Spatiotemporal Data Cleansing for Indoor RFID Tracking Data

Asif Iqbal Baba, Hua Lu, Xike Xie, Torben Bach Pedersen
Department of Computer Science, Aalborg University, Denmark
{asif, luhua, xkxie, tbp}@cs.aau.dk

***Abstract*—The Radio Frequency Identification (RFID) is increasingly being deployed in indoor tracking systems, such as inventory management and airport baggage monitoring, etc. Effective RFID data management is expected to be able to support various applications that range from monitoring to analysis. However, the "dirtiness" in raw RFID readings hinder the progress of applying meaningful high level applications. Hence, it is indispensable to cleansing RFID data in such systems.**

In this paper, we focus on two quality aspects in raw indoor RFID data: temporal redundancy and spatial ambiguity. The former refers to the large number of repeated readings for the same object and the same RFID reader during a short period of time. The latter refers to the undetermined whereabouts of an object due to multiple RFID readings by different readers simultaneously. We investigate the spatiotemporal characteristics of indoor spaces as well as RFID reader deployment, and exploit them in designing effective data cleansing techniques. Specifically, we aggregate raw RFID readings to reduce temporal redundancy; we design a distance-aware graph to resolve spatial ambiguity with respect to the indoor topology and the RFID reader deployment captured in the graph. We evaluate the spatiotemporal data cleansing techniques using both real and synthetic datasets. The experimental results demonstrate that the proposed techniques are effective and efficient in cleansing indoor RFID tracking data.

I. Introduction

The Radio Frequency Identification (RFID) is increasingly being deployed in indoor tracking systems, e.g., airport baggage monitoring. Without physical sight or contact, an RFID tag attached to a moving object (e.g., a bag in an airport) makes the object seen by an RFID reader when the object is in the reader's detection range. As a result, the RFID reader reports the object's presence to the database that manages the object positions. When multiple RFID readers are deployed in an indoor space like an airport, objects like bags with RFID tags are tracked by the reports from those readers.

Effective RFID tracking data management is expected to support various applications that range from monitoring to analysis of indoor moving objects. However, noises and errors abound in raw RFID data. Such "dirtiness" comes from different sources. Essentially, radio frequency waves are not steady and therefore the detection range of an RFID may change from time to time, especially in an indoor space where there are various signal reflecting and/or blocking entities like walls as well as changing flows of people. Such dirtiness hinders the progress of applying meaningful high level applications, and therefore we need to clean indoor RFID data.

In this paper, we focus on two kinds of dirtiness in indoor RFID tracking data.

Temporal Redundancy: An RFID reader report $(readerID, objectID, time)$ means the object identified by $objectID$ is seen by the reader identified by $readerID$ at time point $time$. A tagged object can be read many times by the same reader within a short period, depending on the sampling frequency configured for a reader[1].

Spatial Ambiguity: An tagged object can be read by multiple readers simultaneously. This may result from the unexpected change of the detection range of a reader nearby. Such changes happen due to various reasons, e.g., metal items that reflect the signals or the re-direction of reader antenna(s). As a result, the object is reported by multiple readers that are places at different positions and spatial ambiguities are caused.

In other words, redundant readings arise temporally when an object is detected multiple times by a device. In contrast, ambiguous readings appear spatially where an object is detected by multiple readers simultaneously. In this paper, we focus on these two issues in an indoor setting. Exploiting the spatiotemporal constraints implied by indoor RFID reader deployment, we design data cleansing techniques to remove the temporal redundancy and reduce the spatial ambiguity.

An example is shown in Figure 1. There are two readers r_1 and r_2 in two rooms respectively. An object O_1 moves in the left room from time point t_1 to time point t_5, which yields five reports by r_1 as the trajectory is within r_1's detection range. This causes temporal redundancy in the RFID data. If the times points $t_1, \ldots, t_5$ are very close to each other, we compress the five reports into a single tuple $\langle r_1, O_1, [t_1, t_5] \rangle$.

