Managing Evolving Uncertainty in Trajectory Databases

Hoyoung Jeung, Hua Lu, Saket Sathe, Man Lung Yiu

Abstract—Modern positioning technologies enable collecting trajectories from moving objects across different locations over time, typically containing time-varying measurement errors of positioning systems. Unfortunately, current models on uncertain trajectories are incapable of capturing dynamically changing uncertainty in trajectory data, as well as lacking the support of recent progress made in improving localization accuracy. In order to tackle these problems, we address three important issues centric to uncertain trajectory management. First, we propose a flexible trajectory modeling approach that takes into account model-inferred actual positions, time-varying uncertainty, and nondeterministic uncertainty ranges. Second, we develop three estimators that effectively infer evolving densities of trajectory data. Last, we present an efficient mechanism to evaluate probabilistic range queries on those evolving-density trajectories. Empirical results on two large-scale real datasets demonstrate the quality and efficiency of our approach.

Index Terms—H.2.4.h Query processing; H.2.8.o Spatial databases and GIS; G.3.h Probabilistic algorithms

1 Introduction

Uncertainty management is a central issue in trajectory databases. The research interests, optimization goals, and methodologies in this domain are indeed rich and diverse [14], [34], [33], [12], [21], [39], [11], [10], [20]. Despite this diversity, these studies are generally established upon a common principle—*location uncertainty is captured by a certain range centered on the position recorded in the database*. This principle was initially discussed by Pfoser and Jensen [27] in the database literature, which is longer than a decade ago.

This paper reconsiders this principle. As of today, GPS is no longer the only primary means for positioning, yet a wide spectrum of technologies [22] are being used to produce trajectory data, including RFID sensors, location estimation with 802.11, smart-phone sensors, infrared and ultrasonic systems, GSM beacons, and even vision sensors. These positioning systems typically yield different characteristics of trajectory data, which also exhibit various properties as well as degrees of uncertainty.

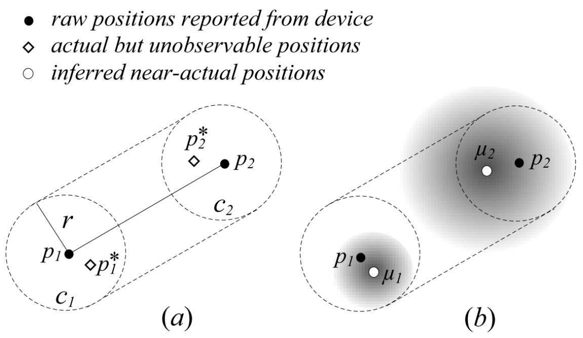
In this paper, we claim that the principle serving as the basis for the uncertain trajectory modelings is incapable of effectively capturing various types of uncertainty caused from different positioning sources. Our claim is based on three key observations:

1. A location reported from a positioning system already bears some positional error. This implies that the exact position to which an object has actually been may not be identical to the reported position. The actual location is typically unobservable, but may be possible to infer a near-actual position by employing various advanced mechanisms, such as filtering [35], smoothing [22], and sensor fusion [13]. This suggests that the uncertainty range should not be centered on the reported location but on the (inferred) actual location.

2. Positional errors may vary over time, e.g., GPS accuracy generally increases or decreases according to the presence of obstacles, such as tunnels and tall buildings. The accuracy of WiFi-based location estimation is also subjective to signal strengths available. As a result, the uncertainty of trajectory data should also change along time.

3. Bounding an area of uncertainty may cause loss of information. Gaussian distributions, for example, are commonly used to model the error distributions of positions, e.g., GPS logs [27], RFID positions [35], WiFi-based localization [22], and GSM-phone positioning [4], which are essentially unbounded. Thus, it is inevitable to miss out some information when data processing is performed using a bounded range of uncertainty over unbounded distributions.

- *H. Jeung is with SAP Australia.*
  E-mail: hoyoung.jeung@sap.com
- *H. Lu is with Aalborg University, Denmark.*
  E-mail: luhua@cs.aau.dk
- *S. Sathe is with EPFL, Switzerland.*
  E-mail: saket.sathe@epfl.ch
- *M. L. Yiu is with Hong Kong Polytechnic University, China.*
  E-mail: csmlyiu@comp.polyu.edu.hk



● raw positions reported from device
◇ actual but unobservable positions
○ inferred near-actual positions
p_2^*
p_2
c_2
r
p_1
p_1^*
c_1
μ_2
p_2
p_1
μ_1
(a)
(b)

Fig. 1: Different uncertain trajectory modelings.

Turning to a popular uncertainty model for trajectories, the *cylinder model* [33], [34] represents a trajectory as a sequence of 'buffered' line segments. The buffered area on each line segment, i.e., uncertainty range, captures all possible locations where an object could visit between two consecutive positions reported. Fig. 1(a) illustrates this modeling in 2D space, where the uncertainty region of position p_i is bounded by a circle c_i with radius r, and the two circles are linked by the outer lines that cover the union of all possible circles between c_1 and c_2. p_i^* represents the actual but unobservable location corresponding to the reported position p_i. Obviously, this modeling cannot address the three concerns discussed beforehand. In contrast, Fig. 1(b) shows another approach that models the uncertainty areas centered on (inferred) near-actual positions μ_1 and μ_2, corresponding to p_1 and p_2, respectively. The uncertainty ranges also exhibit different sizes with blurred boundaries, which represent different degrees of unbounded uncertainty areas.

1.1 Contributions

The goal of this paper is to establish core foundations for uncertain trajectory management, based on the new modeling approach. This requires a wide variety of re-innovations; particularly, we focus on three important problems, and make the following salient contributions:

1. Evolving-density trajectory model.
The first contribution of this paper is to introduce a new uncertain trajectory model that represents a trajectory as time-dependent Gaussian distributions. In each such distribution, the mean represents an actual location, while the standard deviation reflects the degree of an uncertainty range. The beauty of this modeling is to effectively capture the dynamicity of location uncertainty without any unrealistic assumption, while facilitating efficient query processing. We also provide a flexible framework that allows various approaches including domain-specific models to precisely infer such evolving normal distributions.

2. Evolving density estimators.
As the second contribution, this paper proposes three *evolving density estimators* that infer time-varying densities of location data. Computing evolving distributions is in fact a difficult problem, in particular when given data is multivariate, i.e., (t, x, y) coordinates. Existing work on uncertain trajectory processing often assumes that probability density functions are given [6], [7], [32], [33], [30]; however, we go beyond this assumption and develop effective methods for estimating time-dependent probability distributions of multivariate positional data. Our estimators have various advantages over existing motion estimators, e.g., they can efficiently give an entire probability distribution at each time instance, whilst Kalman filters [15] are capable only of giving an expected actual position, and particle filters [18] require significantly higher computation.

3. Efficient query processing.
Our third contribution is to present an effective mechanism that indexes evolving-density trajectories, and efficiently evaluates probabilistic range queries using the indexes. As the uncertainty ranges of evolving-density trajectories are unbounded, and vary over time (e.g., Fig. 1(b)), the prior studies [6], [31], [32], [7] in this domain are limited to fully support our uncertainty model. To address this problem, we employ a temporal R-tree as well as a hash table for quickly identifying a candidate set of uncertain trajectories, by dynamically computing the minimum bound for each data point (distribution) that can satisfy a given query condition. This process does not require any deterministic uncertainty ranges, leading to no information loss during probabilistic query processing. Furthermore, we also offer a concrete solution, including parameter settings, to precisely evaluate a presence probability for each candidate based on a Monte Carlo approach.

The remainder of the paper is organized as follows: Section 2 reviews state-of-the-art uncertain trajectory models and their pitfalls. We then introduce our uncertainty model in Section 3, and offer the details for evolving density estimators in Section 4. Section 5 presents the indexing and query processing mechanisms for probabilistic range queries. In Section 6, we provide experimental results using two real datasets. Section 7 discusses relevant studies. We then conclude in Section 8.

2 Uncertain Trajectory Models

There are two major reasons why uncertainty occurs in trajectory data [27]. One is known as *measurement error* which is caused by limited accuracy of positioning technology, e.g., GPS error. The other is *sampling error* that originates from discrete sampling of continuous movements of an object—the locations of the object between two sampled positions are unknown.

To deal with these uncertainty factors in trajectory data management, a rich body of studies have proposed various uncertainty models. These models commonly represent a trajectory using a sequence of uncertainty areas, so-called *uncertain trajectory*. Each of the uncertainty areas captures the measurement and sampling errors. This section provides an overview of these uncertain trajectory models.

