Querying Uncertain Spatio-Temporal Data

Tobias Emrich [#1], Hans-Peter Kriegel [#1], Nikos Mamoulis [*2], Matthias Renz [#1], Andreas Züfle [#1]

[#]*Institute for Informatics, Ludwig-Maximilians-Universität München*
Oettingenstr. 67, D-80538 München, Germany
[1]{emrich, kriegel, renz, zuefle}@dbs.ifi.lmu.de

[*]*Department of Computer Science, University of Hong Kong*
Pokfulam Road, Hong Kong
[2]nikos@cs.hku.hk

***Abstract*— The problem of modeling and managing uncertain data has received a great deal of interest, due to its manifold applications in spatial, temporal, multimedia and sensor databases. There exists a wide range of work covering spatial uncertainty in the static (snapshot) case, where only one point of time is considered. In contrast, the problem of modeling and querying uncertain spatio-temporal data has only been treated as a simple extension of the spatial case, disregarding time dependencies between consecutive timestamps. We present a framework for efficiently modeling and querying uncertain spatio-temporal data. The key idea of our approach is to model possible object trajectories by stochastic processes. This approach has three major advantages over previous work. First it allows answering queries in accordance with the possible worlds model. Second, dependencies between object locations at consecutive points in time are taken into account. And third it is possible to reduce all queries on this model to simple matrix multiplications. Based on these concepts we propose efficient solutions for different probabilistic spatio-temporal queries for a particular stochastic process, the Markov chain. In an experimental evaluation we show that our approaches are several order of magnitudes faster than state-of-the-art competitors.**

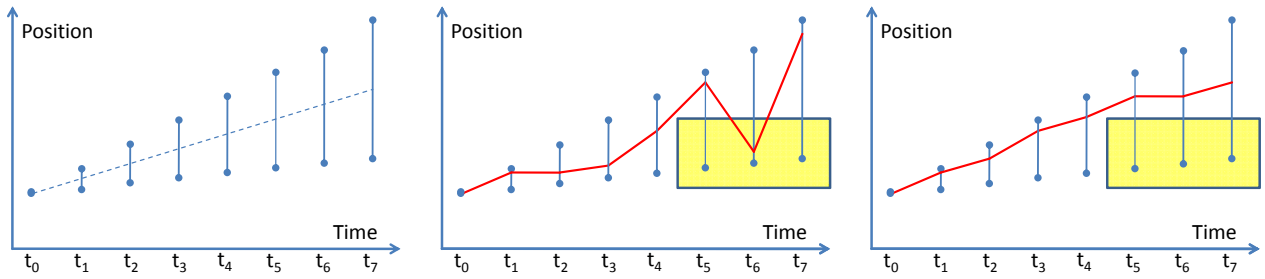
I. Introduction

Uncertain data management has received a lot of attention in the past decade, due to the abundance of uncertain data, typically collected by modern monitoring devices, such as GPS receivers and sensors. There exists a wide range of work studying spatial uncertainty in the static (snapshot) case, where only one point of time is considered, e.g. [1], [2], [3], [4], [5], [6], [7]. On the other hand, the problem of modeling and managing uncertain data with a temporal component has only received limited attention by the community. Most of the existing works treat spatio-temporal uncertainty as a simple extension of spatial uncertainty. For instance, the methods of [8] and [9] assume that for each timestamp in the history and every object, there is a location sample, which carries uncertainty. The samples are then modeled as uncertain regions, using probability density functions (PDFs) and these regions are connected to form the trajectories. Given a spatio-temporal range query [8], we can then estimate the probability that a trajectory intersects the window by the overlap of the corresponding PDFs with the window.

These types of models, however, may not be acceptable in the case where we only have a sample of locations in the moving history of an object. In this case, for timestamps where locations are not sampled, we have to infer the whereabouts of the object with the help of stochastic models. These models are particularly useful when predicting the future locations of objects, with the help of current (or recent) location and movement observations. In addition, these models allow for the economic representation and storage of the data, since only a subset of the object locations need to be sampled and used for the inference of their remaining locations.

As an exemplary application, consider the problem of monitoring iceberg activity in the North Atlantic. Ships transiting between Europe and east coast ports of North America traverse a great circular route that brings them into the vicinity of icebergs carried south by the cold Labrador Current. It was here that the R.M.S. Titanic sank in 1912, after it struck an iceberg. This disaster resulted in the loss of 1517 lives and led directly to the founding of the The International Ice Patrol (IIP) in 1914. The mission of the IIP is to monitor iceberg danger near the Grand Banks of Newfoundland and provide the locations of all known ice to the maritime community. The IIP does this by sighting icebergs, using visual observations from ships and aircrafts, as well as data from buoys and radars. A database stores the recorded positions and extents of observed icebergs and data models are used to predict their movement, based on the uncertainty of the recorded observations. Estimating the position of an iceberg at a time t underlies two sources of error: (i) the observation measurement error and (ii) the obsoleteness of the most recent observation. Using a model to describe this uncertainty, we can derive at each time t and each uncertain object o (i.e. an iceberg) a probability density function (pdf) describing the position of object o at time t, and answer any related queries.

We found that uncertain spatio-temporal data management based on such models has not been adequately studied in the past. This paper aims at filling this gap. Section II discusses related work. In Section III, we formally define the uncertain spatio-temporal data that we deal with in this paper and the queries that we will study. Then, we provide details on modeling uncertain spatio-temporal data with time dependencies (Section IV). The main contribution of the paper is presented in Section V, which includes a framework for processing spatio-temporal queries on such data. Our framework relies



(a) Abstract Model (b) Assuming temporal independence (c) Not assuming temporal independence

Fig. 1. Modeling Spatio-Temporal Data

on efficient analysis of the space of possible worlds by multiplying Markov-Chain transition matrices. Our approach is exploits the power of existing tools for matrix multiplication, e.g. provided by Matlab, in order to accelerate the computation of spatio-temporal data distributions in accordance to possible worlds semantics. We gracefully integrate pruning approaches into the Markov Chain matrices, which results in drastically reducing the search space and the computational effort during query evaluation. The proposed framework is general enough to be applied for cases where an arbitrary number of observations exist for uncertain moving objects and for different spatio-temporal query variants as we demonstrate in Sections VI and VII, respectively. As we show experimentally (Section VIII), we achieve a speed up of multiple orders of magnitude compared to a straightforward solution, which relies on Monte-Carlo simulation over the space of possible trajectories that the objects may follow. Our approach is easily implementable because the tools provided by Matlab libraries are available for all common programming languages (e.g., C/C++, Java, etc.) and, as a result, they can be easily integrated into existing uncertain DBMSs.

II. Related Work

The problem of querying spatio-temporal data has been studied extensively. There exist numerous publications on efficient query evaluation for the case where the attribute values at each time t are known for certain (for a comprehensive coverage, see [10]). From this body of work, our approach is mostly related to spatio-temporal data indexing for predictive querying, for example indexes like [11], [12] and approaches like [13]. Still, these papers neither consider probabilistic query evaluation nor model the data with stochastic processes.

However, in scenarios where data are inherently uncertain, such as sensor databases, answering traditional queries using expected values is inadequate, since the results could be incorrect [14]. In such cases, probabilistic queries that take the full information of the underlying uncertainty into account and that yield results with probabilistic guarantees are required.

One of the first works that deal with uncertainty in trajectories is [15]. This work considers routes that are captured by GPS and assumes that the recorded locations are uncertain. Indexing such data for range queries is considered; the authors use a simple model that sums up the probabilities that the trajectory points are included in the range queries to derive the probabilities of the results. Trajcevski et al. [16], [17] follow a similar approach for the same problem settings. At each point in time the position of an object is modeled as an ellipse. Each trajectory is thus represented by a 3D cylindrical body. Since no assumptions are made about the probability distribution inside the ellipses only binary answers to queries are possible. For example the model can answer if an object is certainly within a query region or could be inside a query region during a time interval but not give a probability to those events.

The work of [9] is based on the same model as [16], [17]; the authors assume a database of (historical) uncertain trajectories, each having a 3D cylindrical body. The objective is to identify the nearest neighbors of an uncertain query trajectory, throughout its lifetime; i.e., to partition the lifetime into intervals, each containing a stable most probable nearest trajectory from the database. Each interval is then partitioned recursively according to the second most probable neighbor, etc. Again, this work falls into the category of papers that do not consider location dependencies between consecutive timestamps and do not rely on stochastic models.