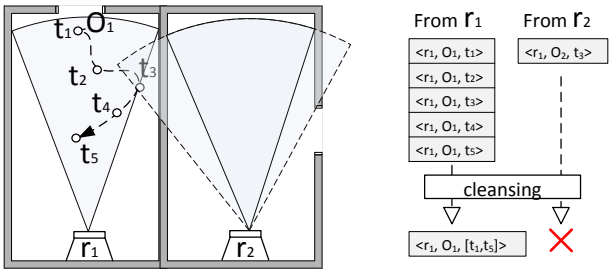


Fig. 1. An example of RFID data cleansing

On the other hand, at time point t_3, object O_1 is detected by both readers. This is due to the unexpected expansion of r_2's

[1] An RFID reader's read rate can be as high as over 100 per second [22].

detection range, as shown by the dashed range. Consequently, from the RFID data, O_1 seems to be in both rooms at same time t_3. Thus spatial ambiguity is caused. However, due to spatial constrains, O_1 can not appear in both locations, as they are separated by walls. Neither can O_1 move from one room to another room if time points t_2 and t_3 are very close and the distance between the two rooms (through the two doors) are long enough. By considering such spatiotemporal constraints, our cleansing technique is able to remove spatial ambiguous reports such as $\langle r_2, O_1, t_3 \rangle$.

We make the following contributions in this paper.

- We formulate the data cleansing problem for RFID data obtained in indoor spaces.
- We design a threshold based temporal cleansing algorithm to eliminate temporal redundancy in indoor RFID data.
- We propose a distance-aware deployment graph to capture indoor spatiotemporal constraints, and design a spatial cleansing algorithm that utilizes the graph to identify and reduce spatial ambiguity in indoor RFID data.
- We conduct extensive experimental studies to evaluate our spatiotemporal cleansing techniques. The experimental results demonstrate that our cleansing techniques are effective, efficient and scalable.

The rest of this paper is organized as follows. Section II reviews the related works on RFID data management/cleansing and symbolic indoor tracking. Section III formulates the indoor RFID data cleansing problem that we tackle in this paper. Section IV details the temporal cleansing algorithm, followed by the spatial cleansing techniques in Section V. Section VI presents extensive experiments on both synthetic and real data sets. Finally, Section VII concludes the paper and discusses the directions for future research.

II. Related Work

In this section we review related work. We cover RFID data management and cleansing in Section II-A, and symbolic indoor tracking in Section II-B.

A. RFID Data Management and Cleansing

In recent years RFID technology has been widely used in many scenarios such as supply chain management [6], [7], [13], health care [5], people and object tracking [2]–[4], [8], [9]. Massive amounts of RFID data is generated by RFID-based applications, e.g., Wal-Mart produces about 7 terabytes of RFID data per day [7]. Roozbeh et al. [19] discuss the general challenges for RFID data management.

As raw RFID data are not clean, cleansing is needed. To handle missing readings, smoothing filters [10], [17] are employed in the steaming context. Mylyy [17] proposes a temporal smoothing filter with a fixed time window. By counting the RFID readings in the filter window and comparing them to given thresholds, this method reduces missing RFID readings from the data stream. However, it is difficult to decide the best window size for varying RFID data. Jeffery et al [10] proposes adaptive SMURF (*Statistical sMoothing for Unreliable RFid data*). Modeling the RFID data streams as statistical samples of RFID tags, SMURF uses sampling estimators to automatically adjust the filter window size.

On the other hand, techniques are also developed to handle duplicates in RFID data. Mahdin et al. [16] use count Bloom Filters to remove duplicate RFID data at the reader level. Assuming that cross reads are always less than normal reads, Bai et al. [1] employs a counting based smoothing filter to remove cross reads in RFID streams. Liao et al. [14] proposes a probability model to estimate the tag density for each RFID reader, and utilize the model to remove cross reads. Tinggaard et al [20] designs a data warehouse for Bluetooth/RFID tracking data obtained from airport passengers, in order to facilitate flow analysis on uncertain passenger movements.

