The VLDB Journal (2011) 20:767–791
DOI 10.1007/s00778-011-0249-3

SPECIAL ISSUE PAPER

Ranking continuous nearest neighbors for uncertain trajectories

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

Received: 30 September 2010 / Revised: 10 April 2011 / Accepted: 7 May 2011 / Published online: 30 August 2011

Abstract This article addresses the problem of performing Nearest Neighbor (NN) queries on *uncertain* trajectories. The answer to an NN query for certain trajectories is time parameterized due to the continuous nature of the motion. As a consequence of uncertainty, there may be several objects that have a non-zero probability of being a nearest neighbor to a given querying object, and the continuous nature further complicates the semantics of the answer. 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 experiments demonstrate that the proposed algorithms yield significant performance improvements when compared with the corresponding naïve approaches.

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 EECS, Northwestern University, Chicago, IL, USA
e-mail: gocez@eecs.northwestern.edu

R. Tamassia
Department of CS, Brown University, Providence, RI, USA
e-mail: rt@cs.brown.edu

I. F. Cruz
Department of CS, The University of Illinois at Chicago, Chicago, IL, USA
e-mail: ifc@cs.uic.edu

Keywords Nearest neighbor · Uncertainty · Moving objects database

1 Introduction

A broad set of applications—such as traffic management, emergency response, disaster remediation, context-aware tourist information, and battlefield management—rely on *Moving Object Databases* (*MOD*) [19] to manage the (*location*, *time*) information of the mobile entities, and efficiently process various queries of interest, e.g., range, density, variants of nearest neighbor (reverse, surface, aggregate), and skyline [5,13,18,24,26,33,34,59,60].

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 [58]. 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., [3,47,48,37]), including spatial and spatio-temporal settings (e.g., [8,39,40,51,55]).

As an illustrative scenario, in order 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 cannot be determined accurately. Hence, the associated uncertainty needs to be taken into account when planning the routes and traffic light sequence. Similarly, 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. Yet another settings are studies related to climate and weather data where, for instance, one may be interested in the spatio-temporal proximity among hurricanes. However, their typical tracking (via satellite) has a rather uncertain location-in-time information of the eye of a hurricane, much less the overall size of the affected region. Motivated by this kind of 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 [21,42], the answer to a *continuous* NN-query in a spatio-temporal setting is *time parameterized* [49] 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 next.

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* $t_2 = 20$.

Using algorithms available in the literature (e.g., [35,49]), 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 $\boldsymbol{Tr}_2$.

Now consider a slightly modified scenario, depicted in Fig. 1, 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 $\boldsymbol{Tr}_2$.

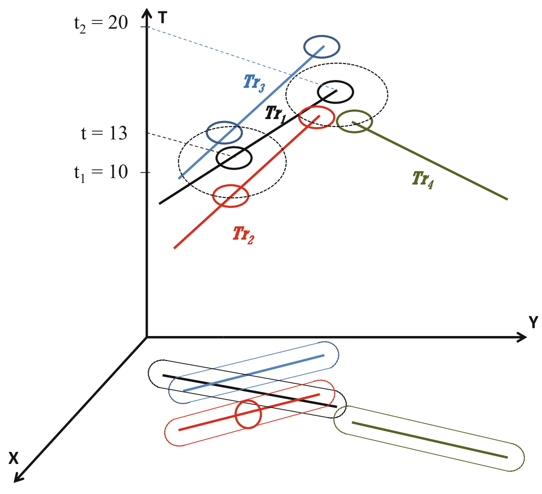

Fig. 1 Uncertainty impact for continuous NN queries

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 Fig. 1) 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 [47] 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 a suite of syntactic constructs that will enable them to express their interests when posing the queries. Toward 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-trajectories' 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 [54]. In addition to the stylistic and structural improvements from the conference version, we introduce the following new contributions:

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

The rest of this article is structured as follows. In Sect. 2, we describe the necessary background. Section 3 presents 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 location uncertainty *pdf* s. Based on these properties, in Sect. 4 we focus on the processing of NN-queries: we present the structure of the answer, along with the corresponding algorithm for constructing it. In Sect. 5, the previous context is changed so as to consider objects that move along a road network. Section 6 addresses different syntactic variants of NN-queries for uncertain trajectories. In Sect. 7 we present a portion of our experimental results, following up with a discussion of related works and concluding remarks.

Due to space limitations, we now present only a subset of our experimental results, which demonstrate the benefits of the proposed methodologies. The complete set with a lengthier discussion, as well as a more detailed presentation of related work and of the concluding remarks were peer reviewed during the submission process and are available elsewhere [53].

2 Preliminary background

We now introduce background material, starting with an overview of the uncertainty models typically used in MOD settings and a formal definition of the model used in this work. Next, we discuss 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) [8] and conclude with observations on *probabilistic completeness* for NN queries.

2.1 Modeling uncertainty of motion

Selecting a model for the motion plan affects the representation of trajectories in a MOD [16,17] and, consequently, the overall strategy of query processing, including indexing [1,2,13,28,33,38,50,52] and pruning/refinement [11,25,49,61]. In addition, the choice of the motion model affects the corresponding uncertainty model that can be associated with it. Three popular models for uncertainty in MOD settings are:

M1: The location and time data of the moving objects are obtained by receiving periodic updates of the form (x, y, t), as in on-board GPS systems [13,34]. In this case, nothing is known between consecutive updates, except that the motion is bounded by some maximum speed $v_{\max}$. Using ideas from time-geography [20] in a MOD context, it has been shown that the projection of the uncertain locations on the XY plane is bounded by an ellipse with foci in two consecutive update points [39]. A subsequent spatio-temporal version of the model [22] names the volume between two update points a *bead* (called *space-time prism* in [29]), which presents the first formal analysis of the properties of the model in terms of query languages (cf. Fig. 2a).

M2: In some applications, each object transmits its *expected* velocity along with its current sampled location [25]. Typically, velocity information is provided to save bandwidth consumption at the expense of imprecision [58]. As long as the sampled location does not

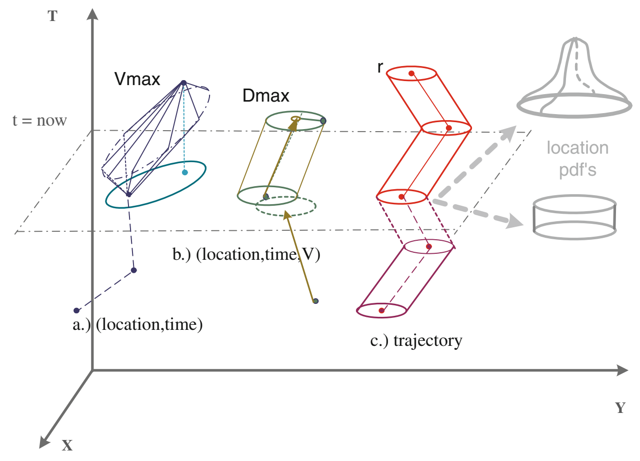

Fig. 2 Motion models and uncertainty

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* [58]. The possible locations of an object under consecutive MOD updates are illustrated in Fig. 2b.

M3: Another model of motion represents the trajectories as sequences of (x, y, t) points and the locations between such points are obtained via linear interpolation [17]. Although this model is typically used for past trajectories, it can also represent future trip-planning trajectories, where 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 Google Maps. The uncertainty model for processing spatio-temporal range queries for "full" trajectories assumes that at each time instant the object's location is bounded [55]. This model is illustrated in Fig. 2c, 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:

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

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

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

Definition 2 (*An uncertain trajectory*) $\boldsymbol{Tr_i^u}$ is a trajectory augmented with: (1) 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. Thus, we have:

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

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 $\boldsymbol{Tr_i^u}$ 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 [6].

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., [8,51]) 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 the *pdf* of the object with ID oid_k and expected location $(x_k(t), y_k(t))$ at time t 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 Sect. 3.

