RPDM: A System for RFID Probabilistic Data Management

Abstract. Data streams are more and more commonly generated in a large number of scenarios by audio and video devices, Global Positioning System (GPS), Radio Frequency Identification (RFID) and other types of sensors. In particular, RFID technology has recently gained significant popularity, especially for real-time people and goods tracking, however the noisy, redundant and unreliable nature of RFID streams, coupled with their huge size, can make their exploitation and management difficult.

In this paper, we present a realtime system for RFID Probabilistic Data Management (*RPDM*). The system manages unreliable and noisy raw RFID data and transforms them into reliable meaningful probabilistic data streams by means of a newly proposed method based on a probabilistic Hidden Markov Model (*HMM*). Moreover, to handle the huge data volume generated by RFID deployments, RPDM proposes and implements a simple on-line summarization mechanism, which is able to provide small space representation for the massive RFID probabilistic data streams while preserving the meaningful information. The results are promptly stored in a probabilistic database, in such a way that a wide range of probabilistic queries can be submitted and answered effectively. The experimental evaluation proves the feasibility of the approach in real-world object tracking scenarios.

Keywords: RFID data streams, Hidden Markov Model, Probabilistic Data Management, Data reduction, Object tracking

1. Introduction

Many computer and communication applications generate data in the form of continuous streams. These data streams are possibly infinite sources of data that stream continuously while observing a physical phenomenon, e.g. temperature or humidity levels, phone conversation records or audio video streaming, and so on. Data streams could be generated in different scenarios by different devices, such as audio and video devices, Global Positioning System (GPS), Radio Frequency Identification (RFID) and other types of sensors. In the last several years, RFID technology has gained significant popularity due to its ability of detecting objects and people carrying small RFID tags in an environment equipped with RFID readers. RFID applications usually rely on RFID deployments to manage high-level events such as tracking the location that products visit for supply-chain management [21], monitoring the location and status of patients in hospital environment [34], indoor people localisation [47] and so on. Furthermore, RFID technology can be used with other sensors to these purposes. For instance, in [11] RFIDs and cameras are used together in order to localize the potential intruders. While camera-based systems can localize all the people in the scene (regardless if they are intruders or not), RFIDs can identify authorized people. Therefore, in a security application scenario, the data produced by those combined systems can be very useful for real-time monitoring and as a hint for further investigations by house detectives.

A fundamental relation for all the above and many other purposes is the location of people and objects over time. On the other hand, RFID readers produce raw data that cannot be directly used to this end: they detect the presence of tags in its vicinity and generate streams of low-level observations in the form of TREs (Tag Read Events), $(tag\_id, antenna\_id, time)$, that show when tags are being sighted and the related antennas. These low-level observations must be transformed into high-level events meaningful to applications. One such sample scenario is shown in Figure 1(a), where John, a user wearing an RFID tag that transmits every second, moves between locations O1, H1 and H2. In this context, any TRE coming from John's tag must be transformed into meaningful relation instances such as "John entered his office O1 at 09:00". Nevertheless, the management of RFID data in transforming low-level streams into high-level events poses a number of challenges [7,24]. In particular, the nature of an RFID data stream is noisy, redundant and unreliable, mainly because of three main reasons:

- *Conflicting Readings* i.e., when an RFID tag is simultaneously detected by two antennas that cover adjacent areas, thus making it difficult to establish the actual location of tag [29];
- *Missing Readings* i.e., loss of reading instances in which RFID tags are not detected by the antenna while actually being present within its coverage area. The incidence of this phenomenon is high and not negligible. Previous studies report that an RFID reader is usually able to detect only 60% -70% of tags that are in its vicinity [19,29];
- *Data-Information Mismatch* i.e., mismatch between the information which the application is concerned to and the data produced by the sensors.

To this end, one possible solution for real-time applications is to generate probabilistic streams by inference on a Hidden Markov Model (HMM) [30,31, 44,46]. Then, probabilistic inference is required in order to extract high-level complex events from the low-level atomic events acquired by the readings. For example, in the sample scenario shown in Figure 1(a), the location of the objects is unknown to the system and observed low level sensor data is translated into precise and more reliable estimates about the location of these objects. In this context, raw data are then transformed into probabilistic data as a probabilistic relation `At(tagID,location,time,prob)` that can be stored in a (probabilistic) database table and queried to detect complex events meaningful to applications [44]. Figure 1(b) shows an example tuple `(John,O1,09:00,0.99)`, which indicates that John at time 09:00 was in his office O1 with probability 0.99.

Another important aspect to be considered is that RFID tags continually send out their IDs at pre-programmed intervals (few seconds) and for each tag read, the number of probabilistic tuples equals the number of reference locations. Therefore, an HMM for RFID deployments produces huge volumes of uncertain data that can reach in practical cases the size of gigabytes in a day. Storing all these probabilistic tuples in the probabilistic database is extremely expensive and, even more important, it is not always useful. For instance, consider again the sample scenario shown in Figure1, having a total duration of 3 hours and 20 minutes. In Figure1(a), John works in his office for three hours. Then, John goes to the coffee room (H2) by

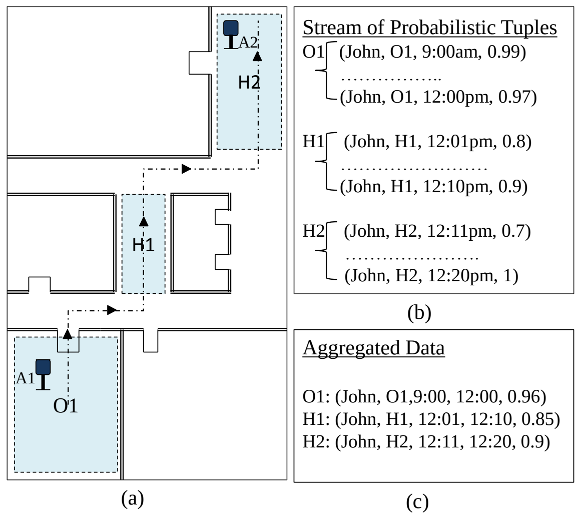
A2
H2
H1
A1
O1
(a)
Stream of Probabilistic Tuples
O1 (John, O1, 9:00am, 0.99)
……………..
(John, O1, 12:00pm, 0.97)
H1 (John, H1, 12:01pm, 0.8)
……………………
(John, H1, 12:10pm, 0.9)
H2 (John, H2, 12:11pm, 0.7)
…………………
(John, H2, 12:20pm, 1)
(b)
Aggregated Data
O1: (John, O1,9:00, 12:00, 0.96)
H1: (John, H1, 12:01, 12:10, 0.85)
H2: (John, H2, 12:11, 12:20, 0.9)
(c)

Fig. 1. (a) A visual representation for John movements; (b) The stream of inferred probabilistic tuples; (c) The stream of aggregated tuples

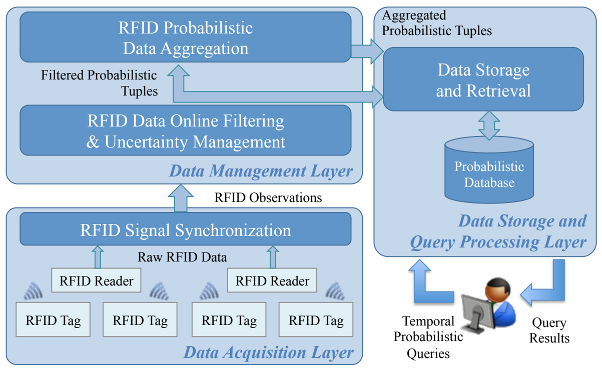
RFID Probabilistic Data Aggregation
Aggregated Probabilistic Tuples
Filtered Probabilistic Tuples
Data Storage and Retrieval
RFID Data Online Filtering & Uncertainty Management
Probabilistic Database
Data Management Layer
RFID Observations
Data Storage and Query Processing Layer
RFID Signal Synchronization
Raw RFID Data
RFID Reader
RFID Reader
RFID Tag
RFID Tag
RFID Tag
RFID Tag
Data Acquisition Layer
Temporal Probabilistic Queries
Query Results

Fig. 2. High level system description

passing through the hall (H1), where he stays for some minutes talking with one of his colleagues. Since the number of locations in this scenario is three, $32,400 =$ (60 seconds $*$ 180 minutes $*$ 3 locations) probabilistic tuples are produced for the first three hours which report more or less the same location information for him (stay in office). This represents a rather realistic scenario, as usually person or good movements are noticeably slower than RFID transmission rates.

