Probabilistic Inference over RFID Streams in Mobile Environments

Thanh Tran†, Charles Sutton‡, Richard Cocci†, Yanming Nie†, Yanlei Diao†, Prashant Shenoy †

†*University of Massachusetts, Amherst* ‡*University of California, Berkeley*

***Abstract*— Recent innovations in RFID technology are enabling large-scale cost-effective deployments in retail, healthcare, pharmaceuticals and supply chain management. The advent of mobile or handheld readers adds significant new challenges to RFID stream processing due to the inherent reader mobility, increased noise, and incomplete data. In this paper, we address the problem of translating noisy, incomplete raw streams from mobile RFID readers into clean, precise event streams with location information. Specifically we propose a probabilistic model based on Dynamic Bayesian Networks to capture the mobility of the reader, object dynamics, and noisy readings. Our model can self-calibrate by automatically estimating key parameters from observed data. Based on this model, we employ a sampling-based technique called particle filtering to infer clean, precise information about object locations from raw streams from mobile RFID readers. Since inference based on standard particle filtering is neither scalable nor efficient in our settings, we propose three enhancements—particle factorization, spatial indexing, and belief compression—for scalable inference over large numbers of objects and high-volume streams. Our experiments show that our approach outperforms a state-of-the-art cleaning approach such as SMURF by XXX% while also being scalable and efficient.**

I. Introduction

RFID deployments have become popular in domains such as retail management [16], healthcare [16], pharmaceuticals [16], and library management [12], [30]. RFID applications enable unique identification of every tagged object and provide real-time monitoring and tracking capabilities. While early RFID deployments used fixed, wall-mounted readers, technology advancements have added mobile, handheld readers to the mix, significantly complicating the tasks of stream processing and querying of RFID data. Recent research has adopted general stream query processing to encode application information needs as declarative queries and evaluate these queries over real-time RFID streams [15], [30], [36], [34]. However, raw RFID data, particularly from mobile readers, is unsuitable for direct querying for two reasons:

- **Data is incomplete and noisy**. Despite technological advances, RFID readings continue to be noisy. Observed read rates in actual deployments are significantly below 100% [20], largely due to the intrinsic sensitivity of radio frequencies (RFs) to environmental factors such as interference from nearby metal objects [13] and contention among tags [14]. Also, it is hard to estimate the data quality in advance, because the read rate depends greatly on the particular characteristics of the deployment. Mobile readers can produce even more noisy data than static readers since they produce readings from arbitrary handheld orientations.
- **Observed data reveals data of interest only indirectly**. Raw RFID readings only contain tag identities and do not contain additional high-level information such as object locations or containments that are needed by tracking and monitoring applications. require locations to be associated with To infer locations, a reading only reveals that the object was in the vicinity of a reader when it was sensed. If the reader location is fixed and known, these readings provide indirect coarse-grained location information for objects. When mobile readers are used instead, their locations vary over time and can be uncertain. Therefore, feeding readings from such readers into a stream processor does not allow a tracking and monitoring application to answer even simple queries concerning object locations.

For these reasons, RFID data streams, particularly those from mobile readers, are not "readily queriable". We further note that hardware technology advances are not expected to address this problem for the foreseeable future. This is because RFID technology is inherently designed for identification rather than high-level information such as locationing or containment. Even with multiple closely-spaced wall-mounted readers, the precision of the acquired information is insufficient for tracking and monitoring tasks such as identifying misplaced inventory in retail stores and computing density of flammable objects in each square foot area [16], [36]. Instead, software solutions are needed to enable rich stream query processing for tracking and monitoring and to ultimately realize the promise of mobile RFID technology.

Therefore, in this paper, we address a fundamental **data cleaning and transformation problem for mobile RFID data streams**, which translates noisy, raw data streams from mobile RFID readers into clean, precise queriable event streams with location information. Informed by the demand of subsequent stream query processing, our work aims to meet three objectives: (i) high precision results of data cleaning and transformation, which are able to simplify probabilistic query processing as shown in [**?**], (ii) producing such results at stream speed, (iii) scaling this process for large tracking and monitoring environments.

Recent research on RFID data cleaning [15], [20] has proposed building an abstraction of device data appropriate for further query processing. This approach focuses on the simpler problem of whether an object is in the (large) read range of a static reader. These smoothing techniques, when applied to mobile readers, provide information such as location only with limited precision, as we shall show in the performance analysis of this paper. The RFID data transformation component of [**?**]

generates low resolution location data, such as a person in a particular office, which is inadequate for many tracking and monitoring tasks as mentioned above. Moreover, both of these studies consider at most one hundred RFID-tagged objects and lack scalable solutions for large-scale environments such as typical warehouses.

In this paper, we present a novel approach for efficient, scalable cleaning and transformation of mobile RFID data streams while offering high precision results. Our approach is based on the view that applications want to query against facts about the true state of the physical world, but these facts are revealed only indirectly through a sensing process that, even for the data that can be generated, is lossy and noisy. The task of data cleaning and transformation is essentially to recover the facts necessary for query processing while mitigating the effects of data loss and sensing noise. Toward this goal, we employ a principled probabilistic approach to (1) *model precisely how mobile RFID data is generated from those facts about the physical world* and (2) *infer likely estimates of the facts as noisy, raw data streams arrive*.

While probabilistic inference is a well-established research area, applying it to clean and transform RFID data streams while meeting the three aforementioned objectives, namely, high precision, stream speed, and scalability, pose considerable challenges. By way of addressing these challenges, we make the following contributions:

Modeling the data generation process (§III). First, we design a probabilistic model that captures the underlying data generation process, including the key components such as reader motion, object movement, and noisy sensing of these objects by the reader. In particular, our model employs a flexible parametric RFID sensor model that can be automatically and accurately configured for a variety of environments using a standard learning technique. In contrast, existing projects resort to manual calibration of the sensor model for each RFID deployment environment [19], [12], [20], precisely because they lacked such a flexible parametric sensor model.

Efficient, scalable inference (IV). To infer object locations from noisy, raw data streams, we apply a sampling-based inference technique, called particle filtering, to the probability distribution developed above. Unfortunately, this inference technique requires a prohibitively large number of samples to cope with the number of objects typical in our target environment, hence inadequate for stream processing. Our second contribution is to enhance particle filtering to *scale to large numbers of objects* and *keep up with high-volume streams* while offering high precision inference results. To do so, we develop a suite of advanced techniques, including particle factorization, spatial indexing, and belief compression. These techniques lead to a solution that uses only a small number of samples at any instant by focusing on a subset of the objects, while maintaining high inference accuracy.

Prototyping and evaluation (§V). Our third contribution is a prototype implementation and detailed performance evaluation of our system for translating mobile RFID data streams into clean, precise event streams with location information. Our results of running a location update query over both real-world traces and large-scale synthetic data show that (1) with an automatically and accurately configured model, our approach estimates object locations with high accuracy, e.g., within a range of a few inches, (2) our approach outperforms the state-of-the-art RFID data cleaning technique [20] by XXX%, (3) our system is robust to noise in both observed tags and observed reader locations, and (4) our system is the first to scale to tens of thousands of objects with small memory usage and at a constant rate of XXX readings per second, which has reached the maximum rate at which a mobile RFID reader can produce readings. In contrast, naive particle filtering can process only 0.1 reading per second when given 20 objects while striving to achieve comparable accuracy.

II. Problem Statement and Overview

In this section, we present a problem formulation, illustrate how this enables rich stream query processing, and briefly discuss our approach.

A. Problem Statement

Given a stream of raw readings of RFID tags and a sequence of reader locations, both of which can be noisy, we wish to derive a clean, precise and queriable event stream where RFID tag observations are augmented with the locations of the corresponding objects. This high-level problem can be further described using the underlying physical world, the data streams from a mobile reader, and the desired output stream.

The Physical World. The physical world being monitored is a large storage area comprising shelves $\mathcal{S}$ and a set of objects $\mathcal{O}$. Both shelves and objects are affixed with RFID tags. Since the shelves are at fixed locations, we assume that the precise locations of their tags are also known a priori. However, the object locations are unknown and must be determined as part of the cleaning and transformation process. Typically, objects stay on the same shelf but can sometimes move from one shelf to another. The facts of interest to the application are the (x, y, z) location of each object O_i at each time instant t.[1]

A mobile RFID reader provides the only means to observe the physical world. Mobile readers come in two flavors—handheld readers that are used by humans to scan and monitor tagged objects (e.g., on store shelves), and readers that be mounted on robots for automated monitoring and order processing (e.g., Kiva systems [22]). The mobile reader periodically scans the storage area. In each round, the reader produces readings that contain the tag ids of observed objects (usually a subset of $\mathcal{O}$) and tag ids of observed shelves (also a subset of $\mathcal{S}$). In addition, the (x, y, z) location of the reader itself at time t is computed using a positioning technology such as indoor GPS or ultrasound [33].