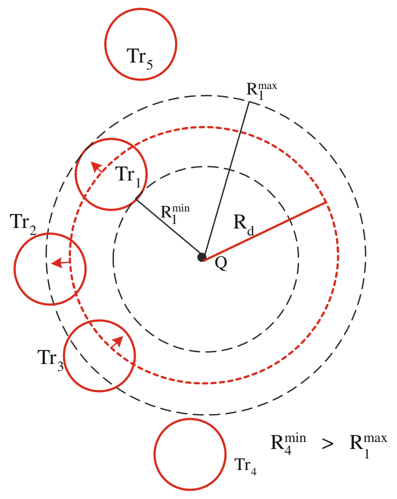

Fig. 3 Uncertain NN-query (crisp $\mathbf{o}_q$)

2.2 Uncertain objects and 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 **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 [8]. In what follows, we present a concise summary along with observations relevant to our work.

Observation O1: We consider bounds on the distances that possible nearest neighbors can have from **Q**. As shown in Fig. 3, the distance between **Q** and the *most distant point of the closest disk*, $R_{\max}$, bounds the distance that any possible nearest neighbor of $\boldsymbol{Tr}_q$ can have. Hence, any object $\mathbf{o}_i$ (a snapshot of a trajectory $\boldsymbol{Tr}_i$) whose closest possible distance to **Q**, denoted with $R_i^{\min}$, is larger than $R_{\max}$ has zero probability of being a nearest neighbor to $\boldsymbol{Tr}_q$ and can be disregarded in the query evaluation. As can be seen from Fig. 3, $R_4^{\min} > R_1^{\max}$ 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 denote with $R_{\min}$ the distance from **Q** to the closest point of the closest disk.

Observation O2: We now turn to the probability of the location along a trajectory $\boldsymbol{Tr}_i^{\mathbf{u}}$ being *within distance* R_d from **Q**. This probability can be expressed as:

$$P_{i,Q}^{WD}(R_d) = \int\int_A pdf_i(x, y)\,\mathrm{d}x\,\mathrm{d}y \tag{3}$$

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

Example 2 [8] *When $pdf_i(x,y)$ is uniform, the probability $P_{i,Q}^{WD}(R_d)$ can be calculated as:*

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

where $\Theta = \arccos\frac{d_{iQ}^2 + r^2 - R_d^2}{2d_{iQ}r}$, $\alpha = \arccos\frac{d_{iQ}^2 + R_d^2 - r^2}{2d_{iQ}R_d}$, *and d_{iQ} is the distance between* **Q** *and the expected location of $\boldsymbol{Tr}_i^{\mathbf{u}}$. We note that modifications are needed to Eq. 4 when* **Q** *is located inside the uncertainty zone of $\boldsymbol{Tr}_i^{\mathbf{u}}$ [8]. Taking the derivative of $P_{i,Q}^{WD}$ yields $pdf_{i,Q}^{WD}(R_d)$ which, in the case of uniform distribution, will be non-zero only when $d_{iQ} - r \le R_d \le d_{iQ} + r$.*

Observation O3: In computing the probability that a given object, $\boldsymbol{Tr}_j^{\mathbf{u}}$, 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 $\le R_d$ from $\boldsymbol{Tr}_q$; (2) The probability that *every other* object Tr_i $(i \ne j)$ is at a distance greater than R_d from the location **Q** of $\boldsymbol{Tr}_q$; and (3) The fact that the distributions of the objects are assumed to be independent from each other.

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

$$P_{j,Q}^{NN} = \int_0^{\infty} pdf_{j,Q}^{WD}(R_d) \cdot \prod_{i \ne j}(1 - P_{i,Q}^{WD}(R_d))\,\mathrm{d}R_d \tag{5}$$

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

By sorting the objects that have a non-zero probability of being nearest neighbors according to the minimal distances of their boundaries from **Q**, one can break the evaluation of the integral from Eq. 5 into subintervals corresponding to each $R_{\min_i}$ and the computation of the $P_{j,Q}^{NN}$ can be performed in a more efficient manner, based on the sorted distances and the corresponding intervals [8]. This is especially important because the evaluation of the integrals like the one specified in Eq. 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 **Q**.

2.3 On the completeness of NN-probabilities

While the ideas in observations **O1–O3** are intuitive and sound, there may be certain issues related to their *completeness*. Namely, the computation of the values $P_i^{NN}(Q)$

using Eq. 5 will not yield a *probability space* [14]. 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:

$$P^{NN}_{i,Q} = P^{NN\_E}_{i,Q} + P^{NN\_J}_{i,Q} \tag{6}$$

- $P^{NN\_E}_{i,Q}$ denotes the *exclusive* probability that $\boldsymbol{Tr^{\mathbf{u}}_i}$ is the nearest neighbor of Tr_q and is calculated in the spirit of Eq. 5.
- $P^{NN\_J}_{i,Q}$, 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:

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

The right-hand side of Eq. 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 **Q**, while all the other moving objects have a smaller probability of that occurring.

By analogy, we have:

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

where the right-hand side of Eq. 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 **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 **Q**:

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

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—a fact that will be exploited in Sects. 4 and 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.

3 Uncertain querying object and convolutions

In this section, we: (1) illustrate the problems that arise when the query object has an uncertainty associated with its location; and (2) show that by using a simple transformation, we can reduce this case to one in which the ideas presented in Sect. 2.2 are applicable for a large class of *pdf*'s, with appropriate modifications.

For the time being, let us 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 Fig. 4. 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 Sect. 2.2 (cf. [8]). 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., [9]) of the uncertainty disk of $\boldsymbol{Tr_q}$ with a disk of diameter R_d.

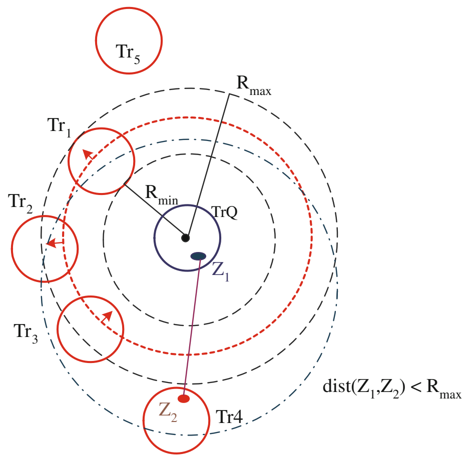

Fig. 4 Uncertain NN-query (uncertain Tr_q)

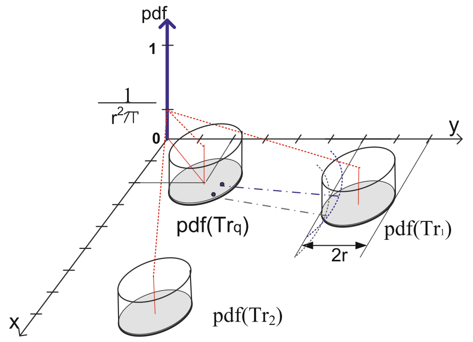

Fig. 5 Evaluating within distance probability

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 $P^{WD}_{q,P}(R_d)$ using, e.g., Eq. 3, and "add" the uncountably many such results—which is, integrate over the area $D_i \cap (D_q \oplus R_d)$, with $\mathrm{d}x_p$ and $\mathrm{d}y_p$ as the extra variables of differentiation.

This method yields a quadruple integration in the corresponding version of Eq. 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 Eq. 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 $\|l_1 l_2\| \le R_d$ (over all such locations l_1 and l_2, and after discretizing the corresponding location pdf's [8,51]). 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 5 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* ≤ 4 from $\boldsymbol{Tr}_q$ (obviously, 0 for $\boldsymbol{Tr}_2$).