In this paper, we present a realtime system for RFID Probabilistic Data Management called *RPDM*, whose high-level architecture is shown in Figure 2. RPDM includes a *Data Management Layer* that manages unreliable and noisy RFID data streams, coming from the *Data Acquisition Layer*, and transforms them into reliable meaningful probabilistic data streams by means of a newly proposed *Online Filtering & Uncertainty Management* mechanism. The developed method includes a specifically designed data model based on a probabilistic HMM. The presented model combines prior domain knowledge about the system behavior in the form of its parameters with the actual observations to infer the location of the people. Furthermore, the proposed model uses a sample based sequential Monte Carlo algorithm, named *particle filtering* to infer the locations of the people or objects in location tracking system.

To handle the huge data volume generated by RFID deployments, RPDM proposes and implements a simple *Probabilistic Data Aggregation* mechanism, which is able to provide small space representation for massive RFID probabilistic data streams while preserving the meaningful information. The on-line summarization mechanism draws inspiration from the field of clustering [26]. The main idea behind the proposed approach is to keep on aggregating tuples until a state transition is detected. This can be seen in Figure1(c): only one tuple shows John's location from 9:00am to 12:00pm i.e. in his office O1. An object or person is said to have state transition if its location changes from one to another, as in Figure1(b) where John moves from O1 to H1 and consequently to H2. In this case, the proposed summarization method stores only 3 probabilistic tuples instead of $36,000$ = (60 seconds * 200 minutes * 3 locations), while these 3 stored probabilistic tuples give enough information about John's movements. Probabilistic tuples are aggregated as they arrive, hence avoiding the use of expensive and offline disk based operations such as sorting and summarization. The result is promptly stored by the *Data Storage and Query Processing layer* in a probabilistic relation `At(tagID,location,from,to,prob)` in the probabilistic database MayBMS [25], in such a way that a wide range of probabilistic queries can be applicable and answered effectively.

The rest of the paper is organized as follows. Section 2 discusses related works. In Section 3, we conceptualize the complete system for RFID probabilistic data management, then we propose the online filtering & uncertainty management and the data aggregation methods (Sections 4 and 5, respectively). In Section 6 we present extensive experiments in complex single- and multi- antenna object tracking scenarios, showing a very good reliability of the proposed system . Finally, Section 7 gives some concluding remarks.

2. Related works

In the last few decades, RFID technology has emerged significantly with many real time applications, such as product tracking and asset management, object and people authentication, health care etc. Nevertheless, data management in these RFID applications poses a number of challenges, including conflicting readings, missed readings and data information mismatch [7]. Our work proposes an infrastructure specifically focused on people tracking, where RFID tags are attached to the people. This infrastructure applies probabilistic methods to filter and transform raw RFID data streams into probabilistic meaningful data streams and efficiently stores them by applying aggregation mechanisms in probabilistic database for query processing purposes. The following related works will basically focus on: (a) deterministic and probabilistic approaches for filtering and smoothing of RFID data streams; (b) techniques for the transformation of RFID low-level data streams into high-level information; and (c) RFID data aggregation.

In literature, different deterministic and probabilistic approaches have been proposed for filtering and smoothing RFID data streams [32,39,12,5,8,41,14]. For instance, in [27] authors proposed a framework named ESP (Extensible Sensor stream Processing) for sensor data filtering and smoothing for use in pervasive applications. ESP introduces the concept of temporal and spatial granules. It is designed as a pipeline by using declarative filtering procedures. These procedures involve deduping, removing outliers and smoothing of data collected from various sources. However, users need to specify a set of instructions or algorithms to clean the data. In [29,28], authors discussed a sliding window based approach called SMURF. It is a declarative, adaptive smoothing filter for RFID data. Based on the characteristics of the underlying data stream SMURF automatically adjusts the window size; therefore, applications do not need to set the size of smoothing window. The main idea of SMURF is to model the unreliability of RFID readings to see RFID data streams as statistical sample of the tags in the physical world, and incorporates sampling theory techniques, such as binomial sampling and π-estimators, to perform its cleaning process. Another, deferred RFID data cleaning framework is introduced in [43], where the RFID cleaning framework uses declarative sequence based rules at the time of query execution to correct RFID data anomalies. Note that the technique used in this work performs RFID data cleaning at the time of query execution, therefore it is not useful in application scenarios (e.g. in sensor fusion applications) where RFID data should be cleaned and filtered before further processing. In [15,16], Deshpande et al. discuss techniques based on probabilistic model in order to handle input errors and inaccuracies. These techniques are based on temporal and spatial correlations to predict missing values, to identify outliers and to approximate answers to queries. Most of them mainly deal with inaccuracy errors present in raw RFID data, filtering and smoothing operations before feeding into higher level applications, thus not dealing (and not exploiting) the "meaning" of the managed information.

A number of probabilistic techniques have also been proposed for the analysis and transformation of RFID low-level data streams into meaningful information in order to deal with data-information mismatch problem. These techniques, exploit the probabilistic nature of RFID data and manage their inherent uncertainty in the form of probabilities and correlations, so to achieve even higher effectiveness in the application scenarios they are applied to [30,33,44,45,46,48]. For instance [31,44] generate probabilistic streams by inference on an HMM. Then, probabilistic inference is required in order to extract high-level complex events from the low-level atomic events acquired by the readings. For example, in tracking applications, the location of the objects is unknown to the system and the observed low level sensor data is translated into precise and more reliable estimates about the location of these objects by implementing an HMM [44,45]. While, similarly to these works, our system also performs online management of uncertainties present in the received RFID data and estimates the correct location of people, it also detects state transitions, thus avoiding the redundant information produced in stable states.

Typically, RFID devices generate data in the form of data streams and then it is stored in a database for further analysis. In literature, in this context two kinds of approaches exist for RFID data compression. One approach is on-line processing which views RFID data as data streams [13,18,6]. The other approach is off-line processing of raw RFID data, i.e. data is first stored and then further processing is performed [20,21,36,10]. For instance, a graph-based model is discussed in [10] for providing the compression in RFID systems. This model captures the possible object locations and their containment relationships. However, high detection rates at the RFID readers are required by graph model in order to have accurate results. In [21], a new model for warehousing RFID data is proposed. The proposed model provides significant data compression and path-dependent aggregates while preserving the object transitions. The proposed work basically takes advantage of object movements in bulk, of data generalization and the merge or collapse of the path segment that RFID objects follow. Though, this work does not handle missing and erroneous data tuples. Furthermore, this technique is not useful if objects do not move together in large groups. Lee et al. have discussed this aspect of RFID data in [36]. They proposed an efficient storage scheme and query processing for supply chain management. They used an effective path encoding method to represent the information flow representing the movements of products. A storage scheme is developed to process tracking queries and path oriented queries efficiently based on path encoding scheme and numbering scheme. Though, this approach performs off-line processing for RFID data aggregation, compression and storage. In [6], authors present a lossy aggregation mechanism for RFID data streams based on temporal and spatial aggregations. The proposed algorithm exploits the time and space dimension to reduce the volume of input RFID data streams. Another approach for RFID data compression specific for supply chain scenario is presented in [13]. This approach takes advantage of the property that, in this context, objects move together, though it is capable of representing aggregations of objects which are not dependent on movements along the supply chain. In particular, this work represents an incremental aggregation approach based on various combinations of attributes describing RFID data other than paths and locations. Using this compression approach, the authors develop a lossless, relational-based storage model which preserves information about both path dependent and path independent items. In [18], a lossy compression technique is proposed for RFID data streams. In particular, the authors define a data structure to represent compressed RFID warehouses. Moreover, they propose an architecture that gathers readings from RFID readers and stores them in a compact way. The majority of the above discussed aggregation approaches are application specific. Many of them propose off-line storage models which are path dependent and thus the transition of single object is lost. On the other hand, our summarization mechanism is capable of performing both off-line and on-line processing of RFID data streams. Moreover, it is path independent and efficiently captures the objects transitions while avoiding the redundant information produced in stable states.

Viewing RFID data as data streams, different deterministic [17,22,51] and probabilistic [35,49,40,3] clustering approaches for very large amount of data can be considered as well. Besides purely deterministic approaches, the vague and uncertain nature of the data stream has recently captured a lot of research attention and many clustering algorithms have been proposed which also take into account the probabilities associated to the involved data. In this context, a fuzzy version of DBSCAN has been presented as FDBSCAN [35]. This algorithm, instead of finding regions with high density, identifies regions with high expected density, based on the probability distributions of the objects. Another probabilistic extension is P-DBSCAN [49], which takes advantage of the probability distribution information of the object locations in the definition and computation of probabilistic core object and probabilistic density-reachability. In [40], an extension of the K-means algorithm is proposed, named UK-means algorithm, which considers expected distance between the object and the representative of the cluster. As UK-means is based on classical K-means algorithm, it can be sensitive to noise. UMicro [3] uses a general uncertainty model and keeps track of the standard errors of each dimension within each cluster, showing that the use of even general uncertainty model during the clustering process is enough to improve the quality of results over purely deterministic approaches. Other similar related approaches are the two-phase clustering algorithm discussed by Zhang et al. in [50], named as LuMicro, and PWStream [23], which has been proposed for the specific problem of sliding windows. Most of the clustering methods discussed above analyze the incoming data and judge on their "certainty", thus producing the highest quality possible clusters both in terms of compactness and high probability, discarding low quality ones. Further, they work on the assumption of knowing specific information characterizing the uncertainty, such as having the entire probability density function or standard error data available. The number of clusters to be produced is also usually known in advance. On the other hand, our methods are targeted for summarization task in a location tracking context but can be applicable to other application scenarios. More specifically, our ultimate goal is to correctly estimate the location of people and to identify and highlight state transitions, while avoiding redundant information produced in stable states. In this context, only one active cluster per tag suffices and we summarize the received data in order to make it available to subsequent modules in a more compact but equally meaningful way.