The beads model [27], [14], [25], [21], [23] is established upon the observation that an object's movements are generally restricted by its (maximum) speed. Specifically, this model uses an ellipse for representing the uncertain locations where an object can possibly travel within two consecutive reported locations, where the two positions form the foci of the ellipse and the thickness of the ellipse is determined by the object's velocity. In 3D x-y-t space, the shape of the ellipse becomes a *bead*, which is an integrated body of an upward and a downward pointing cones. This geometry is also often called *space-time prism* [14], [21] or *pendant* [23]. This model thus represents an uncertain trajectory using a chain of beads. Note that the volume of a bead can become very large when the sampling rate of position is not frequent enough. Fig. 2(a) illustrates an example of this model, which represents an uncertain

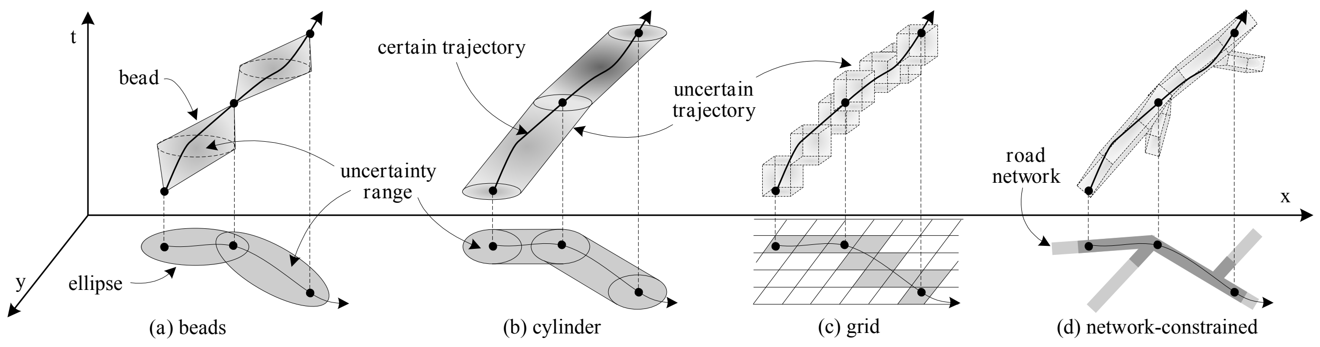


Fig. 2: Graphical comparison of uncertain trajectory models.

trajectory as a sequence of such ellipses in 2D or beads in 3D.

The cylinder model [34], [33], [12] 'buffers' a line segment—which models an object's linear movement between two sampled positions—using a user-specified uncertainty threshold. Thus, this model represents an uncertain trajectory as a sequence of such buffered line segments. In 3D x-y-t space, the uncertainty trajectory is illustrated as a sequence of sheared cylindrical bodies, shown as Fig. 2(b). A salient feature of this model is to offer well-established query semantics that define a set of uncertain movements of an object with respect to the cylindrical uncertainty representation of trajectory (e.g., objects *definitely/sometimes/always* reside within a given query space).

The grid model [26], [38] first partitions a given data space into a set of disjoint cells, and then represents an uncertain trajectory as a sequence of such cells, each of which covers some possible locations of the object in spatial or spatiotemporal space (Fig. 2(c)). Whilst this model is simple and facilitates efficient uncertain trajectory computing [38], finding an appropriate cell size is a difficult problem [17], since the size directly affects both the modeling power for capturing the uncertainty of trajectory and the efficiency of trajectory computing.

The network-constrained model [11], [10], [20], [39] maps a coordinate-based location in a raw trajectory to a linear range on a graph that models a road network. The range captures the possible locations of an object on the graph (map). Such a linear range is typically represented by a line segment, or a sequence of line segments that cover multiple edges in the graph (e.g., when a raw position is around a junction in a road network). Thus, the shape of an uncertain trajectory becomes a subgraph in x-y space (shown as the dark parts of the road network in Fig. 2(d)) or a set of 2D planes in x-y-t space. As shown, this model makes the uncertainty regions of a trajectory relatively tight, meaning that the degree of trajectory uncertainty can be reduced by using the additional information provided by networks (maps).

2.1 Pitfalls of the Uncertainty Models

Despite the various modeling capabilities of the existing uncertain trajectory models, they commonly neglect several important aspects in modeling and managing uncertain trajectories. We briefly discuss about their drawbacks:

First, the uncertain trajectory models generally regard a location measured from positioning technology as a precise, actual location of an object, while modeling an uncertainty range based on the reported position as center. Such a raw position, however, typically bears some measurement error [22], [27], thus the position may not be the exact location where the objet actually resided. This renders the uncertainty range center-shifted from the corresponding actual position that is generally unobservable. When the degree of measurement error is large, this 'shift effect' also becomes significant, which can cause false dismissals or false positives in uncertainty-aware query processing. Unfortunately, none of the uncertain trajectory models takes this into account.

Second, some of the uncertain trajectory models assume that the degree of uncertainty is constant regardless of the change of location or time. This assumption, however, may not always hold in reality. For example, GPS accuracy generally increases or decreases according to the presence of obstacles (e.g., tunnels and tall buildings) and the availability of a sufficient number of satellites for positioning. The accuracy of location estimation with 802.11 is also subjective to signal strengths available [22], which varies as the object moves over time. Therefore, the constant range used in the current uncertain trajectory models is not effective to capture the dynamic property of location uncertainty.

Third, the uncertain trajectory models bound the area of location uncertainty, typically using a circle with a user-specified radius. This approach works well with uniform distributions, however, positioning errors in practice rarely obey uniform distributions [22], [35]. In general, non-uniform distributions are unbounded. Therefore, it is inevitable to miss out some information when data processing is performed over any bounded uncertainty areas on such unbounded distributions.

Forth, most of the models assume that the probability density function (PDF) of a location is given. It is, however, a non-trivial problem to compute the parameters of a PDF (e.g., mean and standard deviation for a Gaussian distribution), in particular when each location of a trajectory requires different parameter values for PDF.

Last, some models require additional data beyond location coordinates: the beads model uses maximum speed of an object to determine the thickness of the ellipse (beads), while the network-constrained model needs map data encompassing the coverage of trajectories. As a result, these models may not be useful for some applications in which such additional information is unavailable; for example, the network-constrained model is unapplicable for freely moving objects' trajectories, e.g., trajectories of non-commercial airplanes and ships.

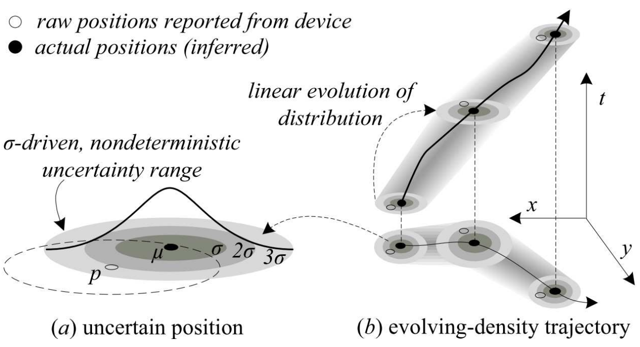

○ raw positions reported from device
● actual positions (inferred)
linear evolution of distribution
σ-driven, nondeterministic uncertainty range
μ
σ
2σ
3σ
p
t
x
y
(a) uncertain position
(b) evolving-density trajectory

Fig. 3: Evolving-density trajectory model.

3 Evolving–Density Trajectory

Covering the pitfalls of the existing uncertain trajectory models, we propose a new model for capturing and representing the uncertainty of trajectory, termed *evolving-density trajectory*. In the sequel, we introduce a set of key principles that establish the new uncertainty model, and then describe our system framework that supports the model.

3.1 Key Principles

Actual position: A wide range of applications employ various approaches for processing noise-contaminated raw location data obtained from positioning technologies. Examples include Kalman filtering for GPS data [22], particle filtering for mobile RFID readings [35], map matching for network-constrained objects' locations [3], cross/auto correlation between multiple locations, and sensor fusion [13]. These approaches provide means that can infer more reliable positions where an object was actually located. The evolving-density trajectory model supports such an inferred location as an actual location of the object, which serves as the center point of an uncertainty range.

Gaussian distribution: Measurement errors in positioning typically obey Gaussian distributions, e.g., RFID positions [35], WiFi-based localization [22], and GPS locations [27]. Therefore, our approach models that an object's location follows Gaussian distributions. Fig. 3(a) illustrates an *uncertain position* that is a basic unit to form an evolving-density trajectory. The uncertain position models the object's actual location as the mean μ, corresponding to the raw position p. This is a key difference from the GPS error model [27] where the mean of an error distribution is a raw GPS position p. In addition, the standard deviation σ reflects the degree of uncertainty with respect to μ.

σ-driven, nondeterministic uncertainty range: A Gaussian distribution is essentially unbounded. There exists no finite radius r for bounding un uncertain region over the distribution. Obviously, a small r (e.g., $r = 1 \cdot \sigma$) leads to missing significant information. Even a large r (e.g., $r = 3 \cdot \sigma$) causes some missing information. Therefore, we do not represent an uncertainty range in a deterministic manner. Instead, we keep only the information of deviation σ, and then dynamically compute the minimum bound for each data point (distribution) that can satisfy a given query condition. This approach does not incur any information loss. We will offer more details in Section 5.3.1.

Time-dependent uncertainty: In order to reflect the time-varying errors caused from positioning systems, we model each uncertain position in an evolving-density trajectory differs from another, meaning that each uncertain position has different values for μ and σ. In our modeling, we do not consider a particular method but various well-known as well as domain-specific models to estimate the values for μ and σ. As default estimators, we will also present three methods in Section 4.

Linear evolution of distribution: In-between two consecutive uncertain positions, we assume that the distributions of actual positions evolve linearly. This assumption is reasonable, since an object's movements between two positions are commonly modeled as linear in certain trajectories. One key advantage of this property is that the linear evolution of probability distribution can facilitate efficient computation of probabilistic queries, as any intermediate position between two consecutive uncertain positions is easily computed by linear algebra. We will offer more details of this processing at Algorithm 2 in Section 5.3.2.

Fig. 3(b) shows an example of evolving-density trajectory, which is built on the key principles described above. The shape of an evolving-density trajectory is similar to that represented by the cylinder model shown in Fig. 2(b); however, our model exhibits an irregular, blurred cylindrical body in order to capture the nondeterministic, time-dependent uncertainty of the trajectory. Table 1 compares the key features of the evolving-density trajectory model with the state-of-the-art uncertain trajectory models.