Cheng et al. [18] consider possible world semantics on static data with multidimensional or interval PDFs. A wide range of queries is studied. In [8], the model is extended to support search in databases with uncertain trajectories. Similar to [16], recorded trajectories (e.g., from GPS data) are spatially extended to capture all possible locations that the object may have passed through. Then, indexing is used to prune regions in space and time (together with the corresponding trajectory data) that do not satisfy the queries and possible worlds semantics are used to refine the overlapped areas in order to determine the probabilistic query results. Again, this work ignores inference based on stochastic models.

Approaches like [19] and [20] consider uncertain time series and data streams, respectively. Similar to other work, they also disregard correlations between points in time; that is, the position of an object at time t is assumed to be independent of its previous position at time $t-1$.

Mokhtar and Su [21] describe a model where the uncertainty region of each object is described by a time dependent stochastic process. Objects are given by MBRs which change their location and extent over time following the stochastic process. The paper shows how to answer certain types of window queries based on this model. However, describing the parameters of the uncertainty regions and not the trajectories

of the objects through a stochastic process yields wrong results regarding to possible worlds semantics. The reason is that location dependency between consecutive timestamps is ignored by this model.

The work described in [22] focuses on the prediction of uncertain trajectories in street networks. The authors propose the use of time-dependent inhomogeneous Markov processes for each crossing. This so called Trajectory Continuous Time Bayesian Network is constructed by analysing training data. Afterwards, it is used to predict the next movement of an object when arriving at a crossing. The proposed algorithm shows very high accuracy rates of around 80% predicting routes of objects. However the system does not consider data management and efficiency in query evaluation, but only tries to predict the routes of single objects.

To illustrate the problem of previous approaches disregarding temporal dependencies, consider Figure 1(a), where an uncertain object trajectory is modeled. Here, it is assumed that the object moves with an uncertain speed upwards. The speed of o may change over time, but will not drop below some minimal speed greater than zero and will not exceed some maximum speed. Therefore, given the position of an object o at time t_0, the future position of an object can be modeled using the expected speed of o (the dashed line in Figure 1(a)), and lower and upper bounds, or alternatively a variance, depicted by the intervals at each point of time. Since at each point of time, the positions of o are modelled as independent random variables, a trajectory such as depicted in Figure 1(b) has a probability greater than zero. However, this trajectory is actually not possible, since o makes a large leap backwards between times t_5 and t_6, which is not possible given knowledge about the movement of o, which this model should incorporate. A possible trajectory is shown in Figure 1(c). Here, the object moves within its speed limits at each points of time.

The flaw of modelling trajectories which are not actually possible becomes a problem when processing spatio-temporal queries based on this model. For example, consider a spatio-temporal window query, which is to return for an object o, the probability that o intersects the query window q, depicted in Figures 1(b) and 1(c). For any model that ignores the dependency between locations at subsequent points of time, the probability that o is always outside the window is the product of many probabilities and gets very small. Thus, for a large number of points of time inside the query region, the probability that o intersects the query window converges to one. However, if the dependency between locations at subsequent points of time is considered, then the probability that o is outside the window at time t_6 depends on the probability at time t_5. If o is not in the window q at t_5, then it cannot be in q at t_6 either, since the object cannot move backwards. Thus, the probability that o intersects the query window q at any time, is equal to the probability that o intersects the query window at time t_5. This is intuitive because the object cannot move back into the window. One of our aims in this work is to properly model such dependencies, instead of simply treating time as an additional dimension in space.

An initial approach to address the problem of temporal dependencies has been made by [23]. Similar to our approach, they assume a discrete state space and the Markov property to allow transition between successive points of time. Their query language *Lahar* allows to formulate queries by stating regular expressions on an alphabet of states, and returns the probability of observing a sequence of state satisfying this regular expression. However, this syntax cannot handle time context as required by many common queries. For example, no regular expression can express the language that contains at least one character x at a given position interval. Such query corresponds to a window query as used above, and formally defined in Section III.

To summarize, all works so far on querying uncertain spatio-temporal data assume that the location probabilities of an object at two different times are independent. However, time-dependence is the main characteristic of temporal data, which cannot simply be ignored and this is the focus of this work.

III. Problem Definition

In this paper, we assume discrete space and time domain $\mathcal{S}$ and $\mathcal{T}$, where space is defined by coordinates and time denotes points in time. Formally, let $\mathcal{S} = \{s_1, ...s_{|\mathcal{S}|}\} \subseteq \mathbb{R}^d$ be a finite set of possible locations in space which we call *states* and let $\mathcal{T} = \mathbb{N}_0^+$ be the time-space. Consequently, a (certain) object o that moves in space is represented by a *trajectory* given by a function $o : \mathcal{T} \rightarrow \mathcal{S}$ that defines the location $o(t) \in \mathcal{S}$ of o at a certain point of time $t \in \mathcal{T}$.

We assume uncertain spatio-temporal objects, i.e. objects $o \in \mathcal{D}$ that are associated with *uncertain object trajectories*. We have to cope with uncertain trajectories when taking future motions of objects (e.g., by extrapolation) into account or if we want to infer locations between subsequent object observations (i.e., by interpolation). In some applications, like the iceberg tracking example in the introduction, observations are rare and we have to infer the object locations at most timestamps in the domain $\mathcal{T}$. An observation at a specific time may be precise or uncertain.

To model uncertain object trajectories, we suppose that the locations of an uncertain spatio-temporal object $o \in \mathcal{D}$ at time t are realizations of a random variable $o(t)$. An uncertain object trajectory of object $o \in \mathcal{D}$ comprises a set of trajectories, each assigned with a probability indicating its likelihood to be the true trajectory of o. This consideration suggests modeling uncertain object trajectories as a realization of a stochastic process [24], formally:

Definition 1 (Uncertain Object Trajectory): Given the spatial domain $\mathcal{S}$ and the time domain $\mathcal{T}$, an *uncertain object trajectory* $o(t) \in \mathcal{S}$of an object $o \in \mathcal{D}$ is a stochastic process $\{o(t) \in \mathcal{S}, t \in \mathcal{T}\}$.

An example of an uncertain object trajectory of an object $o \in \mathcal{D}$ is illustrated in Figure 2. The raster models all possible locations (i.e., states) in $\mathcal{S}$, shown for the time sequence $\langle t_0, \ldots, t_3 \rangle$. Here we assume that the last observed location

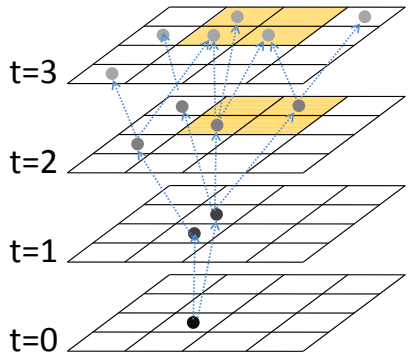


Fig. 2. Querying Uncertain Spatio-Temporal Data

of object o was at time t_0; all locations of o that follow are uncertain. Consequently, the uncertain trajectory of o comprises all possible trajectories starting at $o(t_0)$.

Our goal is to efficiently evaluate probabilistic spatio-temporal queries on uncertain spatio-temporal objects; i.e., queries about objects that are probably located in a given spatial region during a given range in time. Within the scope of this paper, we assume a set of uncertain spatio-temporal objects $\mathcal{D}$, i.e. objects associated with uncertain object trajectories $o(t)$, and focus on spatio-temporal queries specified by the following parameters: (i) a spatial region $S^{\Box} \subseteq \mathcal{S}$, i.e. a set of (not necessarily connected) locations in space, and (ii) a set $T^{\Box} \subseteq \mathcal{T}$ of (not necessarily subsequent) points in time. In the remainder, we use $Q^{\Box} = S^{\Box} \times T^{\Box}$ to denote the query ranges in the space and time domain. The most intuitive definition of a probabilistic spatio-temporal query is given below:

Definition 2: [Probabilistic Spatio-Temporal (Exists) Query] Given a query region $S^{\Box}$ in space and a query region $T^{\Box}$ in time, a *probabilistic spatio-temporal exists query* (PST$\exists$Q), retrieves for each object $o \in \mathcal{D}$ the probability $P(o(t) = s) \in [0, 1]$ that o is located in $S^{\Box}$ at some time $t \in T^{\Box}$.

