Uncertain Range Queries for Necklaces

Goce Trajcevski*(1) Alok Choudhary *(2) Ouri Wolfson †(3) Li Ye * Gang Li *

*Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, Il 60208
Email: goce,chouchar,li.ye, gang.li@eecs.northwestern.edu

†Department of Computer Science
University of Illinois at Chicago
Chicago, Il 60612
Email: wolfson@cs.uic.edu

***Abstract*—We address the problem of efficient processing of spatio-temporal range queries for moving objects whose whereabouts in time are not known exactly. The fundamental question tackled by such queries is, given a spatial region and a temporal interval, retrieve the objects that were inside the region during the given time-interval. As earlier works have demonstrated, when the (location,time) information is uncertain, syntactic constructs are needed to capture the impact of the uncertainty, along with the corresponding processing algorithms. In this work, we focus on the uncertainty model that represents the whereabouts in-between two known locations as a bead, and an uncertain trajectory is represented as a necklace – a sequence of beads. For each syntactic variant of the range query, we present the respective processing algorithms and, in addition, we propose pruning strategies that speed up the generation of the queries' answers. We also present the experimental observations that quantify the benefits of our proposed methodologies.**

I. Introduction

The efficient management of *(location,time)* information of mobile entities is essential for many applications that range from navigation and efficient traffic management, through environmental health monitoring and remediation [1], [2], [7], [15]. Investigating techniques for efficient storage, retrieval and query processing of such data has been a topic of studies of the field of Moving Objects Databases (MOD) [9]. However, as has been observed in the literature [3], [4], [11], [13], [14], [19], [21], [22], [25], almost all the applications that rely on some form of Location Based Services (LBS) [18] need to incorporate the factor of the *uncertainty* of the spatio-temporal information. The uncertainty is an inherent property of the *(location,time)* data in many domains, due to various sources: – the imprecision of the Global Positioning System (GPS) devices [25]; – the imprecision and of the tracking sensors and localization errors [24]; – the imprecision of the roadside sensors [5]; etc. There are other, complementary, motivations for inducing the uncertainty in the system like, for example, trading off the precision for communication cost [8], or protection of the users privacy [16]. As noted in [22], unless the uncertainty is captured in the model and the query language, the burden of factoring it out from the queries' answers is solely on the user. Hence, depending on the chosen model of uncertainty, for each query of interest a proper linguistic construct needs to be provided, along with the algorithm for efficient processing of that query.

(1) Research Supported by the NSF: CNS-0910952
(2) Research Supported by NSF: CNS-0551639, IIS-0536994, HECURA CCF-0621443, OCI 0956311, SDCI OCI-0724599
(1) Research Supported by the NSF: DGE-0549489, IIS-0957394, IIS-0847680

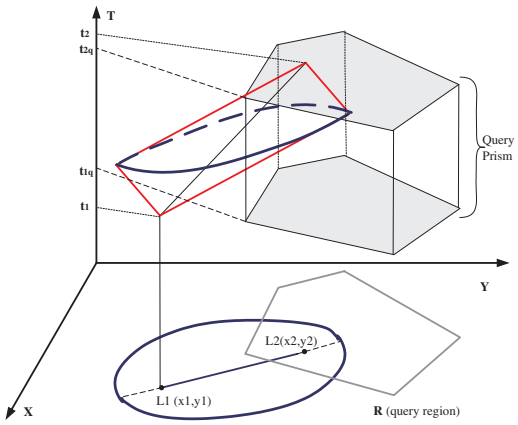


Fig. 1. Spatio-temporal range query for a bead

We focus on processing spatio-temporal *range* for trajectories, for which the basic syntax is **Qb**: *"Retrieve the moving objects that were inside the region R, between t_{1q} and t_{2q}"*. R denotes the boundaries of the region of interest, and $[t_{1q}, t_{2q}]$ is the time-interval of interest for the query. However, when uncertainty is present, **Qb** may have different variants. For example, in tracking settings in wireless sensor networks, one may be interested in **Qb1**: *"Retrieve the moving objects that have a non-zero probability of possibly being inside the region R, sometimes between t_{1q} and t_{2q}"*. Such types of queries have been investigated in [22], however, the location uncertainty adopted in that work was the one of a disk with a given (fixed) uncertainty-radius, around the expected location of the moving object at any time-point. In the 2D space + time system, this model yields a sequence of sheared cylinders representing the possible whereabouts.

In this work, we consider a different uncertainty model – the one in which the possible whereabouts of a given object in-between two consecutive updates are bounded by the so-called *beads*, and the trajectory – represented as sequences of beads, is called a *necklace*. The model and the concepts were introduced under these names in [11], and earlier by

978-0-7695-4048-1/10 $26.00 © 2010 IEEE
DOI 10.1109/MDM.2010.76

time-geography community. Arguably, many properties of the beads can be equivalently stated in terms of the properties of the uncertainty model presented [19] – a 2D ellipse. Recently, a very thorough treatment of this particular model, as well as a logical formalization of trajectories databases and query languages, has been presented in [14]. At the heart of the beads model is the assumption that in-between two location updates $L1(x_1, y_1)$ at t_1 and $L2(x_2, y_2)$ at t_2, the only thing known about the moving object is that its maximum *speed* is bounded, i.e., $|\overline{v}| \leq |\overline{v_{max}}|$. An illustration of the main intuition behind the range queries in beads settings is provided in Figure 1. The query prism has the bases consisting of the query-region R at the times of interest t_{bq} and t_{eq}. The projection of the possible whereabouts of the moving objects in 2D corresponds to an ellipse, with foci at L_1 and L_2 (cf. [19]). When represented in the 3D space (=2D + time), this model yields the bead, which is obtained via an intersection of the two cone-volumes with vertices at (x_1, y_1, t_1) and (x_2, y_2, t_2) [11]. The shape of the bead (equivalently, the shape of the ellipse), depend on the value of[1] $\overline{v_{max}}$ (cf. [14]).

The main contributions of this work can be summarized as follows:

– We formalize the variants of uncertain spatio-temporal range queries presented in [22] in the settings in which the uncertainty is modelled via beads/necklaces, and we demonstrate that some of the properties presented [22] are valid these settings.

– For each query type, we present an efficient processing algorithm. In addition, we propose pruning strategies that can be used wheh processing spatio-temporal range queries.

– We present experimental observations which demonstrate that prunning can yield a speed-up of up to a factor of 2 when processing particular query types.

The rest of this paper is structured as follows. In Section 2 we recollect some necessary background and introduce the foundations of the uncertainty model used throughout this work. In Section 3, we introduce the different query types and present the respective processing algorithms, as well as the pruning strategies. Section 4 presents our experimental results, and Section 5 positions our work with respect to the related literature. Finally, in Section 6 we conclude the paper and outline directions for future work.

II. Preliminaries

We now proceed with introducing the concepts that will be used throughout the rest of this paper. Firstly, we give a formal definition of a trajectory, its uncertainty model and the concepts of beads, necklaces and possible motion curves. Subsequently, we present few properties of the beads that will be used when deriving the algorithms for processing the different variants of the range queries.

The basic concept describing the motion of a given object is the one of a trajectory (cf. [14], [22]):

[1]Whenever clear from the context, when describing the *speed* we will simply use a scalar notation, e.g. v_{max}.