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. 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 $P(\|\overline{V}_i - \overline{V}_q\| \le R_d)$. The key observation is that $\overline{V}_i - \overline{V}_q$ is another random variable, denote it $\overline{V}_{iq}$—known as a *cross-correlation* of $\overline{V}_i$ and $\overline{V}_q$ [32,56]. Also, $\overline{V}_{iq}$ can be viewed as the sum $\overline{V}_i + (-\overline{V}_q)$. Since $\overline{V}_i$ and

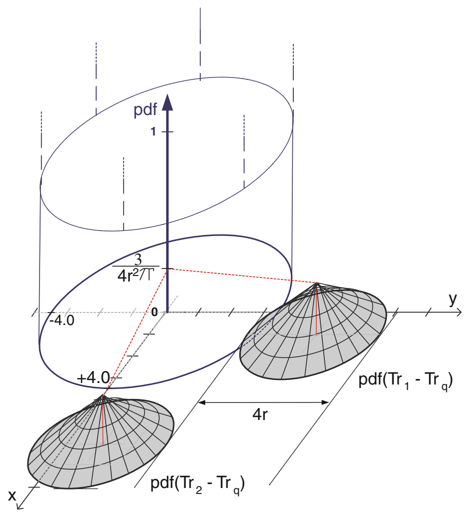

Fig. 6 Within distance probability: convolution

$\overline{V}_q$ (consequently, $-\overline{V}_q$) are independent variables [8,51]), 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 $-\overline{V}_q$) [56]. In other words:

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

Example 4 As one can readily verify (cf. [56]), 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 Fig. 6, 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. Fig. 5), 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 Eq. 2, we have

$$pdf(\overline{V}_{iq}(t)(X, Y)) = \begin{cases} 0, & \sqrt{((x_i(t) - x_q(t)) - X)^2 + ((y_i(t) - y_q(t)) - Y)^2} > 2r \\ \frac{3}{4r^2\pi}\left(1 - \frac{\sqrt{((x_i(t)-x_q(t))-X)^2+((y_i(t)-y_q(t))-Y)^2}}{2r}\right) & \text{otherwise} \end{cases} \tag{7}$$

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* [43]. However, in many practical settings, the *pdf*'s of the objects' locations (e.g., uniform, Gaussian) are *Riemann integrable* [43], 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 outline of the proof of Property 1 and 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: $\int f(\overline{x})d\overline{x}) = \int[\int g(\overline{u}) \cdot h(\overline{x} - \overline{u})d\overline{u})]d\overline{x} = \cdots$ substitute variables $\overline{x} = \overline{x} + \overline{u}$, noting that $d\overline{x}$ remains the same and the Jacobian is "1" (translation)$\cdots = \int g(\overline{u})d\overline{u} \cdot \int h(\overline{x})d\overline{x} = \cdots$ since h and u are *pdf*s, 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: $\overline{x} = \overline{x} + \overline{u} \cdots = \int(\overline{x} + \overline{u})[\int g(\overline{u}) \cdot h(\overline{x})d\overline{u})]d\overline{x} = ((\int \overline{x}h(\overline{x})d\overline{x})$ $\int g(\overline{u})d\overline{u}) + ((\int \overline{u}g(\overline{u})d\overline{u}) \int h(\overline{x})d\overline{x})$.

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 [56]. 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 toward that end, we need to define the concept of a *rotational* (a.k.a *cylindrical*) symmetry [32]. 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 *If 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 ρ with a center at $\overline{C}_c$ and an angle ϕ, 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 ϕ; (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 Jacobian 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* [32], 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., $P_{1,Q}^{NN\_E} > P_{2,Q}^{NN\_E}$, cf. Sect. 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 Eq. 5, we have:

$$\text{(I):}\quad P_1^{NN}(Q) = \int_0^{\infty} pdf_1^{WD}(R_d) \cdot (1 - P_2^{WD}(R_d))\mathrm{d}R_d = \int_{R_1^{\min}}^{R_{\max 1}} pdf_1^{WD}(R_d) \cdot (1 - P_2^{WD}(R_d))\mathrm{d}R_d$$

and, similarly:

$$\text{(II):}\quad P_2^{NN}(Q) = \int_0^{\infty} pdf_2^{WD}(R_d) \cdot (1 - P_2^{WD}(R_d))\mathrm{d}R_d = \int_{R_2^{\min}}^{R_1^{\max}} pdf_2^{WD}(R_d) \cdot (1 - P_1^{WD}(R_d))\mathrm{d}R_d.$$

The claim follows from the observations that for every ν, 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 we are given a collection of moving objects with equal *pdf*s (modulo translation with respect to their centers) that are rotationally symmetric. Let $\boldsymbol{Tr}_q$ denote the (uncertain) querying trajectory. The following theorem summarizes the main result of this section:

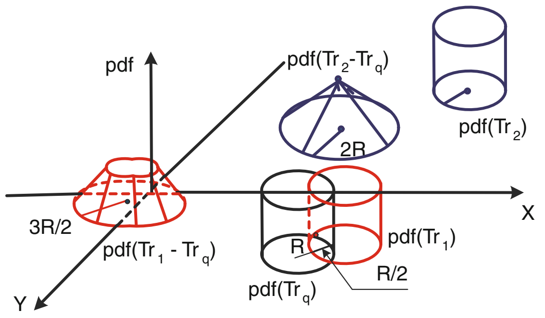

Fig. 7 Convolution of intersecting *pdf*s

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.

As an illustration, recall Fig. 6. Since the centroid of $\boldsymbol{Tr}_1^u - Tr_q^u$ is closer to the coordinate center than the one of $\boldsymbol{Tr}_2^u - \boldsymbol{Tr}_q^u$, we get $P_1^{NN}(Q) > P_2^{NN}(Q)$.

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, Fig. 7 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 Fig. 7, Tr_2 has zero probability of being a nearest neighbor.

4 Uncertain NN-query

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 the motivational Example 1 (Sect. 1), the answer to a continuous NN-query for crisp trajectories was presented 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 Sect. 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 $[t_b, t_1]$, $[t_1, t_2]$, ..., $[t_{n-1}, t_e]$ 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 $[t_{i-1}, t_{(i-1),1}]$, $[t_{(i-1),1}, t_{(i-1),2}]$, ..., $[t_{(i-1),(k-1)}, t_i]$. 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 Sect. 1, assuming that every trajectory has an uncertainty radius of 30 m, 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)]] }

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 (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 (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 generic representation of 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 labeled 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 labeling 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), as illustrated in Fig. 8. Note that the nodes in Fig. 8 also contain another component of their labeling, 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 Sect. 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

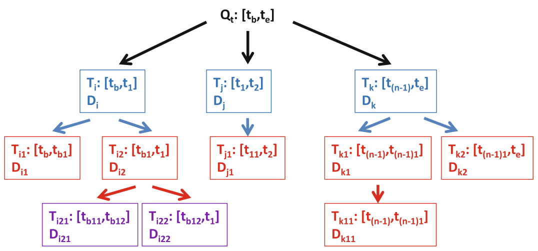

Fig. 8 Answer structure for an NN-query: the IPAC-NN tree

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 Sect. 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 Sect. 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 Sect. 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., [4,41]) 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 $V_{xiq} = v_{xi} - v_{xq}$ and $V_{yiq} = v_{yi} - v_{yq}$. Also, let X_{biq} and Y_{biq} denote the coordinates of the expected location at t_b. We have $X_{biq} = x_{bi} - x_{bq}$ and $Y_{biq} = y_{bi} - y_{bq}$.

The distance of TR_{iq} from the origin, as a function of time, is given by (cf. [4,41]) $d_{iq}(t) = \sqrt{At^2 + Bt + C}$, where:

$$A = V_{xiq}^2 + V_{yiq}^2,$$
$$B = -2(V_{x_{iq}}^2 t_b + V_{x_{iq}} X_{b_{iq}} + V_{y_{iq}}^2 t_b + V_{y_{iq}} Y_{b_{iq}}), \text{ and}$$
$$C = 2X_{b_{iq}} V_{x_{iq}} t_b + V_{x_{iq}}^2 t_b^2 + X_{b_{iq}}^2 + 2Y_{b_{iq}} V_{y_{iq}} t_b + V_{y_{iq}}^2 t_b^2 + Y_{b_{iq}}^2.$$

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 Sect. 3, we know that at any time instant t, the ranking of the probabilities of a given object Tr^u_j being a nearest neighbor to Tr^u_q 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 Fig. 9 illustrate two cases in which pairs of distance functions (corresponding to pairs of TR-like trajectories) intersect in two points and one point, respectively. We call such intersection points *critical time-points*.