This query type has been studied before (e.g. in [16], [17]), albeit over data models that disregard dependencies between locations at consecutive timestamps, as we have discussed in Section II. For our motivating application described in the introduction, an exemplary query could be: find all icebergs that have non-zero probability to be inside the movement range of a particular ship during the ship's movement in the North Atlantic. Another query could be to predict the number of cars that will be in a congested road segment after 10-15 minutes.

In addition, we study the following two interesting probabilistic query variants. Note that the second variant has not been considered in the past:

Definition 3: [Probabilistic Spatio-Temporal For-All Query] A *probabilistic spatio-temporal for-all query* (PST$\forall$Q) retrieves for each object $o \in \mathcal{D}$ the probability $P(o(t) = s) \in [0, 1]$ that o remains in $S^{\Box}$ for *all* times $t \in T^{\Box}$.

Definition 4: [Probabilistic Spatio-Temporal k-Times Query] A *probabilistic spatio-temporal k-times query* (PSTkQ) retrieves for each object $o \in \mathcal{D}$ and each parameter $1 \leq k \leq |T^{\Box}|$ the probability that o is located in $S^{\Box}$ at *exactly* k times $t \in T^{\Box}$.

PST$\forall$Q and PSTkQ are important complements to the PS$\exists$TQ. For example, these queries can progressively determine candidates that remain in a certain region for a while. For example, for a given region somewhere in the north Atlantic we want to retrieve all icebergs that have non-zero probability remaining in this region for a specified period of time, e.g. to be able to make some measurements over a certain time period. Further examples where such queries are useful are for location-based-service (LBS) applications, e.g. a service provider could be interested in customers that remain at a certain region for a while, such that they can receive advertisements relevant to the location.

Note that, although the spatial (temporal) parameters of the queries define contiguous regions (intervals) in the space (time) domain, our query processing approaches are also applicable for any arbitrary subset of the space (time) domain.

IV. Modeling Uncertain Spatio-Temporal Data

In accordance to the previous section, the constituent parts of an uncertain spatio-temporal object are specifications of probability distributions over the space and time domain. Naive models typically restrict the regions that the object can be at each timestamp. All uncertain trajectories are combinations of locations in these regions, giving all these trajectories equal probabilities ([9], [16], [17], [19], [20], [21]). However, as already mentioned, these models disregard any time dependencies between locations, possibly yielding incorrect results.

According to Definition 1, the uncertain motion of an object is defined as a stochastic process. In particular, the (first-order) Markov-Chain model in a discrete time- and state-space is assumed. The state space of the model is the spatial domain $\mathcal{S}$. State transitions are defined over the time domain $\mathcal{T}$. In addition, the Markov-Chain model is based on the assumption that the position $o(t+1)$ of an uncertain object o at time $t+1$ only depends on the position $o(t)$ of o at time t. Formally:

Definition 5: A stochastic process $o(t), t \in \mathcal{T}$ is called a Markov-Chain if and only if

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

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

$$P(o_{t+1} = s_j | o_t = s_i)$$

The conditional probability

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

is the (single-step) *transition probability* of state s_i to state s_j at time t.

Note that in this work we assume that the transition probabilities $P_{i,j}(t)$ are given, e.g. derived from expert knowledge or derived from historical data. For example, the current of the water in the Atlantic ocean can be used to infer the transitions of icebergs. For traffic data, the transition probabilities at each road intersection are usually estimated by historic traffic records.

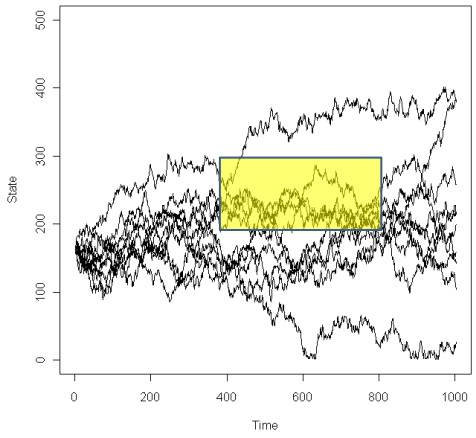


Fig. 3. Some possible worlds of one uncertain object

Definition 6: A Markov Chain is homogeneous if and only if the transition probabilities are independent of t, i.e. $P_{i,j}(t) = P_{i,j}$.

The (single-step) transition matrix $M = P_{i,j} \in \mathbb{R}^{S^2}$ is a *stochastic* matrix, i.e. the following properties hold:

$$\forall i, j \in S : P_{i,j} \geq 0$$

$$\forall i \in S, \sum_{j \in S} P_{i,j} = 1$$

Let $P(o,t)$ be the distribution vector of an object o at time t, such that $(P(o,t) = s_1), \ldots, P(o,t) = s_{|\mathcal{S}|})$, $p_i \in P(o,t)$ corresponds to the probability that o is located at state s_i at time t. The distribution vector of o at time $t+1$ can be inferred from $P(o,t)$ as follows:

Corollary 1:

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

The *m-step transition probability* $P^t_{i,j}$ is the probability that an uncertain object o that is located at state s_i at time t, will be located at state s_j at time $t+m$ and can be computed exploiting the Equations of Chapman-Kolmogorov ([25]) as follows:

$$(P^m_{i,j}) = M^m$$

Given the probability distribution $P(o,t)$ of an uncertain object o at time t, the probability distribution $P(o,t+m)$ of o at time $t+m$ can be computed by

Corollary 2:

$$P(o, t+m) = P(o,t) \cdot M^m$$

In the following, we will model any uncertain object $o \in \mathcal{D}$ by a homogeneous Markov-Chain with an initial probabilistic density function. An instantiation of the object at each point of time is called a *possible world*. Figure 3 depicts a small subset of all possible worlds of an uncertain object for a given Markov-Chain model.

The main challenge in answering probabilistic spatio-temporal queries is to correctly consider the possible world semantics in the model. In other words, the issue at hand is to return the fraction of possible worlds of an object o, for which it holds that at any time $t \in T^{\square}$ it holds that $o(t) \in S^{\square}$. An example is given in Figure 3, where a small subset of the possible worlds of an object o is depicted. Any world, for which the corresponding path intersects the spatio-temporal query window, satisfies the query predicate.

Still, the number of possible worlds is in $O(|S|^T)$; i.e., 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.

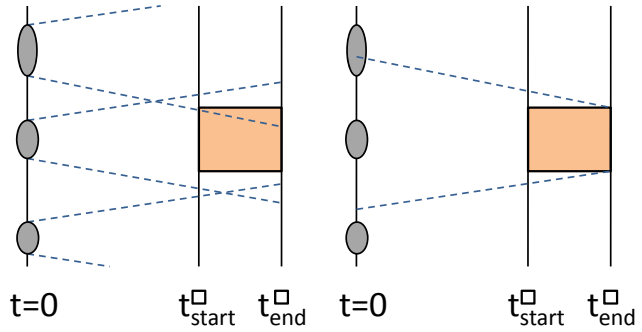
V. Probabilistic Spatio-Temporal Query Processing using the Markov-Chain Model

In this section, we show how the queries that we presented in Section III can be evaluated efficiently. For the ease of presentation, we first assume that for each object, there is a single observation at time $t = 0$ and that we want to predict the result of a probabilistic spatio-temporal exists query (Definition 2) in the future. We generalize our results for the case of multiple observations and other queries in Sections VI and VII, respectively.

We propose two approaches towards efficient query processing. The *object-based approach* (Section V-A) directly computes $P^{\exists}(o, S^{\square}, T^{\square})$ in an efficient way, while the *query-based approach* (V-B) follows a reverse methodology: computation starts at the query window and the transposed Markov-Chain matrix is used to compute, for each state $s \in \mathcal{S}$ the probability that an object starting at s satisfies the query predicate.

Figure 4 shows a high-level example of query evaluation using these two approaches. Consider a spatio-temporal query with query time interval $[t^{\square}_{start}, t^{\square}_{end}]$ illustrated by the shaded rectangle on the right of each subfigure. We would like to predict the result of the query, based on observations about the locations of the objects at time ($t = 0$) and a given model that captures their state transition probabilities (states correspond to spatial locations illustrated by the y-axis in the example). Exemplary objects are shown in oval shapes.