Unlike these works [1], [10], [14], [16], [17] that conduct cleansing in pre-processing, Rao et al. [18] proposes a deferred approach that uses declarative sequenced-based rules to conduct data cleansing at query execution time.

This paper differs from the aforementioned related works in two important aspects. First, it focuses on off-line indoor RFID positioning data rather than general streaming RFID data. Second, it exploits the spatiotemporal constraints implied in an indoor space in data cleansing.

B. Symbolic Indoor Tracking

We briefly review symbolic indoor positioning and tracking using the example floor plan shown in Figure 2. The indoor space has several rooms and a corridor that connects all the rooms. Each room has one or more doors. A number of RFID readers are placed at the doors.

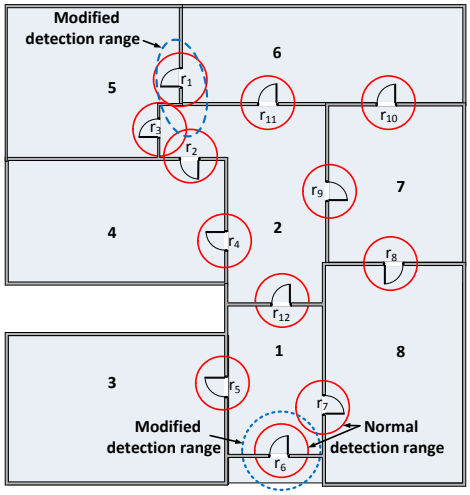


Fig. 2. Positioning device deployment

1) Raw Positioning Data: Symbolic indoor tracking employs the proximity analysis. An object is only seen when it is within the detection range of a positioning device, e.g., an RFID reader. Accordingly, the object's seen position is indicated by the RFID reader's identifier in the raw reading. Each *raw reading* is in the format of (*deviceID*, *objectID*, *t*), which means that the object identified by *objectID* is detected by the device identified by *deviceID* at time *t*.

Usually the detection range of a positioning device is a circular region with a pre-specified radius. A positioning device continuously detects objects that are in its range, with the frequency determined by its sampling rate. All the positioning devices in an indoor space generate large amount of raw readings. For the floorplan example shown in Figure 2, an example table of raw readings is shown in Table I.

TABLE I
RAW READINGS TABLE

deviceID	objectID	t
r_6	$object_1$	t_0
r_6	$object_1$	t_1
r_6	$object_1$	t_2
r_7	$object_1$	t_3
r_6	$object_1$	t_4
r_7	$object_1$	t_5
r_7	$object_1$	t_6
r_6	$object_1$	t_7
r_7	$object_1$	t_8
r_7	$object_1$	t_9
r_7	$object_1$	t_{10}
r_8	$object_1$	t_{14}
r_8	$object_1$	t_{15}
r_8	$object_1$	t_{16}
r_9	$object_1$	t_{16}
r_9	$object_1$	t_{19}

deviceID	objectID	t
r_9	$object_1$	t_{20}
r_9	$object_1$	t_{21}
r_1	$object_1$	t_{24}
r_1	$object_1$	t_{25}
r_1	$object_1$	t_{26}
r_1	$object_1$	t_{27}
r_2	$object_1$	t_{29}
r_2	$object_1$	t_{30}
r_2	$object_1$	t_{31}
r_2	$object_1$	t_{32}
r_3	$object_1$	t_{33}
r_3	$object_1$	t_{34}
r_3	$object_1$	t_{35}
r_3	$object_1$	t_{36}
r_3	$object_1$	t_{37}
r_3	$object_1$	t_{38}

2) Graph Based Indoor Tracking: Jensen et al. [11] propose a graph based approach for indoor tracking based on the raw positioning data generated by devices like RFID readers. A *deployment graph* is constructed to capture the deployment of positioning devices and the indoor topology. In particular, a partitioning device is one that partitions the indoor space into different parts such that an object must be seen by the device when the object moves from one part to the other. All such partitioning devices partition the indoor space into different cells. A device deployment graph G_{dev} is formally defined as labeled graph $G_{dev} = \big(V, E, D, \mathcal{L}_E\big)$, where:

1) V is the set of vertices; each $v_i \in V$ represents a cell.
2) E is the set of edges, where $E = \{(v_i, v_j) \mid v_i, v_j \in V \wedge v_i \neq v_j\}$. An edge between two cells indicate that objects can move from one cell to the other under the surveillance of one or more devices.
3) $D = \mathcal{S}_{device}$ is a set of positioning devices.
4) $\mathcal{L}_E : E \rightarrow 2^D$ maps an edge to a subset of D. Specifically, an edge is labeled by the identifiers of those devices that monitor the edge.

Figure 3 presents the deployment graph for the indoor setting shown in Figure 2. Note that vertex 9 in the graph represents the outdoor space.

In addition to the deployment graph, two mapping structures [11] are defined to facilitate the indoor tracking. First, mapping $D2V : D \rightarrow 2^V$ returns a set of cells $D2V(r_i)$ that have a given device r_i on their edges. Referring to Figure 3, device r_1 is in the label of the edge between cells 5 and 6, so $D2V(r_1) = \{5, 6\}$. In capturing the possible movements of objects the mapping is quite useful. If an object's movement is observed by device r_i at time t, it must have moved from a cell in $D2V(r_i)$. Likewise if an object leaves device r_i, it definitely

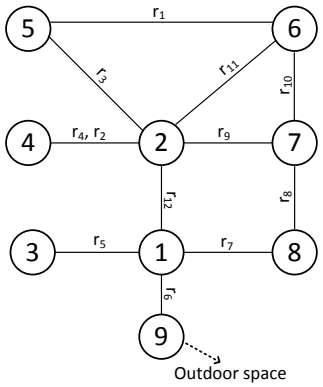


Fig. 3. Deployment graph for the deployment in Figure 2

enters a cell in $D2V(r_i)$. Second, mapping $V2D : V \rightarrow 2^D$ returns the set of devices $V2D(v_i)$ that monitor the edges of the given cell v_i. In the example shown in Figure 3, $V2D(1)$ returns $\{r_6, r_5, r_7, r_{12}\}$.

This paper distinguishes itself from the previous work [11] with several important characteristics. First, this paper considers the overlaping between different positioning devices, whereas the previous work [11] assumes that devices do not overlap. Second, in relation to the first point, this paper is intended to decide where an object really is when it is seen by multiple devices, whereas the previous work [11] focuses on tracking the object when it is not seen by any device. This important difference is illustrated in Figure 4. Third, in order to support data cleansing, this paper proposes a graph (Section V-A) different from the deployment graph [11].

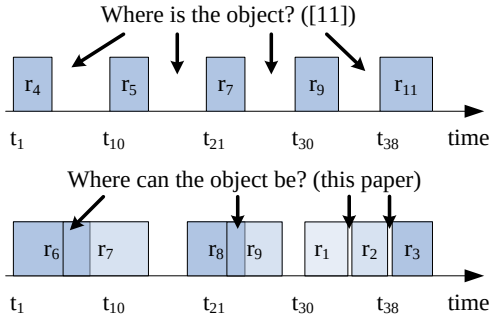


Fig. 4. Difference from the previous work

III. PROBLEM FORMULATION

In this section we formulate the data cleansing problem that we tackle in this work. We present the definitions and the main cleansing tasks in Section III-A. We give an overview of our solution in Section III-B. Table II lists the notation used throughout the paper.

A. Definitions and Tasks

Without loss of generality, we assume that all raw readings are ordered by their detection times in the Raw Reading Table (RRT). We formally define other concepts as follows.

Definition 1: (**Temporal Redundant Readings**) Two raw readings rr_i and rr_j are temporal redundant readings if

$|rr_i.t - rr_j.t| \leq \tau$,[2] $rr_i.deviceID = rr_j.deviceID$ and $rr_i.objectID = rr_j.objectID$.