Data Streams from Mobile Readers. Various readings from a mobile reader have the following characteristics:

No Information about object locations. Since an RFID stream only consists of a sequence of tag ids and observation times, the locations of objects are not observed directly.

[1] In this work, we assume that the facts of interest to the monitoring application only consist of object locations. The extension to also include inter-object relationships is a main task of our future work.

Noisy object readings. Object readings are highly noisy. First, if an object is on the boundary of the sensing area, in what is called the *minor detection range*, the read rate is far less than 100%. Even if the object is close to the reader, in what is called the *major detection range*, objects can be missed due to environment factors such as occluding metal objects, interference from other electronic devices, and contention among tags. Sometimes objects can be read unexpectedly due to reflection of radio waves by obstructing objects. Finally, mobile readers have greater noise and lower read rates than fixed readers—mobile readers tend to read objects from arbitrary orientations, and certain orientations can result in poor read rates.

Uncertainty in reader locations. The exact reader location is usually uncertain. For example, even when handheld readers are coupled with indoor positioning systems such as ultrasound locationing, the reported locations are imprecise (e.g., accuracy is about tens of centimeters for moving objects [33]). As another example, a robotic reader can measure its location using dead reckoning, essentially by counting the number of times that its wheels have revolved. But such location estimates may contain significant noise because the robot can drift sideways due to inertia or forward due to wheel slippage, as we observed in our lab deployment (detailed in §V-C).

While the exact data format varies with the reader, in this work we assume that readings are produced in two separate streams: the RFID reading stream has readings (`time, tag_id of object` O_i `or tag_id of shelf` S_j) and the reader location stream has reports (`time, (x, y, z)`). In practice, these streams may be slightly out-of-sync in time. In our model, however, a time step (also called an *epoch*) is fairly coarse-grained, e.g., a second. This allows us to generate synchronized streams via simple low-level processing, such as assigning the same time to RFID readings produced in one epoch and taking average of multiple location updates in an epoch to produce a single update. Therefore, we consider only synchronized streams in the rest of the paper.

Output Event Stream. Our goal is to translate noisy, primitive data streams from a mobile RFID reader into clean, precise event streams with location information. In the output stream, each event reports the location of an object as follows: (`time, tag_id of` O_i`, (x, y, z) of` O_i`, (statistics)?`). Events are output for not only observed objects but also objects with missed readings. In other words, the output stream not only augments the input streams with object locations but also mitigates the effect of missed readings.[2] In addition, the optional statistics field can be used to report summary information of the estimated location distribution, such as its variance or confidence regions.

Finally note that as the reader moves, it may observe an object several times from different locations. Combining such multiple readings provides valuable information about the object location. To avoid fluctuating values in the output, our system outputs an event for an object only at particular points: for example, within x seconds after an object was read, upon completion of a shelf scan, or upon completion of a full area scan. The choice of when to output reports is left to the discretion of the application.

[2] While transforming raw data streams into an event stream, we can also archive the raw streams for post-facto analysis.

B. Support for Stream Query Processing

We next illustrate the rich stream query processing that our event stream enables but raw streams from mobile RFID readers do not. We write our example queries in the CQL stream query language [1]. The first query reports the location change of each object. It simply reads the event stream, considers the most recent location report of each object, and if the location differs from the previous one, outputs the tag id and the new location of the object.

```
Select Istream(E.tag_id, E.(x, y, z))
From   EventStream E [Partition By tag_id Row 1]
```

The second query detects potential violations of a fire code: display of solid merchandise shall not exceed 200 pounds per square foot of shelf area.

```
Select   Rstream(E2.area, sum(E2.weight))
From     (Select Rstream(*,
                 SquareFtArea(E.(x, y, z)) As area,
                 Weight(E.tag_id) As weight)
          From   EventStream E [Now])
         E2 [Range 5 seconds]
Group By E2.area
Having   sum(E2.weight) > 200 pounds
```

The nested `Select-From` query simply adds two attributes to each event: the square foot area that each object belongs to, computed by a function on its (x, y, z) location, and the weight of the object, retrieved by another function using its tag id. Then the outer query considers events in each 5 second window, groups them based on the square foot area, computes the total weight of the objects in each group. For the groups with the total weight greater than 200 pounds, it reports the area and the total weight in output.

Crucially, both of these queries require reliable knowledge of the object location, which is unavailable without processing and transforming the raw data streams. While the focus of this paper is not sophisticated probabilistic query processing such as [**?**], we view our work as a crucial data cleaning and transformation step that enables such query processing over real-world RFID data streams.

III. A Probabilistic Data Generation Model

In this section, we present a probabilistic model that captures how raw data streams are generated by a mobile RFID reader from the true state of the world. Given the complexity of the problem, our model incorporates the motion of the reader, the object dynamics, and most importantly, the noisy sensing of objects and reader locations.

Formally, the world is modeled as a vector of **random variables**, which are represented as nodes in Figure 1. There are two types of variables: *evidence variables* that we observe in the data, and *hidden variables* that we wish to infer from the information contained in the evidence. In our application, the hidden variables are the true reader location R_t and the object

R_t	True reader location at time t. Vector containing (x, y, z) position and orientation.
$\hat{R}_t$	Noisy observation of reader location at time t.
O_{ti}	True location of object i at time t. Vector containing (x, y, z) position.
$\hat{O}_{ti}$	Binary variable indicating whether object i is observed at time t
S_i	True location of shelf tag i
$\hat{S}_{ti}$	Binary variable indicating whether shelf tag i is observed at time t
$\mathbf{R}$	Matrix of all true reader locations $[R_1 R_2 \dots R_T]$
$\hat{\mathbf{R}}$	Matrix of all observed reader locations $[\hat{R}_1 \hat{R}_2 \dots \hat{R}_T]$
$\mathbf{O}_t$	Matrix of all true object locations at time t
$\mathbf{O}$	Matrix of all true object locations at all time steps
$\hat{\mathbf{O}}_t$	Binary vector $[\hat{O}_{t,1} \dots \hat{O}_{t,M}]$ of all readings at time t
$\hat{\mathbf{O}}$	Matrix of all object readings at all time steps

TABLE I

SUMMARY OF NOTATION IN THIS PAPER.

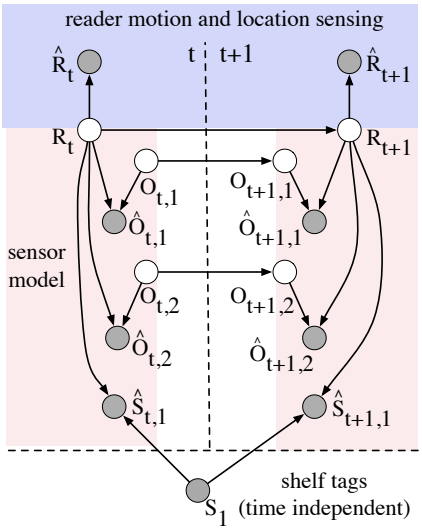


Fig. 1. Model of reader and object locations. The shaded region at top contains the reader motion model and reader location sensing model. The lightly-shaded region at bottom contains the RFID sensor model.

locations O_{ti}, which are represented by the unshaded nodes in Figure 1. The evidence variables are the reported reader location $\hat{R}_t$ and the object readings $\hat{O}_{ti}$, which are indicated by the shaded nodes in Figure 1. (For definitions of all the notation used in this section, see Table I.)

The goal of this section will be to define a joint probability distribution $p(\mathbf{R}, \mathbf{O}, \hat{\mathbf{R}}, \hat{\mathbf{O}})$ over both hidden and observed variables. Then, given observed values $\hat{\mathbf{R}}$ and $\hat{\mathbf{O}}$, this joint model induces a conditional distribution $p(\mathbf{R}, \mathbf{O}|\hat{\mathbf{R}}, \hat{\mathbf{O}})$ over the true locations, which can be used to predict the objects' locations. We describe various components of our model in Section III-A, how we combine them into a single joint distribution in Section III-B, and how we estimate model parameters in Section III-C.

A. Components of the Model

Our joint model over the entire world is divided into four components that separately model different aspects of the domain. We explain each of the components in detail below.