3. System description

For clarity of presentation, in the remainder of the paper we will refer to random variables using uppercase letters and to single values with lowercase letters. Specifically we will refer to the following entities:

- $\mathcal{A}$ as the reading antennas, $\mathcal{A} = (\alpha_{h=1,\ldots,n})$;
- $\mathcal{T}$ as the tags deployed to the people, $\mathcal{T} = (\tau_{i=1,\ldots,m})$;
- $\mathcal{L}$ as the locations in the specified area, $\mathcal{L} = (\lambda_{j=1,\ldots,k})$;
- $\mathcal{O}$ as the signal range of a multi-antenna observation, $\mathcal{O} = (SR_{\alpha_1} \times SR_{\alpha_2} \times \ldots SR_{\alpha_n})$, where SR_{α_h} is the signal range of antenna α_h.

With reference to Fig. 2, at the lowest level of the proposed system there is the *data acquisition layer*, where raw data coming from RFID devices (e.g., RFID tags attached to objects and people) is read through (typically multiple) RFID readers and properly synchronized. RFID readers receive data from these tags in the form of radio signals and convert them into digital form.

Raw RFID data streams are TREs (Tag Read Events) in the form of $(\tau_i, \alpha_h, RSSI, t)$, where α_h is the identifier of the antenna the tag τ_i is seen by, $RSSI \in SR_{\alpha_h}$ and t is the time instant of the reading. Readers' synchronization is required in order to avoid collisions [9,38]. In particular, all the TREs concerning a given tag τ_i and originated at time t are combined into one RFID *observation* $o_t^{\tau_i} \in \mathcal{O}$ by the *RFID signal synchronization module*. In other words, $o_t^{\tau_i}$ represents the signal strengths coming from each of the antennas $\alpha_{h=1,\ldots,n}$ for the tag identifier τ_i at timestamp t.

The second layer is the *data management layer*. This layer is made up of two modules that clean the received RFID observations and give filtered, meaningful and summarized data to applications. More specifically, the *RFID data online filtering & uncertainty management module* elaborates the received observations $o_t^{\tau_i}$ to estimate the probability distribution $P\left(L_t^{\tau_i}\right)$ of the random variable $L_t^{\tau_i}$ over the set of locations $\mathcal{L}$, one for each tag τ_i. In other words, for each location λ_j, $P\left(L_t^{\tau_i} = \lambda_j\right)$ represents the probability that tag τ_i is in λ_j. Finally, the *RFID probabilistic data aggregation module* takes the stream of probability distributions $P\left(L_t^{\tau_i}\right)$ for $t = 1 \ldots now$ as input and, for each

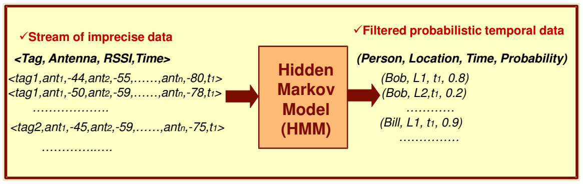
✓Stream of imprecise data
<Tag, Antenna, RSSI,Time>
<tag1,ant1,-44,ant2,-55,……,antn,-80,t1>
<tag1,ant1,-50,ant2,-59,……,antn,-78,t1>
<tag2,ant1,-45,ant2,-59,……,antn,-75,t1>
Hidden Markov Model (HMM)
✓Filtered probabilistic temporal data
(Person, Location, Time, Probability)
(Bob, L1, t1, 0.8)
(Bob, L2,t1, 0.2)
(Bill, L1, t1, 0.9)

Fig. 3. Block diagram of RFID data online filtering & uncertainty management module

tag τ_i, it outputs a stream of probability distributions in an aggregated form $P(L^{\tau_i}_{[t_s,t_e]})$, $1 \leq t_s \leq t_e \leq now$.

The third layer is for the data storage and query processing purposes. In this layer, the *Data Storage and Retrieval Module* is implemented by exploiting the probabilistic database management system MayBMS [1] that is used to store probabilistic RFID data provided by the data management layer and to solve queries. MayBMS is a probabilistic state of the art DBMS and is configured as an extension of the famous open-source relational DBMS Postgres [2]. It is a robust and scalable system for managing data that implements probabilistic and uncertain mechanisms for efficient representation and storage. In this way, users can submit temporal probabilistic queries on the stored RFID probabilistic tuples in MayBMS by using its front-end interface and get query answers with their associated confidence values.

In the following, we will first focus on the different system modules by showing their implementation details then we will provide a detailed account of the performed tests.

4. RFID data online filtering & uncertainty management

Figure 3 represents a block diagram of the RFID data online filtering & uncertainty management module. This module exploits a data model based on a Hidden Markov Model (HMM) [42] to infer the values of $L^{\tau_i}_t$. A HMM is a graphical model typically used in temporal contexts. It infers the attributes of interest which are not directly observable, based on a sequence of events that are observable. For example, consider the reference scenario of location tracking, where the interest of the application is to infer the position of people or objects over time on the basis of RFID readings collected by the reader. The former is not being directly observable itself and considered as the *hidden variable* while the latter actually represent observable events, or simply *observations*. Thus, this module uses a HMM to produce, at each timestamp, a distribution over each tag location (i.e. the hidden variables or *states*) based on the observations coming from the data acquisition layer. The main important feature of this kind of models is that they allow to combine prior domain knowledge about the system behavior with the actual observations to compute the most likely values of the hidden variables. While observations are directly evaluable, the prior knowledge about the system is represented by Conditional Probability Distributions (CPD) which are referenced as the parameters of the HMM.

4.1. Representation

Figure 4 shows the HMM we conceived for the RFID online filtering & uncertainty management module. Graph nodes represent the random variables (*hidden states* and *observations*) of the modeled system, while directional arcs represent the concept of "causality" whose degree is indicated by the corresponding CPD. Specifically, darker nodes in the graph represent the *observations* and thus correspond to measurements collected by RFID antennas, while clear nodes represent *states* and thus coincide with the positions of the people. In Figure 4, time is shown through the use of vertical "lanes"; each lane therefore represents the situation of the system in a single instant in time. In this regard, it is worth noting that, according to the well-known Markov principle, these models typically assume that the variables at time t directly depend on the variables at time t and $t-1$ only and, hence, two consecutive time instances are sufficient for completely representing the whole system. The other parameters of the HMM, or the CPDs that describe the relationship of causality between the variables (represented as directional arcs in Figure 4), are listed below:

- The *initial states distribution* $P(L_0)$, encodes knowledge about the initial state of the system (i.e. the time instant 0);
- The *transition probability distribution* $P(L_{t+1}|L_t)$, encodes the knowledge of how the state of the *hidden variables* at time instant $t+1$ depends on the state at time instant t;
- The *observation probability distribution* $P(O_t|L_t)$, encodes the knowledge of how the *observations* at time instant t depend on the state of the *hidden variables* at time instant t.

