Indexing Uncertain Spatio-Temporal Data

Tobias Emrich[1], Hans-Peter Kriegel[1], Nikos Mamoulis[2],
Matthias Renz[1], Andreas Züfle[1]
[1]Institute for Informatics, Ludwig-Maximilians-Universität-München
[2]Department of Computer Science, University of Hong Kong
{emrich, kriegel, renz, zuefle}@dbs.ifi.lmu.de, nikos@cs.hku.hk

ABSTRACT

The advances in sensing and telecommunication technologies allow the collection and management of vast amounts of spatio-temporal data combining location and time information. Due to physical and resource limitations of data collection devices (e.g., RFID readers, GPS receivers and other sensors) data are typically collected only at discrete points of time. In-between these discrete time instances, the positions of tracked moving objects are uncertain. In this work, we propose novel approximation techniques in order to probabilistically bound the uncertain movement of objects; these techniques allow for efficient and effective filtering during query evaluation using an hierarchical index structure. To the best of our knowledge, this is the first approach that supports query evaluation on very large uncertain spatio-temporal databases, adhering to possible worlds semantics. We experimentally show that it accelerates the existing, scan-based approach by orders of magnitude.

Categories and Subject Descriptors

H.3.3 [**INFORMATION STORAGE AND RETRIEVAL**]: Information Search and Retrieval

Keywords

Uncertain Spatio-Temporal Data, Uncertain Trajectory, Indexing

1. INTRODUCTION

Efficient management of large collections of spatio-temporal data pertaining to mobile entities whose locations change over time is paramount in a large variety of application domains: military applications, structural and environmental monitoring, disaster/rescue management and remediation, Geographic Information Systems (GIS), Location-Based Services (LBS). The technological enabling factors for such applications are advances in sensing and communication/networking, along with the miniaturizations of the computing devices and development of embedded systems. In almost every application domain, the location data at different (discrete) time-instants is obtained via some positioning devices, like GPS-enabled mobile devices, RFID or road-side sensors. In addition, to reduce the communication cost, improve the bandwidth utilization, and cope with storage constraints, often the recorded object trajectories undergo simplification, eliminating some recorded values. Having object trajectories sampled only at discrete time-instants and/or simplified, renders the movement in-between samples *uncertain* and query evaluation challenging.

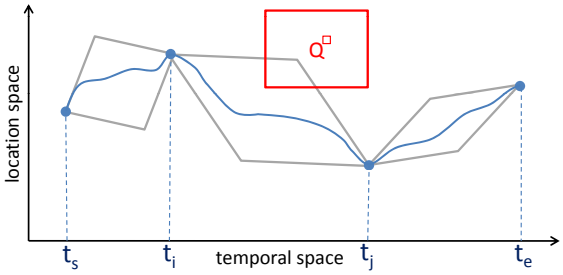


Figure 1: Spatio-Temporal Data.

Consider an object o, moving in a one-dimensional space, as illustrated in Figure 1. Having complete information about this trajectory enables answering a query asking whether the object intersects a spatio-temporal window Q. However, this task becomes difficult if only a few positions (at times $\{t_s, t_i, t_j, t_e\}$ in the example) of the exact trajectory are recorded. A simple interpolation approach, which connects temporally consecutive observations by line segments and assumes a movement with constant direction and speed between these points, is unacceptable for applications where probabilistic analysis of the uncertain movement is required. When taking uncertainty under consideration, the main challenge is that the space of possible $(location, time)$ positions between two observations can grow very large. More importantly, the number of possible trajectories between two observed locations explodes.

A common method to approximate possible locations between two observations is the beads (or necklace) model ([11, 22]). This model is based on some constraints about the motion of an object. In particular, assuming a maximum speed in each direction of each dimension, the possible locations that an object can visit between two exact observations is bounded. Recent work [8] follows the pragmatic assumption that the uncertain movement of an object between consecutive observations can be described by a Markov-Chain model, which captures the time dependencies between consecutive locations. [8] shows how the space of possible worlds (i.e., trajectories between consecutive observations) can be efficiently analyzed by multiplying Markov-Chain transition matrices and that probabilistic query evaluation can be facilitated by integrating pruning mechanisms into the Markov Chain matrices. All these are sufficient for the case where there are few queried objects, following similar movements; however, if there is a large number of objects in the database, with different movements, evaluating a probabilistic spatio-temporal query directly against each object individually (i.e., a scan-based approach) would be very expensive.

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

In this paper, we propose an indexing framework to efficiently cope with large spatio-temporal data bases. Our work also assumes that the movement of each object follows a Markov-Chain model (described in Section 2). The objective of our index (described in Section 4) is to minimize the number of objects for which exact probabilistic evaluation has to be performed. To achieve this goal, in Section 3, we propose a number of *uncertain spatio-temporal (UST) object approximations*, which are stored in the index and a set of corresponding pruning methods, which use the approximations to efficiently eliminate objects that may not qualify a given probabilistic query. Section 5 presents an extensive experimental evaluation, which demonstrates the effectiveness of our indexing approach. Related work is discussed in Section 6. Finally, Section 7 concludes the paper.

2. PRELIMINARIES

This section formally defines the type of spatio-temporal data that we index, the stochastic model that we use for uncertain trajectories derived from the data, and the query types that we handle.

Data: We consider a discrete time and space domain, i.e., the common assumption of many existing works (e.g. [18, 1, 10, 8]), where $\mathcal{S} = \{s_1, ...s_{|\mathcal{S}|}\} \subseteq \mathbb{R}^D$ is a finite set of possible locations, which we call *states*, in a D-dimensional space and $\mathcal{T} = \mathbb{N}_0^+$ is the time domain. Given this spatio-temporal domain, the (certain) movement of an object o corresponds to a *trajectory* represented as function $o : \mathcal{T} \rightarrow \mathcal{S}$ of time defining the location $o(t) \in \mathcal{S}$ of o at a certain point of time $t \in \mathcal{T}$. We consider incomplete (and/or imprecise) spatio-temporal data, where the motion of an object is not recorded by a crisp trajectory. Instead, we are only given a set $o.T_{obs}$ of $(location, time)$ observations for each object o. At any time $t \notin o.T_{obs}$, the position of o is uncertain, i.e., a random variable. In many applications, a stochastic model can be built and used to infer knowledge about this uncertainty.

Uncertain Data Model: We refer to the *spatio-temporal approximation* of a trajectory of an object o in a time interval spanned by two consecutive observations of o at t_i and t_j as a *bead* or *diamond* $\diamond(o, t_i, t_j)$. The diamond can be computed by considering the maximum and minimum (singed) velocities of the object in each dimension [25]. The whole approximation of the trajectory based on a set T_{obs} of observations (i.e., a chain of beads) is referred to as a *necklace*. For example, the movement of the object in Figure 1 is bounded by a chain of diamonds.

Existing studies on modeling uncertain trajectories ([23, 25, 24, 26, 15]) naively consider all possible trajectories bounded by a necklace equi-probable. However, given two consecutive observations $o(t_i)$ and $o(t_j)$ of object o, there are time dependencies between consecutive locations between $o(t_i)$ and $o(t_j)$, which render some locations in the corresponding diamond (e.g., those near the line segment that connects $o(t_i)$ and $o(t_j)$) more probable to be visited by o than others (e.g., those near the boundary of the diamond). Therefore, these models possibly yield incorrect inferences resulting in incorrect answers to probabilistic queries. To overcome this problem, in our proposal each uncertain spatio-temporal (UST) object $o \in \mathcal{D}$ is associated with an *uncertain object trajectory*, which is represented by a stochastic process. The stochastic process assigns to each $t \in T$ exactly one random variable; random variables at consecutive time moments can be mutually dependent. This dependency is vital in most applications, where the locations of an object at two close points of time are highly correlated. Thus, the uncertain trajectory $o(t)$ of object o comprises a set of (crisp) trajectories, each assigned with a probability indicating its likelihood to be the true trajectory of o. Thereby, each trajectory with a non-zero probability is called a *possible world* of o. Assuming that objects are mutually independent, our semantics comply with the classic possible worlds model [6]. If $t \in o.T_{obs}$ (i.e., the exact location of o at time t has been observed), then $o(t)$ corresponds to a (trivial) random variable having one possible location (i.e., state) with probability 1.

The main challenge in answering a probabilistic spatio-temporal query is to correctly consider the possible worlds semantics in the model. In other words, the query results should comply with the possible worlds model. Naively, this can be done by evaluating the query predicate on each possible world, and summing up the probabilities of possible worlds satisfying the query predicate. In general, the number of possible worlds to be considered is $O(|S|^T)$; exhaustively examining all of them requires exponential time, even for finite time and space domains. Clearly, any naive approach that enumerates all possible worlds, is not feasible.

In this work, we model the uncertain movement of an object within a diamond, using a first-order Markov-chain model. This approach models the movement between successive points in time, based on background knowledge (e.g., physical laws) and has proven capable of effectively capturing the behavior of real objects in practice. For instance, [1] and [10] show how Markov-models can effectively capture the movement of vehicles on road networks for prediction purpose. In [18], it is shown that Markov-models can also be used to model the indoor movement of people, as tracked in RFID applications.