RFID sensor model: Given that the read rate of an RFID reader is less than 100%, it is natural to model the reader's sensing region in a probabilistic manner: each point in the sensing region has a non-zero probability that represents the likelihood of an object being read at that location. To determine the probabilistic values for different points, we can represent the sensing region as the *likelihood of reading a tag* based on the factors including the distance and angle to the reader. Since the sensor noise varies with time and location, it is also possible to introduce other parameters into the likelihood, and estimate those parameters based on the specifics of the deployment environment.

Formally, we introduce a flexible parametric model that describes how the read rate of an RFID reader decays with distance and angle. Given the true location R_t of the reader and O_{ti} of the object i, the sensor model is a conditional distribution $p(\hat{O}_{ti}|O_{ti}, R_t)$ that models the probability of reading the tag. If we denote the reader location by the vector $[r_t^x, r_t^y, r_t^z]$, and the reader angle in relation to the reference coordinate frame by r_t^ϕ, then we can compute the distance d_{ti} and the angle θ_{ti} between the reader and the tag as follows:

$$\begin{aligned} \delta &= O_{t,i} - [r_t^x, r_t^y, r_t^z] \\ d_{ti} &= \sqrt{\delta^T \delta} \\ \cos\theta_{ti} &= \frac{\delta^T [\cos r_t^\phi, \sin r_t^\phi]}{d_{ti}} \end{aligned}$$

Empirically, we have found that the read rate decreases approximately quadratically with distance and with angle, so that the probability can be written as a function like $\sum_{c=0}^{2} a_c (d_{ti})^c + \sum_{c=1}^{2} b_c (\theta_{ti})^c$, where the $\{a_c\}$ and $\{b_c\}$ are coefficients that we expect to be negative. But strictly speaking this quadratic function cannot be a probability distribution, because it is not restricted to $[0, 1]$. To fix this, we compose the quadratic function with the sigmoid function $f(x) = 1/(1 + \exp\{-x\})$, which has the effect of squashing a real number into the interval $(0, 1)$. This transformation yields the *logistic regression* model, which is a standard technique for probabilistic binary classification from the statistics literature. Putting this together, the sensor model is:

$$p(\hat{O}_{ti} = 0|d_{ti}, \theta_{ti}) = \frac{1}{1 + \exp\{\sum_{c=0}^{2} a_c (d_{ti})^c + \sum_{c=1}^{2} b_c (\theta_{ti})^c\}}. \quad (1)$$

The coefficients a_c and b_c are real-valued model parameters that are learned from data in a calibration step, discussed in Section III-C. We use the same sensor model for both the object tags and the shelf tags. The only difference is that for the shelf tags, we write the distribution as $p(\hat{S}_i = 0|d_{ti}, \theta_{ti})$, but the same model and coefficients are used in both cases.

As our results in Section V show, our sensor model is a flexible parametric form that can fit a variety of sensing regions, including conical and spherical regions (examples of learned sensor models are shown in Figure 5(b)-5(d)).

Reader motion model: This model describes how the reader moves. We assume that the reader moves with a constant velocity that varies somewhat over time. In other words, the new location is the old location plus a noisy version

of the average velocity. Formally, the new location R_t can be computed from the old location as $R_t = R_{t-1} + \Delta + \epsilon$, where Δ is the average velocity of the reader, and ϵ is the noise. The motion noise ϵ is a Gaussian random variable with mean 0 and diagonal covariance matrix Σ_m.

Reader location sensing model: This model describes the noise in our observations of the reader's location. For example, an RFID-equipped robot may compute its location by dead reckoning, that is, basically by counting how many times its wheels have revolved. We assume that this measurement noise is Gaussian with mean μ_s and covariance Σ_s. A more complex noise model is not necessary here, because errors in the reader location can be corrected by information from the static shelf tags as shown in our experiments in Section V.

Object location model: We further model the fact that objects can change locations in a warehouse scenario. At each time t, we say that an object changes its location with probability α, in which case the new location is distributed uniformly across all shelves; otherwise, the object location does not change. We write this model as a conditional distribution $p(O_{ti}|O_{t-1,i})$. This model contains no distinguishing information about the object's new location, but such information is not needed here: The object location model is used to temporarily create samples that will be weighted based on the actual observations in the inference process; the new object location will be eventually inferred from the readings from that location (details of this process are given in Section IV-A).

B. Formal Definition

Now we describe how the component models can be combined to define a joint model over the entire domain. By way of illustration, we first describe *how the data would be generated if the world behaved according to our model*: Assume that the initial reader location R_1 is known. Sample initial object locations $\mathbf{O}_1$ from a uniform distribution over the shelf. Then for each time step t, perform the following five steps. (1) Generate the new reader location R_t from the previous location R_{t-1} by sampling from the reader motion model $p(R_t|R_{t-1})$. (2) Generate a noisy observation $\hat{R}_t$ of the reader location from the reader noise model $p(\hat{R}_t|R_t)$. (3) Generate new object locations $\mathbf{O}_t$ from the object location model $p(\mathbf{O}_t|\mathbf{O}_{t-1})$. (4) Decide whether each object is observed using the sensor model. Each object i is observed with probability $p(\hat{O}_{ti}|R_t, O_{ti})$. (5) Decide whether each shelf tag is observed using the sensor model. Each shelf tag i is observed with probability $p(\hat{S}_{ti}|R_t, S_t)$.

Now we give the formal description of our model. Any distribution that can be sampled in the manner above can also be factorized into the following product of local probability distributions:

$$\begin{aligned} p(\mathbf{R}, \hat{\mathbf{R}}, \mathbf{O}, \hat{\mathbf{O}}|\mathbf{S}) = & p(R_1, \mathbf{O}_1) \prod_t p(R_t|R_{t-1}) p(\hat{R}_t|R_t) \\ & \times \prod_{i \in \mathcal{O}} p(O_{ti}|O_{t-1,i}) p(\hat{O}_{ti}|R_t, O_{ti}) \\ & \times \prod_{i \in \mathcal{S}} p(\hat{S}_{ti}|R_t, S_t). \end{aligned} \quad (2)$$

The factorization of (2) can be depicted as a directed acyclic graph called a *directed graphical model* or a *Bayesian network*, as shown in Figure 1. Our model is a particular case of a dynamic Bayesian network (DBN) [26], but with conditional probability functions specially designed for our problem.

C. Parameter Estimation using Learning

We next describe the self-calibration step in which we estimate the model parameters from data. The parameters of our model are the coefficients $\{a_c\} \cup \{b_c\}$ of the sensor model, the average reader velocity Δ, the variance Σ_m of the reader velocity, and the mean μ_s and variance Σ_s of the noise in reader location sensing. The sensor model in particular depends not only on the type of reader used, but also on the specifics of the environment such as metal objects and density of tags. One way to calibrate the sensor model is to perform calibration in the lab [19], [12], [20], in which the read rate is measured when a reader and an RFID tag are placed at various known distances and angles. Such manual lab calibration is not only tedious but also inaccurate in real deployments due to the change of environmental factors in those deployments.

An important benefit of having a flexible parametric model is that we can automatically learn the model parameters using a small training data set collected from the same environment in which the system is to be fielded. The training set includes the observed reader locations and readings of a small set of tags, some of which are shelf tags with known locations. We perform parameter estimation using Expectation-Maximization (EM), a standard method for parameter estimation in the presence of hidden variables [3]. In Section V, we show that only a small number of shelf tags (e.g., less than 20) are needed to learn accurate sensor models.

IV. Efficient, Scalable Inference over Streams

As noisy, raw data streams emanate from a mobile RFID reader, the task of translating them into a clean, precise event stream with location information is treated as an inference process in our work. Inference is essentially to estimate the true locations of objects for each time t even if there are no readings returned for some of the objects. Formally speaking, from the joint distribution $p(\mathbf{R}, \mathbf{O}, \hat{\mathbf{R}}, \hat{\mathbf{O}})$ defined previously over both the physical world and noisy readings, inference is to compute the conditional distribution $p(\mathbf{R}, \mathbf{O}|\hat{\mathbf{R}}, \hat{\mathbf{O}})$. This conditional distribution can be used to predict true object locations and optionally the true reader location.

Exact inference for our model is very challenging, because the true conditional distribution has a complex shape. Instead, we sample from the distribution approximately using a generic machine learning algorithm called *particle filtering*. How to apply particle filtering to our particular problem is described in Section IV-A. However, a naive implementation of particle filtering does not scale to the enormous number of objects that would be expected in a real warehouse. To handle this, we augment the basic algorithm in three novel ways:

- We first propose an advanced technique, *particle factorization*, to reduce the number of samples needed for accurate inference for a large number of objects (Section IV-B).