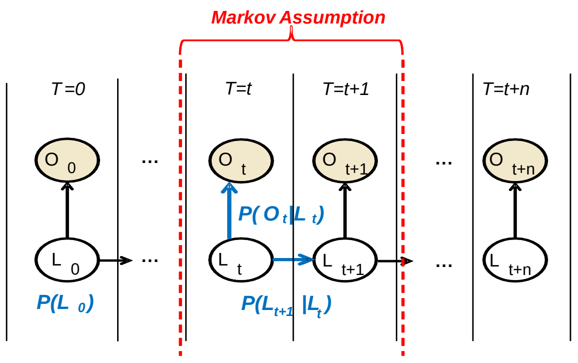

Fig. 4. Graphical representation of Hidden Markov Model used

4.2. Learning

In the specialization of the HMM in the context of location tracking, the involved CPDs are modeled as follow:

- The *initial states probability* $P(L_0)$: we assumed this parameter to be a uniform distribution among all the possible locations;
- The *transition probability*$P(L_t|L_{t-1})$: It is modeled as a matrix whose rows ad columns are associated to the available locations so that each cell $[i, j]$ contains the probability value of having a movement from location i to location j (as an example, if two locations are separated by a wall the corresponding cell contains the value 0);
- Finally, the *observation probability* $P(O_t|L_t)$: This information is typically not available and, thus, has to be learned from training data. In our case, the training data for each of the locations can be represented as points in an (n)-dimensional Cartesian space whose coordinates are the RSSI values for the n antennas. For instance, Figure 5, shows a $3D$ representation of the training data of our sample scenario presented in Section 1: since the number of antennas used to collect training data is two, each training data $(o, \lambda) \in \mathcal{O} \times \mathcal{L}$ is a made up of a tuple o of the RSSI values and the location for the received o (the third dimension). To learn a n-dimensional observation probability, we adopt the popular statistical method called *Maximum Likelihood Estimation (MLE)* which, given the learning data, estimates the value of the probability function parameter that maximizes the likelihood of the observed data (i.e. that makes the learning data "most likely"). Actually, MLE allows us to compute the conjunctive probability $P(O_t, L_t)$, from which observation probability $P(O_t|L_t)$ can be easily computed by applying the Bayes theorem.

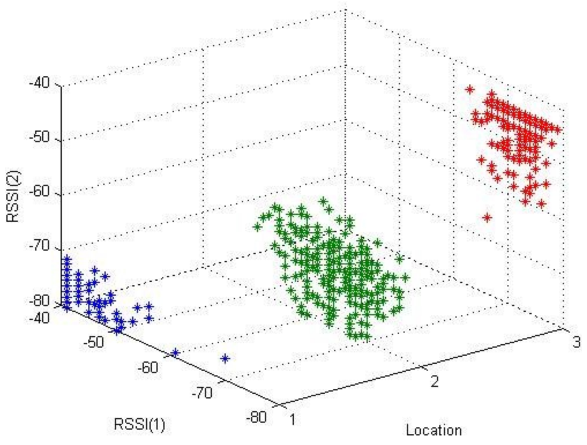

Fig. 5. Representation of the training data of our sample scenario for MLE collected by 2 antennas

4.3. Inference

Our final aim of modelling a stochastic process with an HMM is to obtain the *posterior probability distribution* $P(L_t^{\tau_i})$ over the *hidden variable* $L_t^{\tau_i}$ given the *observed* measurements. This task is called "inference" and different algorithms can be used to this purpose. Among the others, we decided to exploit a popular Monte Carlo algorithm called *Particle Filtering* [4], usually adopted in sample-based inference processes. The algorithm works by computing and constantly maintaining sets of particles to describe the historical and present states of the model. Algorithm 1 represents the steps executed while performing inference at each time instant t. Specifically, given the observed values $o_t^{\tau_i}$ for each identified tag τ_i at time t, the algorithm works by iteratively executing the following steps:

Initialization: (see line 1 of algorithm 1) during this step, an initial set of N particles is created by randomly sampling from the initial states probability $P(L_0)$.

Prediction: (lines $2-5$) during this step, the state of *hidden variables* at time t is estimated by using their state at time $t-1$ and exploiting the parameters of the HMM. More precisely, for each existing particle p_{t-1}^i at time $t-1$ a new particle p_t^i is created for time t by sampling from $P(L_t|L_{t-1})$.

Filtering: (line 6) in this step, the *observation* $o_t^{\tau_i}$ at time t is used to update the states previously estimated for time t. More precisely, each particle p_t^i is assigned a weight based on the values of the observed variables at time t and on the observation probability $P(O_t|L_t)$. This weight is proportional to $P(O_t = o_t^{\tau_i}|L_t = \lambda)$ where λ is the location of p_t^i.

Algorithm 1 :Inference algorithm

```
Require: $N$ number of particles
 1: At time $t = 0$, initialize $p_0^i \sim P(L_0)$ where $i = 1, \cdots, N$
 2: repeat
 3:   $t = t + 1$
 4:   for $i = 1, \cdots, N$ do
 5:     Sample particle $p_t^i \sim P(L_t|L_{t-1})$
 6:     Assign weights to new particles $w_t^i = P(O_t|L_t)$
 7:   end for
 8:   re-sample $N^*$ new particles $p_t^{*i}$ with new equal weights $w_t^{*i}$
 9:   $P(L_t^{\tau_i}) = count(p_t^{*i}|L_t)/N^*$
10: until $t = T$
```

Re-sampling: (line 8) in this step, particles p_t^i, for $i = 1, \ldots, N$, are re-sampled in order to generate a new set of particles p_t^{*i}, all with the same weight w_t^{*i}. This task is necessary in order to avoid degeneracy, i.e. the case where a single particle has all the weight.

Broadly speaking, each particle p_t^i represents a guess about the location of tag τ_i. Then, after a number of iterations, the inference task is performed: to compute the posterior probability $P(L_t^{\tau_i})$ we can indeed simply count the number of particles in each location and divide it by the total number (line 9).

5. RFID probabilistic data aggregation

In this subsection, we describe the details of the online aggregation algorithm (see Algorithm 2) that is implemented in the RFID probabilistic data aggregation module shown in Figure 6.

Given m tags and k locations, the previous module performs inference on an HMM to produce a stream of timestamp ordered probabilistic tuples[1]:

$$X_1^{\tau_1}, X_1^{\tau_2}, \ldots, X_1^{\tau_m}, X_2^{\tau_1}, \ldots, X_2^{\tau_m}, \ldots$$

[1]For ease of presentation and without loss of generality, we assume that tuples arrive in tag order. For the same reason, the discrete probability distribution of the location random variable is represented as one tuple instead of n different tuples.

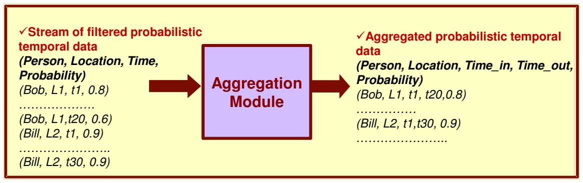
✓Stream of filtered probabilistic temporal data
(Person, Location, Time, Probability)
(Bob, L1, t1, 0.8)
……………….
(Bob, L1,t20, 0.6)
(Bill, L2, t1, 0.9)
…………………..
(Bill, L2, t30, 0.9)
Aggregation Module
✓Aggregated probabilistic temporal data
(Person, Location, Time_in, Time_out, Probability)
(Bob, L1, t1, t20,0.8)
……………
(Bill, L2, t1,t30, 0.9)
…………………..

Fig. 6. Block diagram of RFID probabilistic data aggregation module

where each tuple $X_t^{\tau_i}$ has the form:

$$(\tau_i, t, P(L_t^{\tau_i} = \lambda_1), P(L_t^{\tau_i} = \lambda_2), \ldots, P(L_t^{\tau_i} = \lambda_k))$$

This is received in input by the aggregation algorithm that in turn outputs a stream of probabilistic tuples of the form: $X_{[t_s,t_e]}^{\tau_i} = (\tau_i, t_s, t_e, P(L_{[t_s,t_e]}^{\tau_i} = \lambda_1),$

$$P(L_{[t_s,t_e]}^{\tau_i} = \lambda_2), \ldots, P(L_{[t_s,t_e]}^{\tau_i} = \lambda_k))$$

such that:

– For each pair of tuples on the same tag τ_i, $X_{[t_{s_1},t_{e_1}]}^{\tau_i}$ and $X_{[t_{s_2},t_{e_2}]}^{\tau_i}$, $[t_{s_1}, t_{e_1}] \cap [t_{s_2}, t_{e_2}] = \emptyset$;
– For each source tuple $X_t^{\tau_i}$, a result tuple $X_{[t_s,t_e]}^{\tau_i}$ exists such that $t \in [t_s, t_e]$.

The aggregation algorithm works on the intuition that if a person wearing a tag τ_i is stationary or resides at the same location for a period of time $[t_s, t_e]$, the corresponding probabilistic tuples $X_{t_s}^{\tau_i}, \ldots, X_{t_e}^{\tau_i}$ should show "similar" probability distributions. Therefore, in order to derive $X_{[t_s,t_e]}^{\tau_i}$ we draw inspiration from the large dataset clustering field [22] in that we incrementally group together consecutive "similar" tuples. To this end, at each timestamp t the algorithm maintains at most m clusters, one for each tag τ_i, and for each cluster $c_t^{\tau_i}$, it treats the tuple region collectively through some statistics $stat^{c_t^{\tau_i}}$ providing a summarized description for the cluster. When a new tuple $X_{t+1}^{\tau_i}$ arrives, the algorithm tries to add it to the cluster associated to the corresponding tag, $c_t^{\tau_i}$, by updating the corresponding $stat^{c_{t+1}^{\tau_i}}$ values (see lines 3–5 of algorithm 2). Then, a boundary condition is checked (line 6) and, if it is the case, the tuple is inserted into the cluster by replacing its statistics with the newly computed ones $stat^{c_{t+1}^{\tau_i}}$ (line 7). On the other hand, if a violation is detected, the following steps are performed:

– $c_t^{\tau_i}$ is closed and discarded from the set of current clusters S (line 10);
– a tuple $X_{[t_s,t]}^{\tau_i}$ describing the behavior of the tag τ_i in the time interval corresponding to the period

in which the cluster $c_t^{\tau_i}$ was active is stored in the database (line 11);

- a new cluster for τ_i is created including tuple $X_{t+1}^{\tau_i}$ only, its statistics is computed and it is added to S (lines 12 and 13).