Definition 1: A *trajectory* Tr of a moving object, is a polyline in a 3D space (2D spatial + time), represented as a sequence of points $Tr = (x_1, y_1, t_1), \ldots, (x_n, y_n, t_n)$, where $\forall(i, j)(i < j \Rightarrow t_i < t_j)$. Between two consecutive points (x_i, y_i, t_i) and $(x_{i+1}, y_{i+1}, t_{i+1})$, the object is assumed to move along the straight line-segment $(\overline{(x_i, y_i)(x_{i+1}, y_{i+1})})$, and with a constant *expected speed* $v_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}/(t_{i+1} - t_i)$. The *expected location* of the object at any time-point t ($\in (t_i, t_{i+1})$) is the one obtained via linear interpolation between the endpoints, using the expected speed v_i. The projection of Tr_k in the Euclidian 2D space is called its *route*.

Definition 1 implicitly casts the trajectory as a function from *Time* domain into the 2D Euclidian space (i.e., $f(t) \rightarrow \mathbf{R}^2$). While it gives, in a sense, a crisp description of the whereabouts-in-time of a given object, as we mentioned in Section 1, it often may be the case in practice that the vertices of the polyline are actual samplings (obtained via GPS device or tracking sensors), and little is known about the location of the object in-between two such consecutive samples. Hence, we need to formally characterize the set of those possible locations, for which we have the following (cf. [14]):

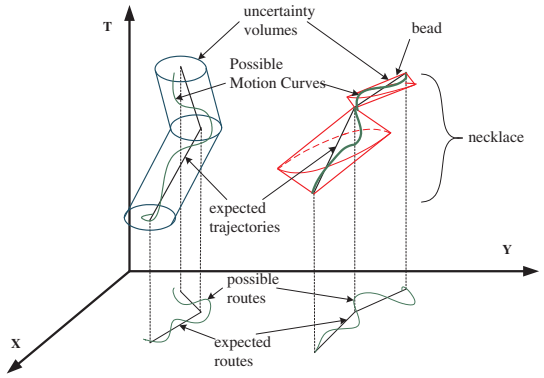


Fig. 2. Uncertain Trajectory (Beads/Necklace)

Definition 2: Let v^i_{max} denote the maximum *speed* that an object can take within the time-interval (t_i, t_{i+1}). A *bead* B_i = $((x_i, y_i, t_i), (x_{i+1}, y_{i+1}, t_{i+1}), v^i_{max})$ is defined as the set of all the points (x, y, t) satisfying the following constraints:

$$\begin{cases} t_i \leq t \leq t_{i+1} \\ (x - x_i)^2 + (y - y_i)^2 \leq [(t - t_i)v^i_{max}]^2 \\ (x - x_{i+1})^2 + (y - y_{i+1})^2 \leq [(t_{i+1} - t)v^i_{max}]^2 \end{cases} \quad (1)$$

The first and the second constraint of Equation 1, when taken together describe a cone emanating upwards from (x_i, y_i, t_i), with a vertical axis and with circles whose radius value at time t is $(t - t_i)v^i_{max}$, whereas the first and the third constraint together, specify a cone emanating downwards from $(x_{i+1}, y_{i+1}, t_{i+1})$, with a vertical axis and with circles whose radius at time t is $(t_{i+1} - t)v^i_{max}$. Hence, the bead B_i can be viewed as volume defined by the intersection of those two

cones. We note that at $t = t_i$ (resp. $t = t_{i+1}$) the locations of the object are crisp (i.e., no uncertainty). We now have:

Definition 3: Given a trajectory Tr, its corresponding *uncertain trajectory* UTr is a *sequence of beads*, B_1, B_2, ..., B_{n-1}, also called a *lifeline necklace* (cf. [11], [14]). A *possible motion curve* $PMC(Tr)$ of UTr is any function *f: Time* $\rightarrow \mathbf{R}^2$ for which every point (x, y, t), is either a vertex of the polyline of Tr, or it satisfies $(x, y) = f(t)$ and is inside the corresponding bead – i.e., $(\forall t)(t_i < t < t_{i+1}) \Rightarrow ((x, y, t) \in B_i)$.

The concepts introduced in Definitions 1-3 are illustrated in Figure 2. For comparison, the left part of Figure 2 illustrates the corresponding concepts when the uncertainty of the object's location at any time t is represented as a disk with a fixed radius r (cf. [22]). We note that Definition 2 and Definition 3 slightly deviate from the corresponding definitions in [14], in the sense that [14] assumes a fixed value of v_{max} for every bead throughout the whole lifetime necklace, whereas we allow different beads B_i to have different bounds on their respective maximum speeds v^i_{max}. This servers the purpose of capturing the possibility that the maximum speed can be different on different terrains or road-segments, and will be of relevance when we discuss the pruning strategies in Section 3.

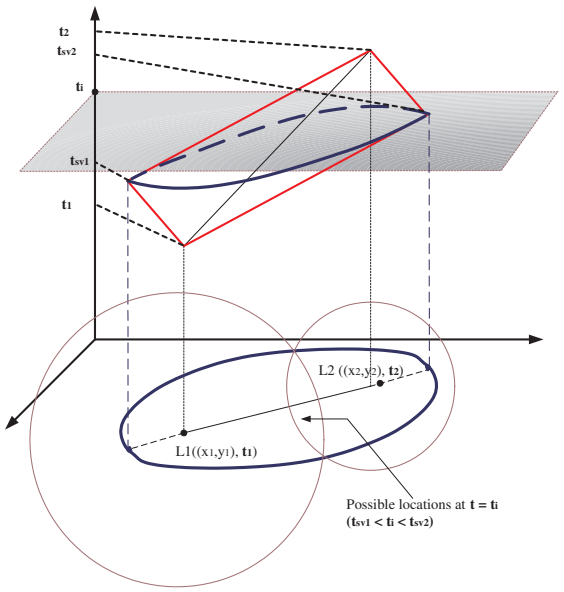


Fig. 3. Components and (X,Y) projection of beads

We now proceed with introducing few more properties[2] of the beads, which will be important for specifying the algorithms for the corresponding predicates specifying the different types of range queries. For a given bead B_i, let $d_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$ denote the distance between locations of the starting location (at t_i) and ending location (at t_{i+1}). Also, let $t_{sv_i} = (t_i + t_{i+1})/2 - d_i/2v^i_{max}$ and $t_{sv_{i+1}} = t_i + t_{i+1})/2 + d_i/2v^i_{max}$. We observe that each bead had two distinct types of volumes:

(1) *Single disk volumes:* For every $t \in [t_i, t_{sv_i}]$, the spatial boundary of the bead at t is a circle with radius $r(t) = v^i_{max}(t - t_i)$, centered at (x_i, y_i). Similarly, for every $t \in [t_{sv_{i+1}}, t_{i+1}]$, the spatial boundary of the bead at t is a circle with radius $r(t) = (t_{i+1} - t)v^i_{max}$, centered at (x_{i+1}, y_{i+1}). Hence, throughout $[t_i, t_{sv_i}]$, the 3D volume of the bead consists of a single cone, with a vertex at (x_i, y_i, t_i) (similarly for $[t_{sv_{i+1}}, t_{i+1}]$).