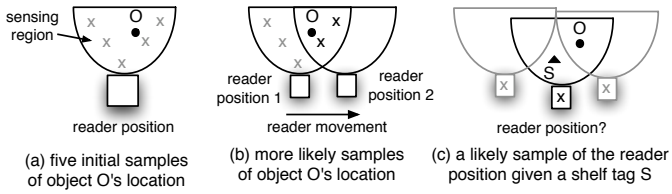


Fig. 2. Weighting samples of object and reader locations.

- We then augment the factorized particle filter with spatial indexing structures to limit the set of objects that are actually processed at each time step (Section IV-C).
- At some point in inference, the particles for an object may stabilize in a small region. In this case, we compress the particle representation of the object's location into a parametric distribution to save both space and time (Section IV-D).

A. Particle Filtering

In this section, we describe the main intuitions behind sampling-based inference for our problem. We also give a formal description of how the generic particle filtering algorithm [11] is applied to our particular probability distribution, which provides a technical context for our later extensions.

The basic idea is to maintain a weighted list of samples, each of which contains a hypothesis about the true location of each object as well as a hypothesis about the true reader location. Each sample has an associated weight, representing the likelihood of the sample being true. The weight of a sample is assigned based on the following intuitions.

As Figure 2(a) shows, if a reader detects the tag of object O once, the tag must be in the vicinity of the reader. We can generate multiple samples about the tag location in the reader's sensing region (or a slightly larger area) but cannot further distinguish these samples. However, if the reader detects the tag again from a nearby position, then the samples that reside in the intersection of the sensing regions at the two reader positions will be assigned higher weights (Figure 2(b)). Regarding the reader location, samples are weighted based on the likelihood of seeing all observed objects from that location. Of particular importance are the shelf tags with known locations. As Figure 2(c) shows, an observed shelf tag S can be used to distinguish good samples of the reader location, from which the reader can detect the shelf tag, from those bad samples of the reader location, from which the reader cannot.

At the next time step, these samples are updated to reflect expected changes of reader and object locations. Their weights are adjusted based on the new observations from that step. At any point, we can use this weighted list of samples as a distribution over the hidden variables, i.e., the true object locations and reader location, given the observations—exactly the result that inference aims to compute.

Formally, we denote a set of samples (termed *particles* in the literature) at time t using $x_t^1, \cdots, x_t^{(J)}$. We denote the j-th particle by a vector $x_t^{(j)} = (R_t^{(j)}, O_{t,1}^{(j)}, \ldots, O_{t,n}^{(j)})$, where $R_t^{(j)}$ is a hypothesis about the reader location and $O_{t,i}^{(j)}$ is a hypothesis about an object location. Let the weight of $x_t^{(j)}$ be $w_t^{(j)}$. The particle filtering algorithm in our application is:

Step 1 Initialization. Generate a set of initial particles $\{x_1^{(j)} | j = 1 \ldots J\}$ from the prior distribution $p(R_1, \mathbf{O}_1)$.

Step 2 Update. Let the vector y_t contain all of the observations at time t. Then for each time step t:

- *Sampling*. For each particle $x_{t-1}^{(j)}$, generate a new particle $x_t^{(j)}$ from a *proposal distribution* $q(x_t | x_{t-1}^{(j)}, y_t)$. The proposal distribution is an arbitrary distribution chosen to be easy to sample from. In this work, we use the reader motion model and object location model for sampling.
- *Weighting*. Compute a new particle weight

$$w_t^{(j)} = C w_{t-1}^{(j)} \cdot \frac{p(x_t^{(j)} | x_{t-1}^{(j)}, y_t)}{q(x_t^{(j)} | x_{t-1}^{(j)}, y_t)}, \tag{3}$$

  where C is a constant with respect to j, chosen so that $\sum_j w_t^{(j)} = 1$. This weight adjusts for the fact that the particles were sampled from the proposal distribution, rather than the true distribution of the model.
- *Re-sample* from the particles to reproduce the highest-weight ones. Each of the new particles is selected by sampling from the set of old particles with replacement. A particle is selected with probability equal to its weight.

Step 3 Inference output. At any time step, the posterior distribution over the hidden variables can be estimated by a weighted average of the particles. More formally,

$$p(O_{ti} \,|\, \hat{\mathbf{R}}_{1\ldots t}, \hat{\mathbf{O}}_{1\ldots t}) \approx \sum_{j=1}^{J} w_t^{(j)} \mathbf{1}_{\{O_{ti} = O_{ti}^{(j)}\}}, \tag{4}$$

where $\mathbf{1}_{\{a=b\}}$ is an indicator function that is 1 if and only if $a = b$. A similar formula is used for the reader location. From these distributions, it is easy to compute any desired statistics, such as the mean, the variance, or a confidence region.

Sensible initialization of the particles is also important, because otherwise many samples will begin far away from the object's actual location. In this work, we create new particles for an object when we see it the first time or at a location far away from the previous location of observing it. At the current location, we initialize the particle locations from a uniform distribution over a cone originating at the reader location. The width of the cone is chosen to be an overestimate of the true range of the reader. We call this initialization *sensor-model based* initialization.

B. Particle Factorization

So far every particle has included a sample of the locations of all objects. To get good accuracy, intuitively we expect to use a large number of particles in the number of objects (which was observed in our experiments presented in Section V-D). This is because even if a particle contains good location estimates for some objects, it may contain bad locations for other objects, simply through random chance in the sampling procedure. Figure 3(a) illustrates an example of this: Particle A (the dark stars) contains a good sampled location for Object 1 but not for Object 2. On the other hand, particle B (the light stars) contains a good sampled location for Object 2 but not Object 1. As the number of objects grows, it becomes more likely that most particles will happen to have sampled a bad

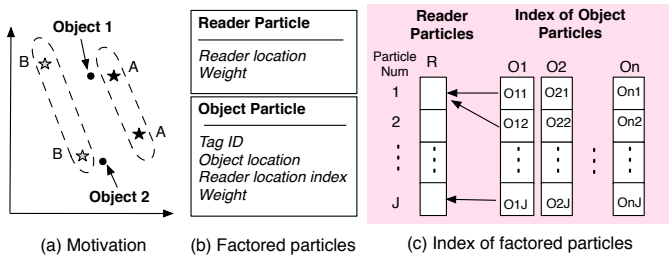


Fig. 3. Motivation and data structures for factored particles.

location for some object. One way to overcome this problem is simply to use more particles, but this becomes prohibitively expensive when there are large numbers of objects.

In this section, we introduce an advanced technique that enables the particle filter to scale dramatically in the number of objects. In this technique, which we call *particle factorization*, we break a large particle over all the objects into smaller particles over individual objects. This allows us to combine good particles from different objects and, essentially, to represent an exponentially large number of unfactored particles in the amount of space linear in the number of objects. The challenge is to ensure that the operations required by the particle filter can still be performed in this factored representation.

First we explain the data structures that we use to maintain these factored particles. As shown in Figure 3(b), we maintain a list of reader particles, each of which contains a hypothesis about the reader location and an associated weight. Each object particle contains a hypothesis of the object location and the reader location (a pointer to the reader particle in our implementation), a weight, and the object's tag id. We also maintain an index of object particles that maps from an object's tag id to the list of object particles for that tag id; further, each object particle refers to the corresponding reader particle via the contained pointer (Figure 3(c)).

In addition to maintaining factorized particles, we also maintain factorized weights. Each reader particle $R_t^{(j)}$ has an associated weight $w_{rt}^{(j)}$. The reader particle also has a list of associated object particles $O_{ti}^{(j,1)} \ldots O_{ti}^{(j,K)}$ for each object i. Each of these object particles has a weight $w_{ti}^{(jk)}$. The semantics of the factored weights is: If we were to expand the factored representation into the exponentially-long list of unfactored particles, then the weight of the unfactored particle is the factored reader weight times all of the factored object weights. In our factorized particle filter presented in the rest of this section, we manipulate these weights without actually constructing unfactored particles.

Now we explain how these data structures can be used to efficiently implement the factorized particle filter. First, the sampling step can be performed entirely on the factored representation. To sample from the proposal distribution, first for each reader particle, we sample a new reader location from the reader motion model, and then for for each associated object particle, we sample a new object location from the object location model.