Algorithm 2 Tuple aggregation algorithm

```
Require: $k$ number of locations, $m$ number of tags , $B$ critical boundary
 1: $S =$ current set of clusters; //$S$ contains at most $p$ elements
 2: repeat
 3:     receive the next stream point $X_{t+1}^{\tau_i}$
 4:     $c_t^{\tau_i}$ = identifyCluster($X_{t+1}^{\tau_i}$, $S$) //$stat^{c_t^{\tau_i}}$ is extracted from $c_t^{\tau_i}$
 5:     $stat^{c_{t+1}^{\tau_i}}$= updateStatistics($stat^{c_t^{\tau_i}}$,$X_{t+1}^{\tau_i}$)
 6:     if testBoundaryCondition($stat^{c_{t+1}^{\tau_i}}$) then
 7:         $c_{t+1}^{\tau_i}$ = add($X_{t+1}^{\tau_i}$,$c_t^{\tau_i}$); //$stat^{c_t^{\tau_i}}$ is replaced with $stat^{c_{t+1}^{\tau_i}}$
 8:         update $S$ with $c_{t+1}^{\tau_i}$;
 9:     else
10:         close and discard $c_t^{\tau_i}$ from $S$;
11:         insert $X_{[t_s,t]}^{\tau_i}$ into the database;
12:         $c_{t+1}^{\tau_i}$ = createNewCluster($X_{t+1}^{\tau_i}$);
13:         add $c_{t+1}^{\tau_i}$ to $S$;
14:     end if
15: until data stream ends
```

Until now, we intentionally left our aggregation model generic. In the following, we show how output tuples and cluster statistics are computed.

5.1. Output tuples

In many clustering applications, the resulting clusters have to be represented or described in a compact form to achieve data abstraction. Basically, the most typical compact description of a cluster is given in terms of cluster prototypes or representative patterns such as the *centroid* [26]. The centroid is the logical center of the cluster, usually computed as the average of all the points in the cluster. The use of the centroid to represent a cluster is a very popular schema, which works well when the clusters are compact, as in our reference scenario.

Therefore, we represent tuples in the k-dimensional Cartesian space as points whose coordinates are the probability values for the k locations. This tuple representation actually exhibits tight clustering as long

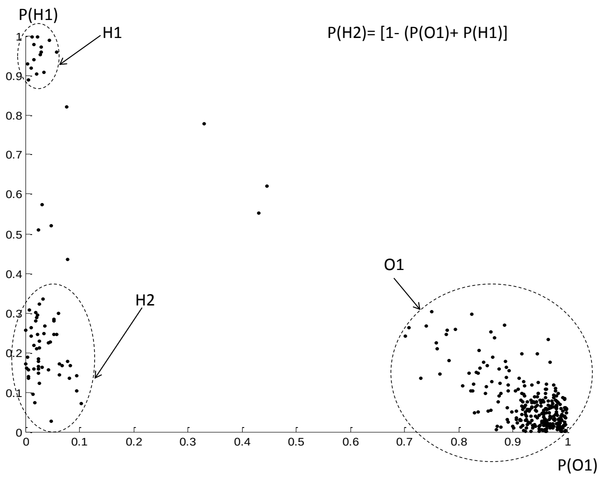

Fig. 7. Cartesian Space representation of the probabilistic tuples of our sample scenario

as the state does not change and a good separation in case of state transition. Figure 7, shows the cartesian plane representation of the sample scenario discussed in Section 1. Since the number of locations is three in this scenario, each tuple generated by the RFID processing module is a point in a 3-dimensional space whose coordinates are the probability values for the locations O1, H1 and H2 (the graph shows only the first two dimensions since the third is linearly dependent from the others). We can see that, since John is residing at a same place (his office) for a long period, a large number of points are concentrated in the O1 region; all these points can be aggregated in one point which will be representative of the behavior of all of them. Instead, when John moves from O1 to H1 and consequently to H2, there is a transition that can be seen in the form of some scattered points on the graph plane. Hereinafter, whenever the context is clear, we will use $X_t^{\tau_i}$ to denote either a probabilistic tuple $(\tau_i, t, P(L_t^{\tau_i} = \lambda_1), P(L_t^{\tau_i} = \lambda_2), \ldots, P(L_t^{\tau_i} = \lambda_k))$ or its representation in the Cartesian space $(P(L_t^{\tau_i} = \lambda_1), P(L_t^{\tau_i} = \lambda_2), \ldots, P(L_t^{\tau_i} = \lambda_k))$.

Then, we incrementally compute the centroid $V_{c_t^{\tau_i}}$ of each cluster $c_t^{\tau_i}$ while it evolves as:

$$V_{c_t^{\tau_i}} = [(V_{c_{t-1}^{\tau_i}} * |c_{t-1}^{\tau_i}|) + X_t^{\tau_i}]/(|c_{t-1}^{\tau_i}| + 1)$$

where $|c_{t-1}^{\tau_i}|$ is the size of cluster $c_{t-1}^{\tau_i}$, and, when it is closed, we store $X_{[t_s,t]}^{\tau_i}$ as $(\tau_i, t_s, t, V_{c_t^{\tau_i}})$.

Table 1

Online filtering and uncertainty management result comparison

				1- ANT		2- ANT	
EXP	Scenario	#Tags		% TAL	AvgLocError	% TAL	AvgLocError
1	Stay	1		89	0.1141	96	0.0372
2	No Stay	1		73.75	0.2719	81.25	0.2471
3	Stay	2	tag1	80	0.2366	96.89	0.0218
			tag2	75.86	0.2711	96.89	0.0235
4	No Stay	2	tag1	73.33	0.2734	93.33	0.0927
			tag2	64.44	0.3797	91.11	0.0816
Mean				76.06	0.2578	92.58	0.0840

5.2. *Boundary conditions*

The main objective of the boundary condition test, is to be able to discriminate when a cluster has to be closed in order to avoid distortion. To this end, we draw inspiration from techniques at the state of the art for cluster validity measurement [37]. Two measurement criteria are typically used for evaluating a clustering schema [37]: compactness and separation. While the former expresses the requirement that the members of each cluster should be as close to each other as possible, the latter refers to the fact that clusters themselves should be widely separated, which is not particularly interesting for our scenario; we thus focus on compactness and consider three different methods for quantifying it. The three models, which provide different indices that can be used in the boundary condition test, are:

– *Maximum Probability Change (MPC)*: it monitors the probability distribution trends. To this end, let $\overline{L}_{X_t^{\tau_i}}$ ($\overline{L}_{c_t^{\tau_i}}$) be the location with the maximum probability value in $X_t^{\tau_i}$ ($c_t^{\tau_i}$). For each cluster $c_t^{\tau_i}$, MPC maintains $\overline{L}_{c_t^{\tau_i}}$ as statistics, and the boundary condition is satisfied when $\overline{L}_{c_t^{\tau_i}} = \overline{L}_{X_{t+1}^{\tau_i}}$. The main disadvantage of this method is that it is very sensitive to noise and thus makes more clusters with fewer points in it;

– *Diameter-oriented (DM)*: it measures how large the cluster shape is. To this end, it uses the cluster diameter as statistics and checks whether the latter is within a threshold B: $\max_{X,Y \in c_{t+1}^{\tau_i}} \{d(X,Y)\} \leq B$. The main disadvantage of this approach is the time and space complexity, due to the fact that the distance between all pairs of points have to be computed and constantly kept updated on the arrival of new data elements. This function is also very sensitive to noise, since the maximum cluster diameter can quickly become large in a noisy environment;

– *Centroid Vs Latest Reading Comparison (CLRC)*: it gives a measure of the mutual distance between the centroid $V_{c_t^{\tau_i}}$ and the latest point $X_{t+1}^{\tau_i}$. To this end, it checks whether $d(V_{c_t^{\tau_i}}, X_{t+1}^{\tau_i}) \leq B$. The main advantage of this method w.r.t. the DM model is that computations are less time- and space-consuming, as $V_{c_t^{\tau_i}}$ can be computed incrementally.