DEFINITION 1. *A stochastic process $o(t), t \in \mathcal{T}$, is called a Markov-Chain if and only if*

$$\forall t \in \mathbb{N}_0 \forall s_j, s_i, s_{t-1}, ...s_0 \in S :$$

$$P(o(t+1) = s_j | o(t) = s_i, o(t-1) = s_{t-1}, ..., o(0) = s_0) =$$

$$P(o(t+1) = s_j | o(t) = s_i),$$

where the conditional probability

$$o.P_{i,j}(t) := P(o(t+1) = s_j | o(t) = s_i)$$

is the (single-step) *transition probability* of object o from state s_i to state s_j at time t. The matrix $o.M(t) := (o.P_{i,j})_{i,j}$ is called *transition matrix*. Let $o.P(t) = (p_1, \ldots, p_{|\mathcal{S}|})$ be the distribution vector of an object o at time t, where p_i corresponds to the probability that o is located at state s_i at time t. The distribution vector $o.P(t+1)$ can be inferred from $o.P(t)$ as follows:

$$o.P(t+1) = o.P(t) \cdot o.M(t)$$

Queries: Within the scope of this paper, we focus on selection queries specified by the following parameters: (i) a spatial window $S^\square \subseteq \mathcal{S}$, (ii) a contiguous time window $T^\square \subseteq \mathcal{T}$, and (iii) a probability threshold τ. In the remainder, we use $\mathcal{Q}^\square = \mathcal{S}^\square \times \mathcal{T}^\square$ to denote the search space of a query. The most intuitive definition of a probabilistic spatio-temporal query is given below:

DEFINITION 2. *[Probabilistic Spatio-Temporal τ Exists Query] Given a query window $\mathcal{S}^\square$ in space and a query window $\mathcal{T}^\square$ in time, a* probabilistic τ spatio-temporal exists query *(PST$\tau\exists$Q), retrieves all objects $o \in \mathcal{D}$ such that $P(\exists t \in \mathcal{T}^\square : o(t) \in \mathcal{S}^\square) \geq \tau$; i.e., the trajectory of o intersects the query window $\mathcal{Q}^\square$ with probability at least τ.*

For example, consider the trajectory of Figure 1 and assume that we only know for certain its observed locations at $\{t_s, t_i, t_j, t_e\}$; a PST$\tau\exists$Q query defined by rectangle $\mathcal{Q}^\square$ would return the depicted trajectory, only if the probability that the trajectory intersects $\mathcal{S}^\square$ at any time within $\mathcal{T}^\square$ exceeds τ. Another query type is

the Probabilistic Spatio-Temporal τ ForAll Query ([8]), denoted as (PST$\tau\forall$Q), which requires an object to remain in the spatial window $\mathcal{S}^{\square}$ for the whole time window $\mathcal{T}^{\square}$. Due to space constraints, we will not discuss this query in this work, but note that our techniques proposed for PST$\tau\exists$Q queries can easily be adapted.

By modeling the movement within a diamond using a Markov-chain model, the true probability that an object o satisfies a PST$\tau\exists$ query, can be computed in PTIME [8], exploiting that the matrix M is generally sparse (only a few states are directly connected to a single state). Still, query evaluation remains too expensive over a large spatio-temporal database, where we have to compute the qualification probabilities of all objects. In view of this, we define a set of approximations of uncertain object trajectories enabling spatio-temporal and probabilistic filtering in Section 3. We then show in Secion 4 how we can organize these approximations in an index in order to perform efficient query evaluation.

3. APPROXIMATING UST-OBJECTS

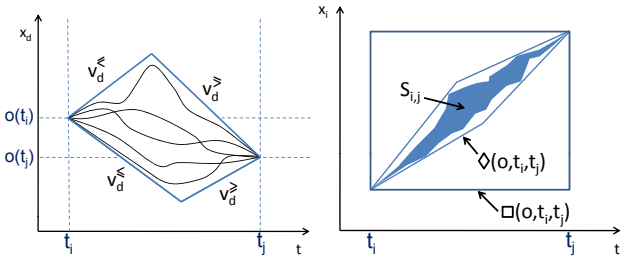
In this section, we introduce (conservative) spatio-temporal as well as probabilistic (conservative) spatio-temporal (UST-) object approximations, which serve as building blocks for our proposed index, to be presented in Section 4.

3.1 Spatio-Temporal Approximation

To bound the possible locations of an object o between two subsequent observations $(o(t_i), o(t_j))$, we need to determine all state-time pairs $(s,t) \in S \times T, t_i \leq t \leq t_j$ such that o has a non-zero probability of being at state s at time t. This is done by considering all possible paths between state $o(t_i)$ at time t_i and state $o(t_j)$ at time t_j. An example of a small set of such paths is depicted in Figure 2(a). Here, we can see a set of five possible trajectories of an object o, i.e., all possible $(state, time)$ pairs of o in the time interval $[t_i, t_j]$. In practice, the number of possible paths becomes very large. Nonetheless, we can efficiently compute the set of possible $(state, time)$ pairs using the Markov-chain model: The set of state-time pairs S_i reachable from $o(t_i)$ can be computed by performing $t_j - t_i$ transitions using the Markov chain $o.M(t)$ of o, starting from state $o(t_i)$ and memorizing all reachable $(state, time)$ pairs. Similarly, we can compute S_j as all state-time pairs $(s,t) \in \mathcal{S} \times \mathcal{T}, t_i \leq t \leq t_j$ such that o can reach state $o(t_j)$ at time t_j by starting from state s at time t. S_j can be computed in a similar fashion, starting from state $o(t_j)$ and using the transposed Markov chain $o.M(t)^T$. The intersection $S_{i,j} = S_i \cap S_j$ yields all state-time pairs which are consistent with both observations. Let us note that in practice, it is more efficient to compute S_i and S_j in a parallel fashion, to reduce the explored space. When the computation of S_i and S_j meet at some time $t_i \leq t \leq t_j$, we can prune any states which are not reachable by both $s(t_i)$ at time t_i and $s(t_j)$ at time t_j. However, the number $|S_{i,j}|$ of possible state-time pairs in $S_{i,j}$ can grow very large, so it is impractical for our index structure (proposed in Section 4) to store all $S_{i,j}$ for each $o \in DB$ in our index structure. Thus, we propose to conservatively approximate $S_{i,j}$. The issue is to determine an appropriate approximation of $S_{i,j}$ which tightly covers $S_{i,j}$, while keeping the representation as simple as possible. The basic idea is to build the approximation by means of both object observations $o(t_i)$ and $o(t_j)$ with the corresponding velocity of propagation in each dimension. To do so, we first compute for the set of state-time pairs S_i to derive the maximum and minimum possible velocity in the time interval $[t_i, t_j]$:

$$v_d^{\leqslant} := max_{(s,t)\in S_i}\left(\frac{s[d] - o(t_i)[d]}{t - t_i}\right)$$

$$v_d^{\leqslant} := min_{(s,t)\in S_i}\left(\frac{s[d] - o(t_i)[d]}{t - t_i}\right)$$



(a) Approximating trajectories (b) Diamond vs MBR

Figure 2: Spatio-Temporal Approximation.

where $s[d]$ ($o(t_i)[d]$) denotes the projection of state s ($o(t_i)$) to the d-th dimension. By definition, we can guarantee that for any $t_i \leq t \leq t_j$ it holds that

$$o(t)[d] \leq o(t_i)[d] + (t - t_i) \cdot v_d^{\leqslant} \text{ and}$$

$$o(t)[d] \geq o(t_i)[d] + (t - t_i) \cdot v_d^{\leqslant}$$

Furthermore, we bound the velocity of propagation at which o can have reached state $o(s_j)$ at time t_j from each location in the state-space S_j:

$$v_d^{\geqslant} := max_{(s,t)\in S_j}\left(\frac{o(t_j)[d] - s[d]}{t_j - t}\right)$$

$$v_d^{\geqslant} := min_{(s,t)\in S_j}\left(\frac{o(t_j)[d] - s[d]}{t_j - t}\right)$$

Again, we can bound the position of o in dimension $1 \leq d \leq D$ at time $t_i \leq t \leq t_j$ as follows:

$$o(t)[d] \leq o(t_j)[d] - (t_j - t) \cdot v_d^{\geqslant}, \text{ and}$$

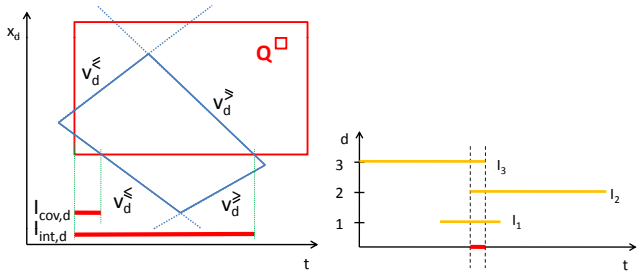
$$o(t)[d] \geq o(t_j)[d] - (t_j - t) \cdot v_d^{\geqslant}$$