3.2 Framework Overview

Fig. 4 illustrates our system framework for managing evolving-density trajectories, consisting of the following key components:

Evolving Density Estimator computes the probability distribution of an object's position at each time. Specifically, this component takes a certain number of recent positions in a trajectory, and infers a Gaussian distribution (i.e. a mean value μ and a standard deviation σ) at a current time. This process can be performed in an online manner; whenever a new position is streamed to the system,

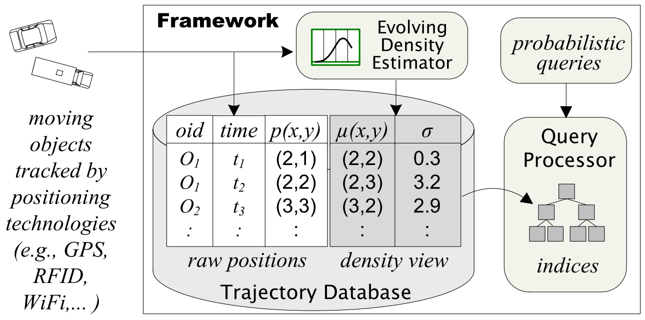
uncertain trajectory model	actual position	varying distribution	trajectory segment shape		index support	requirement except pdf
			2D	3D		
evolving-density [this paper]	✓	✓	blurred line	blurred cylinder	✓	×
beads [27], [14], [25], [21], [23]	×	✓	ellipse	up/downward cones	×	max. speed
cylinder [34], [33], [12]	×	×	buffered line	sheared cylinder	×	×
grid [17], [26], [38]	×	×	square	cube	✓	cell size
network-constrained [11], [39], [10], [20]	✓	✓	line	plane	✓	map data

TABLE 1: Comparison of uncertain trajectory models.

the estimator computes and populates the corresponding probability distribution. The framework also supports any user-given estimator to infer evolving distributions based on domain-specific knowledge.

Trajectory Database manages not only the raw positions reported from moving objects, but also the corresponding probability distributions derived from an evolving density estimator. To this end, we take the approach of *model-based views* [8], in order to preserve the original position data, while facilitating complex data processing over the probability distributions. This approach brings two key benefits to our system: One is to allow users to rerun the estimation process of evolving densities, when they find an incorrect setting for model parameters, or when they develop a new estimator. The other benefit is to enable various post-process of data. For example, integrating different error distributions obtained from heterogeneous positioning systems can substantially increase accuracy and precision, beyond what is possible using an individual location-sensing system [22].

Query Processor supports efficient processing of probabilistic range queries on the evolving-density trajectories managed in the system. In particular, the processor implements the well-known filter-and-refinement paradigm. At the filter step, both a temporal R-tree and a hash table are used to quickly prune those trajectory records whose time or position attributes are irrelevant to a given query condition. At the refinement step, the query processor evaluates whether each candidate resulted from the filtering step actually satisfies the given query (probability) condition. This process is performed by calling probability computation functions that are built-on Monte Carlo approaches, simulating Gaussian densities in a precise manner.


Framework
Evolving Density Estimator
probabilistic queries
moving objects tracked by positioning technologies (e.g., GPS, RFID, WiFi,...)
Query Processor
indices
raw positions
density view
Trajectory Database

oid	time	p(x,y)	μ(x,y)	σ
O_1	t_1	(2,1)	(2,2)	0.3
O_1	t_2	(2,2)	(2,3)	3.2
O_2	t_3	(3,3)	(3,2)	2.9
:	:	:	:	:

Fig. 4: Architecture of the framework.

4 Evolving Density Estimators

In this section, we propose to implement three estimators, which are used as evolving density estimator in our framework (Fig. 4). These estimators extend the GARCH (Generalized AutoRegressive Conditional Heteroskedasticity) model [29] for handling multi-dimensional location data. The GARCH model is a well-established stochastic volatility model that is generally used to assess an investment risk in finance, since volatility represents the degree of deviation from what the data is supposed to be, reflecting a measure of risk for investing. This motivates us to consider using the GARCH model for trajectory data processing—theoretically, the model can assess the deviation of a raw position from where the corresponding actual position is supposed to reside on.

In order to capture time-varying properties of location data, the estimators employ a sliding window that takes a H number of consecutive positions for the estimation, and then repeat the same estimation process using the next H positions. Specifically, given a sliding window of positions $S_{t-1}^{H} \in \mathbb{R}^{H\times 2}$, the estimators infer two quantities: (i) the expected true position $\hat{p}_t = (\hat{x}_t, \hat{y}_t)$ at time t, and (ii) a standard deviation $\hat{\sigma}_t$.

Our estimators, C^2-Est, R-Est, and AR-Est, have different trade-offs between accuracy and efficiency. In the sequel, we offer details for each estimator, and then compare their characteristics in Section 4.4. We also describe a generic method that can measure the accuracy of a given evolving density estimator in the last subsection.

4.1 Conditional Correlation Estimator

We first propose the Conditional Correlation estimator (**C^2-Est**). C^2-Est uses a multivariate mean inference model for estimating the expected true position $\hat{p}_t = (\hat{x}_t, \hat{y}_t)$, where $\hat{x}_t$ and $\hat{y}_t$ are the x- and y-coordinate, respectively. We employ the VAR (Vector AutoRegressive) model, VAR(k), where k represents the model order. The VAR model is a statistical model used to capture the linear interdependencies among multiple time series, contributing to the 2011 Nobel Prize in Economics in applying VAR models to macroeconomic analysis.

In the context of trajectory data, the VAR model can exploit the interdependencies of $\hat{x}_t$ and $\hat{y}_t$ for inferring the actual position $\hat{p}_t$. Specifically, the VAR(k) models $p_i = \hat{p}_i + a_i$ where $t - H \leq i \leq t-1$, and a_i are a series of uncorrelated random vectors with zero mean and covariance matrix Σ_a. Using a VAR(k) model, we infer the expected true position $\hat{p}_t$ as:

$$\hat{p}_t = \phi_0 + \sum_{j=1}^{k} \phi_j p_{t-j}, \tag{1}$$

possible options for evolving density estimator	estimation accuracy	estimation efficiency	actual position	uncertainty range	multi dimensional
C^2-, R-, AR-Est [this paper]	high	high	✓	✓	✓
dynamic density metrics [28]	high	high	✓	✓	×
particle filters [1], [18], [35]	very high	low	✓	✓	✓
Kalman filters [15], [22]	high	medium	✓	×	✓
map matching [3]	very high	medium	✓	×	✓

TABLE 2: Comparison of alternatives for evolving density estimators.

where $\phi_1, \ldots, \phi_k$ are autoregressive coefficients of each size 2×2, ϕ_0 is a 2-dimensional vector, and $t > k$. By default, the model parameter k is fixed to 2. The value for k, however, may vary according to applications; Finding appropriate values for the parameters of VAR models including the choice of k is well guided in Chapter 5 of [29].

For inferring the deviation $\hat{\sigma}_t$, C^2-Est uses the constant conditional correlation (CCC) model [2]. The CCC model is one of the most popular GARCH models, which is relatively simple to estimate. The CCC model adopts the conditional variance matrix of p_i for inferring a multivariate Gaussian distribution at each time step. The 2-by-2 conditional variance matrix of p_i, denoted as Λ_i, is defined as:

$$\Lambda_i = Var(p_i - \hat{p}_i | F_{i-1}), \;\; \Lambda_i = Var(a_i | F_{i-1}), \qquad (2)$$

where $Var(a_i|F_{i-1})$ is the variance matrix of a_i, given all the information F_{i-1} available until time $i-1$. The CCC model uses the errors a_i from the VAR(k) model to represent the conditional variance matrix Λ_i in Equation (2) as follows:

$$a_i = \Lambda_i^{1/2} \epsilon_i, \; \Lambda_{nm,i} = \rho_{nm} \sqrt{\lambda_{nn,i} \lambda_{mm,i}}, \qquad (3)$$

where $\Lambda_{nm,i}$ is the value on row n and column m of matrix Λ_i, $\lambda_{nn,i}$ and $\lambda_{mm,i}$ are defined using an univariate GARCH(1,1) model, ρ_{nm} are the constant conditional correlations with $\rho_{nn} = 1, \forall n$. The matrix formed by ρ_{nm} is symmetric and positive definite.

Given the constant conditional correlations ρ_{nm} and the univariate GARCH(1,1) models for $\lambda_{nn,i}$ and $\lambda_{mm,i}$, C^2-Est first infers $\lambda_{nn,t}$ and $\lambda_{mm,t}$ using the univariate GARCH models [28]. It then infers $\hat{\Lambda}_t$ as:

$$\hat{\Lambda}_t = (\rho_{nm} \sqrt{\lambda_{nn,t} \lambda_{mm,t}}) \qquad (4)$$

In summary, the C^2-Est estimator infers a two-dimensional Gaussian distribution at each time as $\mathcal{N}(\hat{p}_t, \hat{\Lambda}_t)$.

4.2 Radial Estimator

Next, we present the Radial estimator (**R-Est**). Like the C^2-Est estimator, R-Est also uses the VAR model for inferring the expected true position $\hat{p}_t$. R-Est, however, differs in how to estimate deviation. The CCC model used in C^2-Est incurs relatively high computation on inferring the deviation $\hat{\sigma}_t$. To improve the inefficiency of the inference process, the R-Est estimator employs a more efficient model, i.e., Radial GARCH model, for inferring $\hat{\sigma}_t$. To this end, we convert a 2-dimensional variance matrix inference problem (i.e., dealt with by the CCC model) into an one-dimensional variance inference problem.

Specifically, the R-Est estimator starts by computing the Euclidean norms of the errors a_i given by the VAR(k) model, denoted at γ_i. Intuitively, the Euclidean norm of the errors a_i are the possible variations, such that positions p_i are derived from the expected true positions $\hat{p}_i$. This means that γ_i for each i gives the uncertainty, where each position p_i manifests with respect to $\hat{p}_i$. Taking this process, R-Est uses γ_i for inferring the deviation $\hat{\sigma}_t$.