The object-based approach, illustrated in Figure 4(a), examines for each object the possible trajectories (i.e., possible worlds) that the object will follow in the future and finds the probabilities of trajectories that intersect the query window. For instance, the top-most object has a non-zero probability to be a result of the query, the middle object has a higher probability and the object at the bottom has a zero probability. To compute the probabilities of all possible worlds that intersect the window, for each object, we use matrix multiplications. We show how pruning techniques can be incorporated into the matrix multiplications for efficiency. Still, this approach iteratively examines all objects in the database exhaustively. The query-based approach (Figure 4(b)), on the other hand, *reverses* the computation: given that an object intersects the query window, we compute the probability that this object corresponds to any of the objects in the database; this way, examining objects that are irrelevant to the query is avoided, while at the same time the query results are computed in batch. We now describe these two approaches in detail.



(a) Object based approach (b) Query based approach

Fig. 4. Schematic illustration of object-based (OB) and query-based (QB) query processing.

A. Object-Based Query Processing

Given an object o, in order to find the probability that o is part of a query result, we could use the following straightforward approach: compute, for each point of time $t \in T^{\square}$ the probability that o is located in $S^{\square}$ and aggregate these probabilities. Clearly, this approach yields incorrect results (for example, if $|T^{\square}| = 3$ and for each $t \in T^{\square}$,$P(o,t) = 0.5$. The problem of this approach is that a specific possible world (i.e., trajectory) may be considered more than once, if it overlaps with the query at multiple timestamps. Therefore, to correctly process queries, possible worlds which satisfy the query predicate should only be considered once in the computation of the result.

As an example of proper query evaluation, consider the Markov-Chain:

$$\begin{pmatrix} 0 & 0 & 1 \\ 0.6 & 0 & 0.4 \\ 0 & 0.8 & 0.2 \end{pmatrix},$$

for which all possible worlds are depicted in Figure 5(a) for the first four points of time.[1] Additionally, assume a window query defined by $S^{\square} = s_1, s_2\}$ and $T^{\square} = \{2, 3\}$ and assume an object o which has been observed at s_2 at time $t = 0$, i.e. $P(o, 0) = (0, 1, 0)$. To compute the probability $P^{\exists}(o, S^{\square}, T^{\square})$ that o intersects the query window, we first compute the probability distribution $P(o, 2)$ at time $t = 2$, using Corollary 2. The resulting probability vector $P(o, 2) = (0, 0.32, 0.68)$ gives us a lower bound of 32% for $P^{o,\exists}(S^{\square}, T^{\square})$, since any world in which o is located at state s_2 at time $t = 2$ satisfies the query window. Thus, these worlds can already be considered as true hits and must be ignored at future points of time. Thus, we obtain a new probability distribution $P(o, 2)' = (0, 0, 0.68)$ which will be used for the next state transition using Corollary 1. The resulting vector $P(o, 3) = (0, 0.544, 0.136)$ means that out of the remaining 68% of worlds which have not already been reported as true hits, another 54.4% can now be returned as true hits, since in these worlds o is located at s_2 at time $t = 3$. Since $t = 3$ is the last point of time belonging to the query window, the fraction of 0.136 worlds can be reported as true drops, since in these worlds, o has not intersected the query window at any $t \in T^{\square}$. Thus, the result of this query is $0.32 + 0.544 = 0.864$.

The above example gives an intuition on how to answer queries correctly by identifying worlds that satisfy the query and excluding them from further processing. We incorporate this technique directly in the Markov-Chain, to facilitate efficient on-the-fly pruning when matrix multiplication is performed between a state vector and the Markov-Chain. To achieve this goal, we introduce a new state which is denoted by "✓", denoting true hits, i.e. for any world in which an object o reaches ✓, we know that o satisfies the query predicate. This ✓ state satisfies the *absorbing* property, i.e. any world reaching this state cannot leave it. Instead of directly using the Markov-Chain M, we build the following two new matrices

$$M^{-} = \begin{pmatrix} M & zero(|S|) \\ zero(|S|)^T & 1 \end{pmatrix}$$

and

$$M^{+} = \begin{pmatrix} M' & sum(S^{\square}) \\ zero(|S|) & 1 \end{pmatrix},$$

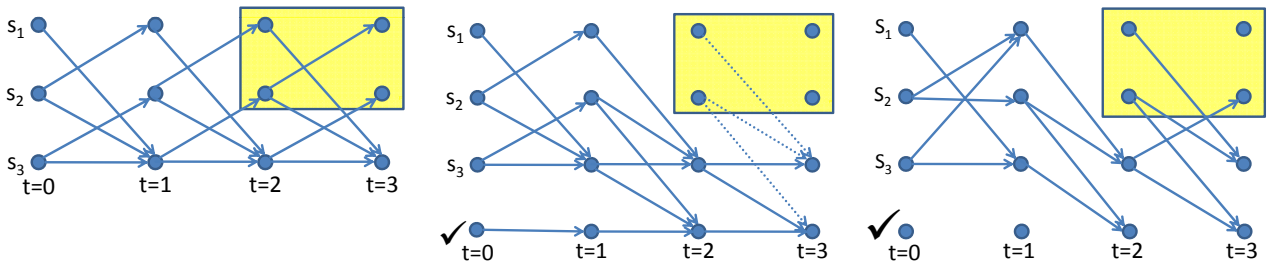
where $zero(|S|)$ is a vector of size S containing all zeroes, $zero(|S|)^T$ is its transposed, M' is derived from M by replacing all columns that correspond to states in $S^{\square}$ by zero vectors, and $sum(S^{\square})$ is a column vector containing for each line in M the sum of values removed this way.

The initial object distribution vector of an object o is now extended by an additional value of zero, corresponding to the fact that initially (at time $t = 0$) we cannot identify any worlds which satisfy the query predicate.[2] At each state transition, where the target state does not belong to $T^{\square}$, we can now use M^{-} instead of M, which has the same effect as M, while preserving the probability of state ✓. If the target state belongs to $S^{\square}$, then M^{+} is used instead. This way, worlds leading into states in $S^{\square}$ are now redirected to state ✓ instead.

Example 1: For the matrix $\begin{pmatrix} 0 & 0 & 1 \\ 0.6 & 0 & 0.4 \\ 0 & 0.8 & 0.2 \end{pmatrix}$ of our running example, the corresponding new matrices are $M^{-} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0.6 & 0 & 0.4 & 0 \\ 0 & 0.8 & 0.2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ and $M^{+} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0.4 & 0.6 \\ 0 & 0 & 0.2 & 0.8 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ The corresponding visualization is depicted in Figure 5(b). Here M^{-} is used for the first transition from $t = 0$ to $t = 1$, while for the transitions from $t = 1$ to $t = 2$ and from $t = 2$ to $t = 3$, M^{+} is utilized as transition matrix. Thus, for an object that has been observed at state s_2 at time $t = 0$, we obtain the initial probability distribution vector $P(o, 0) = (0, 1, 0, 0)$, where the fourth value denotes the initial probability of being a true hit, which is zero since $t = 0$ does not belong to $T^{\square}$. The transition to $t = 1$ yields $P(o, 1) = P(o, 0) \cdot M^{-} = (0.6, 0, 0.4, 0)$. Clearly, the fourth value corresponding to state ✓ is zero, since no state $t \in T^{\square}$

[1] Probabilities are omitted for readability, but can be found in the Markov-Chain.

[2] In the special case where $t = 0$ belongs to $T^{\square}$, we adjust the initial vector by moving all probabilities of states in $S^{\square}$ to state ✓.



(a) All possible worlds (b) Object-based approach (c) Query-based approach

Fig. 5. Procedure of object-based (OB) and query-based (QB) query processing.

has been visited so far. Transition to $t = 2$ yields $P(o, 2) = P(o, 1) \cdot M^+ = (0, 0, 0.64, 0.36)$ and the final transition yields $P(o, 3) = P(o, 2) \cdot M^+ = (0, 0, 0.136, 0.864)$. Therefore, the resulting probability that o intersects the query window is 0.864.

B. Query-Based Query Processing

The object-based approach applies for each object o the methodology described in Section V-A to compute the probability that o is part of the query result. The query-based approach assumes that an object intersects the query. Based on this assumption, this approach starts at the last point of time in the query window, and goes backward in time using the transposed Markov-Chain. This way, we can compute, at any time t, the probability vector $P(t)$ containing for each state s, the probability that an object starting at state s at time t will satisfy the query predicate.