Regarding the distance $d(\cdot,\cdot)$ between tuples, our approach is independent from the actually adopted function. Several alternatives are possible for its implementation, since we only require it is applicable in a n-dimensional space. In our experiments we adopted the Euclidean distance. Finally, note that both for the DM and the CLRC models, we can control the quality of the clustering process by properly selecting the threshold B: low values of B produce a high number of small and tight clusters, while an opposite behavior is observed for high values of B.

6. Experimental evaluation

In this section, we discuss the experiments we have conducted in order to evaluate the effectiveness of the proposed system.

The goal of our evaluation studies is two-fold: (i) to validate the results estimated by the RFID data online filtering & uncertainty management module with the ground truth; and (ii), to validate and compare the effectiveness of each method available in the RFID probabilistic data aggregation module in precisely summarizing the movement behaviors which actually took place in the scenarios. In addition to these, we also performed each experiment with similar setup with two antennas and compared those results with the results obtained from a single antenna.

Table 2

Data aggregation result comparison

					1- ANT				2- ANT			
(a) MPC												
EXP	Scenario	#Tags		#Locs	#Clusters	%SP	%TAL	AvgLocError	#Clusters	%SP	%TAL	AvgLocError
1	Stay	1		3	7 (+133%)	0.0350	89.00	0.1224	3 (=)	0.015	97	0.0422
2	No Stay	1		5	9 (+80%)	0.1125	71.25	0.29850	5 (=)	0.0625	80	0.27404
3	Stay	2	tag1	5	15 (+200%)	0.0517	83.79	0.1771	5 (=)	0.0172	98.27	0.0241
			tag2	5	18 (+260%)	0.0621	82.06	0.1815	5 (=)	0.0172	98.27	0.024
4	No Stay	2	tag1	3	5 (+67%)	0.1111	57.77	0.4416	3 (=)	0.066	93.33	0.0936
			tag2	3	3 (=)	0.0667	53.33	0.4996	3 (=)	0.066	91.11	0.124
Mean					+123%	0.0732	72.87	0.2868	(=)	0.0407	93	0.097

(b) DM												
EXP	Scenario	#Tags		#Locs	#Clusters	%SP	%TAL	AvgLocError	#Clusters	%SP	%TAL	AvgLocError
1	Stay	1		3	7 (+133%)	0.0350	90.00	0.1131	3 (=)	0.015	97	0.0422
2	No Stay	1		5	5(=)	0.0625	72.5	0.29520	5 (=)	0.0625	80	0.27404
3	Stay	2	tag1	5	14 (180%)	0.0483	80.34	0.1818	5 (=)	0.0172	98.27	0.0241
			tag2	5	12 (+140%)	0.0414	85.51	0.1480	5 (=)	0.0172	98.27	0.024
4	No Stay	2	tag1	3	3 (=)	0.0667	62.22	0.3898	3 (=)	0.066	93.33	0.0936
			tag2	3	3 (=)	0.0667	53.33	0.4996	3 (=)	0.066	91.11	0.124
Mean					+75%	0.0534	73.98	0.2713	(=)	0.0407	93	0.097

(c) CLRC												
EXP	Scenario	#Tags		#Locs	#Clusters	%SP	%TAL	AvgLocError	#Clusters	%SP	%TAL	AvgLocError
1	Stay	1		3	5 (+67%)	0.0250	92.00	0.0913	3 (=)	0.015	97	0.0422
2	No Stay	1		5	5 (=)	0.0625	72.5	0.29150	5 (=)	0.0625	80	0.27404
3	Stay	2	tag1	5	11 (+120%)	0.0379	88.62	0.1242	5 (=)	0.0172	98.27	0.0241
			tag2	5	4 (-20%)	0.0138	75.51	0.2050	5 (=)	0.0172	98.27	0.024
4	No Stay	2	tag1	3	3 (=)	0.0667	62.22	0.3898	3 (=)	0.066	93.33	0.0936
			tag2	3	3 (=)	0.0667	53.33	0.4996	3 (=)	0.066	91.11	0.124
Mean					+27%	0.0454	74.03	0.2669	(=)	0.0407	93	0.097

6.1. *Experimental setup*

For evaluating the effectiveness of the presented approach, we conducted several experiments in different scenarios, collecting data from people wearing RFID tags. The experimental scenarios are all set in three indoor locations $\lambda_j = \{1, ..., 3\}$ and capture different possible movement behaviors: 1) " No Stay", where people rapidly move between locations without staying in any specific one; and 2) "Stay", where people move between locations and spend some time in each of them. Both types of scenarios have been tested with one/multiple antennas and tags.

During the training phase, we used a single person as a probe to collect RSSI samples from the tag for each of three locations $\lambda_1, \lambda_2, \lambda_3$, and then performed MLE on them in order to map the locations and to learn observation probability distributions. During the testing phase, instead, particle filtering is applied to infer/track the location of the RFID tags attached to people or objects. Particle filtering has been initialized with 500 particles where initial probability distribution for each location is uniform. Regarding the prediction, a uniform transition matrix has been defined according to a map of locations, where the probability of moving from one location to the others is uniform for all but the case of two locations which are not directly connected, in which case the probability is set to zero.

6.2. *Online filtering & uncertainty management*

The experimental scenarios and the obtained results from a single antenna (1- ANT) and two antennas (2-ANT) are summarized in the left, middle and right parts of Table 1, respectively. For each experiment, we evaluate the results on the basis of two parameters: (a) percentage of time at actual location (%TAL); and (b) average location error (AvgLocError) between estimated and actual locations. Specifically, %TAL is the percentage of time for which the estimated answer reports the same location as the ground truth (the higher the value the better). Moreover, the AvgLocError measure is devised to highlight what we really think is crucial in this evaluation, i.e. how long and how much the estimated value differs from the ground truth: it is calculated by means of an average Euclidean distance between the ground truth and the estimated value over the total time span, only considering those time instants

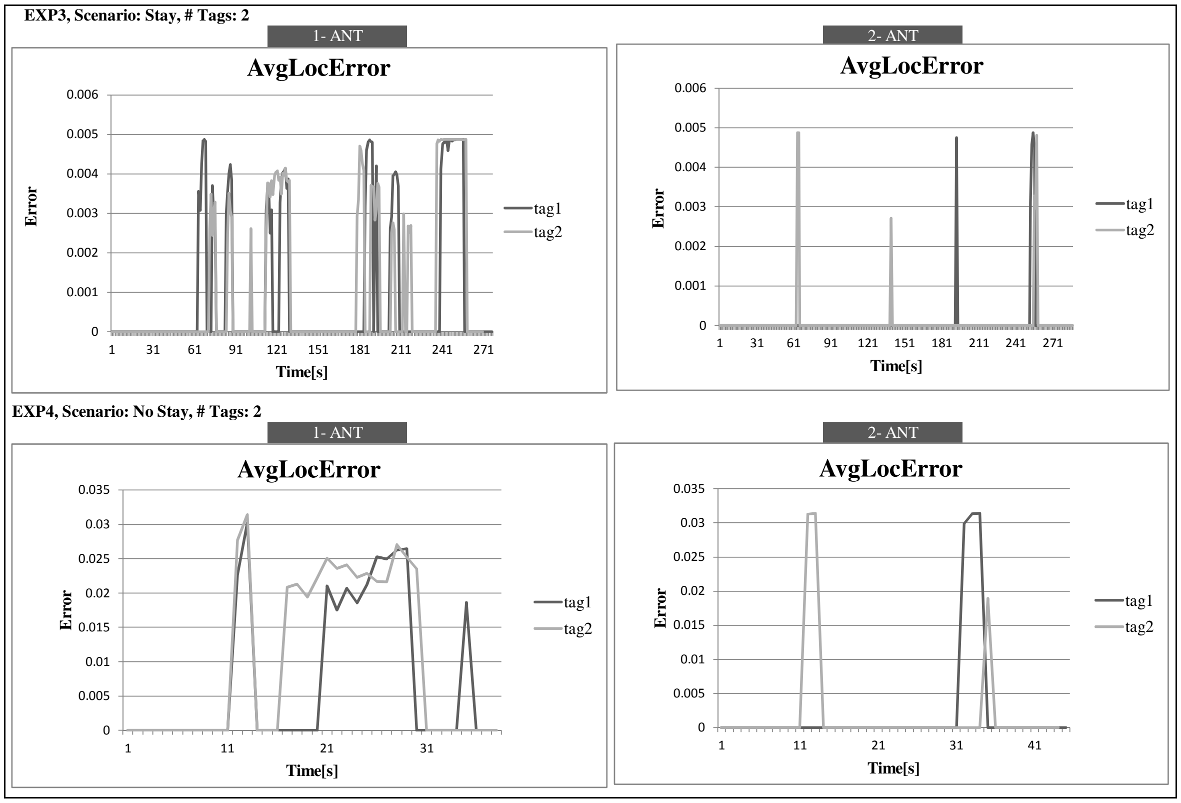

Fig. 8. Average Location Error (AvgLocError) between estimated and actual locations by using one and two antennas

when a "wrong" location is reported. The values of AvgLocError are between 0 and 1, where the lower the value the better the estimate.