In short, the R-Est estimator uses a GARCH(1,1) model for inferring the variance of radial errors γ_i.

4.3 AutoRegressive Radial Estimator

This subsection introduces a variant of the R-Est estimator, called AutoRegressive Radial estimator (**AR-Est**). For inferring the deviation $\hat{\sigma}_t$, this new estimator takes the same RGARCH model used in R-Est.

However, AR-Est takes a different approach for the inference of an actual position. It decomposes the multivariate inference into two separate one-dimensional inferences. While the other two estimators use a multivariate VAR(k) model for inferring $\hat{p}_i$, the AR-Est estimator uses univariate AR (AutoRegressive) models for inferring the $\hat{x}_t$ and $\hat{y}_t$ separately. Since the procedure for inferring $\hat{y}_t$ is exactly the same as estimating $\hat{x}_t$, in the following we describe only the procedure for inferring $\hat{x}_t$.

Given a sliding window S^H_{t-1}, the AR(l) model models $x_i = \hat{x}_i + a_{x,i}$, where $t - H \leq i \leq t-1$, and $a_{x,i}$ are a series of uncorrelated random vectors with zero mean and variance matrix σ^2_{ax}. Given an AR(l) model, we infer the expected true position $\hat{x}_t$ as:

$$\hat{x}_t = \phi_{x0} + \sum_{j=1}^{l} \phi_{xj} p_{t-j}, \qquad (5)$$

where l is a non-negative integer denoting the model order, $\phi_{x1}, \ldots, \phi_{xl}$ are autoregressive coefficients, ϕ_{x0} is a constant, and $t > l$. More details regarding the estimation and choice of model parameter l are described in Chapter 3 in [29].

4.4 Comparison of Estimators

Comparison of C^2-Est, R-Est, AR-Est.

In general, a multivariate model requires a considerably higher number of parameters to estimate than a univariate model. Once all the parameters are set by appropriate values, the multivariate model would perform very well.

On the other hand, a large number of parameters also incur more computation for parameter determination, rendering the estimators based on multivariate models inefficient.

We expect that C^2-Est achieves the most accurate inference for actual position, R-Est offers a low running time, and AR-Est enables a good tradeoff between accuracy and running time. In Section 6.1, we will analyze the accuracy and efficiency of each estimator based experimental results. TABLE 3 summaries the core components of the estimators.

Comparison with Alternatives.
Estimating evolving densities is a non-trivial problem, since it requires to infer the probability distribution of each location in a trajectory, while considering the temporal dependency information of data. Some recent studies [28], [35], [18] have addressed this problem, however, they mainly focused on one-dimensional data (univariate time series).

Kalman filters [15] or particle filters [18] could also be used to implement the evolving density estimator. However, Kalman filters are capable of estimating an expected actual position only, whereas our estimators can give an entire probability distribution at each time instance. In terms of efficiency, our estimators perform significantly faster than particle filters that have a time complexity of $O(N^2 \cdot H)$ for its smoothing [18], where N is the number of samples to generate at each time in a window H. Table 2 compares the key features of our estimators with several representative alternatives available for evolving density estimators.

Accuracy Measure of Estimation.
The effectiveness of our uncertain trajectory model depends on the estimator being used. Therefore, measuring the accuracy of a given estimator is important. However, it is very difficult to measure the accuracy of an estimator, since actual distributions are unobservable, meaning that there are no ground truths to compare.

Fortunately, the literature suggests a solid mathematical means, so-called *density distance* [28]. This distance employs the probability integral transform [9] for measuring the distance between the probability density obtained from an evolving density estimator and its corresponding real (ideal) density. When a ground truth is unavailable, the density distance can serve as an useful measure to assess the effectiveness of an evolving density estimator. We will use the density distance in Section 6.1 to compare the accuracies of our evolving density estimators.

	inferred point $\hat{p}_t = (\hat{x}_t, \hat{y}_t)$	deviation $\hat{\sigma}_t$	time complexity
C^2-Est	VAR(2)	CCC GARCH(1,1)	$O(14 \cdot H)$
R-Est	VAR(2)	radial GARCH(1,1)	$O(9 \cdot H)$
AR-Est	AR(2)	radial GARCH(1,1)	$O(3 \cdot H)$

TABLE 3: Summary of evolving density estimators.

5 Probabilistic Range Query on Evolving-Density Trajectories

Probabilistic range queries are perhaps the most common query type on uncertain trajectories, as they can effectively retrieve uncertain objects or trajectories using solid mathematical foundations in probability theory. In this section, we introduce an efficient mechanism for evaluating probabilistic range queries on evolving-density trajectory databases. To this end, we first extend the definition of probabilistic range queries on evolving-density trajectories. We then present access methods to index evolving-density trajectories, as well as an algorithm for evaluating the queries based the indexes. Note that other probabilistic query types (e.g., probabilistic nearest neighbor queries) can also be evaluated over evolving-density trajectories using existing query-processing methods, as our uncertain trajectory model offers full information in terms of probability distribution at each position, which is sufcient for the probabilistic distance measures used in previous works.

5.1 Definitions

Definition 1: **Presence Probability.**
Given an uncertain object u, a circular query range $\odot(q, r_q)$ centerted at q with radius r_q, the *presence probability* of u in the range $\odot(q, r_q)$ is defined as:

$$Pr(u, \odot(q, r_q)) = \int_{u \cap \odot(q, r_q)} pdf(u, p)\, dp \tag{6}$$

where $pdf(u, p)$ denotes the probability density of object u at point p.

Definition 2: **Snapshot Object.**
Given an uncertain trajectory o and a timestamp t_q, we define $o(t_q)$ as the uncertain object of o at time t_q.

We proceed to define the probabilistic range query on an evolving-density trajectory database as follows.

Definition 3: **Probabilistic Range Query.**
Given a query range $\odot(q, r_q)$, a timestamp t_q, and a probability threshold ρ, a probabilistic range query $\mathcal{R}$ on an evolving-density trajectory database $\mathcal{D}$ returns all trajectories that have presence probabilities in $\odot(q, r_q)$ above ρ.

$$\mathcal{R}_{\mathcal{D}}(\odot(q, r_q), t_q) = \{o \in \mathcal{D} : Pr(o(t_q), \odot(q, r_q)) > \rho\}. \tag{7}$$

Here, we present how to compute the uncertain object $o(t_q)$ for the trajectory o at time t_q. Let $o.t_1, o.t_2, \cdots, o.t_m$ be an increasing sequence of sampling timestamps for the trajectory o, i.e, we have: $o.t_1 \leq o.t_2 \leq \cdots \leq o.t_m$. In addition, let $o.\mu_i$ and $o.\sigma_i$ be the mean and standard deviation of o at time $o.t_i$.

There are two cases of the temporal relationship between t_q and the sampling timestamps of o.

Case 1: $t_q = o.t_i$ for some $i \in [1, m]$: In this case, $o(t_q)$ is an uncertain object with the mean $o.\mu_i$ and deviation $o.\sigma_i$.

$$o(t_q).\mu = o.\mu_i, \quad o(t_q).\sigma = o.\sigma_i$$

Case 2: $o.t_i < t_q < o.t_{i+1}$ for some $i \in [1, m)$: By the linear evolution principle in Section 3.1, we define $o(t_q)$ as an uncertain object with the mean:

$$o(t_q).\mu = o.\mu_i + (o.\mu_{i+1} - o.\mu_i) \cdot \frac{t_q - o.t_i}{o.t_{i+1} - o.t_i}$$

and the deviation:

$$o(t_q).\sigma = o.\sigma_i + (o.\sigma_{i+1} - o.\sigma_i) \cdot \frac{t_q - o.t_i}{o.t_{i+1} - o.t_i}$$

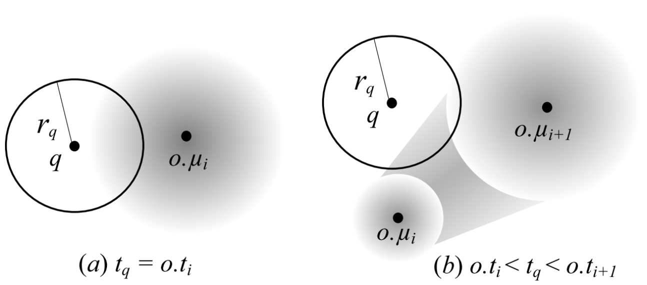


Fig. 5: Different cases in temporal relationship.

Figure 5 illustrates these two temporal relationship cases. For Case 1, as shown in Figure 5(a), we consider only a single record at timestamp t_q. For Case 2, we need to take into account two consecutive records at timestamps $o.t_i$ and $o.t_{i+1}$. This is illustrated in Figure 5(b).

5.2 Indexing Evolving-Density Trajectories

The literature suggests various access methods for efficiently querying uncertain objects. In particular, the U-tree [32] supports multi-dimensional arbitrary densities of objects, which is a desirable property for indexing evolving-density trajectories. Nevertheless, the method cannot be easily adopted for our study, since it does not take temporal information into account. To address this problem, we propose to index evolving-density trajectories with two complementary components: (i) a temporal R-tree [24] on the timestamp of records, and (ii) a hash table that supports efficient record retrieval using id.

A1R-tree: In the two cases discussed in the previous subsection, temporal information plays an important role in computing $o(t_q).\mu$ and $o(t_q).\sigma$. For example, given a trajectory, we need to determine the relevant sampling timestamps of the trajectory o with respect to the query timestamp t_q. Motivated by this, we employ a recent proposal, the A1R-tree [24] (Augmented 1-dimensional R-tree), for indexing the records of all evolving-density trajectories.