Second, the weights of the new particles can also be computed in a factored manner. The important point is that in the factored representation, the weight of a particle for object i does not depend on weights of particles for any other objects. To compute the new weights, the new incremental weight for each reader particle $w_{rt}^{(j)}$ can be computed as $p(\hat{R}_t|R_t^{(j)}) \prod_{i \in \mathcal{S}} p(\hat{S}_{ti}|R_t^{(j)}, S_t)$. The new incremental weight for an object particle $O_{ti}^{(j,k)}$ is $p(\hat{O}_{ti}|R_t^{(j)}, O_{ti}^{(j)})$.

It can be shown that this weighting step is equivalent to the standard particle filtering weight step applied to the full set of unfactored particles. Mathematically, this is because our proposal distribution and our model factorize in the same way as our data structures do. To see this, consider the weight update (3) for unfactored particles:

$$\begin{aligned} w_t^{(j)} &= C w_{t-1}^{(j)} \cdot \frac{p(R_t^{(j)}, \mathbf{O}_t^{(j)} | R_{t-1}^{(j)}, \mathbf{O}_{t-1}^{(j)}, \hat{R}_{t-1}, \hat{\mathbf{O}}_{t-1})}{q(R_t^{(j)}, \mathbf{O}_t^{(j)} | R_{t-1}^{(j)}, \mathbf{O}_{t-1}^{(j)}, \hat{R}_{t-1}, \hat{\mathbf{O}}_{t-1})} \\ &= C w_{t-1}^{(j)} p(\hat{R}_t|R_t^{(j)}) \prod_{i \in \mathcal{S}} p(\hat{S}_{ti}|R_t^{(j)}, S_t) \prod_{i=1}^{N} p(\hat{O}_{ti}|R_t^{(j)}, O_{ti}^{(j)}) \\ &= C w_{t-1}^{(j)} \cdot w_{rt}^{(j)} \prod_{i=1}^{N} w_{ti}^{(j)} \end{aligned} \tag{5}$$

where in the second line, we substitute definitions; and in the last line we simply define $w_{rt}^{(j)}$ and $w_{ti}^{(j)}$ to be the corresponding terms from the previous equation. This equation shows that that the weights can be computed separately for each object, with the same result as if the weight had been computed for the exponential number of unfactored particles that is implicit in our representation.

Finally, performing resampling in this representation is a non-trivial task. For the object particles, we resample them in the usual way while ensuring that their pointers to the associated reader locations are preserved. On the other hand, when we resample reader locations, we instrument resampling to favor reader particles that are associated with good object particles. Details of the resampling algorithm are omitted due to space limitations (but are available in [**?**]).

Our factorization scheme is related to that of [27], but with several important differences. The main difference is that our particle weights are also factorized, while the previous work ignores the weights entirely by resampling at every time step. By maintaining factorized weights, our method avoids both the cost of resampling at most time steps, and the bias introduced by resampling in the factorized representation.

C. Spatial Indexing

Even with factored particles, the inference algorithm presented so far must process all the objects in the world at every time step. This is because the weighting step described in Section IV-A is performed for all objects, whether their tags were read or not. In this section, we introduce spatial indexing as a further approximation that dramatically reduces the processing cost. It is important to note that spatial indexing is possible only after the particles have already been factorized.

The main insight is that even if the number of objects is large, only a much smaller number of them are near the reader at any given time. If we can restrict the processing to only those objects near the reader, a significant amount of work

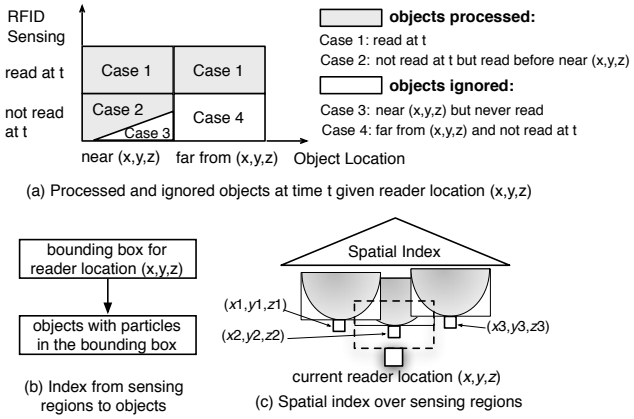


Fig. 4. Intuitions and data structures for spatial indexing.

can be saved. This intuition is more precisely described by the diagram in Figure 4(a), which classifies objects based on their distance from the reader location at time t (x axis) and the result of RFID sensing at t (y axis). There are four cases:

Case 1: If an object is read at time t, no matter how far it is from the reader, it should be processed in inference.

Case 2: If an object is not read at t but was read before near the current reader location, the object needs to be processed so that the particle filter can downweight the particles of the object that are very close to the current reader location.

Case 3: If an object is near the reader but has never been detected from its current location, it is simply invisible to the inference procedure since RFID sensing is the only means of observing the world.

Case 4: Last, the object is far from the reader and indeed not detected at t. According to our sensor model, such objects have a very small (but nonzero) read probability, but rounding this probability to zero appears to be a good approximation.

Therefore, we design a spatial index to distinguish Case 2 from Case 4 so that we can save work for objects belonging to Case 4. For each reported reader location, we construct a bounding box of the sensing region. Then our index has two components. The first component, shown in Figure 4(b), maps from bounding boxes to the set of objects that have at least one particle within the bounding box. The second component, shown in Figure 4(c), is a standard spatial index (a simplified R*-tree [2]) over the bounding boxes.

At each time during inference, we construct a bounding box of the current sensing region and probe the spatial index to retrieve all potentially overlapping bounding boxes inserted in the past. For each of those boxes, we retrieve all contained objects. This gives us the full set of objects belonging to Case 2. Finally, we run particle filtering as usual, but restrict sampling and weighting only to the objects in Cases 1 and 2.

D. Belief Compression

We next present a compression technique that can be embedded in our factorized particle filter to further reduce space consumption and improve inference speed. Recall that a weighted set of particles for each object defines a distribution over the the object's location. The main advantage of the particle representation is the ability to represent arbitrary distributions. For example, when an object is first detected, its location could be anywhere within a large and oddly-shaped area. But as more readings arrive, often the location particles stabilize to a small region. If this occurs, the object location could be represented much more compactly by a parametric distribution. For example, the particle-based representation may require 1000 particles, but a three-dimension Gaussian requires only 9 real numbers to store its parameters. Therefore, compression to the parametric distribution saves considerable space. Compression can also save time as it often allows inference to use fewer particles on the compressed representation.

Per-object based compression. We first describe how an object's particles can be compressed. Suppose that a weighted set of particles over the location of object i defines a distribution $\hat{p}(O_{t,i})$ as in (4), and we wish to compress this into a Gaussian $q(O_{t,i})$ with mean μ and covariance matrix Σ. This can be done by minimizing the KL divergence $\mathrm{KL}(\hat{p}\|q)$, which is a standard measure of "distance" between distributions. When q is Gaussian, the KL amounts essentially to a weighted average of the squared distance between μ and the particles comprising $\hat{p}$. It can be shown that the optimal choice of q uses the sample mean and empirical covariance matrix, that is, $\mu = \sum_j w_{t,i}^{(j)} O_{t,i}^{(j)}$ and $\Sigma = \sum_j w_{t,i}^{(j)} (O_{t,i}^{(j)} - \mu)(O_{t,i}^{(j)} - \mu)^T$. The KL divergence at these parameters measures how much is lost by compression, in the sense of the expected squared error (e.g., in squared feet) of the resulting Gaussian.

Several methods are possible for choosing individual objects to compress. One possibility is to compress an object whenever its tag has not been read for several time steps. This is applicable if an object leaving the read range means that it will not be observed for a long time. An alternative method is to rank the uncompressed objects by the KL of the compressed representation, and compress the objects that would have the least compression error. This method can be further augmented with a threshold. That is, we only compress the particle representation if the KL is above the threshold.

Decompression (sampling) and re-compression. Later on, when a compressed object has its tag read again, we need to perform the particle filtering steps on the compressed representation. To do this, we sample a small number of particles from the Gaussian to decompress the representation. Empirically, we find that many fewer particles are required for accurate inference after decompression than for the original particle filter, because the compressed representation tends to be well-behaved. When the object leaves scope, if its particles are still well-represented by a Gaussian, it can be re-compressed.

Our idea behind compression is similar to the Boyen and Koller method [4]. However, their method performs compression only, while our method embeds compression within a larger sampling procedure. Moreover, we can employ compression on a per-object basis and hence has greater flexibility to explore the benefits of Gaussian and particle-based representations wherever appropriate.

Comment on accuracy. Theoretically, there are no known results (even in the machine learning community) that can quantify the error from compression of arbitrary distributions like our object location distributions. However, our experi-

ments provide empirical evidence that inference accuracy does not degrade because object particles can indeed stabilize to a small region when compression is applied (see Section V-D).