In summary, using the positions $o(t_i)$ at time t_i and $o(t_j)$ at time t_j, and using velocities $v_d^{\leqslant}, v_d^{\leqslant}, v_d^{\geqslant}, v_d^{\geqslant}$, we can bound the random variable of the position $o(t)$ of o at time $t_i \leq t \leq t_j$ by the interval

$$o(t)[d] \in I_d(t) := [max(o(t_i)[d] + (t - t_i) \cdot v_d^{\leqslant}, o(t_j)[d] - (t_j - t) \cdot v_d^{\geqslant})),$$

$$min(o(t_i)[d] + (t - t_i) \cdot v_d^{\leqslant}, o(t_j)[d] - (t_j - t) \cdot v_d^{\geqslant})] \quad (1)$$

Deriving these intervals for each dimension, yields an axis-parallel rectangle, approximating all possible positions of o at time t. In the following, we will call this time dependent spatial approximation of $o(t)$ in the time interval $[t_i, t_j]$ between two observations $o(t_i)$ and $o(t_j)$ a spatio-temporal *diamond*, denoted as $\diamond(o, t_i, t_j)$. A nice geometric property of this approximation is that computing the intersection with the query window at each time t is very fast. Another advantage is that existing spatial access methods (e.g., R-trees) can be easily used to efficiently organize these approximations. To store the approximation, we only need to store the $4 \cdot D$ real values $v_d^{\leqslant}, v_d^{\leqslant}, v_d^{\geqslant}, v_d^{\geqslant}, 1 \leq d \leq D$. A diamond is reminiscent to a time-parameterized rectangle, used to model the worst-case MBR for a set of moving objects in [19]; however, the way of deriving velocities is different in our case. As an example, Figure 2(a) shows for one dimension $d \in D$, positions $o(t_i)$ at time t_i and $o(t_j)$ at time t_j. The diamond formed by the velocity bounds $v_d^{\leqslant}, v_d^{\leqslant}, v_d^{\geqslant}$ and $v_d^{\geqslant}$ conservatively approximates the possible $(location, time)$ pairs. Note that it is possible to use a minimal bounding rectangle $\square(o, t_i, t_j)$ instead of the diamond $\diamond(o, t_i, t_j)$ to conservatively approximate the $(location, time)$ space $S_{i,j}$. In cases, however, where the movement of an object in one dimension is biased



(a) query intersection (b) dimension-wise intersections

Figure 3: Intersection between query and diamond

in one direction, a rectangle may yield a very bad approximation (see Figure 2(b) for an example). Our index employs both approximations $\square(o, t_i, t_j)$ and $\diamond(o, t_i, t_j)$ for spatio-temporal pruning; $\square(o, t_i, t_j)$ is used for high-level indexing and filtering, while $\diamond(o, t_i, t_j)$ is used as a second-level filter.

3.2 Spatio-Temporal Filter

Based on the spatio-temporal approximation of an uncertain object as described in the previous section, it is possible to perform filtering during query processing.

If none of the diamonds assigned to an object $o \in \mathcal{D}$ intersects the query window, then o is safely pruned. In turn, if a diamond of o is inside the query window $\mathcal{S}^\square$ in space, i.e. fully covered by $\mathcal{S}^\square$, at any point of time $t \in \mathcal{T}^\square$, then o is a *true hit* and, thus, o can be immediately reported as result of the query. In order to employ the above spatio-temporal pruning conditions, for a diamond $\diamond(o, t_i, t_j)$ of an object o we need to determine the points of time when it intersect the query window $\mathcal{S}^\square$ in space, as well as the points of time when $\diamond(o, t_i, t_j)$ is fully covered by $\mathcal{S}^\square$. For this purpose, it is helpful to focus on the spatial domain $\mathcal{S}$ and interpret a diamond as well as the query as a time-parameterized (moving) rectangle. By doing so, we can adapt the techniques proposed in [19]: In general, a rectangle R_1 intersects (covers) another rectangle R_2, if and only if R_1 intersects (covers) R_2 in each dimension. Thus, for each spatial dimension d ($d \in \{1, \ldots D\}$), we compute the points of time when the extents of the rectangles intersect in that dimension and the points of time when the extents of the diamond rectangle are fully covered by the query rectangle $\mathcal{S}^\square$.

For a single dimension d, with Equation 1 the query window given by $\mathcal{Q}_d^\square$ intersects the diamond given by $o(t_i)[d]$, $o(t_j)[d]$, $v_d^\leqslant$, $v_d^\leqslant$, $v_d^\geqslant$ and $v_d^\geqslant$ within the points of time

$$I_{int,d} := \{t \in (\mathcal{T}^\square \cap [t_i, t_j] | I_d(t) \cap \mathcal{S}_d^\square\}.$$

Similarly, $\mathcal{Q}_d^\square$ fully covers the diamond within the points of time

$$I_{cov,d} := \{t \in \mathcal{T}^\square \cap [t_i, t_j] | I_d(t) \subseteq \mathcal{S}_d^\square\}.$$

An example is illustrated in Figure 3(a). To compute both sets $I_{int,d}$ and $I_{cov,d}$, we intersect the margins of the diamond with the query window resulting in a set of time intervals, which subsequently have to be intersected accordingly in order to derive $I_{int,d}$ and $I_{cov,d}$. Now, let us consider the overall intersection time interval $I_{int} = \bigcap_{d=1}^{D} I_{int,d}$ (e.g., see Figure 3(b)) and the overall points of covering time $I_{cov} = \bigcap_{d=1}^{D} I_{cov,d}$.

If, for an object $o \in \mathcal{D}$, there is no diamond yielding a non-empty set I_{int}, o can be safely pruned. If any diamond of o yields a non-empty set I_{cov}, o can be reported as result.

In summary, the spatio-temporal filter can be used to identify uncertain object trajectories having a probability of 100% or 0% intersecting (remaining in) the query region $\mathcal{Q}^\square$. Still, the probability threshold τ of the query is not considered by this filter. In addition, the object approximation may cover a lot of dead space if there exist outlier state-time pairs which determine one or more of the velocities, despite having a very low probability. In the following, we show how to exclude such unlikely outliers in order to shrink the approximation, while maintaining probabilistic guarantees that employ the probability threshold τ.

3.3 Probabilistic UST-Object Approximation

We now propose a tighter approximation, based on the intuition that the set of possible paths within a diamond is generally not uniformly distributed: paths that are close to the direct connection between the observed locations often are more likely than extreme paths along the edges of the diamond. Therefore, given a query with threshold τ, we can take advantage of a tighter approximation, which bounds all paths with cumulative probabilities τ to perform more effective pruning.

Based on this idea, we exploit the Markov-chain model in order to compute new diamonds, which are spatio-temporal subregions, called subdiamonds, of the (full) diamond $\diamond(o, t_i, t_j)$, as depicted in Figure 4(a). For each such subdiamond, we will then show how to compute the cumulative probability of all possible trajectories of o passing only through this subdiamond. Let us focus on restricting the diamond at one direction of one dimension; we choose one dimension $d \in D$, and one direction $dir \in \{\wedge, \vee\}$. Direction $\wedge$ ($\vee$) corresponds to the two diamond sides $v_d^\leqslant$ and $v_d^\geqslant$ ($v_d^\leqslant$ and $v_d^\geqslant$). To obtain the subdiamond, we scale the corresponding sides by a factor $\lambda \in [0, 1]$ relative to the average velocity $v_d^{avg} = \frac{o(t_j)[d] - o(t_i)[d]}{t_j - t_i}$. We obtain the adjusted velocity values for direction $\wedge$ as follows:

$$v^{\leqslant\lambda}_d = ((v_d^\leqslant - v_d^{avg}) \cdot \lambda) + v_d^{avg}$$

$$= v_d^\leqslant \cdot \lambda + v_d^{avg} \cdot (1 - \lambda)$$

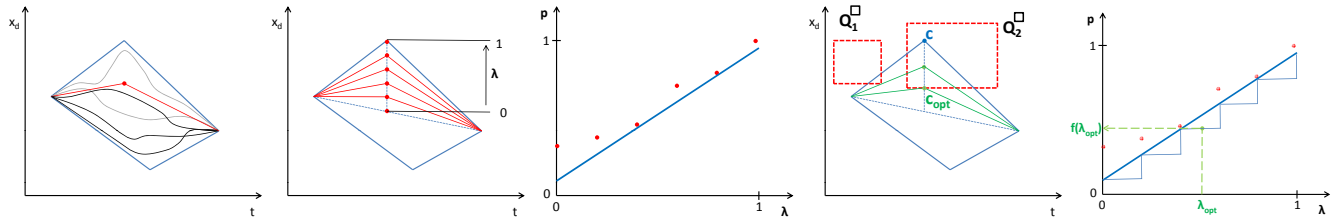
and

$$v^{\geqslant\lambda}_d = ((v_d^\geqslant - v_d^{avg}) \cdot \lambda) + v_d^{avg}$$

$$= v_d^\geqslant \cdot \lambda + v_d^{avg} \cdot (1 - \lambda)$$

The adjusted velocity values for direction $\vee$ can be computed analogously. Thus, for a given diamond $\diamond(o, t_i, t_j)$, dimension $d \in D$, direction $dir \in \{\wedge, \vee\}$ and scalar $\lambda \in [0, 1]$, we obtain a smaller diamond $\diamond(o, t_i, t_j, d, dir, \lambda)$, derived from $\diamond(o, t_i, t_j)$ by scaling direction dir in dimension d by a factor of λ. Figure 4(b) illustrates some subdiamonds for one dimension, the $\wedge$ direction and for various values of λ.