Specifically, each evolving-density trajectory record is captured in a leaf node entry in the form of $(traj\_id, loc\_id, t^{\vdash}, t^{\dashv})$. Here, $traj\_id$ identifies a trajectory o, loc_id identifies the location within the trajectory o. Moreover, $t^{\dashv}$ is the record's time attribute, $t^{\vdash}$ is the previous sampling timestamp on the trajectory, i.e., $t^{\vdash}$ is the time attribute of o's previous record.

A non-leaf node entry is in the form of $(t^{\vdash}, t^{\dashv}, cp)$, where cp is a pointer to a child node and $[t^{\vdash}, t^{\dashv}]$ is the minimum bounding interval that contains all time intervals in that child node.

TrajHash: In addition to the A1R-tree, we also build a hash table TrajHash. While the A1R-tree is used for quickly identifying whether an uncertain trajectory satisfies the spatial and temporal conditions in a given probabilistic range query, TrajHash facilitates efficiently retrieving trajectory records from the evolving-density trajectory database. More precisely, given a trajectory identifier $traj\_id$ and a location identifier loc_id on that trajectory, TrajHash$[traj_id, loc_id]$ returns the corresponding evolving-density trajectory record.

5.3 Query Processing

We apply the well-known filtering-and-refinement approach to process the probabilistic range query stated in Definition 3. At the filtering phase, the A1R-tree is used to prune the trajectory records whose spatiotemporal attributes are irrelevant to a given query. At the refinement phase, we verify whether each candidate reported from the filtering phase meets the probability threshold ρ, by evaluating the concrete presence probability of the candidate. The following subsections discuss more details about these phases.

5.3.1 Filtering Phase

We first present two important lemmas for pruning irrelevant entries in the A1R-tree.

Lemma 1: **Temporal pruning.**
Let e be an entry in the temporal R-tree. If $e.t^{\vdash} \geq t_q$ or $e.t^{\dashv} < t_q$, then the subtree of e cannot contain any result of the query.

Lemma 2: **Spatial pruning.**
Let ϵ_ρ be a value such that the probability of Gaussian distribution within $\mu \pm \epsilon_\rho \cdot \sigma$ equals to ρ. Given an uncertain snapshot object rec with mean $rec.\mu$ and deviation $rec.\sigma$, if $\|rec.\mu, q\| > r_q + \epsilon_\rho \cdot rec.\sigma$, then rec does not qualify as a result of the query.

Fig. 6(a) depicts the spatial pruning mechanism. Given ρ as query input, we can quickly compute ϵ_ρ by finding the closest value to ρ in a lookup table for z-scores, e.g., Fig. 6(b), which can determine the percentile rank (or prediction interval) with known mean and variance.

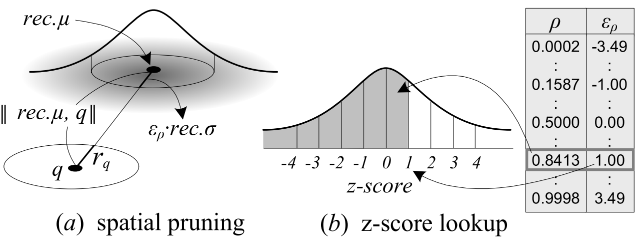


Fig. 6: Illustration of spatial pruning.

In addition to enabling the spatial pruning in query processing, Lemma 2 entitles an important feature to our system. Since ϵ_ρ in Lemma 2 can be dynamically computed during query processing, the framework (Fig. 4) does not need to store any pre-computed uncertainty ranges. In Section 3.1, we described that such pre-computed ranges essentially incur information loss while processing queries over Gaussian distributions. Therefore, Lemma 2 embodies the principle of our uncertain trajectory model, "σ-driven,

Algorithm 1 **RangeQuery** (query point q with radius r_q at time t_q, probability threshold ρ, A1R-tree node $node$)

```
1: $\epsilon_\rho \leftarrow$ ZScoreLookUp($\rho$)
2: $results \leftarrow \emptyset$
3: if $node$ is a non-leaf node then
4:     for each node entry $e \in node$ do
5:         if $e.t^{\vdash} < t_q \leq e.t^{\dashv}$ then                ▷ temporal filtering
6:             RangeQuery($q, r_q, t_q, e.cp$)
7: else
8:     for each node entry $e \in node$ do
9:         if $t_q = e.t^{\dashv}$ then                 ▷ get a single record
10:            $rec \leftarrow$ TrajHash[$e.traj\_id, e.loc\_id$]
11:        else                           ▷ get two consecutive records
12:            $rec_1 \leftarrow$ TrajHash[$e.traj\_id, e.loc\_id - 1$]
13:            $rec_2 \leftarrow$ TrajHash[$e.traj\_id, e.loc\_id$]
14:            $rec \leftarrow$ GetMidPoint($rec_1, rec_2$)
15:        if $\|rec.\mu, q\| < r_q + \epsilon_\rho \cdot rec.\sigma$ then ▷ spatial filtering
16:            if PresenceProb($rec, q, r_q, \rho$)$> \rho$ then
17:                add $rec$ to $results$
```

nondeterministic uncertainty range", without losing any information.

Algorithm 1 presents the overall mechanism for processing probabilistic range queries, consisting of the filtering phase in Lines 1, 3–15 and the refinement phase in Lines 16–17.

The filtering phase has two primary steps. In the first step, the A1R-tree is used to find all relevant evolving-density trajectory records with respect to the given query timestamp t_q. This is processed by an extended point query that fetches all leaf node entries that satisfy $t^{\vdash} < t_q \leq t^{\dashv}$ (Lines 3–6). For each leaf node, we use the hash table TraHash for efficiently retrieving the records whose times are relevant to t_q. For Case 1 where a single trajectory record rec exactly matches the query time, we obtain the record directly via TrajHash (Lines 9–10). For Case 2 where the query time involves two consecutive trajectory records rec_1 and rec_2, we apply Algorithm 2 to compute a snapshot object rec at t_q on the link between $rec_1.\mu$ and $rec_2.\mu$ (Lines 12–14).

Fig. 7(a) shows such a snapshot position $rec.\mu$. Note that the evolving-density trajectory model assumes a linear evolution of probability distribution between two uncertain positions, described in Section 3.1. This allows Algorithm 2 to compute the deviation $rec.\sigma$, based on the linear growth of distribution (Line 7).

In the second step of the filtering phase (Algorithm 1), the record rec obtained from either Case 1 or Case 2

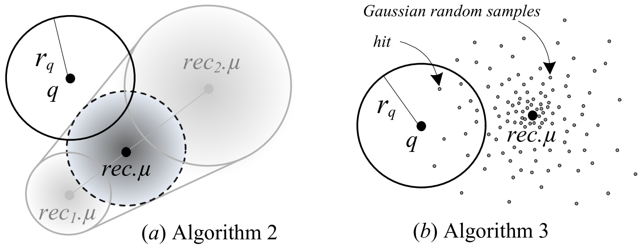


Fig. 7: Illustrations of Algorithms 2 and 3.

is passed for spatial filtering and probability evaluation (Lines 15–17). The spatial pruning of Lemma 2 rules out those trajectory locations that are too far away from the query location q (Line 15). In Line 1, the algorithm already computed the value ϵ_ρ, such that the probability of Gaussian distribution within $\mu \pm \epsilon_\rho \cdot \sigma$ equals to ρ. As described in Fig. 6(b), this computation is performed by simply looking up the z-score table.

5.3.2 Refinement Phase

In the refinement phase, the presence probability (Definition 1) is computed for each trajectory that passes the whole filtering phase (Line 16). The query result includes only those trajectories whose presence probabilities are greater than the given threshold ρ (Line 17). We elaborate on how to compute the presence probability $\odot(q, r_q)$ of a trajectory.

Given an evolving-density trajectory record rec, Algorithm 3 computes the presence probability of rec in the query region $\odot(q, r_q)$ by taking a Monte Carlo approach, illustrated in Fig. 7(b). Specifically, the algorithm first generates N samples using $N = 2\ln(1/\delta)\phi^2\rho$ (Line 1), which allows to compute a presence probability no less than $(1-\phi)\rho$ in the query range $\odot(q, r_q)$ with confidence no less than $1-\delta$. This computation for N is suggested and proved by [38]. In general, we believe that setting N to approximately 500 is reasonable, while considering both precision and efficiency of probability computation—e.g., Kanagal *et al.* [18] show that the inference precision starts to become reliable from when $N = 100$.

For each of N runs, the algorithm generates a random sample point around $rec.\mu$ obeying the deviation $rec.\sigma$ (Line 4). This can be computed by using a Gaussian distribution generator available in various programming libraries for numerical computation. In our implementation, we modified the polar method [19] to generate bivariate (x, y) positions. We omit the details as the modification is straightforward.

Algorithm 2 **GetMidPoint** (evolving-density trajectory record rec_1, next record rec_2)

```
1: $rec \leftarrow$ new record  ▷ snapshot point between $rec_1$ and $rec_2$
2: $t_1 \leftarrow$ time of $rec_1$
3: $t_2 \leftarrow$ time of $rec_2$
4: $f \leftarrow \frac{t_q - t_1}{t_2 - t_1}$
5: $rec.\mu.x \leftarrow rec_1.\mu.x + f \times (rec_2.\mu.x - rec_1.\mu.x)$
6: $rec.\mu.y \leftarrow rec_1.\mu.y + f \times (rec_2.\mu.y - rec_1.\mu.y)$
7: $rec.\sigma \leftarrow rec_1.\sigma + f \times (rec_2.\sigma - rec_1.\sigma)$
8: return $rec$
```

Algorithm 3 **PresenceProb** (evolving-density trajectory record rec, query point q with radius r_q, prob. threshold ρ)

```
1: $N \leftarrow 2\ln(1/\delta)\phi^2\rho$          ▷ suggested by Lemma 4.2 in [38]
2: $hit \leftarrow 0$                                   ▷ hitting count
3: for counter $i$ from 1 to $N$ do        ▷ random point generation
4:     point $s \leftarrow$ GaussianGenerator($rec.\mu, rec.\sigma$)
5:     if $\|s, q\| \leq r_q$ then
6:         $hit \leftarrow hit + 1$
7: return $hit/N$
```

In Lines 5–6, the generated point is verified whether it falls into the query region $\odot(q, r_q)$. Finally, the presence probability is computed and returned in Line 7.