TABLE II
NOTATION

Notation	Meaning
r_i, r_j	RFID readers
rr, rr_i, rr_j	Raw RFID readings
tr, tr'	tracking records
RRT	Raw reading table
ATT	Aggregate tracking table
G_{dd}	Distance-aware deployment graph
dt_i	The minimum dwell time of reader r_i
$d_{i,j}$	The minimum indoor distance between r_i and r_j
$S_{i,j}$	The maximum indoor moving speed between r_i and r_j

Definition 2: (**Tracking Record**) Given a series of temporal redundant readings $rr_1, rr_2, \ldots rr_k$ ($rr_1.t < rr_2.t < \ldots < rr_k.t$), a tracking record tr is a temporal aggregation of them. Formally, tr is in the format ($deviceID$, $objectID$, t_s, t_e), where $tr.deviceID = rr_i.deviceID$, $tr.objectID = rr_i.objectID$, $tr.t_s = rr_1.t$, and $tr.t_s = rr_k.t$ for $1 \leq i \leq k$.

A tracking record states that the object ($objectID$) is detected by a device ($deviceID$) during the time interval $[t_s, t_e]$.

Definition 3: (**Spatial Ambiguous Tracking Records**) Two tracking records tr' and tr are spatial ambiguous if $tr'.deviceID \neq tr.deviceID$ and $tr'.objectID = tr.objectID$, if $tr'.[t_s, t_e] \cap tr.[t_s, t_e] \neq \emptyset$, or if $tr.t_s - tr'.t_e \leq min\_tt$. [3]

Given a raw reading table RRT, we intend to accomplish the following two tasks.

Task 1: Temporal Redundancy Elimination. This task is to aggregate raw readings into more meaningful tracking records. This way is expected to significantly reduce the data size without any information loss.

Task 2: Spatial Ambiguity Reduction. This task is to be done after the first task. Given a large set of tracking records, we identify spatial ambiguous tracking records and reduce such spatial ambiguities by referring to the spatiotemporal constraints imposed by the positioning device deployment as well as the indoor topology.

We call these two tasks together *spatiotemporal data cleansing*. We proceed to give a solution overview for it.

B. Overview of Spatiotemporal Data Cleansing

We design a two-phase solution for spatiotemporal data cleansing for indoor RFID data, as shown in Figure 5.

The first phase, called *Temporal Cleansing*, sequentially scans data from the raw reading table and generates more meaningful tracking records by aggregating on the time. The aggregation results are controlled by the threshold τ. If an object's two consecutive raw readings are apart from each other for a period longer than τ, they will be put into two different tracking records; otherwise, they will be aggregated into the same tracking record. We store all generated tracking records in the *Aggregate Tracking Table* (ATT).

[2] τ is an application-specific threshold [1], [21].

[3] min_tt is the minimum traveling time for an object to move from tr''s device to tr's device. It is to be detailed in Section V-C.

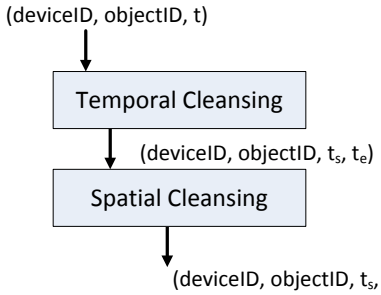


Fig. 5. Two phase spatiotemporal data cleansing

Consider all the raw readings about $object_1$ from Table I, and suppose that threshold τ is specified as four time units. The temporal cleansing on all those readings will generate the aggregate tracking table as shown in Table III. For example, tracking record $(r_6, object_1, t_0, t_7)$ means that $object_1$ is tracked by a reader r_6 from time t_0 to time t_7.

