objects with uncertain velocity. *GeoInformatica*, 13(1), 2009.
32. Yuan-Ko Huang, Zhi-Wei Chen, and Chiang Lee. Continuous k-nearest neighbor query over moving objects in road networks. In *APWeb/WAIM*, pages 27–38, 2009.
33. Zhiyong Huang, Hua Lu, Beng Chin Ooi, and Anthony K. H. Tung. Continuous skyline queries for moving objects. *IEEE Trans. Knowl. Data Eng.*, 18(12), 2006.
34. Glenn S. Iwerks, H. Samet, and Kenneth P. Smith. Maintenance of $K$-nn and spatial join queries on continuously moving points. *ACM Trans. Database Syst.*, 31(2), 2006.
35. Christian S. Jensen, Dan Lin, Beng Chin Ooi, and Rui Zhang. Effective density queries on continuously moving objects. In *ICDE*, page 71, 2006.
36. Dmitri V. Kalashnikov, Sunil Prabhakar, and Susan Hambrusch. Main memory evaluation of monitoring queries over moving objects. *DAPD*, 15, 2004.
37. Menelaos I. Karavelas. Voronoi diagrams for moving disks and applications. In *WADS*, pages 62–74, 2001.
38. George Kollios, Dimitris Gunopulos, and Vassilis Tsotras. Nearest neighbor queries in a mobile environment. In *STDM*, pages 119–134, 1999.
39. George Kollios, Dimitris Gunopulos, and Vassilis Tsotras. On indexing mobile objects. In *ACM PODS*, 1999.
40. Bart Kuijpers and Wallied Othman. Trajectory databases: data models, uncertainty and complete query languages. *J. Comput. Syst. Sci.*, 76(7), 2010.
41. Ralph Lange, Harald Weinschrott, Lars Geiger, André Blessing, Frank Dürr, Kurt Rothermel, and Hinrich Schütze. On a generic uncertainty model for position information. In *QuaCon*, pages 76–87, 2009.
42. Jose A.C. Lema, Luca Forlizzi, Ralf H. Güting, Enrico Nardelli, and Markus Schneider. Algorithms for moving objects databases. *Computing Journal*, 46(6), 2003.
43. Guohui Li, Yanhong Li, LihChyun Shu, and Ping Fan. C-kNN query processing over moving objects with uncertain speeds in road networks. In *APWeb*, 2011.
44. J.S. Lim. *Two-Dimensional Signal and Image Processing*. Prentice Hall, 1990.
45. Nikos Mamoulis, Huiping Cao, George Kollios, Marios Hadjieleftheriou, Yufei Tao, and David W. Cheung. Mining, indexing, and querying historical spatiotemporal data. In *ACM SIGKDD*, 2004.
46. Mohamed F. Mokbel and Walid G. Aref. SOLE: Scalable on-line execution of continuous queries on spatio-temporal data streams. *VLDB Journal*, 17(5):971–985, 2008.
47. Mohamed F. Mokbel, Xiaopeng Xiong, and Walid G. Aref. SINA: Scalable incremental processing of continuous queries in spatio-temporal databases. In *ACM SIGMOD*, 2004.
48. Kyriakos Mouratidis, Man L. Yiu, Dimitris Papadias, and Nikos Mamoulis. Continuous nearest neighbor monitoring in road networks. In *VLDB*, pages 43–54, 2006.
49. Mirco Nanni, Bart Kuijpers, Christine Körner, Michael May, and Dino Pedreschi. Spatiotemporal data mining. In *Mobility, Data Mining and Privacy*. 2008.
50. Jian Pei, Ming Hua, Yufei Tao, and Xuemin Lin. Query answering techniques on uncertain and probabilistic data: tutorial summary. In *ACM SIGMOD*, 2008.
51. Mindaugas Pelanis, Simonas Saltenis, and Christian S. Jensen. Indexing the past, present, and anticipated future positions of moving objects. *ACM Trans. Database Syst.*, 31(1), 2006.
52. Nikos Pelekis, Alessandra Raffaetà, Maria Luisa Damiani, Christelle Vangenot, Gerasimos Marketos, Elias Frentzos, Irene Ntoutsi, and Yannis Theodoridis. Towards trajectory data warehouses. In *Mobility, Data Mining and Privacy*. 2008.
53. Dieter Pfoser and Christian S. Jensen. Capturing the uncertainty of moving objects representation. In *SSD*, 1999.
54. Dieter Pfoser, Nectaria Tryfona, and Christian S. Jensen. Indeterminacy and spatiotemporal data: Basic definitions and case study. *GeoInformatica*, 9(3), 2005.