(2) *Two disks volume:* In-between t_{sv_i} and $t_{sv_{i+1}}$, (i.e., $t \in [t_{sv_i}, t_{sv_{i+1}}]$), the boundary of the bead at t is an intersection of two circles: $C^i_{down}(t)$, centered at (x_i, y_i), with radius $r_{down}(t) = (t - t_i)v^i_{max}$, and $C^i_{up}(t)$, centered at (x_{i+1}, y_{i+1}), with radius $r(t) = (t_{i+1} - t)v^i_{max}$.

We conclude this section with noting one more property of the beads: the projection of the bead B_i onto the the (X, Y) plane is an ellipse (cf. [14], [19]), with foci at (x_i, y_i), and (x_{i+1}, y_{i+1}), and with equation:

$$\frac{(2x - x_i - x_{i+1})^2}{(v^i_{max})^2(t_{i+1} - t_i)^2} + \frac{(2y - y_i - y_{i+1})^2}{(v^i_{max})^2(t_{i+1} - t_i)^2 - (x_{i+1} - x_i)^2 - (y_{i+1} - y_i)^2} = 1 \quad (2)$$

We will use El_i to denote the ellipse resulting from projecting the bead B_i in the (X, Y) plane. Figure 3 provides an illustration of the different components of the (volume of the) bead and its corresponding shapes at different time-points, as projected on the horizontal (X, Y) plane.

III. Processing Uncertain Range Queries

We now focus on the specification of different types of spatio-temporal range queries for beads/necklaces, and the respective processing algorithms. Firstly, we analyze the different categories of queries – their syntax and semantics – noting that in this work we focus on the *qualitative* (cf. [22] queries. Subsequently, we proceed with the details of the queries processing for which we present two pruning strategies, along with the individual refinement-algorithms.

In the sequel, for a given range query with parameters: – R, the spatial region; – and $[t_{bq}, t_{eq}]$, the temporal interval of interest, we will denote the set $\{\forall(x, y, t)|(x, y) \in R$ and $t \in [t_{bq}, t_{eq}]\}$ with QP_R (Query Prism). We reiterate that in this work we focus on query regions that are simple and bounded by convex polygons.

A. Categories of Range Queries

The categories of queries considered in this work are same as the ones considered in [22] – namely, predicates specifying whether a given moving object is *Inside* the query region R, and the reasons that we need more than one predicate are: (1) Due to the continuity of time, one may be interested whether a given moving object is inside R, *sometimes* or *always* throughout $[t_{bq}, t_{eq}]$; (2) Due to the uncertainty of the location, at a given time-point an object may *possibly* be inside R, or *definitely* so. In the spirit of [22], and using the

[2]For detailed exposition, see [14].

notation developed in Section 2, we now proceed with defining the individual predicates for each query category. Note that, without loss of generality, when we quantify over the temporal values, we assume that the domain of the interpretation is limited by the time-interval of interest for the query, $[t_{bq}, t_{eq}]$.

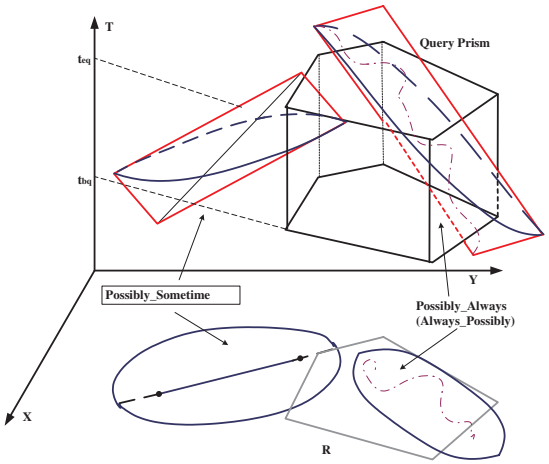


Fig. 4. Queries for $\exists PMC(Tr)$

• *Possibly Sometimes Inside* (Tr, R, t_{bq}, t_{eq}): is a predicate which is true iff there exists some possible motions curve *PMC(Tr)* which, at some time $t \in [t_{bq}, t_{eq}]$ is inside R. In other words, quantifying over the temporal domain and the domain of all the possible motion curves of a given trajectory (i.e., ones within the volume bounded by the lifeline necklace), we obtain: $(\exists PMC(Tr))(\exists t)[Inside(R, PMC(Tr,), t)]$.
– *Sometimes Possibly Inside* (Tr, R, t_{bq}, t_{eq}): as demonstrated in [22], this predicate can be stated as $(\exists t)(\exists PMC(Tr))[Inside(R, PMC(Tr,), t)]$, and is equivalent to *Possibly Sometimes Inside*.
• *Possibly Always Inside* (Tr, R, t_{bq}, t_{eq}): this predicate is true iff there is some (at least one) *PMC(Tr)* that is inside R all throughout $[t_{bq}, t_{eq}]$. Formally: $(\exists PMC(Tr))(\forall t)[Inside(R, PMC(Tr,), t)]$. Observe that now we are focusing on one specific function $f(t) : Time \rightarrow \mathbf{R}^2$, throughout the given time-interval of interest for the query.
• *Always Possibly Inside* (Tr, R, t_{bq}, t_{eq}): is true iff for every time-instance t throughout the time-interval of interest for the query, there exists some – unlike the previous predicate, not necessarily unique – *PMC(Tr)* that is inside R at t. In other words, $(\forall t)(\exists PMC(Tr))[(t \in [t_{bq}, t_{eq}]) \wedge Inside(R, PMC(Tr,), t)]$.

We have the following:

Property 1: Whenever *Possibly Always Inside* is true, so is *Always Possibly Inside*. In addition, when R is convex, the two predicates are equivalent.

Proof: (Sketch, cf. [22]) The first part of Property 1 is true due to the properties of the First-Order Logic – for any predicate $P(x, y, \ldots)$, $(\exists x)(\forall y)P(x, y, \ldots) \Rightarrow (\forall y)(\exists x)P(x, y, \ldots)$. For the second part (equivalence of the predicates for convex polygons), the steps of the proof are almost exactly the same as the ones used in the proof of Theorem 4.1. in [22]. ■

An illustration of the predicates introduced thus far is presented in Figure 4.