6 EXPERIMENTS

In this section, we evaluate the effectiveness and efficiency of the core components in our framework. First, we investigate the performance insights of the evolving density estimators proposed in this paper (Section 4), while comparing with a particle filter as an existing alternative density estimator (TABLE 2). We then compare the effectiveness as well as query-processing performance of our evolving-density trajectory model with the cylinder model (described in Section 2) as a counterpart in the literature.

We implemented the estimators using MATLAB, and tested their performances on an Intel Dual Core 2 GHz machine having 3 GB of main memory. In addition, we used the Java language to implement the query processor including the A1R-tree [24] and TrajHash [Section 5.2], and measured their performances using an Intel Core i7 2.93 GHz system with 12 GB of main memory.

We used two large-scale real datasets in the experiments: (i) ***people*** [40] consists of large number of trajectories obtained from GPS-enabled devices carried by 155 people over two years. (ii) ***car*** [16] contains GPS logs collected from 20 cars over several months, as part of a project that investigated driver response to speeding alerts issued by in-car devices. The following table offers a brief summary of the datasets.

dataset name	*people*	*car*
average sampling interval	2 sec.	1 sec.
number of entities	155 people	20 cars
number of trajectories	56,254	3,190
number of positions	22,591,688	1,778,773

TABLE 4: Summary of datasets.

6.1 Comparing Evolving Density Estimators

This subsection compares the performances of our evolving density estimators (C^2-Est, R-Est and AR-Est) proposed in Section 4 with one of those alternative density estimators listed in TABLE 2. We chose the particle filter as an alternative density estimator, since the other options are limited for apple-to-apple comparison in our experiment. Specifically, dynamic density metircs [28] are unable to handle multi-dimensional location data, Kalman filters [15], [22] cannot estimate uncertainty ranges, while map-matching methods [3] require additional map data. We offered a theoretical comparison among these methods in Section 4.4. For the particle filter implementation, we referred to the well-known tutorial for nonlinear tracking based on particle filters [1]. We set the number of particles to 256 in our experiments, which is suggested in [37].

We first study the estimation accuracies of the evolving density estimators. As mentioned in Section 4.4, we use the density distance [28] for measuring the accuracies, which represents the difference between a probability distribution inferred by an estimator and its corresponding actual (ideal) probability distribution.

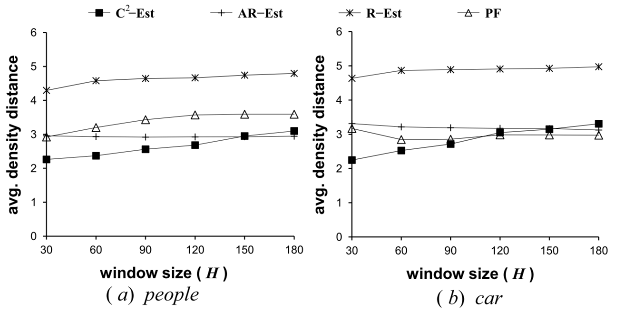


Fig. 8: Accuracy comparison of the estimators (the lower, the more accurate).

Fig. 8 compares the density distances reported from each estimators inference, while increasing the window size. The density distances are computed as averages over the results using all the trajectories in each dataset. Surprisingly, the estimation accuracies of AR-Est and C^2-Est are not worse than the particle filter (PF), which is known as one of the most accurate probabilistic estimation methods. These results imply that the evolving density estimators proposed in this paper are suitable for time-dependent density estimation.

Another observation found from the results is that AR-Est outperform R-Est–the average improvement of AR-Est over R-Est is ranging from a 35% (people) to a 40% (car). Recalling TABLE 3, the only difference between these two estimators is what underlying model is used for inferring an actual position. Therefore, the results suggest that using two separate AR models for inferring the $\hat{x}_t$ and $\hat{y}_t$ is more effective than using one multivariate VAR (2) model for inferring $\hat{p}_i$.

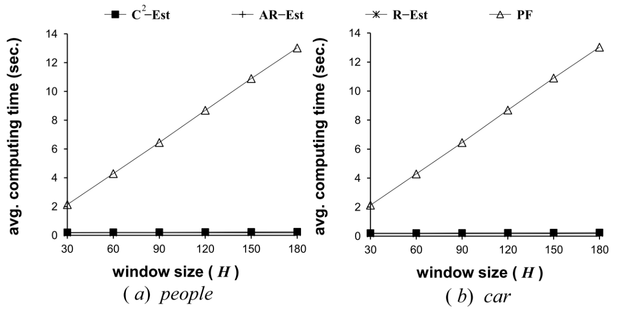


Fig. 9: Efficiency comparison of the estimators with a Particle Filter (PF).

Interestingly, both C^2-Est and R-Est exhibit increasing density distances, as the window size used for inference grows; whilst AR-Est shows a relatively consistent performance regardless of the window size used. The main reason is that the estimation performance of VAR models used in C^2-Est and R-Est is largely affected by the window size, while the performance of autoregressive model used in AREst is robust against the change of window size.

Next, we compare the inference efficiencies of the estimators. Fig. 9 shows the average computing times required to perform one iteration of the evolving-density inference. The results clearly demonstrate the advantages of our evolving density estimators over the particle filter (PF). While the computing times of the particle filter grow linearly as the window size increases, our evolving density estimators show very slight increases in computing time, compared with the particle filter.

To have a closer look to compare the efficiencies among AR-Est, R-Est, and C^2-Est, we excluded the results of PF in Fig. 10. AR-Est is a clear winner in this experiment set; it achieves a factor of 1.5 times speed-up over C^2-Est when H = 180. For small window sizes, R-Est also shows as high efficiency as AR-Est. However, recall that R-Est reported low inference accuracies using small windows in the previous experiments (Fig. 8). Turning to the results from C^2-Est, it shows significantly poor efficiency, compared with the others. The main reason is that C^2-Est uses the CCC model for inferring an uncertainty range, which incurs higher computation than radial GARCH models used in the other estimators.

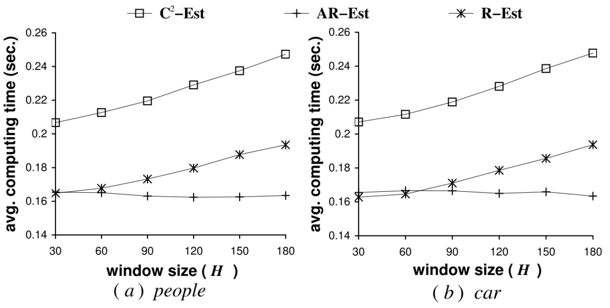


Fig. 10: Efficiency comparison of the evolving density estimators.

To sum up, taking into account both accuracy and efficiency of the estimation process, we conclude that both AR-Est and C^2-Est would be the best choices to infer evolving densities of moving objects. In particular, we recommend AR-Est for applications that manage a large number of objects where density estimation is heavily performed, while we do C^2-Est for applications where the accuracy of density estimation is the first priority.

6.2 Analysis of Evolving-Density Trajectories Driven by AR-Est

This subsection analyzes the characteristics of the evolving-density trajectories estimated by AR-Est that was one of the two winners in the previous experiments. We omit the result from C^2-Est due to the similarity. Fig. 11 demonstrates a part of the trajectories, where empty/dark dots denote raw/inferred positions, and circles represent 1σ ranges. The texts next to the circles show the exact distances in meters of 1σ from each inferred position. The figure clearly shows that our estimator is able to derive time-varying distributions from a given trajectory data. In order to analyze the quality of the estimations by AR-Est, we conducted the following further experiments on studying the inference powers of actual positions as well as uncertainty ranges.

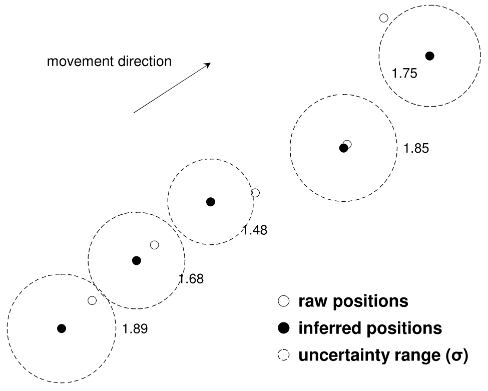


Fig. 11: Uncertain positions estimated by AR-Est.

Actual Positions: As we do not have ground truths, we attempt to compare the accuracies of the actual positions inferred by AR-Est in an indirect manner.

Fig. 12(a) presents Euclidean distances in meters across different positions: $\|$*map-matched positions, GPS positions*$\|$ and $\|$*map-matched positions, inferred positions by AR-Est*$\|$. These results were obtained from *car*. In the results, the average distances of map-matched locations from GPS positions is about 3.8 meters, while that from the inferred positions by AR-Est is around 4.8 meters. If we assume that the map-matched locations are ground truths[1], these results reflect the overall error of AR-Est's inference is higher than that of raw GPS positions. This was in fact unexpected results for us. We then looked inside the dataset, and found out a large number of trajectories in *car* showed sudden changes, such as turning to the left or right at junctions. When such a big change of direction occurred in a trajectory, the estimation of an actual position by AR-Est yielded a large error.