TABLE III
AGGREGATE TRACKING TABLE

deviceID	objectID	t_s	t_e
r_6	$object_1$	t_0	t_7
r_7	$object_1$	t_3	t_{10}
r_8	$object_1$	t_{14}	t_{16}
r_9	$object_1$	t_{16}	t_{21}
r_1	$object_1$	t_{24}	t_{27}
r_2	$object_1$	t_{29}	t_{32}
r_3	$object_1$	t_{33}	t_{38}

TABLE IV
RESULT OF SPATIAL CLEANSING

deviceID	objectID	t_s	t_e
r_6	$object_1$	t_0	t_7
r_7	$object_1$	t_8	t_{10}
r_8	$object_1$	t_{14}	t_{16}
r_9	$object_1$	t_{19}	t_{21}
r_1	$object_1$	t_{24}	t_{27}
r_2	$object_1$	t_{29}	t_{32}
r_3	$object_1$	t_{33}	t_{38}

The second phase, called *Spatial cleansing*, takes the aggregate tracking table as input, identifies possible spatial ambiguities, and reduce them by a distance-aware graph for all positioning devices in the indoor space.

We look at the idle time between tr and tr', i.e., $t_{idle} = tr.t_s - tr'.t_e$, and compare it with the minimum traveling time min_tt on object need to move from device $tr'.deviceID$ to device $tr.deviceID$. If $t_{idle} < min\_tt$, it is impossible for the object to move so fast from one device to the other, and therefore tracking record tr's time interval will be truncated accordingly. Note such a truncation may remove the entire tracking record tr if $tr.t_e$ is also too early to be possible with respect to the minimum movement time. Such minimum times between devices are captured in a distance-aware graph, to be detailed in Section V.

Refer to the running example whose ATT is shown in Table III. Assuming the minimum movement time between device r_6 and r_7 is one time unit, tracking record $(r_7, object_1, t_3, t_{10})$ is truncated to $(r_7, object_1, t_8, t_{10})$ by the spatial cleansing. The result is shown in Table IV.

In the example above, we assume that the first tracking record in ATT is correct and we make the spatial cleansing check each subsequent tracking record in ATT with respect to its previous tracking record tr' for the same object. This assumption holds in the airport scenario because the bags are first handled manually by the staff at check-in desks, and thus it is unlikely for the first readings to be incorrect. As a matter of fact, our spatial cleansing can be extended to other cases

where the known correct tracking record(s) is not the first one. In such a case, we need to make the spatial cleansing work both in the forward and/or backward directions starting from the known correct record. For example, in the airport scenario *"belt loader readers"* may be placed at an airplane and they thus yield known correct ending records.

IV. Temporal Cleansing

In this section, we detail the temporal cleansing that eliminates the temporal redundancy in RFID data.

Algorithm 1 describes the temporal cleansing process. The algorithm starts with the initialization of a new aggregate tracking table ATT (line 1). For each raw reading rr from RRT, all the aggregate tracking records with the same device and object as rr in the current ATT are fetched into set trs (line 4). If trs is not empty and rr is temporally close enough to an existing tracking record tr in trs, i.e., $rr.t - tr.t_e \leq \tau$, tr's time interval is extended to $rr.t$ and rr is processed (lines 5–10). Otherwise, rr cannot be combined to any existing tracking record, and therefore a new tracking record is created for it and inserted to ATT (lines 11–12).

Algorithm 1 TemporalCleansing(Raw reading table RRT, Threshold τ)

```
1:  ATT ← ∅; trs ← ∅
2:  for each rr ∈ RRT do
3:      flag ← false
4:      trs ← {tr ∈ ATT | tr.objectID = rr.objectID ∧
        rr.deviceID = tr.deviceID}
5:      if (|trs| > 0) then
6:          for each tracking record tr ∈ trs do
7:              if (rr.t − tr.t_e ≤ τ) then
8:                  tr.t_e ← rr.t;
9:                  flag ← true;
10:                 break
11:     if flag = false then
12:         insert (rr.deviceID,rr.objectID, rr.t, rr.t) to ATT
```

Note that the temporal cleansing here is characterized by its off-line process manner and the use of threshold τ to control the temporal aggregation. In contrast, the pre-processing employed in the previous work [11] assumes data are online in a stream and it does not use a time threshold.