To use such subdiamonds for probabilistic pruning, we first need to compute the probability $P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda)))$ that object o will remain within $\diamond(o, t_i, t_j, d, dir, \lambda)$ for the whole time interval $[t_i, t_j]$, in a correct and efficient way. The main challenge for correctness, is to cope with temporal dependencies, i.e. the fact that the random variables $o(t_i)$ and $o(t_i + \delta t)$ are highly correlated. Thus, we cannot simply treat all random variables $o(t)$ as mutually independent and aggregate their individual distributions. To illustrate this issue, consider Figure 4(a), where one subdiamond is depicted. Assume that each of the five possible trajectories has a probability of 0.2. We can see that three trajectories are completely contained in the subdiamond, so that the probability $P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda)))$ that o fully remains in the subdiamond $\diamond(o, t_i, t_j, d, dir, \lambda)$ is 60%. However, multiplying for all time instants $t \in [t_i, t_j]$ the individual probabilities that o is located in $\diamond(o, t_i, t_j, d, dir, \lambda)$ at time t produces an arbitrarily small and incorrect result, as time dependencies are ignored. Further-



(a) A sub-diamond (b) Possible sub-diamonds (c) linear optimization (d) Computing λ_{opt} (e) conservative approx.

Figure 4: Probabilistic Diamonds

more, due to the generally exponential number of possible trajectories, $P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda)))$ is too expensive to compute by iterating over all possible trajectories. Instead, we compute this probability efficiently and correctly, as follows.

To compute the probability of possible trajectories between $o(t_i)$ at t_i and $o(t_j)$ at t_j that are completely contained in $\diamond(o, t_i, t_j, d, dir, \lambda)$, an intuitive approach is to start at $o(t_i)$ at time t_i and perform $t_j - t_i$ transitions using the Markov-chain $o.M(t)$. After each transition, we identify states that are outside $\diamond(o, t_i, t_j, d, dir, \lambda)$. Any possible trajectory which reaches such a state is flagged. Upon reaching t_j, we only need to consider possible trajectories in state $o(t_j)$, since all other worlds have become impossible due to the observation of o at t_j. The fraction of un-flagged worlds at state $o(t_j)$ at time t_j yields the probability that o does not completely remain in $\diamond(o, t_i, t_j, d, dir, \lambda)$.

To formalize the above approach, we first rewrite the probability $P(inside(o, \diamond)|o(t_i), o(t_j)))$ that the trajectory of o remains in the subdiamond $\diamond$,[1] given the observations $o(t_i)$, $o(t_j)$ at times $t_i, t_j \in o.T_{obs}$, using the definition of conditional probability:

$$P(inside(o, \diamond)|o(t_i), o(t_j)) = \frac{P(o(t_j)|inside(o, \diamond), o(t_i))}{P(o_{t_j}|o_{t_i})},$$

where $P(o(t_j)|inside(o, \diamond), o(t_i))$ denotes the probability that o reaches the state $o(t_j)$ observed by observation $o(t_j)$, given that o, starting at $o(t_i)$ at time t_i remains inside $\diamond$. $P(o(t_j)|o(t_i))$ denotes the probability that state $o(t_j)$ at time t_j is reached, given that o starts at $o(t_i)$ at time t_i, regardless whether o remains in $\diamond$.

3.4 Finding the optimal Probabilistic Diamond

In the previous section, we described, how to compute the probability of a probabilistic diamond $\diamond(o, t_i, t_j, d, dir, \lambda)$ from a diamond $\diamond(o, t_i, t_j)$, dimension d, direction dir, and scaling factor λ. In this section we will show how to find, for a given query window $\mathcal{Q}^{\square}$ and a given query predicate the subdiamond with the highest pruning power. Let us focus on PST$\tau\exists$ queries first. That is, our aim is to find a value for d, dir and λ, such that the resulting subdiamond $\diamond(o, t_i, t_j, d, dir, \lambda)$ does not intersect $\mathcal{Q}^{\square}$, and at the same time it has a high probability $P(inside(o, \diamond))$. This probability can be used to prune o as we will show later. Formally, we want to efficiently determine

$$argmax_{d \in D, dir \in \{\vee, \wedge\}, \lambda \in [0,1]} [P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda))]$$

constrained to $\mathcal{Q}^{\square} \cap \diamond(o, t_i, t_j, d, dir, \lambda) = \emptyset$.

For a single dimension d, and the *north* direction, a possible situation is depicted in Figure 4(d). Here, the projection $\diamond_d(o, t_i, t_j)$ of the full diamond $\diamond(o, t_i, t_j)$ to the d-th dimension and the projections $\mathcal{Q}_1^{\square}[d]$ and $\mathcal{Q}_2^{\square}[d]$ of two query windows $\mathcal{Q}_1^{\square}$ and $\mathcal{Q}_2^{\square}$ are depicted. The aim is to find the largest values λ_{opt} of λ, such that the corresponding probabilistic diamond $\diamond(o, t_i, t_j, d, \wedge, \lambda_{opt}^{\exists})$ which we call optimal subdiamond, does not intersect $\mathcal{Q}_1^{\square}$ ($\mathcal{Q}_2^{\square}$). To solve this problem, we distinguish between the following cases.

Case 1: the direct line between observations $(o(t_i), t_i)$ and $(o(t_j), t_j)$ in dimension d intersects $\mathcal{Q}^{\square}[d]$. In this case, there cannot exist any $\lambda \in [0, 1]$ such that $\mathcal{Q}^{\square} \cap \diamond(o, t_i, t_j, d, dir, \lambda) = \emptyset$. Therefore, our problem has no solution in dimension d, and d is ignored.

Case 2: the direct line between $(o(t_i), t_i)$ and $(o(t_j), t_j)$ does not intersect $\mathcal{Q}^{\square}[d]$, and we assume without loss of generality that $\mathcal{Q}^{\square}[d]$ is located above this line.[2] In addition, in this case, the time value of the north corner c of $\diamond_d(o, t_i, t_j)$ is located in the interval $\mathcal{T}^{\square}$ (e.g., see $\mathcal{Q}_2^{\square}$ in Figure 4(d)).[3] In this case, the edge $v_{opt}^{\lessdot}$ of the optimal subdiamond $\diamond(o, t_i, t_j, d, dim, \lambda_{opt}^{\exists})$ is given by $(o(t_i), t_i)$ and (s, t) where s corresponds to the lower bound of $\mathcal{S}^{\square}[d]$ and t equals to the time component of c.

Case 3: $\mathcal{Q}^{\square}$ is above the direct line between $(o(t_i), t_i)$ and $(o(t_j), t_j)$ (as in Case 2), but the time value of the north corner c of $\diamond_d(o, t_i, t_j)$ is not located in the time interval $\mathcal{T}^{\square}$ (e.g. $\mathcal{Q}_1^{\square}$ of Figure 4(d)). In this case, the optimal subdiamond must touch a corner of $\mathcal{Q}^{\square}[d]$ due to convexity of both $\mathcal{Q}^{\square}[d]$ and any diamond. If $\mathcal{Q}^{\square}[d]$ is located to the left of c (the right direction is handled symmetrically), then the edge $v_{opt}^{\lessdot}$ of the optimal subdiamond is given by the line between $(o(t_i), t_i)$ and the lower right corner of $\mathcal{Q}^{\square}[d]$ (e.g., see Figure 4(d)).

The optimal value $\lambda_{opt}^{\exists}$ for cases 2 and 3 equals the quotient $\frac{v_{opt}^{\lessdot} - v_{avg}}{v^{\lessdot} - v_{avg}}$, i.e., the fraction of the maximum velocity of the optimal subdiamond and the maximum velocity of the full diamond, both normalized by the average velocity $v_{avg} = \frac{s(t_j)[d] - s(t_i)[d]}{t_j - t_i}$. After identifying the value for $\lambda_{opt}^{\exists}$, for a dimension d and a direction dir, we can compute the probability of the corresponding subdiamond $\diamond(o, t_i, t_j, d, dir, \lambda_{opt}^{\exists})$. Since we can guarantee, that any path in this subdiamond does not intersect the query window, we can obtain a lower bound

$$P_{LB}(never(o, t_i, t_j, \mathcal{Q}^{\square})) = P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda_{opt}^{\exists}))) \quad (2)$$

of the event that o never intersects the query window in the time interval $[t_i, t_j]$. This directly yields an upper bound

$$P_{UB}(sometimes(o, t_i, t_j, \mathcal{Q}^{\square})) =$$

$$1 - P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda_{opt}^{\exists}))) \quad (3)$$

of the probability that the reverse event that o intersects the query

[1] Since the context is clear, we simply use $\diamond$ to denote $\diamond(o, t_i, t_j, d, dir, \lambda)$.