Therefore, we start at time $t^{\square}_{end} := max(T^{\square})$. Clearly, at $t^{\square}_{end}$, a path satisfies the query predicate if and only if it is in state ✓: If it is not in state ✓ at time $t^{\square}_{end}$, then it will never reach state ✓, since at any time after $t^{\square}_{end}$, the matrix M^- will be used which does not allow to enter state ✓. Thus, the vector $P(t^{\square}_{end})$ has the form $(0, ..., 0, 1)$. Intuitively, this vector corresponds to the assumption, that a path satisfies the query. Now we go back in time using this assumption by using the transposed matrices $(M^-)^T$ and $(M^+)^T$: If the current state belongs to $S^{\square}$, we use $(M^+)^T$, otherwise, we use $(M^-)^T$. This procedure is repeated until $t = 0$, the last point of time at which an object o has been observed, is reached. The resulting vector v yields, for each state $s \in \mathcal{S}$, the probability that an object starting at s (with a probability of one) satisfies the query. Vector multiplication of the probability distribution $P(o, 0)$ with v yields the result probability $P^{\exists}(o, S^{\square}, T^{\square})$.

Example 2: Again, consider the example depicted in Figure 5(c), where we want to compute the probability that an object o intersects the query $S^{\square} = \{s_1, s_2\}, T^{\square} = \{2, 3\}$. By transposing M^- and M^+, we obtain:

$$(M^-)^T = \begin{pmatrix} 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0.8 & 0 \\ 1 & 0.4 & 0.2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ and } (M^+)^T = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0 & 0 \\ 1 & 0.4 & 0.2 & 0 \\ 0 & 0.6 & 0.8 & 1 \end{pmatrix}.$$

The query-based approach starts by assuming the probability distribution $P(t = 3) = (0, 0, 0, 1)$. Since $t = 3 \in T^{\square}$, we compute

$$P(t = 2) = P(t = 3) \cdot (M^+)^T = (0, 0.6, 0.8, 1).$$

Since $t = 2 \in T^{\square}$ as well, we repeat this computation:

$$P(t = 1) = P(t = 2) \cdot (M^+)^T = (0.8, 0.92, 0.96, 1).$$

Finally, since $t = 1 \notin T^{\square}$, we get

$$P(t = 0) = P(t = 1) \cdot (M^-)^T = (0.96, 0.864, 0.928, 1)$$

This vector contains, for each state, the probability that an object started at this position will intersect the query. Let us again, assume that initially, the object is located at s_2, i.e. $P(o, 0) = (0, 1, 0, 0)$ we finally obtain:

$$P^{\exists}(o, S^{\square}, T^{\square}) = P(o, 0) \cdot P(t = 0)^T = 0.864.$$

This result equals the result that we derived using the object-based approach.

C. Discussion

The advantage of the query-based approach is that we only have to compute $P(t = 0)$ once, and then compute, for each object o the $P^{\exists}(o, S^{\square}, T^{\square})$ by one single vector multiplication, which can be performed in $O(|P(o, 0)|)$, where $|P(o, 0)|$ is the number of non-zero elements in $P(o, 0)$, i.e. the number of possible positions of o at $t = 0$. In particular, if we assume that the number of possible states observed at $t = 0$ is small (which is realistic even for inaccurate observation types),we approach a total CPU cost of $O(1)$ per object. The initial computation of $P(t = 0)$ has to perform time-transitions using the transposed Markov-Chain. Thus, a vector-matrix multiplication is required for each transition from $t^{\square}_{end}$ to $t = 0$. Thus, the total runtime of the query-based approach is $O(|\mathcal{D}| + |S_{reach}|^2 \cdot \delta t)$, where $|\mathcal{D}|$ is the number of database objects, $|S_{reach}|$ is the number of states that have to be explored, and δt is the number of transitions between $t = 0$ and $t^{\square}_{end}$.

In contrast, the object-based approach has to perform time-transitions using the Markov-Chain *for each object*. Although it is possible in some cases to stop these transitions early using the inherent true-hit detection (computation can be stopped as soon as the probability of state ✓ becomes sufficiently large), in the worst case, all transitions from $t = 0$ to $t^{\square}_{end}$ have to be performed, and for each such transition, a vector-matrix multiplication has to be performed in $O(|S_{reach}|^2)$ time, where $|S_{reach}|$ is the total number of states reachable by o in the time interval $[0, t^{\square}_{end}]$. The total runtime of the object-based approach is thus $O(|\mathcal{D}| \cdot |S|^2 \cdot \delta t)$.

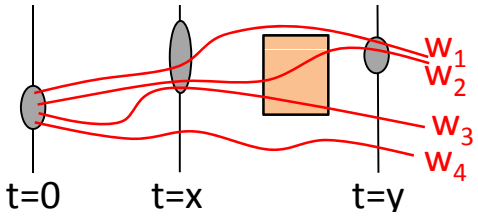


Fig. 6. Multiple observations of an object

Nevertheless, the query-based approach makes the assumption that all objects follow the same model, i.e. all have the same Markov-Chain. In many applications, this assumption is reasonable: The movements of all icebergs are subject to the same currents - the form and shape of an iceberg can be assumed to have negligible impact on the icebergs movement. In road networks, objects may indeed follow different models. In the worst case, we may have to perform the query-based approach once for each object, to compute $P(t=0)$ for it. If the objects can be partitioned to classes (e.g. buses, trucks and cars), the query-based approach can naturally be applied once for each class. In general, if objects follow different Markov-Chains, a technique to speed up the query-based approach is to cluster objects with similar Markov-Chains, and represent each cluster by one approximated Markov-Chain, where each entry is a probability interval instead of a singular probability. This approximated Markov-Chain can be used to perform pruning by detecting clusters of objects which must have (or cannot possibly have) a sufficiently high probability to satisfy the query predicate. Only clusters which cannot be decided as a whole need their objects to be considered individually.

VI. Multiple Observations

So far, we have assumed that there is only one observation per object and that this observation happened at a time that (temporally) proceeded the query time. In this section, we first show how to compute $P^{\exists}(o, S^{\square}, T^{\square})$ given two observations, one before query time and one after query time. Abstractly, this approach can be seen as a time-interpolation, whereas so far we have only considered time-extrapolation. The aim is to incorporate knowledge of both observations, in order to exclude all worlds which are not possible, given both observations, and properly re-weight the probabilities of the remaining worlds. Our proposed technique is applicable for both the object-based as well as the query-based approach. Later, we will give an intuition why considering additional observations, which are further apart from the query window than a considered observation, may still provide additional information. Finally, we outline how these techniques can be easily adapted to incorporate information from an arbitrary number of observations.

Given multiple observations, we have to distinguish between three classes of worlds:

A Worlds that become impossible due to the given observations,
B possible worlds that satisfy the query predicate and
C possible worlds that do not satisfy the query predicate.

The example in Fig. 6 shows 4 possible worlds (trajectories) of an object o observed at a point of time t_x. In addition, two further observations of o exists at time t_y and t_z. The possible locations of o at each time of the observations are illustrated by the ellipses. Due to the observation made at t_z, worlds w_3 and w_4 become impossible because they include impossible states at time t_z, i.e. both worlds belong to class A. In contrast, w_2 belongs to class B, since this world has a non-zero probability since it includes only possible states at all three observations and satisfies the query predicate of intersecting the query window. Finally, w_1 belong to class C, since, albeit possible, it does not intersect the window.

Intuitively and according to the possible worlds semantics, the probability P_{total} that an object satisfies the query predicate, given some observations, is the fraction of possible worlds that satisfy the query predicate, i.e. the fraction

$$P_{total} = \frac{P(B)}{P(B) + P(C)}. \tag{1}$$

In the following, we show how the object-based approach can be adapted to consider multiple observations $OBS^o = \{obs_1^o, \ldots, obs_n^o\}$ of the same object o. Each observation obs_x^o is given by a time $t_{obs_x^o} \in \mathcal{T}$ and a probability distribution $P_{obs_x^o}$ representing the observation. Then, the derived matrices M^- and M^+ can be used for the query-based approach. The approach of Section V-A cannot be applied directly, because now, worlds which have reached the query window are no longer equivalent, and can no longer be unified in a single state ✓. The reason is that the current state of such a world now effects the probability of reaching the state observed at time $t = 3$. Therefore, we need to maintain, for each world that has intersected the query, information about its current state at each time. Therefore, we replace the state ✓ by a set of states s_1✓,...,s_3✓. Each state s_i✓ corresponds to the probability, that o has intersected the query window *and* is currently located in state s_i. The transition matrices M^- and M^+ need to be adjusted accordingly. The shape of M^- is clear: For a transition from t to $t+1$, where $t+1 \notin T^{\square}$ (the case where M^- is used), states are simply transitioned, and worlds which have (not) intersected the query at t must (not) have done so at $t+1$. We obtain:

$$M^- = \begin{pmatrix} M & 0 \\ 0 & M \end{pmatrix}$$

In the case where M^+ is used, that is the case where $t+1 \in T^{\square}$, we have to ensure that any transition to a state $s \in S^{\square}$ leads to the corresponding state s✓. This yields the matrix

$$M^+ = \begin{pmatrix} M - M' & M' \\ 0 & M \end{pmatrix},$$

where M' is derived from M by setting all columns to zero, where the corresponding state $s \notin S^{\square}$, and $M - M'$ is derived by setting all columns to zero for which $s \in S^{\square}$.