To determine how each of the 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:

1. 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};
2. Checks which ones of the solutions are inside $[t_1, t_2]$, defining the critical time-points;
3. 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;
4. Returns the result.

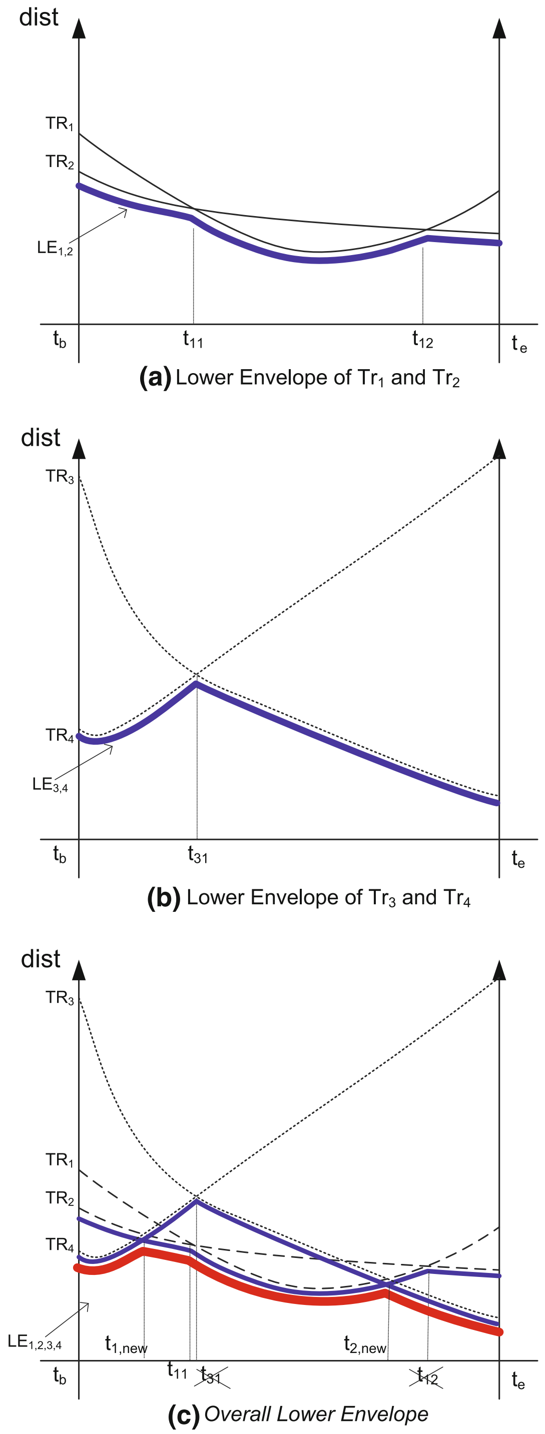

Fig. 9 Constructing the lower envelope

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 Fig. 9a, 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

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

settings of Fig. 9b, 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 [9,46]. 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 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 $[(TR_j, [t_{j1}, t_{j3}])]$.

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,$t_b$, $t_e$));
```

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

Due to the properties of the Davenport–Schinzel sequences [46], 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.

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 }
```

Example 6 Observe Fig. 9, 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 Fig. 9c) 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 added to the existing $LE_{1,2,3,4}$.

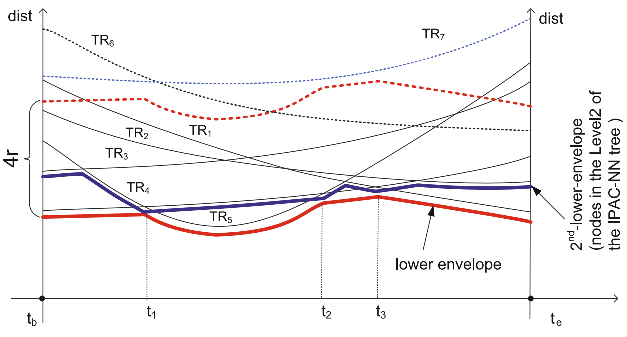

Fig. 10 Envelopes and IPAC-NN tree

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 Fig. 10, TR_7 can be safely pruned from any consideration, because its distance from the lower envelope at any time instant is greater than $4r$.

Algorithm 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.