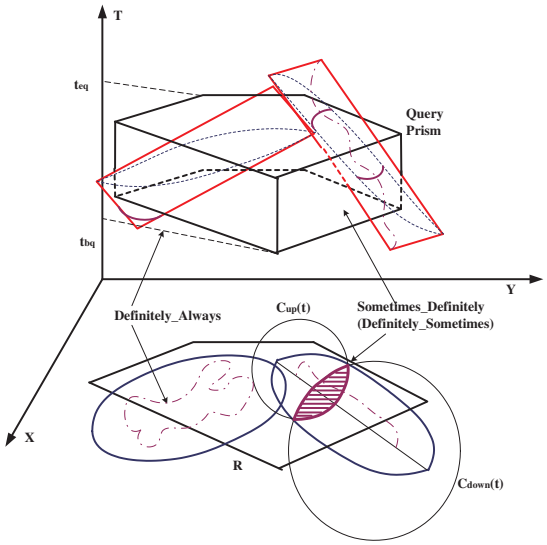


Fig. 5. Queries for $\forall PMC(Tr)$

The rest of the predicates are defined as follows:
• *Always Definitely Inside* (Tr, R, t_{bq}, t_{eq}): is true iff for every time-instance t throughout the time-interval of interest for the query, every single *PMC(Tr)* is inside R. In other words, $(\forall t)(\forall PMC(Tr))[Inside(R, PMC(Tr,), t)]$. We note that this predicate is equivalent to *Definitely Always Inside* (Tr, R, t_{bq}, t_{eq}), because switching the order of two universal quantifiers does not affect the truth of the formula (cf. [22]
• *Definitely Sometimes Inside* (Tr, R, t_{bq}, t_{eq}): is true iff for every possible motion curve *PMC(Tr)* (i.e., for every different $f(t) : Time \rightarrow \mathbf{R}^2$), there exists some time-instance (not necessarily unique), at which that *PMC(Tr)* is inside the query region R. Formally: $(\forall(PMC(Tr)))(\exists t)[Inside(R, PMC(Tr,), t)]$
• *Sometimes Definitely Inside* (Tr, R, t_{bq}, t_{eq}): is true iff there exists (at least one) a time-instance $t \in [t_{bq}, t_{eq}]$ at which every possible motion curve *PMC(Tr)* is inside the query region R. Formally: $(\exists t)(\forall PMC(Tr))[Inside(R, PMC(Tr,), t)]$
Similarly to Property 1, we have:

Property 2: Whenever *Sometimes Definitely Inside* is true, so is *Definitely Sometimes Inside*.

Proof: (sketch – cf. [22]) For any predicate $P(x, y, \ldots)$, $(\exists x)(\forall y)P(x, y, \ldots) \Rightarrow (\forall y)(\exists x)P(x, y, \ldots)$ ■

.

The semantics of each of the last three predicates is illustrated in Figure 5.

For a given circle C, let $D(C)$ denote the *disk* bounded by C, together with the boundary. As indicated on the right portion of the Figure 5, in order to guarantee that the predicate *Sometime Definitely Inside* is true, in general, it is sufficient

to find a *lens*, i.e. a region bounded by the intersection of $D(C_i^{down}(t))$ and $D(C_i^{up}(t))$ for some t, that is entirely inside R at the time t. We elaborate on the details below.

Note that, although we discussed a total of 8 predicates ($2^2 \cdot 2!$), in effect we have 6 different ones, because of two pairs of equivalences *Possibly Sometimes Inside* $\equiv$ *Sometimes Possibly Inside*, and *Definitely Always Inside* $\equiv$ *Always Definitely Inside*. We conclude this section with a summary observation that the predicate *Definitely Always Inside* is strongest among all, in the sense that whenever it is true, so is any other predicate. On the contrary, the predicate *Possibly Sometimes Inside* is the weakest one, in the sense that whenever any other predicate is true, so it *Possibly Sometimes Inside*.

B. Query Processing

Typically [9] the processing of spatio-temporal queries is done in stages:
(1) *Filtering*, where an index is used to eliminate the retrieval (often, from the disk) of the data items that are guaranteed not to satisfy the query [20], [21];
(2) *Pruning*, where certain properties may be used to further filter out a portion of the data set already inside the main memory (without introducing any false negatives) [3]; and
(3) *Refinement*, where corresponding algorithms are used eliminate all the *false positives* that were not filtered out during the previous stage(s).

In this work, we do not address the problem of indexing and, in the sequel, we focus on the last two stages of the query processing.

1) Pruning Stage: One of the modelling issues addressed in [11] was representing the lifeline necklace at different levels of granularity – i.e., encompassing several beads' volumes into a larger volume. Motivated by this, in the current work we propose modifications of the approaches presented in [11] as different types of pruning strategies.

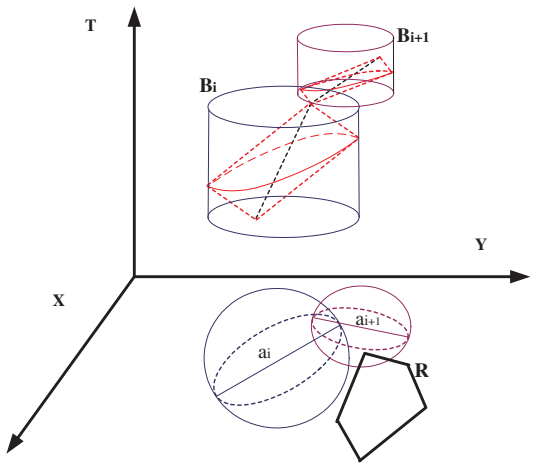


Fig. 6. Cylinder-based pruning approximations

Large bound (LB): In this extreme approach, the entire necklace is bounded by a single minimal vertical cylinder[3]. As can be expected, this strategy is prone to introducing large volumes of a, so-called, *dead-space* and completely eliminate any benefit of the pruning for the purpose of speeding up the refinement stage.

Individual Bead Bounds (IBb): This pruning strategy approximates each bead in a given necklace with its minimum bounding vertical cylinder, thereby approximating the necklace with a sequence of vertical cylinders. In effect, the ellipse – which is the projection of a given bead on the (X, Y) plane, becomes approximated by a circle centered in the center of the respective ellipse, and with a diameter equal to the major-axis of the ellipse. Based on Equation 2, the radius of the approximation-disk Ad_i corresponding to the bead B_i is: $r(Ad_i) = 1/2(v^i_{max})(t_{i+1} - t_i)$.

Uniform Bead Bound (UBb): This pruning strategy still focuses on applying vertical cylinders to approximate each individual bead in a given necklace, however, instead of using different radii for the basis of each cylinder, we simply approximate every bead with a uniform base. In other words, each ellipse is approximated by a circle with a same radius: $r = max_i(r(Ad_i))$. The *UBb* pruning strategy provides a balance between keeping an extra storage item for each bead (consequently, $O(n)$ extra storage for a necklace of size $O(n)$) as done with *IBb*, as opposed to keeping one single parameter for the entire necklace (albeit, introducing more dead space). Figure 6 illustrates the *IBb* pruning strategy for a partially-shown necklace (two consecutive beads). Naturally, since the circle approximating B_i does not intersect with the query-region R, one cannot expect that B_i will intersect the corresponding query-prism. Hence, the bead B_i can be safely pruned from any further consideration regarding a particular predicate.

2) Refinement Algorithms: Before proceeding with the details of the individual algorithms, few remarks are in order. Firstly, given the temporal parameter of the query $[t_{bq}, t_{eq}]$, since each necklace is ordered by the temporal component, using binary search we can find the sub-sequence of the beads that needs to be considered for the purpose of generating the answer to a given query in $O(\log n)$, for a necklace with n beads. We note that if $[t_i, t_{i+1}] \cap [t_{bq}, t_{eq}] = \emptyset$ for any algorithm, then the answer to the corresponding predicate is *false*. In the sequel, without loss of generality we will focus on the processing algorithms for individual beads. For each of the respective algorithms, the input consists of a bead B_i, given as $((x_i, y_i, t_i), (x_{i+1}, y_{i+1}, t_{i+1}), v^i_{max})$, a region R bounded by a convex polygon (∂R) which, together with $[t_{bq}, t_{eq}]$ defines the query-prism QP_R. Also, for a given time-instance t, we use $C_i^{down}(t)$ (resp. $C_i^{up}(t)$) to denote the lower (resp.) upper circles of the *boundaries* of the moving object's whereabouts at t, based exclusively on the (x_i, y_i, t_i) (resp. $(x_{i+1}, y_{i+1}, t_{i+1})$). Their respective interiors (the disks at time t) are denoted with $D(C_i^{down}(t))$ and $D(C_i^{up}(t))$.