We start by using the probability distribution $P_{obs_1^o}$ of an object o observed at $t_{obs_1^o}$. As in the previous sections, we iteratively apply the modified matrices M^- and M^+

until we reach $t_{obs_2^o}$. At this point, we have two probability distributions: One distribution derived using the observation obs_1^o, which has been transitioned to $t_{obs_2^o}$, as well as the probability distribution $P_{obs_2^o}$. These observations can be unified by exploiting independence between observations.

Lemma 1: Let $X(t) := \{P_{obs_1^o}(t), \ldots, P_{obs_n^o}(t)\}$ be a set of pdfs of an object o at time t, derived from independent observations. The joint probability distribution $P(o,t)$ of o at time t is given by:

$$P(o,t) = N(\prod_{x \in X} x_1, ..., \prod_{x \in X} x_{|\mathcal{S}|}),$$

where $N(\cdot)$ is the vector normalization function, i.e. $N(x) = (\frac{x_1}{\sum x}, ..., \frac{x_{|\mathcal{S}|}}{\sum x})$

Proof: For each $i \in 1, ..., |\mathcal{S}|$, $P(o,t)_i$ is, by definition of a probability density function, the probability of the random event that object o is located in state s_i. Without loss of generality, let $a = (a_1, ..., a_{|\mathcal{S}|}) \in X$ be the first observation. Given this observation only, then clearly $P(o,t) = a$ holds. Given further observations, the probability of this event becomes conditioned to

$$P(o,t)_i = P(a_i | X \setminus \{a\}).$$

Since all observations are mutually independent, we get

$$P(o,t)_i = a_i \cdot \prod_{x \in X} x_i,$$

which is the fraction of all worlds, including worlds which are no longer possible given the observations X, in which o is located in state s_i at time t. Due to possible worlds semantics (c.f. Equation 1), we are only interested in the fraction of possible worlds in which o is located in state s_i at time t. Since v contains all possible worlds, the normalization $N(v)$ yields the correct result. ∎

As an example, consider Figure 7, where we again use our running example Markov-Chain

$$M = \begin{pmatrix} 0 & 0 & 1 \\ 0.5 & 0 & 0.5 \\ 0 & 0.8 & 0.2 \end{pmatrix}$$

and assume that an object O has been observed at state s_1 at time t_1 and at state s_3 at time t_4. We obtain the transition matrices

$$M^- = \begin{pmatrix} M & 0 \\ 0 & M \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0.8 & 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0.8 & 0.2 \end{pmatrix}$$

and

$$M^+ = \begin{pmatrix} M - M' & M' \\ 0 & M \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.2 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0.8 & 0.2 \end{pmatrix}$$

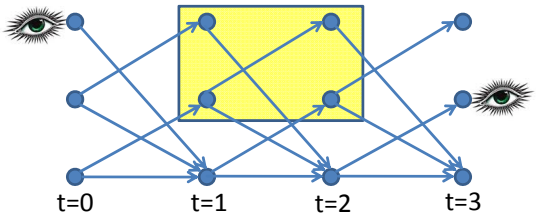


Fig. 7. Two observations of an object

Since at time $t = 0$, o has been observed in state s_1, and since o cannot have reached the window yet, the initial distribution of o is $P_{obs_1^o} = P(o,0) = (1,0,0,0,0,0)$. Transition to $t = 1$ using M^+ (since $t = 1 \in T^\square$) yields $P(o,1) = (0,0,1,0,0,0)$. The next transition using M^+ yields $P(o,2) = (0,0,0.2,0,0.8,0)$. The intuition of this vector is that, at time $t = 2$, object o is located in state s_3 while having reached the query window with a probability of 20%, and otherwise is located in state s_2 having reached the query window. The next transition uses M^- (since $t = 3 \notin T^\square$) and yields $P(o,3) = (0, 0.16, 0.04, 0.4, 0, 0.4)$. Now, at time $t = 3$, the second observation was made. We assume that this observation only has information about the state at $t = 3$, but no information whether o has intersected the query window. Thus, the observation vector has the form $obs_2^o = (0, 0.5, 0, 0, 0.5, 0)$. Due to Lemma 1, and since we assume that obs_1^o (from which $P(o,3)$ was derived) and obs_2^o are independent observations, we can directly multiply the entries of $P(o,3)$ and obs_2^o, yielding $P(o,t)\prime = (0, 0.08, 0, 0, 0, 0)$. Normalization yields $P(o,t) = N(P(o,t)\prime) = (0,1,0,0,0,0)$, which means that at $t = 3$, given both observations, o must be in state s_2 and must not have intersected the query window. This is intuitive, since the only path between s_1 at $t = 0$ and s_2 at $t = 3$ does not intersect the query window.

VII. Additional Spatio-Temporal Queries

Based on the proposed concepts for answering spatio-temporal ∃-queries, we will now show how different query predicates can be answered efficiently.

PST∀Q:: In some applications, it may be interesting to compute the probability that o is in the query window at *all* times $t \in T^{Box}$. Clearly, the probability that o is located in $S^\square$ at all times in $T^\square$ complements the probability that o is outside $S^\square$ at any time in $T^\square$, i.e.

$$P^\forall(o, S^\square, T^\square) = 1 - P^\exists(o, \mathcal{S} \setminus S^\square, T^\square).$$

Although, in general, $\mathcal{S} >> S^\square$, the time required to compute $P^\exists(o, \mathcal{S} \setminus S^\square, T^\square)$ is generally not larger than the time required to compute $P^\forall(o, S^\square, T^\square)$. The only difference between these two computations is the content of the matrix M^+, since different sets of columns of matrix M are merged. In most cases, the computation of $P^\exists(o, \mathcal{S} \setminus S^\square, T^\square)$ is actually faster, since more columns of M^+ are zero.

PSTkQ:: So far, an object o satisfies the query predicate in any world where it intersects the query window, regardless for how long o remains in the query window. In the following, we will show how the probability distribution of the number

of times that o is located in the query window (c.f. Definition 4) is computed.

To answer this query, we can extend the idea of adding new virtual states, to capture the number of times an object has visited the query window; we use the set of states $\mathcal{S}' = \mathcal{S} \times \{0, ..., |T^\square|\}$. Intuitively, an object in state $s' = (s \in \mathcal{S}, k \in \{0, ..., |T^\square|\}) \in \mathcal{S}'$ is currently located in state s and has been in the query window at k points of time. At any point of time $t \in T^\square$, all possible worlds located at a state $s \in S^\square$ are transitioned in order to increase their k value by one. The resulting matrices are

$$M^- = \begin{pmatrix} M & 0 & ... & 0 \\ 0 & M & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & M \end{pmatrix}$$

$$M^+ = \begin{pmatrix} M - M' & M' & 0 & ... & 0 \\ 0 & M - M' & M' & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & ... & M - M' & M' \end{pmatrix}$$

Since initially, an object o visited the query window zero times[3], the initial distribution of an object is concatenated with $k \cdot |\mathcal{S}|$ zeroes. If we perform time transitions until we reach $t^\square_{end}$, we obtain for each $s' = (s \in \mathcal{S}, k \in \{0, ..., |T^\square|\}) \in \mathcal{S}'$ the probability that o will visit the query window exactly k times and will finally be in state s. Grouping this result by k, we obtain for each $k \in \{0, ..., |T^\square|\}$, the probability that o visits the query window exactly k times.