From the results shown in Table 1, we see that, in the simplest scenarios such as the one of EXP1, the system is able to obtain good results even with a single antenna (%TAL of 89%). Moreover, as one would expect, the use of two antennas give more accurate and reliable results, reaching very satisfying levels in each of the tested scenarios. The main reason behind this is that, thanks to our method, two antennas give better coverage of the mapped area by reducing noise and resolving conflicting readings in overlapping areas as compared to single antenna. On average, %TAL and AvgLocError for a single antenna is 76% and 0.25, respectively, and for two antennas is %92 and 0.084. For instance, in EXP4 when using one antenna %TAL for tag1 and tag2 is 73% and 64% respectively, while using two antennas gives greatly improved results, i.e. 93% and 91% for tag1 and tag2 respectively. Similarly, the highest reported average location error with a single antenna is 0.37 in EXP4 for tag2. On the other hand, for two antennas it is only 0.084. Figure 8 shows a detailed graphical comparison of average location error for one and two antennas. For instance, looking at EXP4, two antennas have very few time instances having an erroneous estimated value.

6.3. Data aggregation

In each data aggregation experiment (see Table 2), we measureD the effectiveness of the methods based on four parameters: (a) number of output clusters (#Cluster); (b) percentage of occupied space w.r.t. non-aggregated data (%SP); (c) percentage of time at actual location (%TAL); and (d) average location error (AvgLocError) between clustered and actual locations. The basic intuition for (a) is that the nearer it is to the number of actually visited locations, the more effective is the method; (b) provides a clear quantification of the space required by the aggregated tuples (the smaller the percentage the higher the saved space). Beyond these "overview" approaches, (c) and (d) provide us with more detailed information on the actual

Q1. Find who was at location 'L1' in first minute?	
Filtered Probabilistic Data	**Aggregated Probabilistic Data**

SELECT TagId, conf()
FROM non_agg_data
WHERE LocationId='L1'
AND instant<=(select starttime() +
interval '00:01:00')
GROUP BY TagId;
	SELECT TagId, conf()
	FROM agg_data
	WHERE LocationId='L1'
	AND time_in<=(select starttime() +
	interval '00:01:00')
	GROUP BY TagId;

Q2. Find where was 'P1' in the last 20 seconds?	
Filtered Probabilistic Data	**Aggregated Probabilistic Data**

SELECT LocationId, conf()
FROM non_agg_data
WHERE TagId='P1'
AND instant >= (select endtime() -
interval '00:00:20')
GROUP BY LocationId;
	SELECT LocationId, conf()
	FROM agg_data
	WHERE TagId='P1'
	AND time_out>=(select endtime()-
	interval '00:00:20')
	GROUP BY LocationId;

Q3. Was 'P2' at 'L2' one minute ago?	
Filtered Probabilistic Data	**Aggregated Probabilistic Data**

SELECT conf()
FROM non_agg_data
WHERE LocationId ='L2'
AND TagId= ‘P2'
AND instant = 'T' - interval
'00:01:00' ;
	SELECT conf()
	FROM agg_data
	WHERE LocationId ='L2'
	AND TagId= 'P2'
	AND time_in< 'T'- interval
	'00:01:00'
	and time_out> 'T'- interval
	'00:01:00';

Fig. 9. Three sample queries in plain text and MayBMS form

contents of the generated clusters. More specifically, in this case (c) gives us an idea about the promptness of each method to adjust the output to the ground truth over the experiment duration (the higher the value the better), while (d), similarly to the previous experiments, informs us of how long and how much the ouput of each method differs from the ground truth.

We start by evaluating the experimental results obtained with one antenna (middle part of Table 2 (a, b, c)). We found that MPC is very sensitive to noise and thus performs poorly in the presence of noisy data. On average it makes 123% more clusters than expected (up to 260% more in EXP3), while average location error is quite high, for instance with values of 0.44 and 0.49 for EXP4 (0.2868 on mean for all the experiments). %TAL is about 72% on mean, with the lowest values being 53% (EXP4). DM performs better than MPC but its diameter can quickly become very large in presence of noisy data. DM has an average location error of 0.2713 and average %TAL of approximately 73%, while it makes 75% more clusters than expected. CLRC shows superior performance to MPC and DM, giving good results even in noisy environments. The average %TAL is about 74%, whereas the average location error is approximately 0.2669; on average, it only makes 27% more clusters than expected, which, together with the other figures, represents a very encouraging result. The same holds for the very consistent space savings produced by all methods (ranging from 0.07% of the space required by non-aggregated data to the most compact 0.04%, given by MPC and CLRC, respectively).

As to the experimental results obtained from two antennas (right part of Table 2 (a, b, c)), as expected the system is able to successfully exploit the additional resources achieveing results which outperform the ones from a single antenna. In this case, all three methods performed well due to the fact that multiple antennas allow the system to gain better coverage of the mapped area and to estimate more accurate values, since noise values are reduced and conflicting readings are more easily solved. When using multiple antennas, the #Cluster are equal to number of actually visited locations. This means that transitions are much easier to trace while using two antennas. Furthermore, %TAL is quite high, 93% on mean for all the experiments, with the highest values being 98% and 97% for EXP3 and EXP1, respectively. Average location error is quite low (0.097 on mean for all the experiments) as compared to the error reported in case of a single antenna.

Finally, as mentioned in section 3, the system stores the final output in the data storage and query processing layer, where we can perform temporal probabilistic queries on the stored probabilistic data. Some examples of the supported temporal probabilistic queries for online filtering & uncertainty management (filtered probabilistic data) and data aggregation (aggregated probabilistic data) modules are shown in Figure 9. Each query shows its plain text form and its MayBMS (SQL) form. Note that `conf()` is the MayBMS function for calculating the confidence of the answer, while `starttime()` and `endtime()` are user-defined functions for retrieving the startup time and end time of the used data set, respectively.

7. Conclusions

RFID data streams are becoming more and more widespread, however their noisy, redundant and unreliable nature can make their exploitation and management difficult. In particular, avoiding missing/conflicting readings and being able to extract high-level complex events from the huge volumes of low-level atomic events acquired by the sensors, while always avoiding to store unnecessary information, are particularly critical and challenging tasks.

In this paper, we presented the RPDM realtime system for RFID probabilistic data management which achieves these goals (a) by transforming raw RFID data into reliable meaningful probabilistic data streams and (b) by providing a small space representation for the resulting data while preserving meaningful information. This is accomplished by means of newly proposed mechanisms for online filtering, uncertainty management and on-line summarization.

The highlights of the proposed realtime system for RFID probabilistic data management, also in the light of the succesful experimental evaluation we performed in real-world object tracking scenarios, are as follows:

– Contrary to application specific summarization / aggregation techniques [10,20,21,36], it is capable of performing both off-line and on-line processing of RFID data streams;
– It conveys the RFID data to higher level data information modules, in addition to filtering inaccuracy errors and smoothing raw RFID streams. Instead, most filtering approaches, such as [15,16], filter and smooth data before feeding it into higher level applications, thus not dealing (and not exploiting) the "meaning" of the managed information.
– It works without knowing in advance any specific information characterizing data uncertainty, such as the entire probability density function or standard error data available, or the number of clusters to be produced (this is not true for most other clustering approaches, such as [3,35,49,40]);
– Differently from many off-line only storage models (e.g. [13,18]), it detects state transitions, thus avoiding the redundant information generated in stable states;
– It is ultimately able to correctly handle the online management of RFID data uncertainties and to estimate the correct states.