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
```

The combinatorial complexity of the lower envelope is $O(N)$ and its construction takes $O(N \log N)$ time. 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 10 illustrates the first two levels of lower envelopes for a set of (distance functions of) uncertain trajectories. We summarize the results of this section with the following theorem:

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

We conclude this section with a discussion on the complexity of the algorithms, removing the assumption that all the trajectories consist of single line-segments, since in practice each trajectory may have a different number of segments. Assume that a given trajectory $\boldsymbol{Tr_j^u}$ 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 $\boldsymbol{Tr_j^u}$. Generalizing this observation, if every trajectory $\boldsymbol{Tr_i^u}$ ($i \in \{1, 2, \ldots, M\}$) 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, following with uncertainty model and its implications 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 [10,18,35,44,60], 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),

$$\textbf{\textit{RN-Tr}}_i = \{oid_i, (x_{i_1}, y_{i_1}, \ \ldots, (x_{i_k}, y_{i_k}, t_{i_k})\}$$

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

– 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 $v_{sk}^{\max}$.

The concepts introduced in Definition 3 are illustrated in Fig. 11, which shows three road network trajectories: ***RN-Tr***$_q$, ***RN-Tr***$_1$ and ***RN-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 ***RN-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 ***RN-Tr***$_1$ is an edge that can be traversed in 15 time units; however, the trip along that edge for ***RN-Tr***$_1$ takes $\Delta = 55 - 35 = 20$ time units.

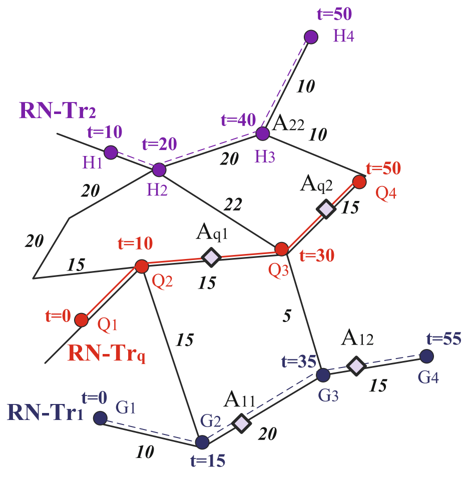

Fig. 11 Trajectories on Road Networks

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 (e.g., [23,35,44,60]):

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 traveled.

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 [10].

In the example scenario depicted in Fig. 11, we have used shortest travel-time distance for labeling 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 [23]:

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, following two key assumptions ([23]) 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 Fig. 11, the shortest path between trajectories ***RN-Tr***$_q$ and ***RN-Tr***$_1$ will consist of their individual corresponding portions, along with the edge (Q_2, G_2) for as long as:

- ***RN-Tr***$_q$ is anywhere along the segments (Q_1, Q_2) and $(Q2, A_{Q1})$
- ***RN-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 ***RN-Tr***$_q$, o_1 denote the object traveling along ***RN-Tr***$_1$, and o_2 denote the object traveling along ***RN-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.

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. [23]). In addition to the 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

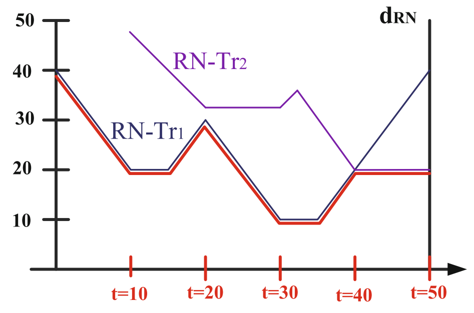

Fig. 12 Continuous travel-time distance

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 Fig. 11, this happens to the distance between o_Q and o_1 when o_Q reaches location A_{q1} along ***RN-Tr***$_q$, at $t = 20$.

Figure 12 shows the network distances of ***RN-Tr***$_1$ and ***RN-Tr***$_2$ from ***RN-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. Sect. 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 [46] 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 ***RN-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 ***RN-Tr***$_i$ and the querying trajectory ***RN-Tr***$_q$ when calculating the lower envelope. However, adding a factor of 2 retains the upper bound $O\left(\left(\sum m_i\right) \cdot N \log N\right)$ 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 one of the distance function used. Consider the following two cases:

1. The network distance is based on the *length of the edges* representing the road segments, and the uncertainty of object's location at each time instant is bounded by a *fixed length*.
2. The network distance 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 of the above cases, for as long as the instantaneous *pdf* satisfies the (appropriately modified) requirement from Sect. 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 Sects. 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 Sect. 5.1, the only modification is the appropriate definition of the distance function in Algorithms 1 and 2.

An especially interesting scenario occurs when the physical nature of the distance function used is different from the one of the uncertainty. 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 Fig. 13, 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 Fig. 11. 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 Fig. 4, the object o_1 can have different *pdf*'s along that line segment. We make the following observations:

I: 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

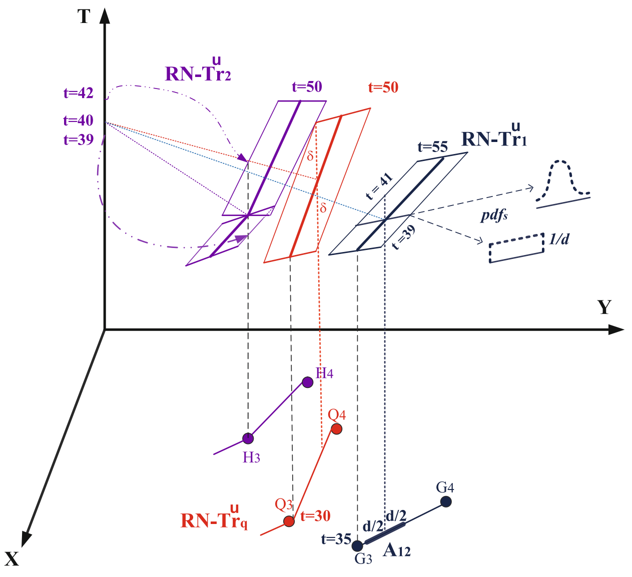

Fig. 13 Uncertain trajectories in road networks

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) [56].

II: 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 $\boldsymbol{RN\text{-}Tr}_1^{\mathbf{u}}$.

III: 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 Fig. 13.

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 δ_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 δ_i denote the corresponding bounds for the trajectory $\boldsymbol{RN\text{-}Tr}_i^{\mathbf{u}}$. Depending on the (relative) values of δ_0 and δ_i (for $i = 1, 2, \ldots, N$),

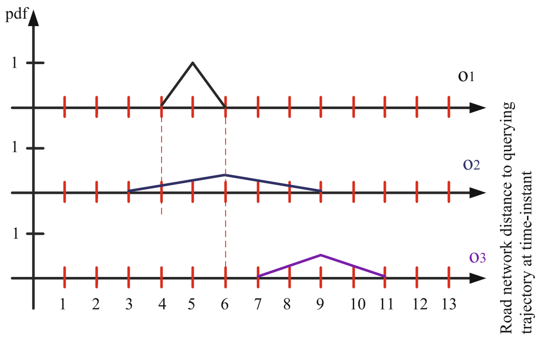

Fig. 14 Travel-time distance and within-distance *pdf*s

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 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 Fig. 14, 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 τ. 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 14 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 τ. 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 τ, 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 Eqs. 3 and 5 from Sect. 2 to the current settings of motion along a road network. Since the travel-time distance is one dimensional, Eq. 3 needs to be modified as follows:

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

where the variable T_d (respectively, t_d) denotes 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 τ, we need to solve the following equation:

$$\begin{aligned}&\int_0^\infty pdf_{o_1,o_Q}^{WD}(T_d)\cdot(1 - P_{o_2,o_Q}^{WD}(T_d))\, dT_d\\ &= \int_0^\infty pdf_{o_2,o_Q}^{WD}(T_d)\cdot(1 - P_{o_1,o_Q}^{WD}(T_d))\, dT_d\end{aligned} \tag{11}$$

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. Fig. 14), 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 Sect. 2.3, the main reason that we are justified to use Eq. 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 Sect. 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 Fig. 15, which shows the expected distances and uncertainty regions for four trajectories $\boldsymbol{RN\text{-}Tr}_1^{\mathbf{u}}$, $\boldsymbol{RN\text{-}Tr}_2^{\mathbf{u}}$, $\boldsymbol{RN\text{-}Tr}_3^{\mathbf{u}}$ and $\boldsymbol{RN\text{-}Tr}_4^{\mathbf{u}}$ from the querying trajectory $\boldsymbol{RN\text{-}Tr}_q^{\mathbf{u}}$ 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 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 $\boldsymbol{RN\text{-}Tr}_4$

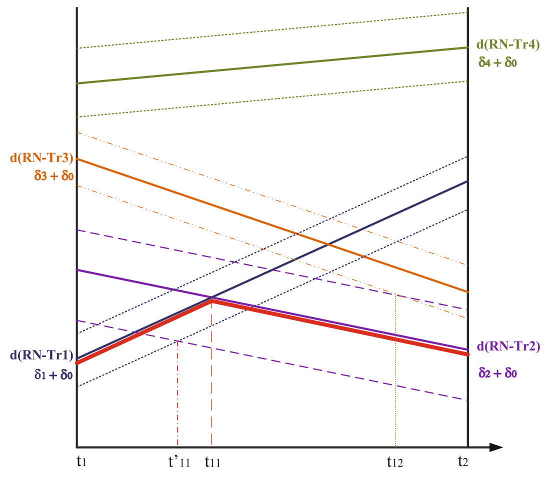

Fig. 15 Lower envelope and nearest neighbor in road network settings

has zero probability of being a nearest neighbor to ***RN-Tr****$_q$* throughout $[t_1, t_2]$. Complementary to this, we observe that there exists 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 ***RN-Tr***$_3$ and ***RN-Tr***$_2$. This, in turn, implies that ***RN-Tr***$_3$ *does* have a non-zero probability of being a nearest neighbor to ***RN-Tr****$_q$* throughout $[t_1, t_2]$.

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

$$(\delta_j + \delta_0) + (\delta_i + \delta_0) < d(\textbf{\textit{RN-Tr}}_i, \textbf{\textit{RN-Tr}}_j) \tag{12}$$

then it has zero probability of being a nearest neighbor of ***RN-Tr****$_q$*.

The impact of the road network settings on the pruning aspects of the lower envelope, when compared to the corresponding results in Sect. 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 fact, we 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 δ'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 Fig. 15. Namely, although ***RN-Tr***$_2$ begins to define the lower envelope of the distance from ***RN-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 Fig. 15, the leftmost node of the first level of the IPAC-NN tree, [***URN-Tr***$_1$, (t_1, t_{11})], will be augmented with the following two fields:

- **Condition:**

$$\begin{aligned} &pdf^{WD}_{o_1,o_Q}(T_d) \cdot (1 - P^{WD}_{o_2,o_Q}(T_d))\, dT_d \\ &< pdf^{WD}_{o_2,o_Q}(T_d) \cdot (1 - P^{WD}_{o_1,o_Q}(T_d))\, dT_d \end{aligned} \tag{13}$$

- **Duration:** $[t'_{11}, t_{11}]$

to indicate that throughout $[t'_{11}, t_{11}]$, although the distance of the expected location along ***RN-Tr***$_2$ is still further than the corresponding value of ***RN-Tr***$_1$ from o_Q, it may be the nearest neighbor with the highest–probability value, provided that **Condition** attribute (Eq. 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 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:

– A function *RN-Env2*$(RN\text{-}Tr_i, RN\text{-}Tr_j, t_1, t_2)$, that calculates the lower envelope of two line segments between times t_1 and t_2 takes $O(1)$ time.

– 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:

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

Regarding the construction of the collections of lower envelopes used to generate the complete answer to the NN-query throughout the time-interval of interest, the main complexity results from Sect. 4 hold also 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 reiterate that a peculiarity of the current settings is that the adapted versions of the Algorithms from Sect. 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 we present a suite of new predicates along with the algorithms for their processing. Essentially, we explore different variations of the SQL query presented in Sect. 1, reflected in the corresponding argument-signatures of the predicates. We first 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 [57], we will assume that symbols that start with a lowercase letter and those that have subscript/index denote constants from the respective domains. We will use symbols starting with an uppercase 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 by 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—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*:

- $\mathbf{C}_{11}$: PossibleNN($\boldsymbol{Tr}_i^{\mathbf{u}}, \boldsymbol{Tr}_q^{\mathbf{u}}, t_b, t_e, t$)
  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.
- $\mathbf{C}_{12}$: PossibleNN($\boldsymbol{Tr}_i^{\mathbf{u}}, \boldsymbol{Tr}_q^{\mathbf{u}}, t_b, t_e, t, k$)
  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 $\mathbf{C}_{11}$ PossibleNN($\cdots$) 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 locations 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 $\mathbf{C}_{11}$ PossibleNN($\cdots$) can be achieved in $O(1)$ time.
2. For the $\mathbf{C}_{12}$ PossibleNN($\cdots$) 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 kth 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 labeled $\boldsymbol{Tr}_i^{\mathbf{u}}$ at time t, is at depth k. This amounts to traversing a single path (the time-interval containing t) along the *IPAC-NN* tree. Assuming a fanout 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 *ϕ-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($\boldsymbol{Tr}_i^{\mathbf{u}}, \boldsymbol{Tr}_q^{\mathbf{u}}, t_b, t_e, \phi$)
  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 ϕ-portion of the interval $[t_b, t_e]$.
- $\mathbf{C}_{22}$: PossibleNN-Int($\boldsymbol{Tr}_i^{\mathbf{u}}, \boldsymbol{Tr}_q^{\mathbf{u}}, t_b, t_e, \phi, \ell$)

this predicate verifies whether $\boldsymbol{Tr}_i^{\mathbf{u}}$ has the ℓth highest–probability of being the nearest neighbor of $\boldsymbol{Tr}_q^{\mathbf{u}}$, for at least a ϕ-portion of the interval $[t_b, t_e]$.

Algorithm 4 evaluates predicate $\mathbf{C}_{21}$: PossibleNN-Int ($\boldsymbol{Tr}_i^{\mathbf{u}}, \boldsymbol{Tr}_q^{\mathbf{u}}, t_b, t_e, \phi$). 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 $O(N)$.

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