V. Performance Evaluation

We have implemented all our inference techniques in a prototype system in Java. In this section, we present a detailed analysis of our system using both real RFID traces and large-scale synthetic data. Our results show that (1) our system can offer clean, precise event streams with accurate location information, e.g., within a range of a few inches, and is robust to noise; (2) our system outperforms SMURF [20], a state-of-the-art RFID data cleaning technique by XX%; (3) our system scales to tens of thousands of objects with small memory usage and at a constant rate of XXX readings per second, while naive particle filtering cannot scale beyond 20 objects.

A. Experimental Setup

Query. In all experiments, we ran the location update query described in Section II over the event stream generated by our system. Recall that this query examines the most recent event of each object, and if the location in this event differs from the previous event, outputs the tag id and new object location. We ran this query over both real RFID traces and simulated streams in a warehouse scenario (detailed below). To avoid fluctuating values in output, our system produced a location event 60 seconds after an object was read the first time in one round of scan (although inference was running in real-time).

Metrics. The accuracy of query output can be measured using two related metrics: The first is the average distance between reported object locations and true object locations, called the *inference error*. Assuming the application has a precision requirement, e.g., within half a foot from the true location, the second metric measures the percentage of location updates that fail to satisfy the requirement, called the *error rate*. The performance metric is the average time that our system takes to process each RFID reading.

Simulator. To obtain insight into various factors on performance and perform scalability tests, we developed a simulator for a warehouse scenario that produces synthetic RFID streams with various controlled properties. The simulated warehouse consists of consecutive shelves aligned on the y axis, with objects evenly spaced on the shelves. Both shelves and objects are affixed with RFID tags. For simplicity, we assume the same height for all tags and hence ignore the z axis. An RFID reader is mounted on a robot that moves down the y axis facing the shelves. In every epoch, it travels about 0.1 foot (which can be varied), stops, senses its current location and reads objects on the current shelf with added noise, and sends both its sensed location and the RFID readings to our system.

RFID readings were generated using a cone-shaped sensor model as shown in Fig. 5(a) (where white is for good read rate). The sensor model has a 30 degree open angle for the major detection range, in which the read rate is uniform and its value is controlled by a parameter, RR_{major}, and an additional 15 degree angle for the minor detection range, in which the read rate degrades from RR_{major} down to 0. The parameters for data generation include: (1) RR_{major}, by default 100%, (2) read frequency RF, by default once every second, (3) the Gaussian model for reader motion, whose default is μ_m=0, σ_m=.01 for both x and y dimensions, and (4) the Gaussian model for reader location sensing, whose default is μ_s=0, σ_m=.01 for both x and y. Each trace contains readings obtained from a single pass of scan of all the tags.

Real lab deployment. To evaluate our system in real-world settings, we also created a lab deployment with a robot-mounted RFID reader, which we detail in §V-C.

B. Model Calibration and Initial System Testing

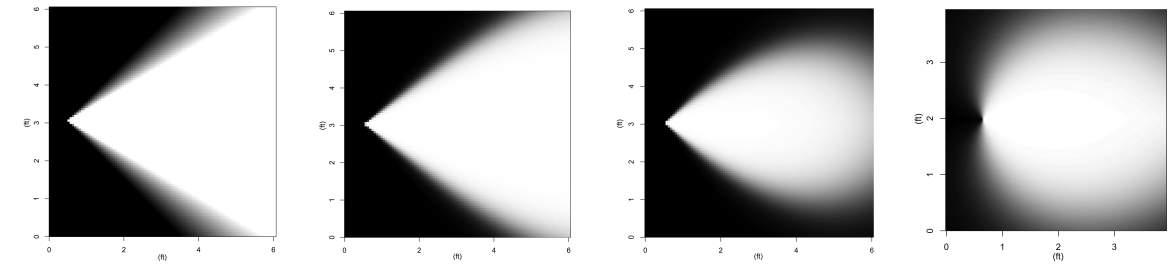
In the section, we evaluate our system for its ability to calibrate the probabilistic model based on the characteristics of the RFID deployment, and test the sensitivity of our system to various factors. We used simulation in this set of experiments. As a baseline, we also ran a method, called *uniform*, that uniformly randomly samples an object's location over the overlapping area of the sensor model and the shelf. This baseline is used as a bound on the worse-case inference error.

Learning RFID sensor model. As noted in Section III-A, the most challenging part of modeling is the sensor model because it varies with the type of reader, environmental noise, etc. To test the flexibility and accuracy of our probabilistic sensor model, we used a small trace consisting of readings of 20 tags for learning the model using EM. To investigate the amount of information needed for accurate learning, we varied the number of tags with known locations, assumed to be shelf tags, in the training data from 0 to 20. When fewer than 20 tags were used as shelf tags, the rest of the tags were treated as object tags whose true locations are unknown.

Fig. 5(a) shows the true sensor model in simulation and Fig. 5(b) and 5(c) show the sensor models learned with 20 shelf tags and 4 shelf tags, respectively. Most importantly, our sensor model learned from 20 shelf tags is very close to the true model. Such approximation degrades only gradually as we reduce the number of shelf tags. When 4 shelf tags or fewer are used, the learned sensor model starts to deviate fast from the true model, because EM in this case is likely to be stuck in some local maxima.

After training, we used the learned sensor models to perform inference over a test trace with 10 object tags and 4 shelf tags, using 1000 particles per object. Most learned models (except those from 0 and 4 shelf tags) result in small inference errors that are comparable to the results using the true model, and much better than the baseline, as shown in Fig. 5(e). This shows that our system can indeed learn accurate sensor models from small traces with a few tags of known locations.

Handling RFID sensing noise. We then investigate the sensitivity of our system to RFID sensing noise by varying the read rate in the reader's major detection range, RR_{major}, from 100% to 50%. Fig. 5(f) shows the results using a trace with 16 object tags and 4 shelf tags. Our system again performs much better than the baseline, and degrades its accuracy only slowly as RR_{major} is reduced. This is because inference can intelligently exploit the facts from the past to smooth noisy

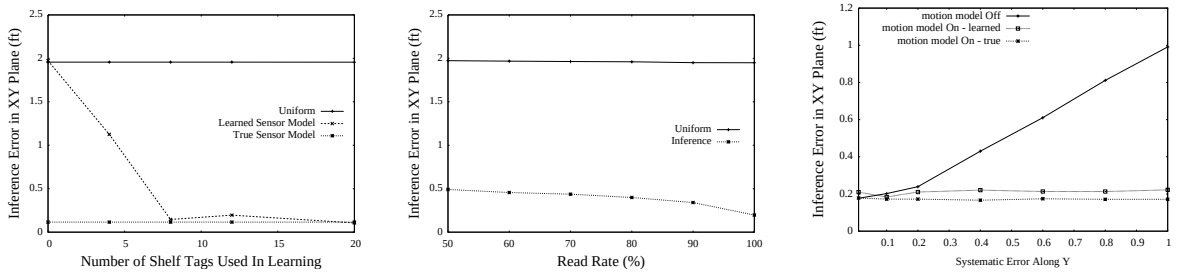


(a) True sensor model used by the simulator

(b) Learned sensor model using 20 shelf tags

(c) Learned sensor model using 4 shelf tags

(d) Learned sensor model for our lab RFID reader



(e) Using sensor models learned with varying numbers of shelf tags

(f) Varying read rates in the reader's major detection range

(g) Varying systematic error along Y (μ_s^y) in reader location sensing (σ_s^y=.2)

Fig. 5. Results of model calibration and initial system testing

object readings and derive their locations, hence not highly sensitive to the changes of the read rate.

Handling reader location noise. We next evaluate our system's ability to handle reader location noise. We generated traces by varying the parameters of the reader location sensing model: the systematic error along the y axis $\mu_s^y \in [0, 1]$, indicating a constant distance between the measured location and the true location; the random noise σ_s^y = 0.01 or 0.2, representing little or high variation. Given the amount of noise present, we used 5000 particles per object to stabilize the performance. Fig. 5(g) shows the results of σ_s^y = 0.2 (the figure for σ_s^y = 0.01 is similar, hence omitted).

Our system's ability to overcome reader location noise is demonstrated by the difference between the bottom curve ("motion model On - true"), which is our system using the true parameters of the reader location sensing model, and the curve ("motion model Off'), which is a simplified technique that always uses the reported location as true location in inference. As μ_s^y increases, our system is very effective in correcting the systematic error, mostly through the evidence of the shelf tags. In contrast, the lack of motion model leads to degradation almost linearly in μ_s^y. Moreover, the curve ("motion model On - learned") shows that we can very well approximate the best system performance by learning the parameters of the location sensing model from a small training trace.