[2] If $\mathcal{Q}^{\square}[d]$ is below the line, we consider direction $dir = \vee$ symmetrically.

[3] Corner c is given by the intersection of lines $(o(t_i), t_i) + v^{\lessdot}$ and $(o(t_j), t_j) + v^{\gtrdot}$.

window at least once in $[t_i, t_j]$. This bound can be used for probabilistic pruning for PST$\tau\exists$ queries, as we will see in Section 3.6.

3.5 Approximating Probabilistic Diamonds

The main goal of our index structure, proposed in Section 4, is to avoid expensive probability computations for subdiamonds. Since the query window is not known in advance, $2D$ computations (i.e., one for each dimension and direction) have to performed in order to identify the optimal subdiamond for a given query and candidate object o. To avoid these computations at run-time, we propose to precompute, for each diamond $\diamond(o, t_i, t_j)$ in $\mathcal{D}$, probabilistic subdiamonds for each dimension and direction and for a set Λ of λ-values. This yields a catalogue of probability values, i.e. a probability for each $\diamond(o, t_i, t_j, d, dir, \lambda), d \in D, dir \in \{\vee, \wedge\}, \lambda \in \Lambda$.

Given a query window, the optimal value λ_{opt} computed in Section 3.4 may not be in Λ. Thus, we need to conservatively approximate the probability of probabilistic diamonds $\diamond(o, t_i, t_j, d, dir, \lambda)$ for which $\lambda \notin \Lambda$. We propose to use a conservative linear approximation of $P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda)))$ which increases monotonically with λ, using the precomputed probability values. For example, Figure 4(c), shows the $(\lambda, probability)$-space, for six values $\Lambda = \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. The corresponding precomputed pairs $(\lambda, P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda))))$ are depicted. Our goal is to find a function $f(\lambda)$ that minimizes the error with respect to $P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda))$, while ensuring that $\forall\lambda \in [0, 1] : f(\lambda) \leq P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda))$. The latter constraint is required to maintain the conservativeness property of the approximation, which will be required for pruning. We model this as a linear programming problem: find a linear function $l(\lambda) = a \cdot \lambda + b$ that minimizes the aggregate error with respect to the sample points, under the constraint that the approximation line does not exceed any of the sample values (e.g., the line in Figure 4(c)). That is, we compute:

$$argmin_{a,b}(\sum_{\lambda \in \Lambda} P(\lambda) - (a + b \cdot \lambda))$$

$$\text{subject to: } \forall\lambda \in \Lambda : P(\lambda) \geq a + b \cdot \lambda$$

We use the simplex algorithm to solve fast this optimization problem. In summary, a probabilistic spatio-temporal object o is approximated by a set of $|o.T_{obs}| - 1$ diamonds, one for each subsequent time points $t_i, t_j \in o.T_{obs}$. Each diamond approximation contains its spatio-temporal diamond $\diamond(o, t_i, t_j)$, consisting of four real values $v^{\leqslant}, v^{\leqslant}, v^{\geqslant}, v^{\geqslant}$, and a set of $2 \cdot D$ linear approximation functions $f_{d,dir}(\lambda)$, one for each dimension $d \in D$ and each direction $dir \in \{\vee, \wedge\}$. Next, we will show how to use these object approximations for efficient query processing over uncertain spatio-temporal data.

3.6 Probabilistic Filter

For each dimension $d \in D$ and direction $dir \in \{\vee, \wedge\}$, we now have a linear function to approximate all $(\lambda, P(o, t_i, t_j, d, dim, \lambda))$. However, using this line directly, may violate the conservativeness property, since the true function may have any monotonic increasing form, and thus, for a value λ_Q located in between two values λ_1 and λ_2 ($\lambda_1, \lambda_2 \in \Lambda, \lambda_1 < \lambda_Q < \lambda_2$) the probability is bounded by $P(\lambda_1) \leq P(\lambda_Q) \leq P(\lambda_2)$. To avoid this problem, we can exploit that the catalogue Λ is the same for all diamonds, dimensions and directions. Thus, we chose the function $f(\lambda) = l(\lfloor\lambda\rfloor)$, where $\lfloor\lambda\rfloor$ denotes the largest element of Λ such that $\lfloor\lambda\rfloor \leq \lambda$. In our running example, the function $f(\lambda)$ is depicted in Figure 4(e). In this example, assume that we have computed an optimal value λ_{opt} in the previous steps. The corresponding conservative approximation $f(\lambda_{opt})$ is shown.

Now, we show how these probability bounds can be used to bound the probability that an object (i.e. its corresponding chain of diamonds) satisfies the query predicate. This is done by probing each uncertain trajectory approximation (each necklace) on the query region $\mathcal{Q}^{\square}$. Obviously, we only have to take into account diamonds intersecting the query time range $\mathcal{T}^{\square}$. In turn, when probing an uncertain trajectory approximation $\diamond(o, t_i, t_j)$ on the query range $\mathcal{Q}^{\square}$, we only have to take the time range $[t_i, t_j]$ into account; i.e., if the time range $\mathcal{T}^{\square}$ of the query spans beyond $[t_i, t_j]$, we truncate $\mathcal{T}^{\square}$ accordingly. Consequently, in the case where more than one diamonds of an object intersect $\mathcal{T}^{\square}$, we can split $\mathcal{Q}^{\square}$ at the time dimension and separately probe the object diamonds on the corresponding query parts. The resulting probabilities obtained for individual diamonds can be treated as independent.

LEMMA 1. *Let $\diamond(o, t_i, t_j)$, $\diamond(o, t_j, t_k)$ be two successive diamonds of object o and $\diamond_1 := \diamond(o, t_i, t_j, d_1, dir_1, \lambda_1)$, $\diamond_2 := \diamond(o, t_j, t_k, d_2, dir_2, \lambda_2)$ be probabilistic subdiamonds, associated with respective probabilities $P(\diamond_1)$ and $P(\diamond_2)$ that o intersects these subdiamonds. Then, the probability $P(\diamond_1 \wedge \diamond_2)$ that o intersects both subdiamonds, is given by*

$$P(inside(o, \diamond_1) \wedge inside(o, \diamond_2)) =$$

$$P(inside(o, \diamond_1)) \cdot P(inside(o, \diamond_2))$$

PROOF. We first rewrite $P(inside(o, \diamond_1) \wedge inside(o, \diamond_2))$ using conditional probabilities.

$$P(inside(o, \diamond_1) \wedge inside(o, \diamond_2)) =$$

$$P(inside(o, \diamond_1)) \cdot P(inside(o, \diamond_2) | inside(o, \diamond_1))$$

Furthermore, we exploit the knowledge that object o is at the observed location $o(t_j)$ at time t_j

$$P(inside(o, \diamond_1) \wedge inside(o, \diamond_2)) =$$

$$P(inside(o, \diamond_1)) \cdot P(inside(o, \diamond_2) | inside(o, \diamond_1) \wedge o(t_j))$$

Based on the Markov model assumption, we know that, given the position at t_j, the behavior of o in the time interval $[t_j, t_k]$ is independent of any position at times $t < t_j$. Thus, we obtain:

$$P(\diamond_1 \wedge \diamond_2) = P(\diamond_1) \cdot P(\diamond_2 | o(t_j))$$

Finally, the lemma is proved based on the fact that the position $o(t_j)$ has been observed, and thus, is not a random variable. □

Lemma 1 shows that the random events of two successive probabilistic diamonds of the same object are conditionally independent, given the observation in between them. This observation allows us to compute the probability $P^{\exists}(o)$ that the whole chain of diamonds of o intersects a query window $\mathcal{Q}^{\square}$. Let $\{t_i, t_j\} \subseteq_{seq} o.T_{obs}$ be the set of pairs of subsequent observations in $o.T_{obs}$. Then,

$$P^{\exists}(o) = P(\bigvee_{\{t_i,t_j\} \subseteq_{seq} o.T_{obs}} sometimes(o, t_i, t_j, \mathcal{Q}^{\square})$$

That is, o satisfies a PST$\tau\exists$ query, if and only if at least one diamond of o intersects $\mathcal{Q}^{\square}$ at least once. Rewriting yields

$$P^{\exists}(o) = 1 - P(\bigwedge_{\{t_i,t_j\} \subseteq_{seq} o.T_{obs}} never(o, t_i, t_j, \mathcal{Q}^{\square}).$$

Exploiting Lemma 1 yields

$$P^{\exists}(o) = 1 - \prod_{\{t_i,t_j\} \subseteq_{seq} o.T_{obs}} P(never(o, t_i, t_j, \mathcal{Q}^{\square}))$$

Using our probability bounds derived in Section 3.4, we obtain

$$P^{\exists}(o) \leq 1 - \prod_{\{t_i,t_j\} \subseteq seq^o.T_{obs}} P_{LB}(never(o,t_i,t_j,\mathcal{Q}^{\square}))$$

which can be used to prune o, if

$$1 - \prod_{\{t_i,t_j\} \subseteq seq^o.T_{obs}} P_{LB}(never(o,t_i,t_j,\mathcal{Q}^{\square})) < \tau \quad (4)$$

If Equation 4 cannot be applied for pruning, we propose to iteratively refine single diamonds of o. Thus, the exact probability $P(sometimes(o,t_i,t_j,\mathcal{Q}^{\square}))$ is computed using the technique proposed in [8]. This exact probability of a single diamond can then be used to re-apply the pruning criterion of Equation 4 , by using the true probability as lower bound. When all diamonds of o have been refined, Equation 4 yields the exact probability $P^{\exists}(o)$.