```
PossibleNN-Int ($Tr^{u}_{i}$, $Tr^{u}_{q}$, $t_b$, $t_e$, $\phi$)
Input: Uncertain trajectory $Tr^{u}_{i}$, Uncertain querying trajectory $Tr^{u}_{q}$, 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 $Tr_i$ and $Tr_q$ with the boundaries of the zone $LE_B$;
Let count = 1, Total = 0;
If ($\overline{Tr_i Tr_q}$ 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;
```

The evaluation of the $\mathbf{C}_{22}$: PossibleNN-Int(Tr^{u}_{i}, Tr^{u}_{q}, t_b, t_e, ϕ, ℓ) predicate is specified in Algorithm 5.

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

```
PossibleNN-Int ($Tr^{u}_{i}$, $Tr^{u}_{q}$, $t_b$, $t_e$, $\phi$, $\ell$)
Input: Uncertain trajectory $Tr^{u}_{i}$, Uncertain querying trajectory $Tr^{u}_{q}$, 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 labeled with $Tr^{u}_{i}$;
Let $[t^{IN}_{i1}, t^{IN}_{i2}]$ denote the time interval in the label of the $i$th such node;
Let j = 1, Total = 0;
while ($j \leq s$)
    { Total = Total + $t^{IN}_{j2}$—$t^{IN}_{j1}$;
      count++; }
If (Total $\geq \phi \cdot (t_e - t_b)$)
    return True;
return False;
```

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 Tr^{u}_{i} is encountered at depth $l < \ell$, then the entire subtree rooted at that node can be eliminated.

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

- $\mathbf{C}^{RN}_{11}$: PossibleNN((Tr^{u}_{i}, Tr^{u}_{q}, t_b, t_e, t), RoadNet) this predicate verifies whether ***RN-Tr***$^{u}_{i}$ has a non-zero probability of being the nearest neighbor of ***RN-Tr***$_{q}$ at the time instant t, when the motion of both objects is constrained by a given road network RoadNet.
- $\mathbf{C}^{RN}_{12}$: PossibleNN((Tr^{u}_{i}, Tr^{u}_{q}, t_b, t_e, t, ℓ), RoadNet) this predicate verifies whether ***RN-Tr***$^{u}_{i}$ has the ℓth highest–probability of being the nearest neighbor of ***RN-Tr***$_{q}$ at the time instant t, when the motion of both objects is constrained by a given road network RoadNet.
- $\mathbf{C}^{RN}_{21}$: PossibleNN-Int((Tr^{u}_{i}, Tr^{u}_{q}, t_b, t_e, ϕ), RoadNet) this predicates verifies whether ***RN-Tr***$^{u}_{i}$ has a non-zero probability of being the nearest neighbor of ***RN-Tr***$^{u}_{q}$, for at least a ϕ-portion of the interval $[t_b, t_e]$, where the motion of both objects is constrained by a given road network RoadNet.
- $\mathbf{C}^{RN}_{22}$:PossibleNN-Int((Tr^{u}_{i}, Tr^{u}_{q}, t_b, t_e, ϕ, ℓ), RoadNet) this predicate verifies whether ***RN-Tr***$^{u}_{i}$ has the ℓth highest–probability of being the nearest neighbor of ***RN-Tr***$^{u}_{q}$, for at least a ϕ-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 $\mathbf{C}^{RN}_{11}$, $\mathbf{C}^{RN}_{12}$, $\mathbf{C}^{RN}_{21}$ and $\mathbf{C}^{RN}_{22}$ 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 a semantic difference that stems from the nature of LE_B in road network settings. Namely, as we discussed in Sect. 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 Sect. 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 Sect. 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 [19]; 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 $\boldsymbol{Tr}_q^{\mathbf{u}}$ 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($\boldsymbol{Tr}_i^{\mathbf{u}}$, $\boldsymbol{Tr}_q^{\mathbf{u}}$, 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 **C**$_{11}$: PossibleNN($\boldsymbol{Tr}_i^{\mathbf{u}}$, $\boldsymbol{Tr}_q^{\mathbf{u}}$, 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 $\boldsymbol{Tr}_1^{\mathbf{u}}$ throughout the entire interval $[t_b, t_e]$?*

which can be specified as:

SELECT $*$ FROM MOD
WHERE PossibleNN-Int($\boldsymbol{Tr}_i^{\mathbf{u}}$, $\boldsymbol{Tr}_q^{\mathbf{u}}$, $t_b, t_e, 1$)

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 for processing **Q1**, which is $O(N)$ 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 $\boldsymbol{Tr}_q^{\mathbf{u}}$ at any time between t_b and t_e.*

SELECT **Tr**
FROM MOD
WHERE PossibleNN(**Tr**, $\boldsymbol{Tr}_q^{\mathbf{u}}$, t_b, t_e, T)
AND (T BETWEEN t_b AND t_e) where we use again 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 $\boldsymbol{Tr}_q^{\mathbf{u}}$ throughout the entire interval $[t_b, t_e]$.*

SELECT **Tr**
FROM MOD
WHERE PossibleNN-Int(**Tr**, $\boldsymbol{Tr}_q^{\mathbf{u}}$, t_b, t_e, ϕ)

Both 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 ℓth highest–probability of being a nearest neighbor of $\boldsymbol{Tr}_q^{\mathbf{u}}$, throughout at least ϕ-fraction of the interval $[t_b, t_e]$.*

SELECT **Tr**
FROM MOD
WHERE PossibleNN-Int(**Tr**, $\boldsymbol{Tr}_q^{\mathbf{u}}$, t_b, t_e, Φ, *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 demonstrate, these gains are significant when compared to the corresponding brute-force approaches.

7 Experiments

We have developed a prototype implementation of our algorithms in Java and we have performed experiments on an Intel Core-2 3.0 GHz machine with RedHat Enterprise Linux. The source code and the datasets used for our experiments are available at http://www.eecs.northwestern.edu/~goce/UncertainNN and we note that, as stated in the Introduction, we only report a small subset of experiments here (cf. [53]).

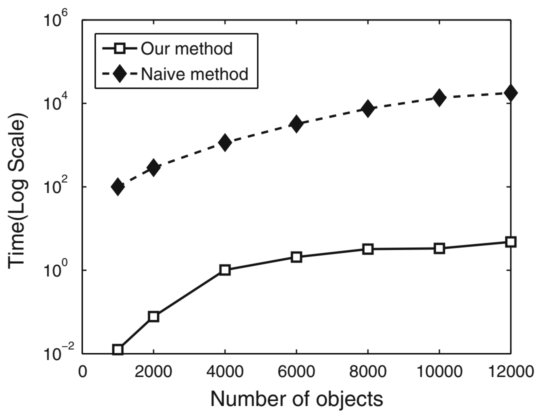

Fig. 16 Processing existential query **Q1**

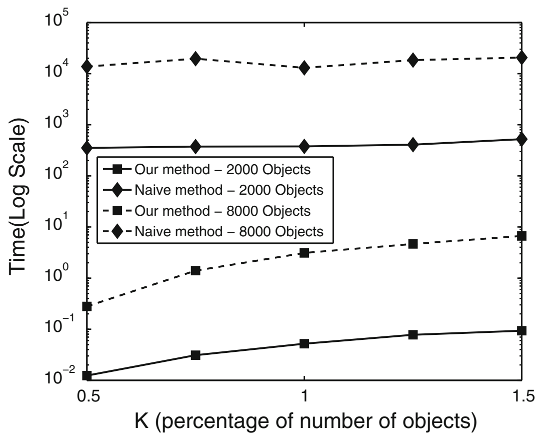

Fig. 17 Impact of the ranking parameter on existential query **Q3**

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 each object starts its motion at a random (x, y) location at a given time. The objects randomly pick a direction and speed between 15 and 60 mph, and change their velocity vectors simultaneously every 3 min, in a manner that satisfies the following constraints: (1) The new direction of motion is within $\pm 60^\circ$ from the previous one; (2) The new speed is again within the range [15,60] mph.

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 reiterate 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. [12,15,23]). In the sequel, we present the results of two groups of experiments that we conducted [53]. Since our algorithms outperform the corresponding naïve approaches by several orders of magnitude, the time axis is represented using a logarithmic scale. For each chart, 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 *every intersection point* between two distance functions throughout the time-interval. Clearly, this incurs a major overhead caused by unnecessary computations, as illustrated in Fig. 16.

Among other things, we investigated the speed-up offered by our approach when processing a variant of query **Q3** with ranking, as a function of k. The results for two datasets, of sizes 2,000 and 8,000 trajectories, respectively, are shown in Fig. 17. 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.

8 Related work

Processing nearest neighbor queries is of interest in a variety of applications, ranging from clustering and data mining, to machine learning and computer vision [45]. The problem of scalable and efficient processing of k-NN queries in the context of spatial databases has been addressed with a branch-and-bound approach [42] and with an incremental technique [21], both of which use R-trees as an index.

Recent years have witnessed tremendous growth in MOD research [19], 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, 2,13,28,33,38]. Specifically, for efficient processing of k-NN queries in spatio-temporal settings, a dual transformation (points to lines) is explored for the settings in which the objects are moving in one dimension [27]. Generic methodologies for processing spatio-temporal queries for trajectories, based on a rich algebra of types, are presented in [30]. 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 [49]. One specific context is when the motion is constrained to road networks [10]. Techniques for efficient processing of NN queries in such setting have been presented in [35,60]. Closely related to our current work, results on the efficient processing of continuous k-NN queries on trajectories are presented in [12] and in [15], and k-NN query processing on road networks is addressed in [23].

Two works very similar in spirit to ours, considering a collection of hyperbolae representing the distance functions from a querying object, are [4,41]. In particular, [41] 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 [4] 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.

The uncertainty model adopted in our work is used for processing range queries in MOD settings [55], 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 [8] 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 [23], 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. Although no formal complexity analysis is given, it appears that the construction of the upper envelope takes quadratic time. More recently [31] 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 in-between 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 [23,31] do not explicitly consider the consequences of mixing in the travel-time distance with the location-based uncertainty of the objects' whereabouts in a given time instant.

9 Conclusions and future work

In summary, we addressed the problem of processing efficiently the time-parameterized 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 traveling in 2D space, and by a line segment for motions 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 [4]) and compare the semantics of traditional *Top-k* NN queries [47] for crisp trajectories with that for uncertain trajectories. In this work, we focused on the qualitative ranking of the answers; however, 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* [7].