[3]Equivalently, one could use a minimum bounding box which is more appropriate for indexing.

Recall that El_i denotes the ellipse which is the (X, Y) projection of the bead B_i, and let F_i^l (= (x_i, y_i)) and F_i^u (= (x_{i+1}, y_{i+1})) denote its *lower* and *upper* (in temporal sense) foci. For complexity analysis, assume that the region R has m edges/vertics, and an one-time pre-processing cost of $O(m)$ has been performed to determine the angles in-between its consecutive vertices with respect to a given point in R's interior [17].

Possibly Sometimes Inside – The algorithm for processing this predicate is as follows:

Algorithm_PSI

1. **If** $(t_i \in [t_{bq}, t_{eq}] \wedge F_i^l \in R) \vee (t_{i+1} \in [t_{bq}, t_{eq}] \wedge F_i^u \in R)$
2. **return true**
3. **else if** $(El_i \cap R \neq \emptyset)$
4. **return true**
7. **return false**

Each of the disjuncts in line 1. can be verified in $O(\log m)$ due to the convexity of R (after the one-time pre-processing cost of $O(m)$) [17]. Similarly, by splitting the ellipse in monotone pieces (e.g., with respect to the major axis), one can check its intersection with R in $O(\log m)$, which is the upper bound on the complexity of Algorithm_PSI .

Possibly Always Inside – Since the *Inside* predicate needs to be satisfied by some $PMC(TR)$ for every time-point, the algorithm for processing this predicate proceeds as follows:

Algorithm_PAI

1. **If** $(C_i^{down}(t_{bq}) \cap C_i^{up}(t_{bq})) \cap R = \emptyset$
2. **return false**
3. **else if** $(C_i^{down}(t_{eq}) \cap C_i^{up}(t_{eq})) \cap R = \emptyset$
4. **return false**
5. **return true**.

Essentially, *Algorithm_PAI* checks whether the intersection of the bead B_i with R at each end-point of the query-interval is empty – in which case the predicate *Possibly Always Inside* cannot hold. On the contrary, if both of those intersections are non-empty then, one can construct a line segment with endpoints at the respective intersections which, due to the convexity of R, will be entirely contained in R. The "certificate" *PMC(Tr)*, can be constructed by applying a linear motion along the segment, and with a constant speed between t_{bq} and t_{eq}.

We note that t_{bq} (resp. t_{eq}) may be inside the time-interval during which B_i consists of a single-cone volume, e.g., $[t_i, t_{sv_i}]$ (cf. Section 2). In such cases, the tests in lines 1. and 3. of Algorithm_PAI reduce to simply testing for (non) intersection of a circle and polygon. Since a lens (intersection of 2 circles) can be split into two monotone arcs (one from each circle), the complexity of *Algorith_PAI* is bounded by $O(\log m)$.

Note that, due to Property 1, *Algorithm_PAI* can be used for checking *Always Possibly Inside*.

Definitely Always Inside – given its definition, the algorithm for processing the *DAI* predicate is straightforward:

Algorithm_DAI

1. **If** $((t_i \in [t_{bq}, t_{eq}]) \wedge (t_{i+1} \in [t_{bq}, t_{eq}]))$

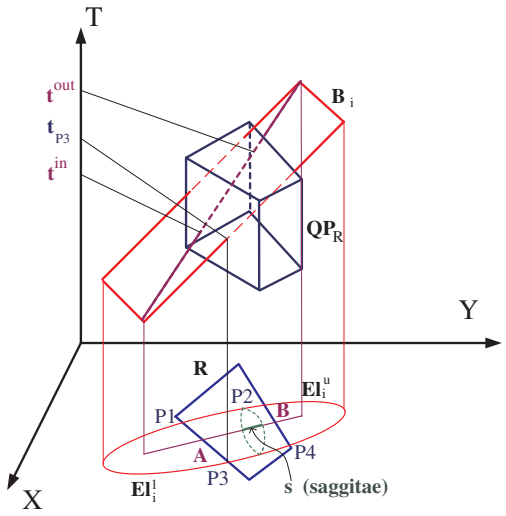


Fig. 7. Critical Points for *DSI* and *SDI* predicates

$\wedge((F_i^l \in R) \wedge (F_i^u \in R)) \wedge (El_i \cap R = \emptyset)$

2. **return true**
3. **else**
4. **return false**

The complexity of the *Algorithm_DAI* is affected by checking the containment of point(s) inside a polygon, and (non) intersection-checking of a convex polygon with an ellipse. Hence, we have the bound of $O(\log m)$.

Definitely Sometimes Inside – recall that, for this predicate we must verify that *every PMC(Tr)* intersects the query-prism QP_R at some (not necessarily unique) time-instance between t_{bq} and t_{eq}. Hence, unless the expected-trajectory $(F_i^l, t_i), (F_i^u, t_{i+1})$ has a non-empty intersection with QP_R, we know that the predicate is *false*. Let t^{in} denote the earliest time that the i-th segment of the expected trajectory enters QP_R, and let t^{out} denote the time that it exits QP_R (cf. Figure 7. Clearly, $[t^{in}, t^{out}] \cap [t_{bq}, t_{eq}] \neq \emptyset$ is a necessary, but by no means a sufficient condition for satisfying the predicate *DSI*. To proceed with the criteria, consider the ellipse El_i. Let El_i^l and El_i^u denote the lower and upper monotonic segments with respect to its major axis, and in direction of the object's motion. Unless both El_i^l and El_i^u have a non-empty intersection with R, then we can construct a $PMC(Tr)$ whose 2D projection is entirely inside $El_i \setminus R$, bypassing R, which will be a "certificate" that the predicate *DSI* is false. In Figure 7, this is illustrated with the points P_1 and P_2 for El_i^u, and P_3 and P_4 for El_i^l. However, this also is not a sufficient criteria. Namely, El_i is only a 2D-projection of the bead B_i, and it may very well be the case that some of the time-instances for the 2D intersection points – denote them by $t(P_k)$, may be outside $[t_{bq}, t_{eq}]$. In terms of the illustration in Figure 7, we need to verify that, e.g., $t(P_3) \in [t_{bq}, t_{eq}]$.

To obtain the time-value of each intersection point $t(P_i)$, we substitute the coordinates of the $El_i \cap R$ in the equations obtained from the last two inequalities from Definition 2. Note that those inequalities specify $D(C_i^{down}(t))$ (resp.

$D(C_i^{up}(t)))$, hence, using "=" instead of "$\leq$" will yield $C_i^{down}(t)$ (resp. $C_i^{up}(t)$). Next, we need to verify that at least one (of the two, since the equations are quadratic in t) solutions is common for both of them and, lastly, we need to verify that the common solution is inside $[t_{bq}, teq]$. We have:

Algorithm_DSI

1. **If** $((t_i \in [t_{bq}, t_{eq}]) \wedge (F_i^l \in R))$
$\vee((t_{i+1} \in [t_{bq}, t_{eq}]) \wedge (F_i^u \in R))$
2. **return true**
3. **else if** $(El_i^u \cap R \neq \emptyset) \wedge (El_i^u \cap R \neq \emptyset)$
$\wedge(\forall k \in \{1,2,3,4\})t(P_k) \in [t_{bq}, t_{eq}]$
4. **return true**
5. **return false**

Similarly to the previous algorithms, we need to find an intersection between a convex polygon and a monotone arc of an ellipse (at most 4 of them), which is $O(\log m)$, except now we have to solve (at most $4 \cdot 2$) quadratic equations and pair-wise compare their solutions, which is $O(1)$. We note that there are some special cases for *Algorithm_DSI* like, for example:
(1) $F_i^l \in R$ and $t_i < t_{bq}$, in which case, $t^{in} = t_{P_1} = t_{bq}$.
(2) $El_i^l \subset R$ (i.e., axis-monotone part of the ellipse is entirely inside R), which is slightly different from the illustration in Figure 7, where the intersection is actually between El_i^l and ∂R. However, such cases can be checked in constant time.