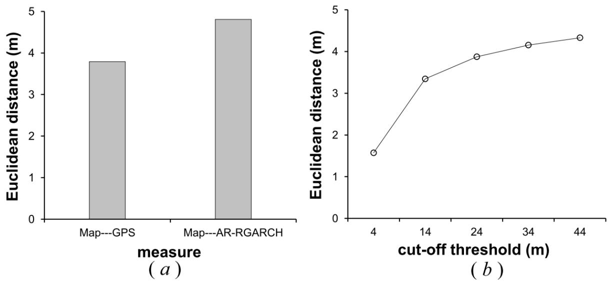


Fig. 12: Comparing map-matched positions with raw GPS positions and inferred positions.

In order to closely look at the effect of sudden changes in AR-Est estimation, we applied cut-off thresholds to the inference results. Note that sudden changes of direction in objects' movements are natural, thus analyzing the effect in AR-Est estimation can offer an important insight into understanding and improving the performance of our

1. Note that the map-matched locations include errors, and thus cannot serve as ground truths; however, they are useful to show the relative differences from GPS positions and from AR-Est-inferred locations, respectively.

density estimator. Fig. 12(b) shows the recomputed average distances along with increasing threshold values. As the threshold value grows, the average distance between the map-matched locations and the inferred positions increases dramatically at first, and then slightly later. Even for 44-meter threshold, the average distance is still smaller than that of "Map—AR-Est" in Fig. 12(a). These results indicate that a small number of large errors, which are caused by the sudden change effect, increase the average distance. Therefore, we expect that the inference error of AR-Est can be substantially reduced, if subsequent research improves the estimator on handling sudden changes of objects' movements.

Uncertainty Ranges: Fig. 13 shows the histograms of 3σ ranges of the distributions estimated by AR-Est. Since 3σ captures a 99.7 % of a Gaussian distribution, it can reflect the practical bound of an uncertainty range in an evolving-density trajectory. Clearly, overall sizes of 3σ–ranges are very small; the peaks of the histograms reside within about 1.2 meters (*people*) and 3.2 meters (*car*). When we consider the fact that horizontal one-sigma errors of GPS positions are about 10.6 meters (refer to Figure 2.9 in [22]), the sizes of uncertainty ranges derived from AR-Est are significantly smaller. This is an important feature of our evolving-density trajectory model as well as AR-Est, since tight uncertainty bounds imply reduced uncertainty in trajectory data management. For example, the cylinder model takes a fixed-size radius to determine the degree of positional uncertainty, which is typically set to the size of GPS errors. Therefore, our uncertain trajectory model can have various benefits in data processing with smaller uncertainty bounds.

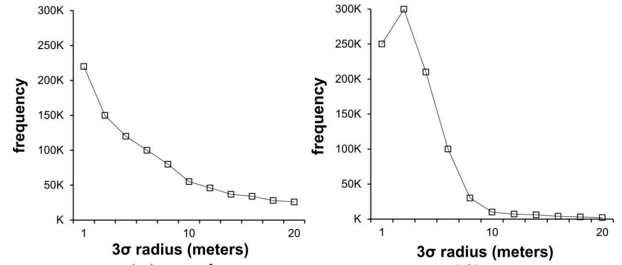


Fig. 13: Histograms of 3σ ranges.

Another key finding from this experiment set is that different movements of objects (e.g., people vs. cars) can result in different sizes of uncertainty ranges, although their positions are reported from the same type of positioning device, i.e., GPS. This dymanicity of uncertainty range can be captured by only our uncertainty model, as our approach is data-driven, while the current uncertain trajectory models use pre-specified uncertainty ranges.

6.3 Comparing Uncertain Trajectory Models in Query Processing

Next, we compare our evolving-density trajectory model with and the cylinder model [33] as an existing uncertaintrajectory model, in terms of query processing. In Section 3.2, we explained that our uncertain trajectory model can accept an arbitrary density estimator. To embody the evolving-density trajectory model, we used two different applications of evolving-density estimator, which are AR-Est and the particle filter. For the implementation of cylinder model, we set the uncertainty range (radius) of each position to 10.6 meters, which is the size of horizontal one-sigma errors in GPS positioning [22]. Note that the uncertainty range of our evolving-density trajectory model is not fixed but time-dependent, described in Section 3.1.

We issued 50 arbitrary probabilistic range queries defined in Section 5.1 on each dataset, and averaged the results. To this end, we built the A1R-tree as well as the TrajHash introduced in Section 5.2 for each dataset. We loaded the indexes into the main memory before processing queries, while the two datasets were kept on disk. We set page size to 4K, and did not buffer data pages. For each query, we randomly picked a query point q and a query time t_q within the spatial and temporal domains of each dataset, respectively. In addition, we specified a query range r_q as a ratio to the average standard deviation $\overline{\sigma}$ across all uncertain positions computed by AR-Est in each dataset. For example, we set $r_q = 1000 \cdot \overline{\sigma}, 2000 \cdot \overline{\sigma}, ..., 5000 \cdot \overline{\sigma}$, which define query ranges with radii of approximately from one to ten kilometers. We then measured the number of results as well as elapsed times in each test, while varying query positions, size of query ranges, and minimum probability threshold ρ.

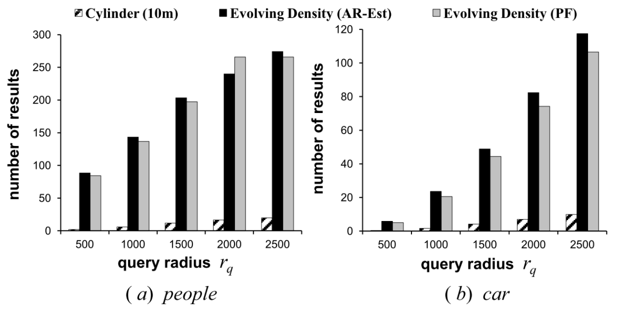
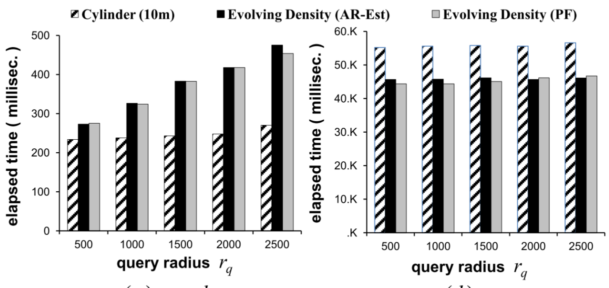


Fig. 14: Numbers of query results.

Figure 14 demonstrates the number of trajectory records returned as query results, along with different applications of query ranges. We set different ratio values to r_q for the two datasets, as each dataset shows a very different data distribution from each other.

Clearly, more numbers of trajectory records are returned when large values are set to r_q, since a wide query range is likely to overlap more trajectories. On both datasets, the numbers of query results precessed over the uncertain trajectories of the cylinder model are much lower than those from our uncertain trajectory model including both AE-Est and particle filter (PF) applications. The main reason is the irregular sampling ratios of data. Our uncertain trajectory model takes the principle of linear evolution of distribution described in Section 3.1, and generates uncertain positions at every time point when no data reported from device. In contrast, the cylinder model considers uncertain positions only for the time when position was reported. Therefore, query processing on our uncertain trajectory model is likely to capture more information, given a query range with time.



(a) *people* (b) *car*

Fig. 15: Query processing times.

We also measured the average elapsed times of query processing, plotted in Fig. 15. Interestingly, the differences of the elapsed times between the cylinder model and our models are not significant, when considering the differences in number of query results in Fig. 14. This is because the cylinder model produces many candidates after the filtering step in query processing, which turn out to be unqualified in the refinement phase, due to the probability threshold. These results indicate that the query processing mechanisms for evolving-density trajectories introduced in Section 5 are effective.

7 Related Work

Uncertain models and density estimators: As discussed in Section 2, existing models [27], [14], [34], [33], [12], [21], [39], [11], [10], [20] for uncertain trajectories suffer from several drawbacks. First, they set the center of uncertain range of an object as its measured position, which may deviate considerably from its actual position. Second, they assume a fixed size for the uncertain range, which may not hold in reality as measurement accuracy varies dynamically. Third, they use a bounded uncertain range, causing information loss and missing query results. To avoid these drawbacks, we propose the evolving density model to represent uncertain trajectories in Section 3.

Our evolving density model applies a density estimator to infer the actual location and the uncertain range of an object. We have compared various estimators (including our estimators) in Section 4.4 and summarized their features in Table 2. All of them can infer the actual position of an object. Kalman filters [15], [22] and map matching [3] cannot infer the uncertain range of an object. While particle filters [18], [35] can infer the uncertain range, they incur much higher running time than our estimators. Both dynamic density metrics [28] and our proposed density estimators apply an auto-regression model [29], [28]. However, the models in [29], [28] are designed for one-dimensional times series (x_t). Our proposed estimators (C^2-Est, R-Est, AR-Est) are developed specifically for multi-dimensional (x_t, y_t) time series (i.e., trajectories), and they have not been studied in [29], [28] either.

Xue et al. [36] studied how to predict the destination of a query user by matching it with historical sub-trajectories obtained from other users. With such additional knowledge, we may refine the Gaussian distribution of the uncertainty range (in Figure 3a) towards the predicted destination, i.e., locations closer to the destination have higher density.