4. THE UST-TREE

In the previous section, we showed that we can precompute a set of approximations for each object, which can be progressively used to prune an object during query evaluation. In this section, we introduce the UST-tree, which is an R-tree-based hierarchical index structure, designed to organize the object approximations and efficiently prune objects that may not possibly qualify the query; for the remaining objects the query is directly verified based on their Markov models, as described in [8] (*refinement step*). Section 4.1 describes the structure of the UST-tree and Section 4.2 presents a generic query processing algorithm for answering $PST\tau\exists$ queries.

4.1 Architecture

The UST-tree index is a hierarchical disk-based index. The basic structure is illustrated in Figure 5. An entry on the leaf level corresponds to an approximation of an object o represented by a quadruple $(\square(o,t_i,t_j), \diamond(o,t_i,t_j), \{f_{d,dir} : d \in D, dir \in \{\vee,\wedge\}\}, oid)$, containing (i) the MBR approximation $\square(o,t_i,t_j)$ (cf. Section 3.1), (ii) the diamond approximation $\diamond(o,t_i,t_j)$ (cf. Section 3.1), (iii) a set $\{f_{d,dir} : d \in D, dir \in \{\vee,\wedge\}\}$ of $2 \cdot D$ linear approximation functions for the precomputed probabilistic diamonds of o (cf. Section 3.5), and (iv) a pointer oid to the exact uncertain spatio-temporal object description (raw object data). Intermediate node entries of the UST-tree have exactly the same structure as in an R-tree; i.e., each entry contains a pointer referencing its child node and the MBR of all MBR approximations stored in pointed subtree. Note that the necklace of each object is decomposed into diamonds, which are stored independently in the leaf nodes of the tree. Since the directory structure of the UST-tree is identical to that of the R-tree, the UST-tree uses the same methods as the R*-tree [2] to handle updates.

4.2 Query Evaluation

Given a spatio-temporal query window $\mathcal{Q}^{\square}$, the UST-tree is hierarchically traversed starting from the root, recursively visiting entries whose MBRs intersect $\mathcal{Q}^{\square}$; i.e., the subtree of an intermediate entry e is pruned if $e.mbr \cap \mathcal{Q}^{\square} = \emptyset$. For each leaf node entry e, we progressively use the spatio-temporal and probabilistic diamond approximations stored in e to filter the corresponding object.

In the spatio-temporal filter step, we first use $\square(o,t_i,t_j)$ (ST-MBR Filter) using simple rectangle intersection tests. If this filter fails, we proceed using $\diamond(o,t_i,t_j)$ (ST-Diamond filter) by performing intersection tests against $\mathcal{Q}^{\square}$ as described in Section 3.2. Note that sometimes multiple leaf entries associated with an object are required to prune an object or confirm whether it is a *true hit*. Therefore, candidates are stored in a list until all their diamond approximations have been evaluated.

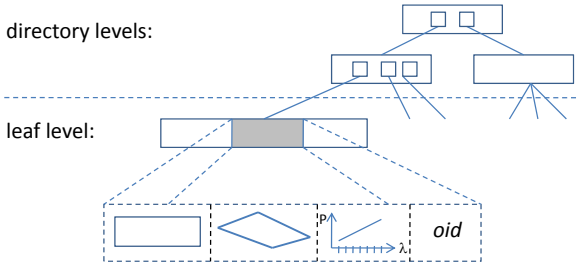


Figure 5: The UST-tree.

Finally, for the remaining candidates we exploit the probabilistic filter (Probabilistic Diamond Filter) as described in Section 3.6. Thereby, we use the linear approximation functions $\{f_{d,dir} : d \in D, dir \in \{\vee,\wedge\}\}$ stored in the leaf-node entry in order to derive an upper bound of the qualification probability $P(\exists t \in (\mathcal{T}^{\square} \cap [t_i,t_j]) : o(t) \in \mathcal{S}^{\square})$ or $P(\forall t \in (\mathcal{T}^{\square} \cap [t_i,t_j]) : o(t) \in \mathcal{S}^{\square})$ (depending on the query predicate). For each object o which is not pruned (or reported as true hit), we accumulate in a list $L(o)$ all upper bounds of its qualification probabilities from the leaf entries that index the diamonds of o. After collecting all candidate objects, the qualification probabilities stored in the list $L(o)$ for each candidate o are aggregated in order to derive the upper bound of the overall qualification probability $P(\exists t \in \mathcal{T}^{\square} : o(t) \in \mathcal{S}^{\square})$ for o, as described in Section 3.6. If this probability falls below τ we can skip o, otherwise we have to refine o by accessing the exact object data referenced by oid.

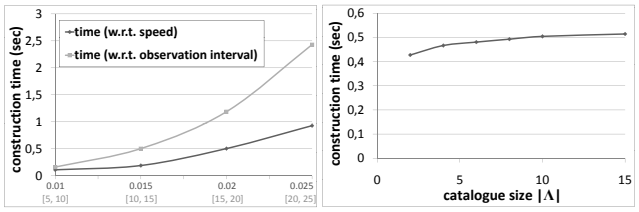
5. EXPERIMENTAL EVALUATION

In order to evaluate the proposed techniques we used data derived from a real application and several synthetic data sets.

Real Data. As a basis for the real world data served the trajectory data set containing one-week trajectories of 10,357 taxis in Beijing from [27]. This heterogeneous dataset contains trajectories having different samples rates, ranging from one sample every fives seconds to one sample every 10 minutes. The average time between localization updates (observations) is 177 seconds. The average distance between two observations is 623 m. We applied the techniques from [4] to obtain both a set of possible states (mostly corresponding crossroads) and a transition matrix reflecting the possible movements of the taxis. We only included data of taxis where the time between two GPS signals (observations) is no more than two minutes to train the Markov chain. The resulting data set consists of 3008 states and 11699 possible transitions between theses states.

Synthetic Data. In order to demonstrate the behavior of the proposed techniques depending on the underlying data we also generated a set of synthetic data sets with different characteristics. For the possible states, we generated n points uniformly distributed in the $[0,1]^2$ space. Each point was then connected to the points which have an Euclidean distance smaller than ϵ. Those connections correspond to the possible movements of an object in the space and we randomly assigned probabilities to each connection such that the sum of all outgoing edges sums up to 1. These values are the entries for the corresponding transition matrix. As a default for this dataset we generated 1000 objects each with 100 observations (= 99.000 probabilistic diamonds) and the parameters were set to $n = 10000, \epsilon = 0.02$ and the catalogue size $|\Lambda| = 10$.

Observations. Additionally to the positions of the states and the transition matrix, we further need observations from each object in order to build a database. The observations were constructed



(a) diamond parameter (b) catalogue size

Figure 6: diamond construction

by a directed random walk through the underlying graph (states = vertices and non-zero transitions = edges). At some time steps we memorize the current position of the object and take these time-state-tuples as an observation of the object. The time steps between two successive observations was randomly chosen from the interval [10,15] if not stated otherwise. For the observations of the real dataset we used the GPS data of taxis, where the time between two signals is between 2 and 20 minutes.

All experiments were run on a Quad Intel Xeon server running Windows Server 2008 with 16 GB RAM and 3.0 GHz. The UST-tree was implemented in Java. For all operations involving matrix operations (e.g., the refinement step of queries) we used MATLAB for efficient processing. All query performance evaluation results are averaged over 1000 queries. The spatial extent of the query windows in each dimension was set to 0.1 and the duration of the queries was set to 10 time steps by default. Unless otherwise stated, we experimented with PST$\tau\exists$ queries, with $\tau = 0.5$. The page size of the tree was set to 4 KB. Our experiments assess the construction cost of the UST-tree structure and its performance on query evaluation. For experiments regarding the effectivity of the Markov-chain model, and an evaluation of its capability to capture the real world in various applications, we refer to previous work, e.g., [1, 4, 10, 18], where Markov Chains were proved successful in modeling spatio-temporal data.