Sometimes Definitely Inside – We now address the processing of the last remaining predicate – *SDI*. Recall (cf. Section III-A) that we need a time-instance $t \in [t_{bq}, t_{eq}]$ for which the entire lens $D(C_i^{down}(t) \cap D(C_i^{up}(t))$ is inside QP_R. Note that as a consequence of Property 2 (via contrapositive), we know that if the predicate *DSI* is *false*, then so is *SDI*. Hence, to process the *SDI* predicate, we need to verify the *DSI*, plus some extra-criteria. We proceed as follows. Let $[t_i^{min}, t_i^{max}] = [t(P_1), t(P_2)] \cap [t(P_3), t(P_4)] \cap [t^{in}, t^{out}]$. Observe that if $[t_i^{min}, t_i^{max}] = \emptyset$ then the predicate *SDI* is false, because there exists no time interval during which the expected location and both ends of the lens can be inside R. Assume $[t_i^{min}, t_i^{max}] \neq \emptyset$.
(1) If $D(C_i^{down}(t_i^{min})) \cap D(C_i^{up}(t_i^{min})) \subset QP_R$, or $D(C_i^{down}(t_i^{max})) \cap D(C_i^{up}(t_i^{max})) \subset QP_R$, then *SDI* predicate is *true*, with a "witness" t_i^{min} (resp. t_i^{max}).
(2) Else, if the portion of $C_i^{down}(t)$ (or $C_i^{up}(t)$) inside El_i is *not entirely contained* in R in both t_i^{min} and t_i^{max}, we know that the predicate *SDI* is *false*.
(3) If neither (1) nor (2) above can be established, let $t_i^{new} = t_i^{min} + ((t_i^{min} + t_i^{max})/2)$, and repeat (1) and (2) for each of the $[t_i^{min}, t_i^{new}]$ and $[t_i^{new}, t_i^{max}]$.
Clearly, the procedure outlined above, denote it *Cascade*$(t_i^{min}, t_i^{max}, B_i, QP_R)$ is recursive in nature, and the first question is whether it terminates. The answer to that is affirmative and the worst case for the number of invocations can be bounded as follows:
Let $A = (x(t^{in}), y(t^{in}))$ and $B = (x(t^{out}), y(t^{out}))$. Also, let $s_{min}(t)$ denote the smallest sum of the two *sagitta*'s at any $t \in [t_i^{min}, t_i^{max}]$, where: – the first sagitta is defined as the perpendicular distance from the midpoint of the $C_i^{down}(t)$ arc bounded by $C_i^{down}(t) \cap C_i^{up}(t)$ to the radical axis of $C_i^{down}(t)$ and $C_i^{up}(t)$; – the second sagitta is defined as the perpendicular distance from the midpoint of the $C_i^{up}(t)$ arc bounded by $C_i^{down}(t) \cap C_i^{up}(t)$ to the radical axis of $C_i^{down}(t)$ and $C_i^{up}(t)$. Then, the procedure *Cascade*$(t_i^{min}, t_i^{max}, B_i, QP_R)$ can execute at most $\lceil \overline{AB}/s_min(t) \rceil$ times. An illustration for the sum of two sagitta's is provided in Figure 7. We now have:

Algorithm_SDI

1. **If** $\neg$ *DSI*
2. **return false**;
3. **else**
4. *Cascade*$(t_i^{min}, t_i^{max}, B_i, QP_R)$

Due to the assumption on convexity of R, we have that the complexity of the *Algorithm_SDI* is bounded by $O(\lceil \overline{AB}/s_{min}(t) \rceil \log m)$.

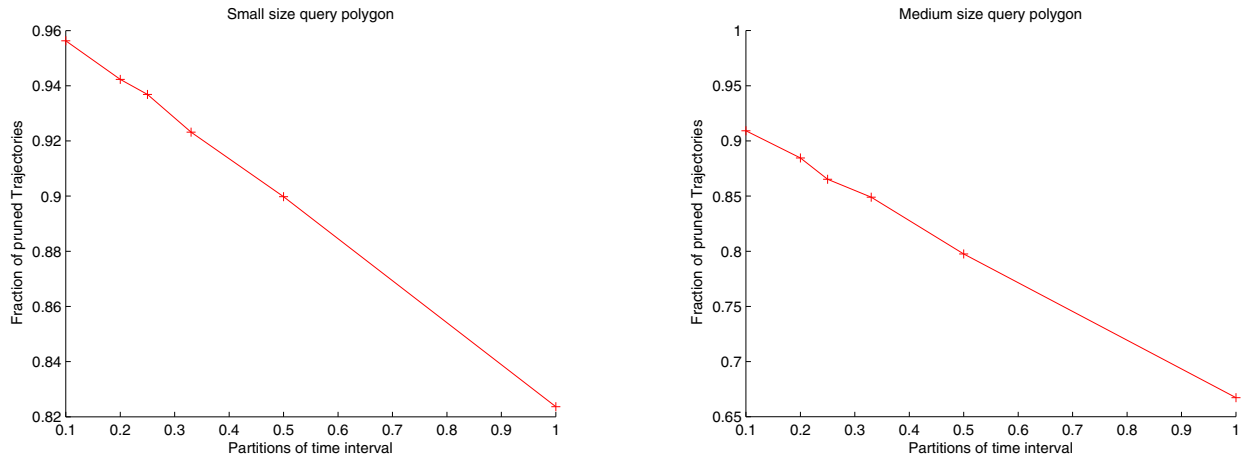
We conclude this section with a broad remark about the complexity results presented. Namely, in each of the algorithms, we focused on an individual bead B_i. However, in the worst case, the time interval of interest for the query, $[t_{bq}, t_{eq}]$ may be spanning the entire time interval of the lifeline necklace. Hence, assuming that a given necklace has n beads, each of the above complexity results needs to be multiplied by n – either because we may have to check each bead to test for satisfiability, or because we need to ensure that the satisfiability of a given predicate is maintained from the previous and into the next bead.

IV. Experimental Evaluations

We performed several experiments, in order to evaluate the benefits of the pruning approaches presented in Section 3. Specifically, for each of the query predicates, we implemented the corresponding refinement algorithm, and compared the total execution times when the particular query was evaluated with pruning (pruning_time + refinment_time) and without pruning (directly applying the refinement algorithms). For implementations of intersections of the ellipses and circles with polygons, we used the publicly available CGAL[4] (Computational Geometry Algorithms Library), which contains robust C++ implementations of data structures and algorithms for a wide variety of geometric computing.