References

[1] http://www.cs.cornell.edu/bigreddata/maybms/.
[2] http://www.postgresql.org/.
[3] C.C. Aggarwal and P.S. Yu. A framework for clustering uncertain data streams. In *Proceedings of the 24th international conference on Data Engineering*, pages 150–159. IEEE, 2008.
[4] Doucet Arnaud, Nando de Freitas, and Gordon Neil. *Sequential Monte Carlo Methods in Practice*. Springer, 2005.
[5] Y. Bai, F. Wang, and P. Liu. Efficiently filtering RFID data streams. In *CleanDB Workshop*, pages 50–57, 2006.
[6] D. Bleco and Y. Kotidis. RFID Data Aggregation. *GeoSensor Networks*, pages 87–101, 2009.
[7] S. S Chawathe, V. Krishnamurthy, S. Ramachandran, and S. Sarma. Managing RFID data. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 1189–1195, 2004. ISBN 0120884690.
[8] H. Chen, W.S. Ku, H. Wang, and M.T. Sun. Leveraging spatio-temporal redundancy for RFID data cleansing. In *Proceedings of the 2010 international conference on Management of data*, pages 51–62. ACM, 2010.
[9] H. Cho, S. Son, J. Kim, and Y. Baek. Time Synchronization of Distributed Readers for a Large-Scale Active RFID Network. In *11th International Conference on Computer and Information Technology (CIT)*, pages 204–211. IEEE, 2011.
[10] R. Cocci, T. Tran, Y. Diao, and P. Shenoy. Efficient data interpretation and compression over RFID streams. In *24th International Conference on Data Engineering, ICDE'08.*, pages 1445–1447. IEEE, 2008.
[11] R. Cucchiara, M. Fornaciari, R. Haider, F. Mandreoli, R. Martoglia, A. Prati, and S. Sassatelli. A reasoning engine for intruders' localization in wide open areas using a network of cameras and RFIDs. In *Proceedings of 1st IEEE Workshop on Camera Networks and Wide Area Scene Analysis*. IEEE, 2011.
[12] P. Darcy, B. Stantic, and A. Sattar. Applying a neural network to recover missed RFID readings. In *Proceedings of the Thirty-Third Australasian Conferenc on Computer Science-Volume 102*, pages 133–142. Australian Computer Society, Inc., 2010.
[13] R. De Virgilio, P. Sugamiele, and R. Torlone. Incremental aggregation of RFID data. In *Proceedings of the 2009 In-*

ternational Database Engineering & Applications Symposium, pages 194–205. ACM, 2009.

[14] R. Derakhshan, M.E. Orlowska, and X. Li. RFID data management: challenges and opportunities. In *IEEE International Conference on RFID*, pages 175–182. IEEE, 2007.

[15] A. Deshpande, C. Guestrin, and S. Madden. Using probabilistic models for data management in acquisitional environments. In *Proceedings of Conference on Innovation Data Systems Research (CIDR)*, pages 317–328, 2005.

[16] A. Deshpande, C. Guestrin, S. R Madden, J. M Hellerstein, and W. Hong. Model-based approximate querying in sensor networks. *The VLDB Journal*, 14(4):417–443, 2005. ISSN 1066-8888.

[17] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining*, volume 1996, pages 226–231. Portland: AAAI Press, 1996.

[18] B. Fazzinga, S. Flesca, E. Masciari, and F. Furfaro. Efficient and effective RFID data warehousing. In *Proceedings of International Database Engineering & Applications Symposium*, pages 251–258. ACM, 2009.

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

[20] H. Gonzalez, J. Han, and X. Li. Flowcube: constructing RFID flowcubes for multi-dimensional analysis of commodity flows. In *Proceedings of the 32nd international conference on Very large data bases*, pages 834–845. VLDB Endowment, 2006.

[21] H. Gonzalez, J. Han, X. Li, and D. Klabjan. Warehousing and analyzing massive RFID data sets. In *22nd International Conference on Data Engineering, ICDE'06.* IEEE Computer Society, 2006.

[22] S. Guha, R. Rastogi, and K. Shim. CURE: an efficient clustering algorithm for large databases. In *ACM SIGMOD Record*, volume 27, pages 73–84. ACM, 1998.

[23] W. C. Hu and Z. L. Cheng. Clustering algorithm for probabilistic data streams over sliding window. In *Proceedings of the 9th International Conference on Machine Learning and Cybernetics (ICMLC)*, pages 2065–2070. IEEE, 2010.

[24] Y. Hu, S. Sundara, T. Chorma, and J. Srinivasan. Supporting RFID-based item tracking applications in Oracle DBMS using a bitmap datatype. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 1140–1151. VLDB Endowment, 2005.

[25] J. Huang, L. Antova, C. Koch, and D. Olteanu. MayBMS: a probabilistic database management system. In *Proceedings of the 35th SIGMOD international conference on Management of data*, pages 1071–1074. ACM, 2009.

[26] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Compututing Survey*, 31:264–323, September 1999. ISSN 0360-0300.

[27] S. Jeffery, G. Alonso, M. Franklin, W. Hong, and J. Widom. Declarative support for sensor data cleaning. *Pervasive Computing*, pages 83–100, 2006.

[28] S. R Jeffery, M. J Franklin, and M. Garofalakis. An adaptive RFID middleware for supporting metaphysical data independence. *The VLDB Journal: The International Journal on Very Large Data Bases*, 17(2):289, 2008. ISSN 1066-8888.

[29] S. R Jeffery, M. Garofalakis, and M. J Franklin. Adaptive cleaning for RFID data streams. In *Proceedings of the 32nd international conference on Very large data bases*, pages 163–174, 2006.

[30] Love Kalra, Xinghui Zhao, Axel J. Soto, and Evangelos E. Milios. Detection of daily living activities using a two-stage markov model. *JAISE*, 5(3):273–285, 2013.

[31] B. Kanagal and A. Deshpande. Online filtering, smoothing and probabilistic modeling of streaming data. In *IEEE 24th International Conference on Data Engineering (ICDE'08)*, pages 1160–1169, 2008.

[32] N. Khoussainova, M. Balazinska, and D. Suciu. Towards correcting input data errors probabilistically using integrity constraints. In *Proceedings of the 5th ACM international workshop on Data engineering for wireless and mobile access*, pages 43–50, 2006. ISBN 1595934367.

[33] N. Khoussainova, M. Balazinska, and D. Suciu. Probabilistic event extraction from RFID data. In *Proceedings of the 24th International Conference on Data Engineering (ICDE)*, pages 1480–1482, 2008.

[34] D.S. Kim, J. Kim, S.H. Kim, and S.K. Yoo. Design of RFID based the Patient Management and Tracking System in hospital. In *30th Annual International Conference of Engineering in Medicine and Biology Society, EMBS.*, pages 1459–1461. IEEE, 2008.

[35] H.P. Kriegel and M. Pfeifle. Density-based clustering of uncertain data. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 672–677. ACM, 2005.

[36] C.H. Lee and C.W. Chung. Efficient storage scheme and query processing for supply chain management using RFID. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 291–302. ACM, 2008.

[37] C. Legány, S. Juhász, and A. Babos. Cluster validity measurement techniques. In *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, pages 388–393, 2006.

[38] K.S. Leong, M.L. Ng, A.R. Grasso, and P.H. Cole. Synchronization of RFID readers for dense RFID reader environments. In *International Symposium on Applications and the Internet Workshops, SAINT Workshops*. IEEE, 2006.

[39] H. Mahdin and J. Abawajy. An approach to filtering RFID data streams. In *10th International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN)*, pages 742–746. IEEE, 2009.

[40] W.K. Ngai, B. Kao, C.K. Chui, R. Cheng, M. Chau, and K.Y. Yip. Efficient clustering of uncertain data. In *Proceedings of the 6th International Conference on Data Mining(ICDM),*, pages 436–445. IEEE, 2006.

[41] X. Peng, Z. Ji, Z. Luo, E.C. Wong, and CJ Tan. A P2P collaborative RFID data cleaning model. In *3rd International Conference on Grid and Pervasive Computing Workshops, GPC Workshops' 08.*, pages 304–309. IEEE, 2008.

[42] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Readings in speech recognition*, pages 267–296. 1990.

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

[44] C. Ré, J. Letchner, M. Balazinksa, and D. Suciu. Event queries on correlated probabilistic streams. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of*

data, pages 715–728, 2008.

[45] T. Tran, C. Sutton, R. Cocci, Y. Nie, Y. Diao, and P. Shenoy. Probabilistic inference over RFID streams in mobile environments. In *IEEE International Conference on Data Engineering*, pages 1096–1107, 2009.

[46] T. L. M. van Kasteren, G. Englebienne, and B. J. A. Kröse. Activity recognition using semi-markov models on real world smart home datasets. *JAISE*, 2(3):311–325, 2010.

[47] J. Vongkulbhisal and Y. Zhao. R-labs: An rfid-based indoor localisation system using antenna beam scanning. *JAISE*, 5(2): 251–266, 2013.

[48] E. Welbourne, N. Khoussainova, J. Letchner, Y. Li, M. Balazinska, G. Borriello, and D. Suciu. Cascadia: a system for specifying, detecting, and managing RFID events. In *Proceeding of the 6th international conference on Mobile systems, applications, and services*, pages 281–294. ACM, 2008.

[49] H. Xu and G. Li. Density-based probabilistic clustering of uncertain data. In *Proceedings of International Conference on Computer Science and Software Engineering*, pages 474–477. IEEE, 2008.

[50] C. Zhang, M. Gao, and A. Zhou. Tracking high quality clusters over uncertain data streams. In *25th International Conference on Data Engineering, ICDE'09.*, pages 1641–1648. IEEE, 2009.

[51] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *ACM SIGMOD Record*, volume 25, pages 103–114. ACM, 1996.