Query processing on uncertain trajectories: Querying on uncertain trajectories have been extensively studied in the last decade. Cheng *et al.* [5], [6] highlighted the importance of probabilistic approaches to querying uncertain moving objects, and presented efficient processing mechanisms for range [6] as well as nearest neighbor queries [5]. Tao *et al.* [31], [32] developed the U-tree that utilizes pre-computed presence probabilities of objects for processing probabilistic range queries efficiently. Others dealt with probabilistic range queries on uncertain trajectories in different spaces, such as road-network spaces [10], [39], and near-future spaces [38]. However, all the works above only consider objects with bounded uncertainty range. In contrast, our proposed indexes and query processing techniques are designed for unbounded uncertainty ranges.

8 Conclusions

This paper revisits state-of-the-art approaches to modeling the uncertainty of trajectories, as their modeling powers are insufficient to capture several important properties of trajectory data. To complement this, we proposed the *evolving-density trajectory* model that represents a trajectory as time-dependent Gaussian distributions. We then introduced three *evolving density estimators* that effectively infer time-varying densities of location data. We also presented an efficient mechanism to process probabilistic range queries on indexed evolving-density trajectories. We believe that this work can serve as an important basis in further studies on managing uncertain trajectory databases.

Acknowledgement

Man Lung Yiu was supported by ICRG grant A-PL99 from the Hong Kong Polytechnic University.

References

[1] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Trans. Sig. Proc.*, 50(2):174-188, 2002.

[2] L. Bauwens, S. Laurent, and J. Rombouts. Multivariate GARCH models: a survey. *Journal of applied econometrics*, 21(1):79-109, 2006.

[3] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *VLDB*, pages 853-864, 2005.

[4] M. Y. Chen, T. Sohn, D. Chmelev, D. Haehnel, J. Hightower, J. Hughes, A. Lamarca, F. Potter, I. Smith, and A. Varshavsky. Practical metropolitan-scale positioning for GSM phones. In *UbiComp*, pages 225–242, 2006.

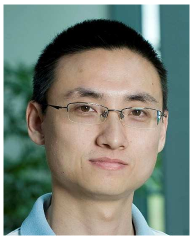
[5] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Querying imprecise data in moving object environments. *TKDE*, 16:1112-1127, 2004.

[6] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. S. Vitter. Efficient indexing methods for probabilistic threshold queries over uncertain data. In *VLDB*, pages 876-887, 2004.

[7] X. Dai, M. L. Yiu, N. Mamoulis, Y. Tao, and M. Vaitis. Probabilistic spatial queries on existentially uncertain data. In *SSTD*, pages 400-417, 2005.

[8] A. Deshpande and S. Madden. MauveDB: supporting model-based user views in database systems. In *SIGMOD*, pages 73-84, 2006.
[9] F. Diebold, T. Gunther, and A. Tay. Evaluating Density Forecasts with Applications to Financial Risk Management. *International Economic Review*, 39(4):863-883, 1998.
[10] Z. Ding. UTR-Tree: An index structure for the full uncertain trajectories of network-constrained moving objects. In *MDM*, pages 33-40, 2008.
[11] Z. Ding and R. H. Güting. Uncertainty management for network constrained moving objects. In *DEXA*, pages 411-421, 2004.
[12] E. Frentzos, K. Gratsias, and Y. Theodoridis. On the effect of location uncertainty in spatial querying. *TKDE*, 21:366-383, 2009.
[13] J. Hightower and G. Borriello. Location systems for ubiquitous computing. IEEE Computer, 34(8):570-66, 2001.
[14] K. Hornsby and M. J. Egenhofer. Modeling moving objects over multiple granularities. *Annals of Mathematics and Articial Intelligence*, 36:177-194, 2002.
[15] A. Jain, E. Y. Chang, and Y.-F. Wang. Adaptive stream resource management using Kalman filters. In *SIGMOD*, pages 11-22, 2004.
[16] C. S. Jensen, H. Lahrmann, S. Pakalnis, and J. Runge. The infati data. TIMECENTER Technical Report, 2008.
[17] H. Jeung, H. T. Shen, and X. Zhou. Mining trajectory patterns using hidden markov models. In *DaWaK*, pages 470-480, 2007.
[18] B. Kanagal and A. Deshpande. Online filtering, smoothing and probabilistic modeling of streaming data. In *ICDE*, pages 1160-1169, 2008.
[19] D. E. Knuth. The art of computer programming, volume 3: (2nd ed.) sorting and searching. Addison Wesley Longman Publishing Co., Inc., 1998.
[20] B. Kuijpers, B. Moelans, W. Othman, and A. A. Vaisman. Analyzing trajectories using uncertainty and background information. In *SSTD*, pages 135-152, 2009.
[21] B. Kuijpers and W. Othman. Trajectory databases: Data models, uncertainty and complete query languages. *J. Comput. Syst. Sci.*, 76:538-560, 2010.
[22] A. LaMarca and E. de Lara. Location Systems: An Introduction to the Technology behind Location Awareness. Morgan and Claypool Publishers, 2008.
[23] H. Liu and M. Schneider. Querying moving objects with uncertainty in spatio-temporal databases. In *DASFAA*, pages 357-371, 2011.
[24] H. Lu, B. Yang, and C. S. Jensen. Spatio-temporal joins on symbolic indoor tracking data. In *ICDE*, pages 816-827, 2011.
[25] H. J. Miller. A measurement theory for time geography. Geographical Analysis, 37:17-45, 2005.
[26] N. Pelekis, I. Kopanakis, E. E. Kotsifakos, E. Frentzos, and Y. Theodoridis. Clustering trajectories of moving objects in an uncertain world. In *ICDM*, pages 417-427, 2009.
[27] D. Pfoser and C. S. Jensen. Capturing the uncertainty of moving object representations. In *SSD*, pages 111-132, 1999.
[28] S. Sathe, H. Jeung, and K. Aberer. Creating probabilistic databases from imprecise time-series data. In *ICDE*, pages 327-338, 2011.
[29] R. Shumway and D. Stoffer. Time series analysis and its applications. Springer-Verlag, New York, 2005.
[30] S. Singh, C. Mayeld, R. Shah, S. Prabhakar, S. E. Hambrusch, J. Neville, and R. Cheng. Database support for probabilistic attributes and tuples. In *ICDE*, pages 1053-1061, 2008.
[31] 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 *VLDB*, pages 922-933, 2005.
[32] Y. Tao, X. Xiao, and R. Cheng. Range search on multidimensional uncertain data. *TODS*, 32:15, 2007.
[33] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. Managing uncertainty in moving objects databases. *TODS*, 29(3):463-507, 2004.
[34] G. Trajcevski, O. Wolfson, F. Zhang, and S. Chamberlain. The geometry of uncertainty in moving objects databases. In *EDBT*, pages 233-250, 2002.
[35] T. Tran, C. Sutton, R. Cocci, Y. Nie, Y. Diao, and P. Shenoy. Probabilistic inference over RFID streams in mobile environments. In *ICDE*, pages 1096-1107, 2009.
[36] A. Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Huang, and Z. Xu. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In *ICDE*, 2013.
[37] J. Yu, W.-S. Ku, M.-T. Sun, and H. Lu. An RFID and particle filter-based indoor spatial query evaluation system. In *EDBT*, pages 263-274, 2013.
[38] M. Zhang, S. Chen, C. S. Jensen, B. C. Ooi, and Z. Zhang. Effectively indexing uncertain moving objects for predictive queries. *PVLDB*, 2:1198-1209, 2009.
[39] K. Zheng, G. Trajcevski, X. Zhou, and P. Scheuermann. Probabilistic range queries for uncertain trajectories on road networks. In *EDBT*, pages 283-294, 2011.
[40] Y. Zheng, X. Xie, and W.-Y. Ma. GeoLife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, 33(2):32-39, 2010.

Hoyoung Jeung is a senior researcher at SAP Research Australia, as well as an adjunct senior fellow at the University of Queensland. Prior to SAP, he worked as a postdoctoral researcher in the Swiss Federal Institute of Technology (EPFL). He earned his PhD in computer science at the University of Queensland, while assisting various research projects carried out by NICTA as well as Aalborg university in Denmark. His research areas cover a wide spectrum of data- and computing-intensive systems in computer science: databases, sensor networks, data mining, cloud computing, distributed systems, and in-memory computing.



Hua Lu received the BSc and MSc degrees from Peking University, China, in 1998 and 2001, respectively, and the PhD degree in computer science from National University of Singapore, in 2007. He is an associate professor in the Department of Computer Science, Aalborg University, Denmark. His research interests include databases, geographic information systems, as well as mobile computing. Recently, he has been working on indoor spatial awareness, complex queries on spatial data with heterogeneous attributes, and location privacy in mobile services. He has served on the program committees for conferences and workshops including ICDE, ACM GIS, SSTD, MDM, PAKDD, APWeb, and MobiDE. He is PC cochair or vice chair for ISA 2011, MUE 2011 and MDM 2012. He is a member of the IEEE.

Saket Sathe received the bachelor's degree in instrumentation and control engineering from Shivaji University, India in 2002, the master's degree in electrical engineering from IIT Bombay, India in 2006, and the PhD degree in computer science from EPFL, Switzerland in 2013. He is currently a post-doctoral researcher at the Distributed Information Systems Laboratory at EPFL. His research focuses on the management of time-series data, mobile/sensor data mining, probabilistic databases, and distributed computing.

Man Lung Yiu received the bachelor's degree in computer engineering and the PhD degree in computer science from the University of Hong Kong in 2002 and 2006, respectively. Prior to his current post, he worked at Aalborg University for three years starting in the Fall of 2006. He is now an assistant professor in the Department of Computing, Hong Kong Polytechnic University. His research focuses on the management of complex data, in particular query processing topics on spatiotemporal data and multidimensional data.