55. Katerina Raptopoulou, Apostolos Papadopoulos, and Yannis Manolopoulos. Fast nearest-neighbor query processing in moving-object databases. *GeoInformatica*, 7(2):113–137, 2003.
56. Rachel Rosmarin. Making money from online maps. Forbes online issue, April 2006. http://www.forbes.com/home/digitalentertainment/2006/04/13/google-aol-yahoo-cx_rr_0417maps.html.
57. Nick Roussopoulos, Stephen Kelley, and Fredéric Vincent. Nearest neighbor queries. In *ACM SIGMOD*, 1995.
58. Halsey L. Royden. *Real Analysis*. Macmillan Co., 1963.
59. Simonas Saltenis and Christian S. Jensen. Indexing of moving objects for location-based services. In *ICDE*, 2002.
60. Joachim Schiller and Agnes Voisard. *Location-based Services*. Morgan Kaufmann Publishers, 2004.
61. Cyrus Shahabi, Mohammad R. Kolahdouzan, and Mehdi Sharifzadeh. A road network embedding technique for k-nearest neighbor search in moving object databases. *GeoInformatica*, 7(3):255–273, 2003.
62. Gregory Shakhnarovich, Trevor Darrel, and Piotr Indyk, editors. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. MIT Press, 2006.

63. Micha Sharir and Pankaj K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, 1995.
64. Mohamed A. Soliman, Ihab F. Ilyas, and Kevin C.-C. Chang. Top-k query processing in uncertain databases. In *ICDE*, 2007.
65. Dan Suciu and Nilesh N. Dalvi. Foundations of probabilistic answers to queries. In *ACM SIGMOD*, 2005. tutorial.
66. Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.
67. Yufei Tao and Dimitris Papadias. Spatial queries in dynamic environments. *ACM Trans. Database Syst.*, 28(2), 2003.
68. Yufei Tao, Dimitris Papadias, and Qiongmao Shen. Continuous nearest neighbor search. In *VLDB*, 2002.
69. Yufei Tao, Dimitris Papadias, and Jimeng Sun. The TPR$*$-tree: An optimized spatio-temporal access method for predictive queries. In *VLDB*, 2003.
70. Yufei Tao, Xiaokui Xiao, and Reynold Cheng. Range search on multidimensional uncertain data. *ACM Trans. Database Syst.*, 32(3):15, 2007.
71. Yannis Theodoridis, Timos Sellis, Apostolos Papadopoulos, and Y. Manolopoulos. Specifications for efficient indexing in spatiotemporal databases. In *SSDBM*, 1998.
72. Goce Trajcevski, Roberto Tamassia, Hui Ding, Peter Scheuermann, and Isabel F. Cruz. Continuous probabilistic nearest-neighbor queries for uncertain trajectories. In *EDBT*, 2009.
73. Goce Trajcevski, Ouri Wolfson, Klaus Hinrichs, and Sam Chamberlain. Managing uncertainty in moving objects databases. *ACM Trans. Database Syst.*, 29(3), 2004.
74. Kishor S. Trivedi. *Probability and Statistics with Reliability, Queueing and Computer Science Applications*. Wiley, 2002.
75. Jeffrey D. Ullman. *Principles of Database and Knwoledge – Base Systems*. Computer Science Press, 1989.
76. Michalis Vazirgiannis and Ouri Wolfson. A spatiotemporal model and language for moving objects on road networks. In *SSTD*, 2001.
77. Ouri Wolfson, A. Prasad Sistla, Sam Chamberlain, and Y. Yesha. Updating and querying databases that track mobile units. *Distr. and Parallel Databases*, 7(3), 1999.
78. Tian Xia and Donghui Zhang. Continuous reverse nearest neighbor monitoring. In *ICDE*, 2006.
79. Xiaopeng Xiong, Mohamed F. Mokbel, and Walid G. Aref. Sea-cnn: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In *ICDE*, pages 643–654, 2005.
80. Man Lung Yiu, Nikos Mamoulis, and Dimitris Papadias. Aggregate nearest neighbor queries in road networks. *IEEE Trans. Knowl. Data Eng.*, 17(6), 2005.
81. Man Lung Yiu, Yufei Tao, and Nikos Mamoulis. The $B^{dual}$-tree: indexing moving objects by space filling curves in the dual space. *VLDB Journal*, 17(3), 2008.
82. Xiaohui Yu, Ken Q. Pu, and Nick Koudas. Monitoring k-nearest neighbor queries over moving objects. In *ICDE*, pages 631–642, 2005.