5.1 UST-tree Construction

The first experiment investigates the cost of index construction. In particular, we evaluate the cost for generating the spatio-temporal and probabilistic diamond approximations used to build the entries of the leaf level; this is the bottleneck of constructing and updating the tree, since restructuring operations always take at most 1ms. On the other hand, constructing the probabilistic diamonds is typically 2-3 orders of magnitude costlier, as illustrated in Figure 6(a). Still, this cost is reasonable, since the construction of a probabilistic diamond is comparable to the construction of $2 \cdot D \cdot |\Lambda|$ subdiamonds, which in turn corresponds to one refinement step (considering the subdiamond as a query window). Construction times pay off, when the query load on the database is reasonable. Figure 6(a) illustrates the construction time as a function of the speed of the objects (upper x-axis values) and the number of time steps between successive observations (lower x-axis values). From a theoretical point of view, both parameters linearly increase the number of reachable states, i.e., the density of the sparse vectors representing the uncertain position of an object at one point of time. The results reflect the theoretical considerations showing a quadratic runtime behavior with respect to both parameters. In a streaming scenario with several updates/insertions per second and large probabilistic diamonds (due to high speed of objects or large intervals between observations), the construction of probabilistic diamonds can be performed in parallel and is therefore still feasible. Figure 6(b) shows the construction cost as a function of parameter $|\Lambda|$ (which determines the number of subdiamonds). Theoretically this parameter should have a linear impact on the construction time. However our implementation exploits the monotonicity of the uncertain trajectories regarding probabilistic subdiamonds; a trajectory which is not included in the probability of a subdiamond, is also excluded from larger subdiamonds in the same dimension and direction. This explains the sublinear runtime w.r.t. $|\Lambda|$.

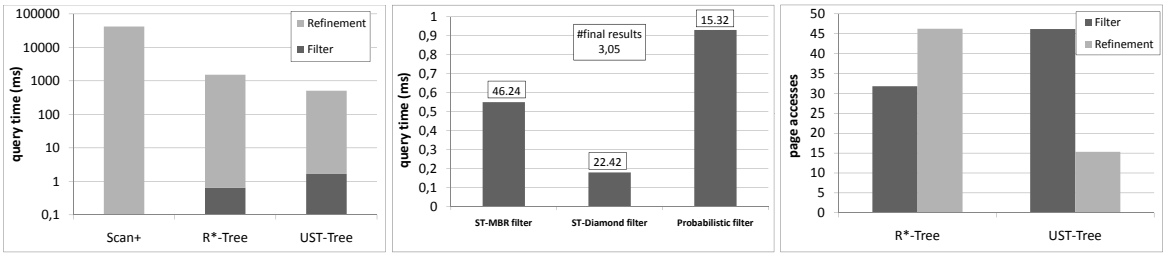
5.2 Query Performance

In the first set of query performance experiments, we compare the cost of using UST-tree with two competitors on synthetic data (see Figure 7). *Scan+* is a scan based query processing implementation, i.e., without employing any index [8]. For each pair of two successive observations of an object, refinement is performed immediately, i.e., there is no filter cost. We enhanced the implementation of *Scan+* by prepending a simple temporal filter, which only considers observation pairs which temporally overlap the query window. The *R*-Tree* competitor approximates all possible locations (i.e. state-time pairs) between two successive observations of an object using only $\Box(o, t_i, t_j)$. These MBRs are then indexed using a conventional R*-Tree [2]. In Figure 7(a), we show the average CPU cost per query (I/O cost is not the bottleneck in this problem), for the three competitors. The cost are split into filter and refinement costs. Although the R*-Tree has lower filter cost, the overall query performance of the UST-tree is around 3 times better than that of the R*-Tree (note the logarithmic scale). This is attributed to the effectiveness of the different filter steps used by the UST-tree; the overhead of the UST-tree filter is negligible compared to the savings in refinement cost.

Figure 7(b) shows the cost and the effectiveness of the individual filter steps of the filter-refinement pipeline used by the UST-tree. The bars show the overall runtime (query time) of each filter and the numbers on top of the bars show the effectiveness of the filter in terms of remaining (observation pair) candidates after the corresponding filter has been applied. We clearly see that the spatio-temporal filters reduce the number of candidates and, thus, the number of required refinements, drastically. We can also observe that the probabilistic filter can reduce the number of refinements by 30% after applying the sequence of spatio-temporal filters. Comparing the cost of the probabilistic filter (which is comparable to that of the spatio-temporal filter) to the cost of candidate refinement, we can observe that the cost required to perform the probabilistic filter can be neglected. This experiment shows that each of the filters incorporated in the UST-tree indeed pays off in terms of CPU cost.

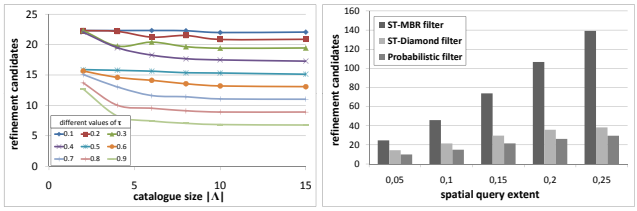
Although I/O cost is not the bottleneck under our setting, the I/O costs of *R*-Tree* and the UST-tree are illustrated in Figure 7(c) for completeness. Filter cost here means all costs which occur during the traversal of the corresponding index structure, i.e., access to intermediate and leaf nodes. Refinement cost includes the number of page accesses to refine the observation pairs that pass the filter step, assuming one I/O per such pair. Note, that the cost of a refinement can be much higher than one page access (e.g. if the Markov Chain, which can become very large does not fit in one disk page) under different settings. The UST-tree has higher filtering cost, since the representation of the probabilistic diamonds requires more space and the tree is larger than the R*-Tree, which only stores MBR approximations but incurs much higher I/O cost for refinements.

The above experiments unveil that the most costly operation is the refinement of spatio-temporal diamonds; thus, we now take a closer look at the effectiveness of the three different methods on pruning spatio-temporal diamonds. The next experiments measure the number of spatio-temporal diamonds which have to be refined at the refinement step; these results can be directly translated to runtime differences of the different approaches.

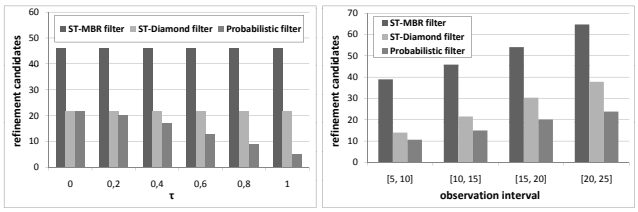


(a) CPU Comparison (b) CPU Time of Filter (c) Page Accesses

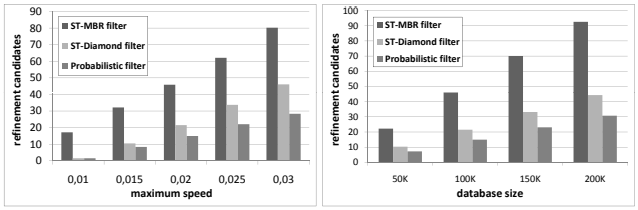
Figure 7: Overall Performance (Synthetic Data Set)



(a) $|\Lambda|$ for different values of τ (b) query extent



(c) varying τ (d) observation time interval

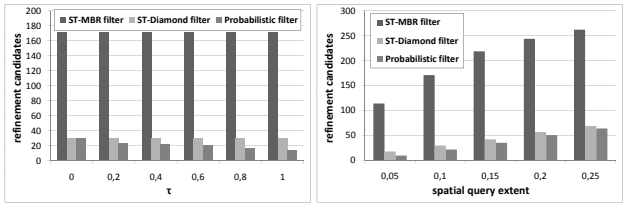


(e) varying maximum speed (f) database size

Figure 8: Experiments on Synthetic Data

Size of the Catalogue $|\Lambda|$. An important tuning parameter for the index is the size of the catalogue which is used for building the probabilistic diamond approximations. In Figure 8(a), it can be observed that the filter effectiveness converges at around $|\Lambda| = 10$ (default value for the experiments). Depending on the query parameter τ, a too small catalogue yields up to twice as much candidates which have to be refined. Note that the number of refinement candidates does not decrease monotonically in $|\Lambda|$. In general a larger catalogue results in a linear function with a larger approximation error. However, the step-function for the conservative approximation becomes smoother which results in a smaller approximation error. Because of these two contrary effects a larger catalogue does not always result in higher filter effectiveness.

Query Parameters. The characteristics of the query have different implications on the index performance. Increasing the spatial extent of the query obviously yields more candidates since more diamonds in the database are affected (cf. Figure 8(b)). The spatio-temporal filter utilizing the diamond approximations becomes more effective in comparison to the ST-MBR-Filter. The percentage of the diamonds which can be pruned using the probabilistic filter remains rather constant (at around 30%) in comparison to the spatio-temporal filter. Another query parameter is the temporal extent of the query. Increasing the length of the query time window $\mathcal{T}^{\square}$ increases the number of refinement candidates. The results are very similar to the results when increasing the spatial extent of the query; we do not include the comparison plot due to space limitations.



(a) varying τ (b) query extent

Figure 9: Experiments on Real Data ($\forall$)

Changing the value of τ obviously only affects the probabilistic filter (cf. Figure 8(c)). The higher τ is set, the more candidates can be pruned by the probabilistic filter. From a value of around 20%, the candidates which have to be refined decrease linearly with τ.

Influencing Variables of ST Diamonds. The size of the spatio-temporal diamonds is generally affected by two parameters. One is the time interval between successive observations, since a larger interval increases the space that can be reached by the moving object between the two observations. For this experiment, the number of time steps between successive observations in the data set was chosen randomly from the intervals on the x-axis in Figure 8(d). The second parameter is the speed of the object and has a similar effect. The speed corresponds to the parameter ϵ, which reflects the maximum distance of points which can be reached by an object within one time step (cf. Figure 8(e)). Since larger spatio-temporal diamonds usually result in more objects which intersect the query window, the number of candidates to be refined increase when increasing these two parameters. Interestingly, the effectiveness of the ST-Diamond Filter decreases over the ST-MBR-Filter, whereas the pruning effectiveness of the Probabilistic Filter increases. This shows, that the probabilistic filter copes better with more uncertainty in the data than the other two filters.