Obviously the above approach is not very memory efficient, since M^- and M^+ each blow up the memory requirement of the original transition matrix M by a factor of $|T^\square|$. Thus, we suggest a more memory efficient algorithm for this problem. Due to space limitations we will give a brief description here. For processing an object $o \in \mathcal{D}$, an additional $(|T^\square|+1) \times |\mathcal{S}|$-Matrix $C(t)$ is necessary. Each entry $c_{i,j}(t) \in C(t)$ corresponds to the probability that o is currently located in state s_j and has been located in $S^\square$ at exactly i points of time $t' \leq t \in T^\square$. We begin by setting the first row of $C(0)$ to $P(o, 0)$ and all other entries to zero, since at $t = 0$, o cannot possibly have entered the query region. At each state transition from t to $t+1$, a transition using M is performed for each row. This is simply achieved by computing $C'(t+1) = C(t) \cdot M$. If t is not in $T^\square$, then we set $C(t+1) = C'(t+1)$. Otherwise, if $t \in T^\square$, then we additionally shift each column corresponding to a state $s_i \in \mathcal{S}$ down by one, and fill the top entry of this line with zero. That is

$$c_{i,j} = \begin{cases} c'_{i,j}, & \text{if } j \notin S^\square \\ 0, & \text{if } j \in S^\square \wedge i = 0 \\ c'_{i-1,j} & \text{otherwise.} \end{cases}$$

When $t^\square_{end}$ is reached, the probability for o to be in the query exactly k-times can be obtained by summing up all probabilities in row k of $C(t^\square_{end})$. Thus the probability

[3]The special case where $t = 0$ belongs to $\mathcal{T}^\square$ is omitted. In that case, the initial distribution of an object simply starts at $k = 1$, i.e. is shifted by $|\mathcal{S}|$.

parameter	value range	default
$\|\mathcal{D}\|$	1,000 - 100,000	10,000
$\|\mathcal{S}\|$	2,000 - 100,000	100,000
$object\_spread$	5	5
$state\_spread$	1 - 20	5
max_step	10 - 100	40

TABLE I

PARAMETERS FOR THE SYNTHETIC DATASETS

$P^{k-times}(o, S^\square, T^\square)$ that o has visited the query window exactly k times is given by

$$P^{k-times}(o, S^\square, T^\square) = \sum_j c_{k,j}(t^\square_{end})$$

Considering our running example we start with the matrix $C(0)$ and transition as along as $t \notin T^\square$:

$$C(0) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot M^2} C(2) = \begin{pmatrix} 0 & 0.32 & 0.68 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Now we shift down the probabilities of the states in $S^\square$ by one row: $C(2) = \begin{pmatrix} 0 & 0 & 0.68 \\ 0 & 0.32 & 0 \\ 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot M} C(3) = \begin{pmatrix} 0 & 0.544 & 0.136 \\ 0.192 & 0 & 0.128 \\ 0 & 0 & 0 \end{pmatrix}$

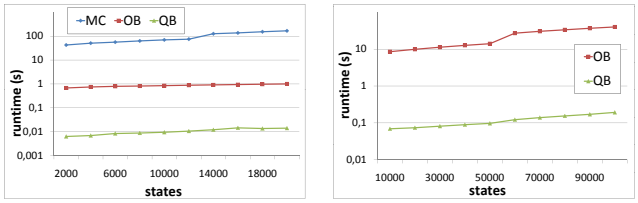
After performing the last shift we obtain $C(3) = \begin{pmatrix} 0 & 0 & 0.136 \\ 0 & 0.544 & 0.128 \\ 0.192 & 0 & 0 \end{pmatrix} \xrightarrow{rowsum} \begin{pmatrix} 0.136 \\ 0.672 \\ 0.192 \end{pmatrix}$. The resulting vector reflects the probability of o to be in the query exactly 0 (0.136), 1 (0.672) and 2 (0.192) times.

VIII. EVALUATION

A. Experimental Setup

For our experimental evaluation we used synthetic as well as real datasets. In order to observe the influence of data characteristics we used several parameters for the construction of the synthetic datasets. Each experiment is performed using a database of $|\mathcal{D}|$ objects. The location of each object at time t_0 is given by a PDF over a certain number of states. This value is controlled by the parameter $object\_spread$ and characterizes the amount of uncertainty in the database. The total number of states of the system is characterized by $|\mathcal{S}|$. To control the density of the transition matrix (which corresponds to the number of possible transitions in the modeled system) we used the parameter $state\_spread$. From each state it is possible to transition into $state\_spread$ states. To model locality of the transitions within the system we also introduced a parameter which bounds the possible states which can be reached by one transition. An object in state s_i can only transition into states $s_j \in [s_{i-max\_step/2}; s_{i+max\_step/2}]$. All parameters for the synthetic datasets are summarized in Table I.

As real datasets we used two road network datasets. The first is the road network of North America which consists of 175,813 nodes and 179,102 edges. As this is a rather sparse graph we also extracted the road network from Munich which has 73,120 nodes and 93,925 edges. The transition matrix is



(a) Small state space. (b) Large state space.

Fig. 8. Query processing runtime w.r.t. the number of states.

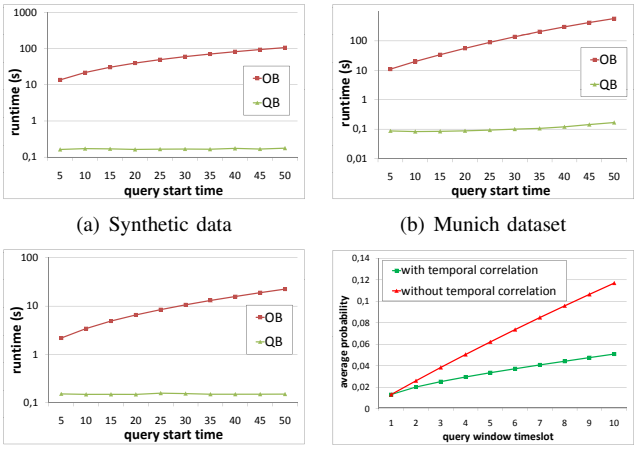
equivalent to the adjacency matrix of the corresponding graph. This means each node is treated as a state and each edge corresponds to two non-zero entries in the transition matrix. The value of the non-zero entries of one line in the matrix are set randomly and sum up to one. In this way they reflect the transition probabilites from one node in the road network to its directly connected neighbors.

In our experiments we evaluate object-based (**OB**) and query-based (**QB**) query processing using several query predicates ($\exists, \forall$ and k-count). Additionally we compare these approaches to a Monte-Carlo based method (**MC**). The **MC** approach samples paths of each object and outputs the fraction of the sampled paths which fulfill the query predicate. Sampling the path of an object requires first drawing a start state from the objects distribution. Afterwards for each timestep a state from the successor states of the current state is chosen according to the probability distribution given by the transition matrix. Note that **MC** only returns approximate results, where the accuracy can be improved if more paths are sampled. Since the sampling of paths is equivalent to a Bernoulli sequence, the standard deviation between the sampled probability ($\hat{p}$) and the true probability (p) is given by $\sigma_{\hat{p}} = \sqrt{\frac{p \cdot (1-p)}{n}}$. For 100 samples, the standard deviation between p and $\hat{p}$ is thus at least 5% and gets worse for small and large values of p.

All experiments were run on a single 64-Bit machine with an Intel Xeon 5160 processor with 3.0 GHz and 32GB of RAM. The computations where performed using MATLAB R2011a. Unless mentioned otherwise, we generated 10,000 objects randomly distributed across the state space and the query window is defined by the states [100, 120] and time interval [20, 25]. For the Monte-Carlo based approach the number of drawn samples was set to 100.