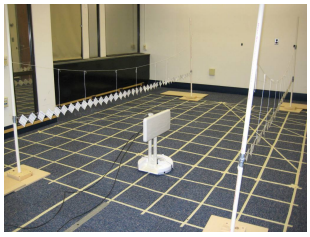
Other experiments. We also explored other factors such as the read frequency, the noise in reader motion, and the size of shelf area for particle initialization. Overall, we observed little sensitivity of our system to these factors. Details are omitted in the interest of space but available in [**?**].

C. Evaluation using Real Traces

To evaluate our system in real-world settings, we generated a lab RFID deployment. As Fig. 6(a) shows, we erected two parallel shelves (assumed to be along the y axis), containing 80 EPC Gen2 Class 1 tags spaced four inches apart. Each shelf has five evenly-spaced reference tags whose true positions are known a priori. We constructed a mobile reader by mounting a bi-static antenna connected to a ThingMagic Mercury5 RFID reader on an iRobot Create robot (Fig. 6(a)). The robot was programmed to scan one row of tags, and turn around to scan the other row, at a speed of 0.1 foot/sec with readings performed once every second. The robot computed its location using dead reckoning. We observed significant error in reported location, e.g. up to 1 foot off from the true location. To emulate various read rates, we varied the reader's timeout setting—the amount of time a tag is given to respond after the initial signal is sent by the reader—from 0.25 to 0.75 second.

We used the reference shelf tags to create a training trace to run EM and learn the physical sensor model for our antenna. The result, shown in Fig. 5(d), reveals that our antenna's read area is akin to a balloon with a wide minor range, whose read rate is inversely related to an observed object's angle from the center of the antenna; this agrees well with manually calibrated sensor models for similar Thingmagic readers [24].

We next compare the accuracy of our system to SMURF



(a) A robot-mounted reader scanning two rows of tags

Timeout (ms)	Our System			SMURF (improved)			Uniform Sampling		
	X(ft)	Y(ft)	XY(ft)	X(ft)	Y(ft)	XY(ft)	X(ft)	Y(ft)	XY(ft)
250 (SS)	0.29	0.33	**0.44**	0.33	0.60	**0.70**	0.33	3.96	3.97
500 (SS)	0.28	0.39	**0.48**	0.33	0.88	**0.94**	0.33	3.98	3.97
750 (SS)	0.27	0.35	**0.44**	0.33	0.76	**0.83**	0.33	3.95	3.97
250 (LS)	0.17	0.41	**0.44**	1.31	0.60	**1.44**	1.31	3.98	4.19
500 (LS)	0.35	0.36	**0.51**	1.31	0.58	**1.43**	1.31	3.98	4.19
750 (LS)	0.55	0.31	**0.64**	1.31	0.69	**1.48**	1.31	3.98	4.19

(b) Inference error of our system, an improved version of SMURF, and Uniform Sampling. SS denotes a small imagined shelf (0.66x4ft) and LS a large imagined shelf (2.6x4ft).

Fig. 6. Evaluation results using real traces.

[20] using traces collected from our lab. SMURF is essentially an adaptive smoothing technique that for each epoch, decides if a tag has moved away from the sensing region when there is a missed reading. However, it cannot directly translate RFID readings into location events. For effective comparison to our system, we augmented SMURF with additional sampling: In each epoch, if SMURF decides that the tag is still in range as a result of smoothing, a location of the tag is obtained by randomly sampling over the intersection of the current read range and the shelf. At some point, when SMURF decides that the tag is no longer in scope, all sampled locations generated in those consecutive epochs are averaged to produce the final estimate. Finally, SMURF does not support learning of the sensor model, so we approximate the read range based on our learned model to enable sampling of the tag location.

Fig. 6(b) shows results of our system, improved SMURF, and uniform sampling, with the first three rows generated from a small imagined shelf, and the next three rows from a large imagined shelf. Since the read range can be large, such shelf information helps restrict the area for object location sampling in all three algorithms. As can be seen, the accuracy of our system is within 0.44 to 0.63 foot. In comparison, the error of SMURF is 1.6 to 2.0 times of our system when the shelf area is small and is over 2.3 times when the shelf area is increased.

These differences can be explained by two main reasons: First, SMURF can not correct the error in reported reader location as observed in our traces. While smoothing may be affective, sampling of object location is always performed from the reported reader location. This explains the difference between our system and SMURF along the y axis where the robot drifted significantly away from the expected location. Second, object location sampling that we added to SMURF is rather primitive compared to the sampling-based inference employed in our system. Their different effects are shown by the error in x; the error of SMURF is strictly half of the shelf size in x, as inaccurate as uniform sampling.

D. Scalability Tests using Simulation

Finally, we show how our system improves over basic particle filtering (§IV-A) in scalability while maintaining high accuracy. We examine our proposed techniques, particle factorization (§IV-B), spatial index (§IV-C), and belief compression (§IV-D), each including the preceding technique(s) in that order. For scalability tests, we assume that the application has an accuracy requirement of within 0.5 foot from the true location. We used our simulator to create synthetic streams. All CPU measurements were obtained from a 3Ghz dual-core xeon processor with 6GB memory available for use in Java.

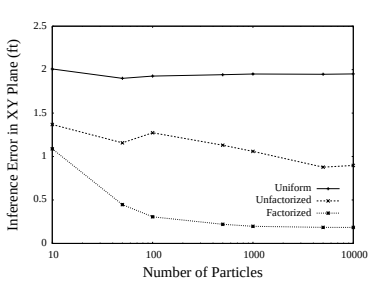
Varying number of particles. First, we investigate how many particles are needed to meet the accuracy requirement. Intuitively, a technique that needs more particles is less likely to scale. We ran the basic particle filter (unfactorized) and our factorized particle filter. The parameter varied is the number of particles P: P unfactorized particles for the basic filter and P particles per object for the factorized filter, which consume the same amount of space. As Fig. 7(a) shows, our factorized filter quickly stabilizes its accuracy at around 0.2 foot after P reaches 500. In contrast, the basic filter slowly improves its accuracy and has not reached 0.5 foot even with 10,000 particles. As explained in §IV-B, the basic filter cannot scale due to its excess need of particles to achieve decent accuracy.

Varying number of objects. We next increased the number of objects from 10 to 20,000 and ran all three advanced techniques as well as the basic filter. Fig. 7(b) and 7(c) report on the inference error and the average time taken to process each reading (on a log scale). As can be seen, given 20 objects, the basic filter takes about 10 second to process each reading, by using XXX particles yet still violating the accuracy requirement. The factorized filter, by using XXX particles per object, well meets the accuracy requirement and improves processing cost significantly. However, this cost still degrades fast as the object count increases. Augmenting the factorized filter with an spatial index reduces the objects processed at each time to a small number, yielding a much reduced cost at a constant 10 msec per reading. Finally, belief compression drastically reduces the cost to 0.1 msec per reading due to the ability to use fewer particles (in this case only 10 particles) after decompression. Neither spatial indexing nor belief compression causes obvious degradation of accuracy.

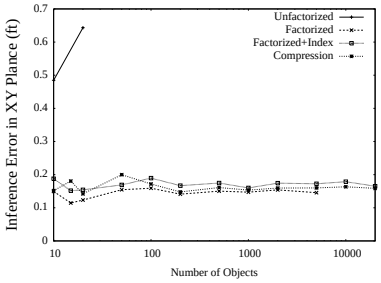
Other experiments. We also ran tests with more noise in reader location, which requires more particles to handle the noise. Belief compression still achieved a constant throughput of 1500 readings per second while meeting the accuracy requirement. Moreover, the memory usage of belief compression for these tests was within 20MB. Details are available in [**?**].

VI. Related Work

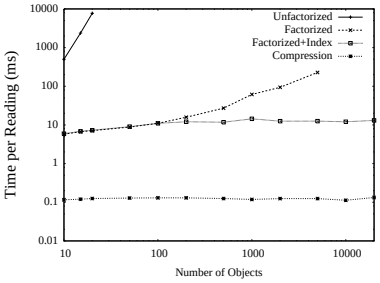
Directly relevant research has been addressed in previous sections. We survey broader areas of related work below.