Database Size. We evaluated the scalability of the UST-tree by increasing the amount of observations (cf. Figure 8(f)). The number of results increases linearly with the database size. The experiment also shows that the number of refined candidates increase linearly.

Real Data. The experiments on the real world data, show similar behavior as those on the synthetic data. Due to space limitations, we only show excerpts from the evaluation. Figure 9(a) illustrates the results for PST$\tau\forall$ queries when varying the value of τ. It is notable that the ST-Diamond Filter seems to even perform better (compared to the ST-MBR-Filter) on the real dataset. The reason for this is that the real dataset has much more inherent irregular-

ity (regarding the locations and the movement of obejcts). This favors the ST-Diamond filter over the MBR approximation (since diamonds are more skewed as in Figure 2(b)). The probabilistic filter is apparently not affected. When varying the query extent (cf Figure 9(b)) the results resemble the results on the synthetic dataset.

6. RELATED WORK

The problem of managing, mining and querying spatio-temporal data has received continuous attention over the past decades (for a comprehensive coverage, see [9]). Specifically for efficient query processing a vast amount of indexing structures for different purposes and data characteristics has been developed (an overview and a classification can be found in [14] and [16]). From this body of work, our approach is mostly related to spatio-temporal data indexing for predictive querying, for example indexes like [19, 12] and approaches like [20]. Still, these papers neither consider probabilistic query evaluation nor model the data with stochastic processes.

However, in scenarios where data is inherently uncertain, such as in sensor databases, answering traditional queries using expected values is inadequate, since the results could be incorrect [3]. One of the first works that deal with uncertainty in trajectories is [17]. This work reviews the sources of error which yield to uncertain trajectories and proposes a filter refinement approach for simple query types. The prevalent approach is to bound the possible positions of an object at each point of time by simple a spatial structure resulting in a spatio-temporal approximation. Examples include static ellipses [25, 24, 23], dynamic MBRs (Minimum Bounding Rectangles) [15] and dynamic ellipses [22, 13] yielding skewed cylinders, diamonds and beads, respectively. To answer queries most of the existing works restrict the possible queries. Often no specific assumption is made about the probability density function (pdf) of the object positions over time ([17, 25, 24, 23, 22, 28, 7]). Thus quantifiers such as "always", "sometimes", "definitely" and "possibly" are used to indicate whether an object intersects a given spatio-temporal query window. These types of queries can be answered by only considering the spatio-temporal approximations. As a consequence, these approaches do not compute probabilities for objects to qualify the queries. A possibility for returning probabilistic results is to restrict the temporal window of the query, such that queries refer to exactly one point in time (cf. [5, 28]). All the aforementioned approaches avoid modeling and considering the time dependencies between successive object locations (see Section 2). These dependencies were first considered in [8], where a Markov Chain model is used, and [18], where certain event detection is the main focus (this work does not handle window queries).

Our work is also inspired by methods for indexing uncertain spatial data. The U-Tree [20, 21] and its extension [29] bound each spatial uncertain object with an MBR and additionally associate it with a set of "probabilistically constrained regions" (PCR). These PCRs can be used for probabilistic pruning during query processing. For efficiency reasons the set of PCRs are conservatively approximated by a linear function over the parameters of the PCRs.

7. CONCLUSIONS

In this work, we proposed the UST-tree which is an index structure for uncertain spatio-temporal data. The UST-tree adopts and incorporates state-of-the art techniques from several fields of research in order to cope with the complexity of the data. We showed how the most common query types (spatio-temporal $\exists$- and $\forall$-window queries) can be efficiently processed using probabilistic bounds which are computed during index construction. To the best of our knowledge, this is the first approach that supports query evaluation on very large uncertain spatio-temporal databases, adhering to possible worlds semantics. Outside the scope of this work is the consideration of an object's location before its first and after its last observation. In both cases, the resulting diamond approximation would be unbounded. An approach to solve this problem is to define a maximum time horizon for which diamond approximations are computed. Beyond this horizon, we can use the stationary distribution of the model M to infer the location of an object.

8. REFERENCES

[1] D. Ashbrook and T. Starner. Using gps to learn significant locations and predict movement across multiple users. *Personal Ubiquitous Comput.*, 7:275–286, 2003.

[2] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-Tree: An efficient and robust access method for points and rectangles. In *Proc. SIGMOD*, pages 322–331, 1990.

[3] G. Beskales, M. A. Soliman, and I. F. IIyas. Efficient search for the top-k probable nearest neighbors in uncertain databases. *Proc. VLDB Endow.*, 1(1):326–339, 2008.

[4] Z. Chen, H. T. Shen, and X. Zhou. Discovering popular routes from trajectories. In *Proc. ICDE*, pages 900–911, 2011.

[5] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Querying imprecise data in moving object environments. *TKDE*, 16(9):1112–1127, 2004.

[6] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4):523–544, 2007.

[7] Z. Ding. Utr-tree: An index structure for the full uncertain trajectories of network-constrained moving objects. In *MDM*, pages 33–40, 2008.

[8] T. Emrich, H.-P. Kriegel, N. Mamoulis, M. Renz, and A. Züfle. Querying uncertain spatio-temporal data. In *Proc. ICDE*, 2012.

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

[10] R. Hariharan and K. Toyama. Project lachesis: parsing and modeling location histories. In *In Geographic Information Science*, pages 106–124, 2004.

[11] K. Hornsby and M. J. Egenhofer. Modeling moving objects over multiple granularities. *Annals of Mathematics and Artificial Intelligence*, 36:177–194, 2002.

[12] C. S. Jensen, D. Lin, and B. C. Ooi. Query and update efficient b+-tree based indexing of moving objects. In *Proc. VLDB*, pages 768–779, 2004.

[13] B. Kuijpers and W. Othman. Trajectory databases: Data models, uncertainty and complete query languages. *J. Comput. Syst. Sci.*, 76(7):538–560, 2010.

[14] M. F. Mokbel, T. M. Ghanem, and W. G. Aref. Spatio-temporal access methods. *IEEE Data Eng. Bull.*, 26(2):40–49, 2003.

[15] H. Mokhtar and J. Su. Universal trajectory queries for moving object databases. In *Mobile Data Management*, 2004.

[16] L.-V. Nguyen-Dinh, W. G. Aref, and M. F. Mokbel. Spatio-temporal access methods: Part 2 (2003 - 2010). *IEEE Data Eng. Bull.*, 33(2):46–55, 2010.

[17] D. Pfoser and C. S. Jensen. Capturing the uncertainty of moving-object representations. In *Proc. SSD*, 1999.

[18] C. Ré, J. Letchner, M. Balazinksa, and D. Suciu. Event queries on correlated probabilistic streams. In *Proc. SIGMOD*, pages 715–728, New York, NY, USA, 2008. ACM.

[19] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez. Indexing the positions of continuously moving objects. In *Proc. SIGMOD*, pages 331–342, 2000.

[20] Y. Tao, R. Cheng, X. Xiao, W. K. Ngai, B. Kao, and S. Prabhakar. Indexing multi-dimensional uncertain data with arbitrary probability density functions. In *Proc. VLDB*, pages 922–933, 2005.

[21] Y. Tao, X. Xiao, and R. Cheng. Range search on multidimensional uncertain data. *ACM TODS*, 32(3):15, 2007.

[22] G. Trajcevski, A. N. Choudhary, O. Wolfson, L. Ye, and G. Li. Uncertain range queries for necklaces. In *Mobile Data Management*, pages 199–208, 2010.

[23] G. Trajcevski, R. Tamassia, H. Ding, P. Scheuermann, and I. F. Cruz. Continuous probabilistic nearest-neighbor queries for uncertain trajectories. In *Proc. EDBT*, pages 874–885, 2009.

[24] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. Managing uncertainty in moving objects databases. *ACM Trans. Database Syst.*, 29(3):463–507, 2004.

[25] G. Trajcevski, O. Wolfson, F. Zhang, and S. Chamberlain. The geometry of uncertainty in moving objects databases. In *Proc. EDBT*, pages 233–250, 2002.

[26] M.-Y. Yeh, K.-L. Wu, P. S. Yu, and M. Chen. PROUD: a probabilistic approach to processing similarity queries over uncertain data streams. In *Proc. EDBT*, pages 684–695, 2009.

[27] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *Proc. KDD*, pages 316–324, 2011.

[28] M. Zhang, S. Chen, C. S. Jensen, B. C. Ooi, and Z. Zhang. Effectively indexing uncertain moving objects for predictive queries. *PVLDB*, 2(1):1198–1209, 2009.

[29] Y. Zhang, X. Lin, W. Zhang, J. Wang, and Q. Lin. Effectively indexing the uncertain space. *IEEE TKDE*, 22(9):1247–1261, 2010.