B. Experiments

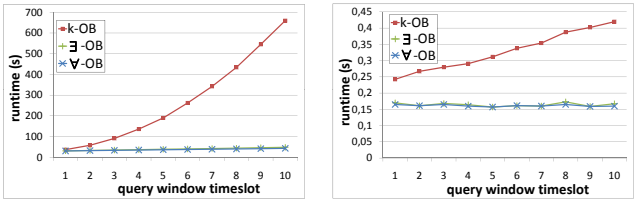
In the first experiment, we vary the size of the state space $|\mathcal{S}|$ and measure the cost of query evaluation for a PST$\exists$Q. In Figure 8(a), we used a relatively small synthetic dataset ($|\mathcal{D}| = 1,000$, $|\mathcal{S}| = [1,000; 10,000]$). The **MC** approach is computationally very demanding in comparison to the other two algorithms. The reason is, that even for such a small setting the Monte-Carlo based approach has to draw a very high amount of samples. Note that already for a small number of sampled paths (we used 100 in this setting) this approach becomes expensive, because the sampling of one path already requires to draw as many samples as the considered stamps in the query. This corresponds to 2,500 samples for one object



(a) Synthetic data (b) Munich dataset

(c) NA dataset (d) accuracy experiment

Fig. 9. Query processing runtime w.r.t. the size of the query time frame



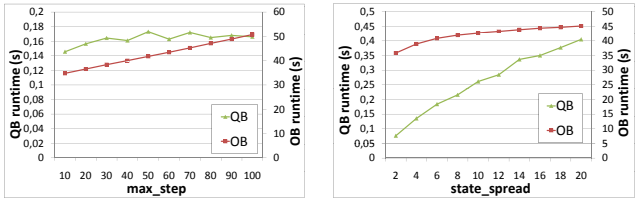
(a) OB approach (b) QB approach

Fig. 10. Performance evaluation of query predicates.

in the database. Due to these high costs we excluded the **MC** algorithm from the remaining experiments. As expected the **QB** approach is much faster than the **OB** approach. Figure 8(b) shows how these two methods scale at larger datasets ($|\mathcal{D}| = 100,000$, $|\mathcal{S}| = 10,000$).

The experiments shown in Figures 9(a) and 9(b) show the dependency of the PST$\exists$Q algorithms on the timeslot we want to query on synthetic and real data sets. The runtime of **OB** is increasing much faster than the runtime of **QB**. As expected the runtimes of both algorithms suffer from a longer glance in the future as the vectors to be multiplied become less sparse with each time stamp. Besides this, **QB** should not be influenced by the number of timestamps whereas the runtime of the **OB** approach scales linearly to that parameter. To justify the used model, we also performed an experiment which compared our model to a model where temporal correlations are ignored (cf. Figure 9(d)) w.r.t. accuracy. In this experiment we posed PST$\exists$ queries with an increasing query time window and measured the average probability of objects having a non-zero probability to fulfill the query predicate. It is clearly visible that ignoring the temporal dependency returns a biased result and the error compared to the correct result even increases for larger query windows.

In Figure 10(a) we compare the three proposed query types PST$\exists$Q, PST$\forall$Q and PSTkQ query using the object-based approach. Obviously for the PSTkQ we have to maintain not only one but multiple vectors (as many as the number of times in $T^{\square}$) per object, which leads to an increased runtime. The PST$\exists$Q and PST$\forall$Q had equal runtime in all experimental



(a) Impact of max_step (b) Impact of state_spread

Fig. 11. Comparison of QB and OB behavior with scaling parameters

settings. Using the **QB** approach all queries run in a fraction of a second and the runtime of PSTkQ seems to scale rather linearly with k (cf. Figure 10(b)) .

In the next set of experiments, the runtime behavior of the two approaches w.r.t. different locality parameters is tested. Figure 11(a) shows the runtime for increasing the max_step parameter whereas Figure 11(b) shows results for increasing $state\_spread$ parameter (Note that the algorithm runtimes are marked at different axes). Both algorithms scale at most linearly with those parameters.

IX. Conclusion

In this paper, we studied the problem of probabilistic query evaluation over uncertain spatio-temporal data. We consider uncertain trajectories, for which some points are sampled via observations, while the remaining points are instantiated by a stochastic process. To our knowledge, this is the first paper that studies such queries over uncertain moving object data, which are modeled by stochastic processes, specifically Markov-Chains. This approach has three major advantages over previous work. First it allows answering queries in accordance with the possible worlds model. Second, dependencies between object locations at consecutive points in time are taken into account. And third, it allows us to infer the probability an object reaches a certain location (state) by matrix multiplications that can be processed extremely efficient. Based on this method we propose a framework for processing queries over such data, which injects pruning techniques into the Markov Chain matrices. An object-based and a query-based approach are proposed; the latter is always more efficient, typically by orders of magnitude. In our experiments we show that we are able to answer queries on settings with 100,000 location states and 10,000 objects in a fraction of a second on a single machine in contrast to state-of-the-art solutions that are multiple orders of magnitude slower, e.g. the Monte-Carlo approach requires several hours for the same query. We show how the framework can be applied for the cases where there exist one or multiple observations per object and for various probabilistic spatio-temporal query variants.

We believe that many more interesting queries and applications can be set on top of this model. To support this we release the MATLAB framework which was developed during the process of this paper online[4]. In the future, we plan to apply our framework for data analysis tasks over spatio-temporal data (e.g. find areas that are expected to become congested together with the time periods of this expectation).

[4]The project page can be found at http://www.dbs.ifi.lmu.de/cms/Publications/UncertainSpatioTemporal

X. Acknowledgements

This work was supported by a grant from the Germany/Hong Kong Joint Research Scheme sponsored by the Research Grants Council of Hong Kong (Reference No. G HK030/09) and the Germany Academic Exchange Service of Germany (Proj. ID 50149322)

References

[1] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. Vitter, "Efficient indexing methods for probabilistic threshold queries over uncertain data," in *Proc. VLDB*, 2004.

[2] 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*, 2005.

[3] H.-P. Kriegel, P. Kunath, M. Pfeifle, and M. Renz, "Probabilistic similarity join on uncertain data," in *Proc. DASFAA*, 2006.

[4] C. Böhm, A. Pryakhin, and M. Schubert, "The Gauss-tree: Efficient object identification of probabilistic feature vectors," in *Proc. ICDE*, 2006.

[5] Y. Iijima and Y. Ishikawa, "Finding probabilistic nearest neighbors for query objects with imprecise locations," in *Proc. MDM*, 2009.

[6] G. Cormode, F. Li, and K. Yi, "Semantics of ranking queries for probabilistic data and expected results," in *Proc. ICDE*, 2009.

[7] T. Bernecker, H.-P. Kriegel, N. Mamoulis, M. Renz, and A. Züfle, "Scalable probabilistic similarity ranking in uncertain databases," *IEEE TKDE*, vol. 22, no. 9, pp. 1234–1246, 2010.

[8] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, "Querying imprecise data in moving object environments," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1112–1127, 2004.

[9] G. Trajcevski, R. Tamassia, H. Ding, P. Scheuermann, and I. F. Cruz, "Continuous probabilistic nearest-neighbor queries for uncertain trajectories," in *EDBT09*, 2009.

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

[11] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez, "Indexing the positions of continuously moving objects," in *Proc. SIGMOD*, 2000.

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

[13] Y. Tao, C. Faloutsos, D. Papadias, and B. Liu, "Prediction and indexing of moving objects with unknown motion patterns," in *Proc. SIGMOD*, 2004.

[14] G. Beskales, M. Soliman, and I. Ilyas, "Efficient search for the top-k probable nearest neighbors in uncertain databases," *PVLDB*, vol. 1, pp. 326–339, 2008.

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

[16] G. Trajcevski, O. Wolfson, F. Zhang, and S. Chamberlain, "The geometry of uncertainty in moving objects databases," in *EDBT02*, 2002.

[17] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain, "Managing uncertainty in moving objects databases," *ACM Trans. Database Syst.*, vol. 29, no. 3, pp. 463–507, 2004.

[18] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, "Evaluating probabilistic queries over imprecise data," in *SIGMOD03*, 2003.

[19] J. Aßfalg, H.-P. Kriegel, P. Kröger, and M. Renz, "Probabilistic similarity search for uncertain time series," in *Proc. SSDBM*, 2009.

[20] 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*, 2009.

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

[22] S. Qiao, C. Tang, H. Jin, T. Long, S. Dai, Y. Ku, and M. Chau, "Putmode: prediction of uncertain trajectories in moving objects databases," *Appl. Intell.*, vol. 33, no. 3, pp. 370–386, 2010.

[23] C. Ré, J. Letchner, M. Balazinska, and D. Suciu, "Event queries on correlated probabilistic streams," in *Proc. SIGMOD*, 2008, pp. 715–728.

[24] S. Karlin and H. M. Taylor, *A First Course in Stochastic Processes*. Academic Pr Inc, 1975, vol. 2.

[25] A. Papoulis, *Probability, Random Variables, and Stochastic Processes, 2nd ed.* McGraw-Hill, 1984.