Our experiments were performed on a Pentium Intel Core 2 Duo 2.26GHZ, 3G MB memory, with Windows XP platform. As parameters of the experiments, we considered:
(1) Geographic area of size 40x40 miles^2.
(2) Total of 1800 moving objects that were generated using a modified version of the *random way-point model* such that each object: – starts at a randomly selected position in the region of interest; – selects a random direction; – selects a random speed between 35mph and 80mph (until it stops); – has a randomly selected value of v_{max} along a given segment, which can range between 20% and 90% from the selected speed of motion.
(3) The total length of trajectories varied from 4 to 120 miles,

[4]http://www.cgal.org



(a) (b)

Fig. 8. Pruning effects

normally distributed within that interval, and in 1 mile we have between 4 and 8 segments.
(4) Query regions that were rectangles of different shapes, and placed at randomly selected positions. We varied their size from small (approx. 5% of the size of the geographic area), to medium (approx. 20% of the entire geographic area).
(5) We varied the duration of the time-interval of interest for the range queries, as percentage of the total time between the earliest start-time for all the trajectories and the latest end-time, randomly selecting starting time-points.

In the sequel, we report the averaged values over 100 runs for different combinations of values of the parameters above. Before we proceed with the different observations, we note that:
(1) When applying the **LB** (Large Bound) pruning, we observed that in the best case, over 90% of the trajectories was typically retained for the refinement stage.
(2) When applying **IBb** and **UBb** pruning strategies, we observed a very little difference between the benefits of the two – specifically, in one particular run (medium size query polygon, and time-interval of 30%), **IBb** pruned 895 trajectories, whereas **UBb** pruned 899 (out of 1800 in the dataset). Hence, in the sequel, we report our observations for **IBb** pruning strategy.

Figure 8 illustrates the benefits of the pruning as a function of the duration of the time-interval of interest for a given query, for the cases when small (Figure 8.a) and medium (Figure 8.b) sized query regions were used. As can be seen, approximately 90% of the trajectories can be pruned when processing queries whose region is of a medium size (relative to the whole region of 40x40 miles^2), for queries whose duration is about 35% of the entire time-interval of active trajectories. However, as can be expected, the effects of the pruning are diminished as the duration of the time-interval increases, and similarly for increasing the size of the query region.

Another illustration of this trend is presented in Figure 11,

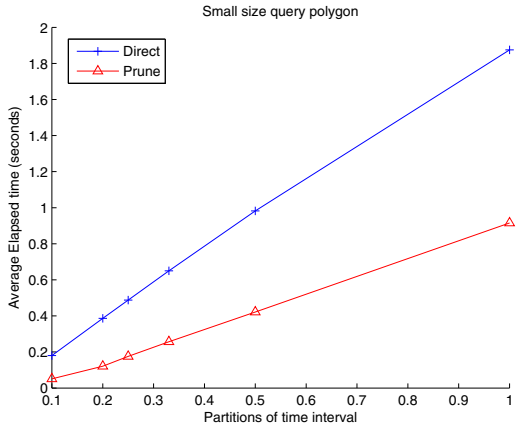


Fig. 9. Pruning – running time for small size query regions

which shows the fraction of trajectories of a given length that are pruned when the query region is small, as opposed to the ones when the query region is medium (the numbers on each bar indicate the length of the respective trajectories).

Next, we compared the benefits of pruning in terms of the overall execution times for the queries. Figure 9 and Figure 10 present the observations for the cases of small and medium size query region, when processing the *Sometimes Definitely Inside* predicate. The abscissa shows the duration of the time-interval of the query, and each graph compares the direct approach (i.e., no pruning) with the approach that uses pruning. When pruning was used, we added the overall time taken by the pruning stage and yet, as can be seen, speed ups of up to a factor of 2 can be achieved.

An interesting observation is that, as much as pruning does yield non-negligible processing speed ups, in some of our experiments we observed that it does not yield a uniform speed-up when processing different predicates. This

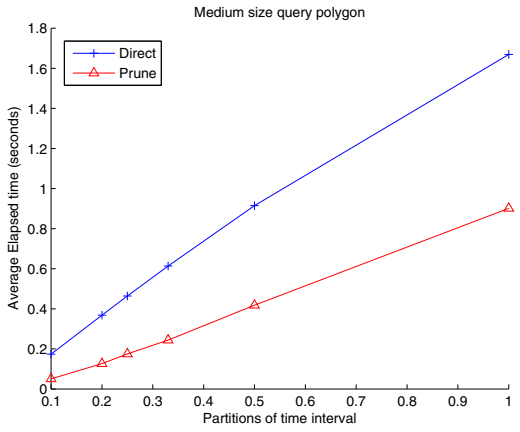


Fig. 10. Pruning – running time for medium size query regions

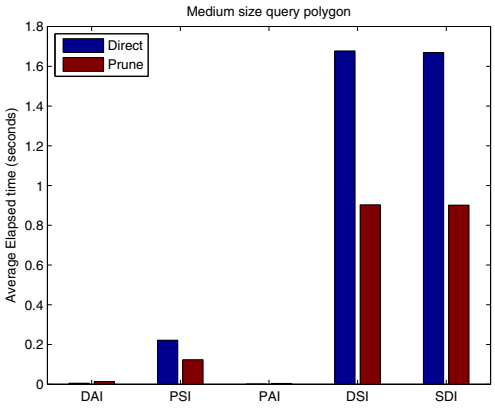


Fig. 12. Pruning – impact on different queries

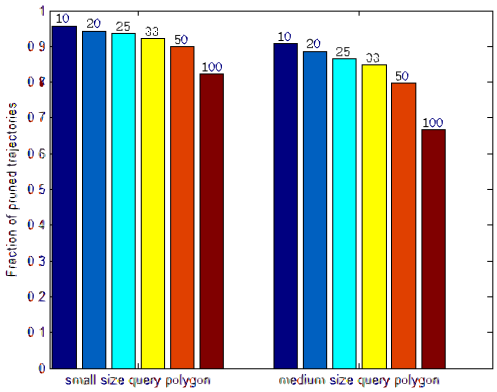


Fig. 11. Pruning – impact on different queries

is illustrated in Figure 12, where the bars indicate the total processing time for individual queries. On the abscissa, we indicate a particular query using the abbreviations from the names of their respective refinement algorithms in Section 3.3. As shown, the queries that use the universal quantifier (*Always*) in the temporal domain had the least benefit – which, in a sense, is to be expected because every single bead of the necklace has to be tested in such queries, thereby diminishing the benefits of the pruning.

V. Related Work

The field of Moving Objects Databases [9] has generated a large body of works and, in particular, quite a few recent results have been focusing on efficient management of uncertainty in spatio-temporal settings.

In [3], an uncertainty model similar to the one used in this work has been analyzed for the purpose of processing range and NN-queries. Various results were presented with quantitative measures of actual probabilistic values. However, in the context of the terminology introduced in this paper, [3] used only the bottom-cone emanating from a particular update point, and did not consider the cone emanating downwards from the next update point in the sequence. Subsequently, [4] have addressed the problem of probabilistic verifiers in dynamic settings, however, the domain of the possible values considered can range along 1 dimension. More recently, similar settings have been used by the time-series community, where the uncertainty of the values can range within a bounded interval, at fixed time-points only [26], and modelling of spatial uncertainty has been addressed in the context of urban studies [6].