(a) Varying numbers of particles



(b) Inference error for little noise traces



(c) Inference time per reading

Fig. 7. Results of scalability tests

RFID stream processing. The HiFi project[15] offers a declarative framework for RFID data cleaning and processing. Its techniques focus on temporal and spatial smoothing of readings generated by a fixed set of static readers. SMURF [20] is a particular cleaning approach employed in HiFi and we experimentally demonstrate the benefits of our approach over SMURF for mobile readers. Architectural issues for probabilistic RFID processing have discussed in context of Data Furnace [17] but the research is still underway.

Sensor data management [25], [8], [10], [32] mostly considers environmental phenomena such as temperature and light. Techniques for data acquisition [25], [7], model-based processing [8], and approximation [6], are geared towards queries (e.g., selection and aggregation) natural to such data. In contrast, RFID data captures object identification and to support querying, such noisy, primitive data needs to be first transformed to clean, rich location data. Model-based views over sensor data streams [9] employ probabilistic inference but are restricted to GPS readings (which already reveal object locations) and small numbers of objects.

RFID warehouses. Issues in RFID data management including inference are discussed in [5]. A high-level design of a large-scale RFID system is presented in [35]. Application-specific rules are used to archive and compress RFID data into databases [34]. Inside RFID databases, advanced data compression techniques are available [18] and data cleansing can be integrated with query processing [29].

Object and person tracking [23], [28], [31] focuses on tracking moving targets when the association is uncertain between the observed features of an object and the object's true identity. In the RFID setting, however, object identities are given as part of the readings; the challenge is to infer locations. Therefore, models from this research area are not appropriate for our problem. Some probabilistic models are based on GPS readings [23], [28], but these do not apply to our problem, because unlike GPS, RFID readings do not reveal locations directly. Moreover, the work in this area is designed for small numbers of objects (on the order of 10 objects), and does not scale to a warehouse setting.

RFID-equipped robots have previously been used to estimate locations of robots [21] or of RFID-affixed objects [19], [12]. However, this work differs from ours in several important aspects. First, in previous work the sensor model is manually calibrated [19], [12], which is problematic because reader performance depends greatly on the characteristics of the environment. Second, in [12], the reader motion model is omitted, which significantly simplifies inference. Third, and most important, this line of work is not designed to produce queriable streams for scalable stream query processing—existing work does not support online inference over RFID streams [12] and does so only for a small set of objects [19]. In contrast, our work employs several techniques to produce queriable streams and scales inference to large numbers of objects at stream speeds.

VII. Conclusions

In this paper, we presented a probabilistic approach to translate noisy, primitive data streams from mobile RFID readers into clean, rich event streams with location information, and further, to do so at stream speed for large numbers of objects. Our results show that with an automatically and accurately learned probabilistic model, our approach can augment object identities with estimated locations with high accuracy, e.g., within a range of a few inches, and is robust to noise in observed tags and reader locations. Such inference further scales to large numbers of objects (e.g., 10's of thousands of them) with small memory usage and at a constant rate of over 1000 readings per second.

We plan to extend our research in several directions. We will extend our work to include inter-object relationships such as containment of interest to monitoring applications. We will also extend it to handle handheld readers that are operated by humans and do not offer reader locations. Finally, we will explore stream query processing over the inferred data to support a range of monitoring applications.

References

[1] A. Arasu, S. Babu, and J. Widom. CQL: A language for continuous queries over streams and relations. In *DBPL*, pages 1–19, 2003.

[2] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. In *SIGMOD '90: Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, pages 322–331, New York, NY, USA, 1990. ACM.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *UAI*, pages 33–42, 1998.

[5] S. S. Chawathe, V. Krishnamurthy, S. Ramachandran, and S. E. Sarma. Managing RFID data. In *VLDB*, pages 1189–1195, 2004.

[6] J. Considine, F. Li, G. Kollios, and J. W. Byers. Approximate aggregation techniques for sensor databases. In *ICDE*, pages 449–460, 2004.

[7] A. Deshpande, C. Guestrin, W. Hong, and S. Madden. Exploiting correlated attributes in acquisitional query processing. In *ICDE*, pages 143–154, 2005.

[8] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *VLDB*, pages 588–599, 2004.

[9] A. Deshpande and B. Kanagal. Online filtering, smoothing and probabilistic modeling of streaming data. In *ICDE*, 2008. To appear.

[10] A. Deshpande and S. Madden. MauveDB: supporting model-based user views in database systems. In *SIGMOD*, pages 73–84, 2006.

[11] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo in Practice*. Springer-Verlag, 2001.

[12] I. Ehrenberg, C. Floerkemeier, and S. Sarma. Inventory management with an RFID-equipped mobile robot. In *IEEE Conference on Automation Science and Engineering (CASE)*, pages 1020–1026, 2007.

[13] K. Finkenzeller. *RFID handbook: radio frequency identification fundamentals and applications*. John Wiley and Sons, 1999.

[14] C. Floerkemeier and M. Lampe. Issues with RFID usage in ubiquitous computing applications. In *Pervasive*, pages 188–193, 2004.

[15] M. J. Franklin, S. R. Jeffery, S. Krishnamurthy, F. Reiss, S. Rizvi, E. W. 0002, O. Cooper, A. Edakkunni, and W. Hong. Design considerations for high fan-in systems: The HiFi approach. In *CIDR*, pages 290–304, 2005.

[16] S. Garfinkel and B. Rosenberg, editors. *RFID: Applications, Security, and Privacy*. Addison-Wesley, 2005.

[17] M. N. Garofalakis, K. P. Brown, M. J. Franklin, J. M. Hellerstein, D. Z. Wang, E. Michelakis, L. Tancau, E. W. 0002, S. R. Jeffery, and R. Aipperspach. Probabilistic data management for pervasive computing: The data furnace project. *IEEE Data Eng. Bull.*, 29(1):57–63, 2006.

[18] H. Gonzalez, J. Han, X. Li, and D. Klabjan. Warehousing and analyzing massive RFID data sets. In *ICDE*, page 83, 2006.

[19] D. Hähnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose. Mapping and localization with RFID technology. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[20] S. R. Jeffery, M. J. Franklin, and M. N. Garofalakis. An adaptive RFID middleware for supporting metaphysical data independence. *VLDB Journal*, 2007. To appear.

[21] G. A. Kantor and S. Singh. Preliminary results in range-only localization and mapping. In *Proceedings of the IEEE Conference on Robotics and Automation (ICRA '02)*, volume 2, pages 1818 – 1823, May 2002.

[22] Kiva. `http://www.kiva.edu/`.

[23] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artif. Intell.*, 171(5-6):311–331, 2007.

[24] X. Liu, M. D. Corner, and P. J. Shenoy. Ferret: Rfid localization for pervasive multimedia. In *Ubicomp*, pages 422–440, 2006.

[25] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *SIGMOD*, pages 491–502, 2003.

[26] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD thesis, University of California, Berkeley, July 2002.

[27] B. Ng, L. Peshkin, and A. Pfeffer. Factored particles for scalable monitoring. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2002.

[28] D. J. Patterson, L. Liao, D. Fox, and H. A. Kautz. Inferring high-level behavior from low-level sensors. In *Ubicomp*, pages 73–89, 2003.

[29] J. Rao, S. Doraiswamy, H. Thakkar, and L. S. Colby. A deferred cleansing method for rfid data analytics. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 175–186. VLDB Endowment, 2006.

[30] S. Rizvi, S. R. Jeffery, S. Krishnamurthy, M. J. Franklin, N. Burkhart, A. Edakkunni, and L. Liang. Events on the edge. In *SIGMOD*, pages 885–887, 2005.

[31] D. Schulz, W. Burgard, D. Fox, and A. Cremens. People tracking with mobile robots using sample-based joint probabilistic data association filters. *International Journal of Robotics Research (IJRR)*, 22(2):99–116, 2003.

[32] A. Silberstein, R. Braynard, and J. Yang. Constraint chaining: on energy-efficient continuous monitoring in sensor networks. In *SIGMOD*, pages 157–168, 2006.

[33] A. Smith, H. Balakrishnan, M. Goraczko, and N. B. Priyantha. Tracking Moving Devices with the Cricket Location System. In *2nd International Conference on Mobile Systems, Applications and Services (Mobisys 2004)*, Boston, MA, June 2004.

[34] F. Wang and P. Liu. Temporal management of RFID data. In *VLDB*, pages 1128–1139, 2005.

[35] E. Welbourne, M. Balazinska, G. Borriello, and W. Brunette. Challenges for pervasive rfid-based infrastructures. In *IEEE PerCom Workshop on Pervasive RFID/NFC Technology and Applications (PERTEC'07)*, 2007.

[36] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *SIGMOD*, pages 407–418, 2006.