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; a spatio-temporal variant of the U-tree structure [51] for free 2D motion is desirable. In addition, we would like to explore the possibility of using the indexing structures from [23,31] to couple the uncertainty due to speed with that due to location. Lastly, given recent trends in MOD research, we plan to investigate how our results can be applied to spatio-temporal data warehousing and mining [36].

Acknowledgments We thank the anonymous reviewers for their constructive suggestions, Abraham Haddad and Ajit Tamhane for their useful comments, and Hui Ding for helping with the experiments presented in the conference version [54].

References

1. Agarwal, P.K., Arge, L., Erickson, J.: Indexing moving points. In: ACM PODS (2000)

2. Aggarwal, C.C., Agarwal, D.: On nearest neighbor indexing of nonlinear trajectories. In: ACM PODS (2003)
3. Aggarwal, C.C., Yu, P.S.: A survey of uncertain data algorithms and applications. IEEE Trans. Knowl. Data Eng. **21**(5), 609–623 (2009)
4. Benetis, R., Jensen, C.S., Karciauskas, G., Saltenis, S.: Nearest and reverse nearest neighbor queries for moving objects. VLDB J. **15**(3), 229–249 (2006)
5. Böhm, C., Ooi, B.C., Plant, C., Yan, Y.: Efficiently processing continuous k-NN queries on data streams. In: ICDE (2007)
6. Cao, H., Wolfson, O., Trajcevski, G.: Spatio-temporal data reduction with deterministic error bounds. VLDB J. **15**(3), (2006)
7. Cheng, R., Chen, J., Mokbel, M.F., Chow, C.-Y.: Probabilistic verifiers: evaluating constrained nearest-neighbor queries over uncertain data. In: ICDE (2008)
8. Cheng, R., Kalashnikov, D.V., Prabhakar, S.: Querying imprecise data in moving objects environments. IEEE Trans. Knowl. Data Eng **16**(9), 112–1127 (2004)
9. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational geometry: algorithms and applications. Springer, New York (2001)
10. Demiryurek, U., Pan, B., Kashani, F.B., Shahabi, C.: Towards modeling the traffic data on road networks. In: GIS-IWCTS (2009)
11. Ding, Z., Güting, R.H.: Managing moving objects on dynamic transportation networks. In: SSDBM (2004)
12. Gao, Y., Li, C., Chen, G., Chen, L., Jiang, X., Chen, C.: Efficient *k*-nearest-neighbor search algorithms for historical moving object trajectories. J. Comput. Sci. Technol. **22**(2), 232–244 (2007)
13. Gedik, B., Wu, K.-L., Yu, P.S., Liu, L.: Processing moving queries over moving objects using motion-adaptive indexes. IEEE Trans. Knowl. Data Eng. **18**(5), 651–668 (2006)
14. Gnedenko, B.V.: Course of Probability Theory. Nauka, Moscow (1988)
15. Güting, R.H., Behr, T., Xu, J.: Efficient *k*-nearest neighbor search on moving object trajectories. VLDB J. **19**(5), 687–714 (2010)
16. Güting, R.H., Böhlen, M.H., Erwig, M., Jensen, C.S., Lorentzos, N., Nardelli, E., Schneider, M., Viqueira, J.R.R.: Spatio-temporal models and languages: an approach based on data types. In: Spatio-Temporal Databases—The CHOROCHRONOS Approach (2003)
17. Güting, R.H., Böhlen, M.H., Erwig, M., Jensen, C.S., Lorentzos, N.A., Schneider, M., Vazirgiannis, M.: A foundation for representing and querying moving objects. ACM Trans. Database Syst. **25**(1), 1–42 (2000)
18. Güting, R.H., de Almeida, V.T., Ding, Z.: Modeling and querying moving objects in networks. VLDB J. **15**(2), 165–190 (2006)
19. Güting, R.H., Schneider, M.: Moving Objects Databases. Morgan Kaufmann, Los Altos (2005)
20. Hägerstrand, T.: What about people in regional science?. Papers Reg. Sci. Assoc. **24**, 7–21 (1970)
21. Hjaltason, G.R., Samet, H.: Distance browsing in spatial databases. ACM Trans. Database Syst. **24**(2), 265–318 (1999)
22. Hornsby, K., Egenhofer, M.J.: Modeling moving objects over multiple granularities. Ann. Math. Artif. Intell. **36**(1-2), 177–194 (2002)
23. Huang, Y.-K., Chen, Z.-W., Lee, C.: Continuous k-nearest neighbor query over moving objects in road networks. In: APWeb/WAIM, pp. 27–38 (2009)
24. Huang, Z., Lu, H., Ooi, B.C., Tung, A.K.H.: Continuous skyline queries for moving objects. IEEE Trans. Knowl. Data Eng. **18**(12), 1645–1658 (2006)
25. Iwerks, G.S., Samet, H., Smith, K.P.: Maintenance of K-nn and spatial join queries on continuously moving points. ACM Trans. Database Syst. **31**(2), (2006)
26. Jensen, C.S., Lin, D., Ooi, B.C., Zhang, R.: Effective density queries on continuously moving objects. In: ICDE, p. 71 (2006)
27. Kollios, G., Gunopulos, D., Tsotras, V.: Nearest neighbor queries in a mobile environment. In: STDM, pp. 119–134 (1999)
28. Kollios, G., Gunopulos, D., Tsotras, V.: On indexing mobile objects. In: ACM PODS, pp. 261–272 (1999)
29. Kuijpers, B., Othman, W.: Trajectory databases: data models, uncertainty and complete query languages. J. Comput. Syst. Sci. **76**(7), (2010)
30. Lema, J.A.C., Forlizzi, L., Güting, R.H., Nardelli, E., Schneider, M.: Algorithms for moving objects databases. Comput. J. **46**(6), (2003)
31. Li, G., Li, Y., Shu, L., Fan, P.: C-kNN query processing over moving objects with uncertain speeds in road networks. In: APWeb (2011)
32. Lim, J.S.: Two-Dimensional Signal and Image Processing. Prentice Hall, Englewood Cliffs (1990)
33. Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y., Cheung, D.W.: Mining, indexing, and querying historical spatio-temporal data. In: ACM SIGKDD (2004)
34. Mokbel, M.F., Aref, W.G.: SOLE: scalable on-line execution of continuous queries on spatio-temporal data streams. VLDB J. **17**(5), (2008)
35. Mouratidis, K., Yiu, M.L., Papadias, D., Mamoulis, N.: Continuous nearest neighbor monitoring in road networks. In: VLDB, pp. 43–54 (2006)
36. Nanni, M., Kuijpers, B., Körner, C., May, M., Pedreschi, D.: Spatiotemporal data mining. In: Mobility, Data Mining and Privacy (2008)
37. Pei, J., Hua, M., Tao, Y., Lin, X.: Query answering techniques on uncertain and probabilistic data: tutorial summary. In: ACM SIGMOD (2008)
38. Pelanis, M., Saltenis, S., Jensen, C.S.: Indexing the past, present, and anticipated future positions of moving objects. ACM Trans. Database Syst. **31**(1), (2006)
39. Pfoser, D., Jensen, C.S.: Capturing the uncertainty of moving objects representation. In: SSD (1999)
40. Pfoser, D., Tryfona, N., Jensen, C.S.: Indeterminacy and spatiotemporal data: basic definitions and case study. GeoInformatica **9**(3), (2005)
41. Raptopoulou, K., Papadopoulos, A., Manolopoulos, Y.: Fast nearest-neighbor query processing in moving-object databases. GeoInformatica **7**(2), 113–137 (2003)
42. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest neighbor queries. In: ACM SIGMOD (1995)
43. Royden, H.L.: Real Analysis. Macmillan Co, New York (1963)
44. Shahabi, C., Kolahdouzan, M.R., Sharifzadeh, M.: A road network embedding technique for k-nearest neighbor search in moving object databases. GeoInformatica **7**(3), 255–273 (2003)
45. Shakhnarovich, G., Darrel, T., Indyk, P. (eds.): Nearest-Neighbor Methods in Learning and Vision: Theory and Practice. MIT Press, Cambridge (2006)
46. Sharir, M., Agarwal, P.K.: Davenport–Schinzel Sequences and Their Geometric Applications. Cambridge University Press, Cambridge (1995)
47. Soliman, M.A., Ilyas, I.F., Chang, K.C.-C.: Top-k query processing in uncertain databases. In: ICDE (2007)
48. Suciu, D., Dalvi, N.N.: Foundations of probabilistic answers to queries. In: ACM SIGMOD (2005) (tutorial)
49. Tao, Y., Papadias, D.: Spatial queries in dynamic environments. ACM Trans. Database Syst. **28**(2), 131–139 (2003)
50. Tao, Y., Papadias, D., Sun, J.: The TPR$*$-tree: an optimized spatio-temporal access method for predictive queries. In: VLDB, pp. 790–801 (2003)
51. Tao, Y., Xiao, X., Cheng, R.: Range search on multidimensional uncertain data. ACM Trans. Database Syst. **32**(3), 15 (2007)
52. Theodoridis, Y., Sellis, T., Papadopoulos, A., Manolopoulos, Y.: Specifications for efficient indexing in spatiotemporal databases. In: SSDBM (1998)

53. Trajcevski, G., Tamassia, R., Cruz, I.F., Scheuermann, P., Hartglass, D., Zamierowski, C.: Ranking continuous nearest neighbors for uncertain trajectories: full and Peer Reviewed Accepted Version. Technical Report NWU-EECS-11-06, Dept. of EECS, Northwestern University (2011)
54. Trajcevski, G., Tamassia, R., Ding, H., Scheuermann, P., Cruz, I.F.: Continuous probabilistic nearest-neighbor queries for uncertain trajectories. In: EDBT (2009)
55. Trajcevski, G., Wolfson, O., Hinrichs, K., Chamberlain, S.: Managing uncertainty in moving objects databases. ACM Trans. Database Syst. **29**(3), 463–507 (2004)
56. Trivedi, K.S.: Probability and Statistics with Reliability, Queueing and Computer Science Applications. Wiley, London (2002)
57. Ullman, J.D.: Principles of Database and Knwoledge—Base Systems. Computer Science Press, Rockville (1989)
58. Wolfson, O., Sistla, A.P., Chamberlain, S., Yesha, Y.: Updating and querying databases that track mobile units. Distr Parallel Databases **7**(3), 257–387 (1999)
59. Xia, T., Zhang, D.: Continuous reverse nearest neighbor monitoring. In: ICDE (2006)
60. Yiu, M.L., Mamoulis, N., Papadias, D.: Aggregate nearest neighbor queries in road networks. IEEE Trans. Knowl. Data Eng. **17**(6), 820–833 (2005)
61. Yu, X., Pu, K.Q., Koudas, N.: Monitoring k-nearest neighbor queries over moving objects. In: ICDE, pp. 631–642 (2005)