The model of uncertainty for moving objects based on the assumption of a maximum velocity being the only parameter known in-between two consecutive (location,time) updates, resulting in beads (and necklaces) was introduced in [11]. In addition, the authors presented several possibilities of presenting a sequence of beads in different levels of granularity, however, the work focused on the modelling aspects, and did not present any formal mathematical analysis of the key properties of the beads. This work was slightly preceded by [19], which introduced the equivalent model of the location-uncertainty of a given moving objects – focusing on the 2D (X, Y) projection which, as mentioned, is an ellipse. A recent work that presents an extensive formal treatments of the beads-based uncertainty model for trajectories is [14], where the data model is accompanied by a complete query language. The features of the uncertainty model when the motion is restricted to a road-network has been presented in [13]. In this work, we built upon these existing results, and we focused on formalizing the processing of *qualitative* spatio-temporal range queries with uncertainty, for different syntactic variants. We considered different pruning strategies, and for each syntactic variant we presented efficient processing algorithms.

The categories of queries considered in this work are equivalent to the ones in [22], however, the uncertainty model that we used is different. Namely, [22] used a fixed-radius disk as a boundary of the possible whereabouts of the moving object at every time-point. This resulted in a volume of possible trajectories corresponding to a sequence of sheared cylinders,

all with equal horizontal projections at individual time-points. In this paper, we considered the model of beads where different segments can have different 3D shapes due to the variations in the maximum speed. The specifics of the model required novel processing algorithms, which is what separates this work from [22] – namely, the concept of the Minkowski sum used in [22], which yielded compact algorithmic approaches for processing range queries, can no longer be applied in beads/necklaces settings. However, the beads model provided us with the opportunity to investigate the impact of the pruning as a factor in queries processing.

VI. Concluding Remarks and Future Work

We addressed the problem of efficient processing of spatio-temporal range queries for uncertain trajectories, where the uncertainty of the motion was modelled as a necklace – a sequence of beads. The main motivation for this model is that in many realistic settings, the positions of the moving objects are sampled only periodically, and little is known about their locations in-between – and this model provides a realistic setting, by assuming only one restricting parameter: the maximum velocity in-between consecutive updates. The model was initially introduced in [11] and, from a complementary perspective [19], however, the thorough treatment of all the different aspects of the model has been presented only recently [14].

Focusing on the qualitative aspects of the uncertainty, we presented pruning strategies which, as demonstrated by our experiments, can significantly speed up the processing of some of the queries. We also demonstrated that some of the properties of other uncertainty models (cf. [22], [25]) hold in the current settings. For each different query predicate that we defined, we presented the corresponding processing algorithms for the refinements stage.

There are several immediate extensions of our work. Currently, we are working on *quantitative* range queries, where actual probabilistic value is assigned for the existing predicates, in a similar spirit to [3]. Concurrently, we are addressing the problems of uncertain spatio-temporal Nearest-Neighbor (NN) queries [12], [23] for beads/necklaces model. The beads appear to be an attractive uncertainty model for trajectories obtained by tracking via periodic sampling in wireless sensor networks. One of the main challenges, however, is to develop distributed and energy-efficient algorithms for processing the spatio-temporal range queries in (near) real time [2], [24].

We note that an important part of the query processing techniques is the indexing [20], and the uncertainty of the spatial whereabouts has been incorporated into indexing schemas for processing probabilistic range queries [21]. However, this is beyond the scope of the current paper, and we leave it for the future work to investigate whether some of the techniques that we proposed here for pruning, can actually benefit some indexing schema(s). Specifically, an important question would be to find an optimal balance of the size of the "pruning-cylinder", in the spirit of [10].

Acknowledgment

We thank Dino Pedreschi for his constructive suggestions.

References

[1] S. Bhattacharya, N. Atay, G. Alankus, C. Lu, O.B. Bayazit, and G.-C. Roman. Roadmap query for sensor network assisted navigation in dynamic environments. In *International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2006.

[2] C. Buragohain, S. Gandhi, J. Hershberger, and S. Suri. Contour approximation in sensor networks. In *DCOSS*, 2006.

[3] R. Cheng, D. Kalashnikov, and S. Prabhakar. Querying imprecise data in moving objects environments. *IEEE-TKDE*, 16(9), 2003.

[4] Reynold Cheng, Jinchuan Chen, Mohamed F. Mokbel, and Chi-Yin Chow. Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data. In *ICDE*, 2008.

[5] A. Civilis, C.S. Jensen, and S. Pakalnis. Techniques for efficient road-network-based tracking of moving objects. *IEEE Transactions on Knowledge and Data Engineering*, (5), 2005.

[6] S. Dodge, R. Weibel, and E. Forootan. Revealing the physics of movement: Comparing the similarity of movement characteristics of different types of moving objects. *Computers, Environments and Urban Systems*, 2009. doi:10.1016/j.compenvurbsys.2009.07.008.

[7] Sorabh Gandhi, Rajesh Kumar, and Subhash Suri. Target counting under minimal sensing: Complexity and approximations. In *ALGOSENSORS*, pages 30–42, 2008.

[8] B. Gedik and L. Liu. Mobieyes: A distributed location monitoring service using moving location queries. *IEEE Transactions on Mobile Computing*, 5(10), 2006.

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

[10] Marios Hadjieleftheriou, George Kollios, Vassilis J. Tsotras, and Dimitrios Gunopulos. Efficient indexing of spatiotemporal objects. In *EDBT*, 2002.

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

[12] Y.-Yk. Huang, C.-C. Chen, and C. Lee. Continuous k -nearest neighbor query for moving objects with uncertain velocity. *GeoInformatica*. to appear (online available DOI).

[13] B. Kujipers and W. Othman. Modelling uncertainty on road networks via space-time prisms. *Int.l Journal on GIS*, 23(9), 2009.

[14] B. Kujipers and W. Othman. Trajectory databases: data models, uncertainty and complete query languages. *Journal of Computer and System Sciences*, 2009. doi:10.1016/j.jcss.2009.10.002.

[15] K. Lam, E. Chan, T. Kuo, S. Ng, and D. Hung. Retina: A real time traffic navigation system. In *SIGMOD*, 2001.

[16] Mohamed F. Mokbel, Chi-Yin Chow, and Walid G. Aref. The new casper: Query processing for location services without compromising privacy. In *VLDB*, 2006.

[17] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, 2000.

[18] M. Scholl P. Rigaux and A. Voisard. *Location-based Services*. Morgan Kaufmann Publishers, 2001.

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

[20] Y. Tao, D. Papadias, and J.Sun. The tpr∗-tree: An optimized spatio-temporal access method for predictive queries. In *VLDB*, 2003.

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

[22] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. Managing uncertainty in moving objects databases. *ACM TODS*, 29(3), 2004.

[23] Goce Trajcevski, Roberto Tamassia, Hui Ding, Peter Scheuermann, and Isabel F. Cruz. Continuous probabilistic nearest-neighbor queries for uncertain trajectories. In *EDBT*, 2009.

[24] H. Wang, K. Yao, and D. Estrin. Information-theoretic approaches for sensor selection and placement for target localization and tracking in sensor networks. *journal of Communications and Networks*, 7(4), 2005.

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

[26] Mi-Yen Yeh, Kun-Lung Wu, Philip S. Yu, and Ming-Syan Chen. Proud: a probabilistic approach to processing similarity queries over uncertain data streams. In *EDBT*, 2009.