We use the formula shown in Eq. 1 to set the threshold τ.

$$Threshold(\tau) = \frac{detection\ range\ diameter}{object\ moving\ speed} \quad (1)$$

Here, we estimate how long it takes an object to move through a circular detection range of a reader. An example is shown in Figure 6. In Section VI, we experimentally test the effect of this method for setting τ.

V. Distance-Aware Spatial Cleansing

In this section, we elaborate on spatial cleansing, i.e., reducing the spatial ambiguity by exploiting spatiotemporal constraints imposed by the RFID reader deployment and the indoor topology. In Section V-A, we describe a graph that captures those indoor spatiotemporal constraints. In Section V-B, we detail how to construct such a graph given the RFID reader deployment in an indoor space. In Section V-C, we propose the spatial cleansing algorithm using the graph.

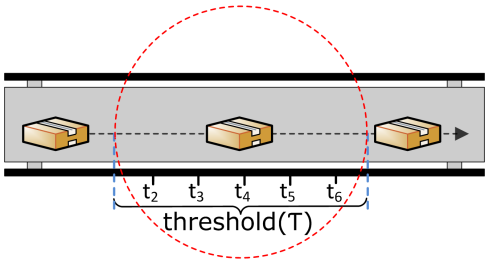


Fig. 6. An example of threshold setting

A. Distance-Aware Deployment Graph

As described in Section III, spatial ambiguity takes place when two tracking records temporally overlap or are too close to each other. For such cases, we need to check if the two involved readers are close enough in the deployment. If they are not close enough for the object to move from one to the other during the time gap or for it to be seen simultaneously by the two readers, the two tracking records are dirty as they tell wrong information with respect to the reality. In order to reduce such spatial ambiguity, it is necessary to know the distances between readers. However, such information is not captured in the deployment graph [11] as shown in Figure 3.

Motivated as such, we propose a distance-aware deployment graph in this section. The goal of such a graph is to enable deriving the minimum travel time from one reader to another. The basic idea is to model the readers as graph vertices and capture such minimum travel times in the corresponding graph edges. For the sake of flexibility, we do not use the travel time directly as the graph weights. Instead, for each edge connecting two readers r_i and r_j, we store the minimum indoor walking distance between them and the maximum walking speed allowed by the physical conditions.

Formally, the distance-aware deployment graph is a weighted graph $G_{dd} = (V, E, \mathcal{L}_V, \mathcal{L}_E)$, where

1) V is a set of vertices. Each vertex $v_i \in V$ represents a deployed positioning device r_i.
2) E is the set of edges, where $E = \{(r_i, r_j) \mid r_i, r_j \in V \wedge r_i \neq r_j \wedge D2V(r_i) \cap D2V(r_j) \neq \emptyset\}$.
3) $\mathcal{L}_V : V \rightarrow \mathcal{R}$ assigns to a vertex v_i the minimum dwell time that an object o should spend in device r_i's detection range such that o is detected by device r_i.
4) $\mathcal{L}_E : E \rightarrow \mathcal{R} \times \mathcal{R}$ assigns to a edge (r_i, r_j) the minimum indoor walking distance between two devices r_i and r_j and the maximum speed with which an object can move between them. Specifically, $\mathcal{L}_E((r_i, r_j)) = (d_{i,j}, S_{i,j})$.

The distance-aware deployment graph corresponding to floor plan in Figure 2 is shown in Figure 7. Our specific design on the weights above are justified by practical needs. Different RFID readers (and other positioning devices) usually imply different minimum dwell times for detection. Even for one particular reader, its minimum dwell time may be changed

due to the tuning of its physical parameters (e.g., sampling frequency). Therefore, our design of individual graph vertex weights can support such differences and possible changes.

On the other hand, the travel time between two readers does not solely depend on the distance between them. For example, the moving speed of the conveyor belt system in an airport is tunable, in order to cope with different baggage traffic loads. As a result, the minimum travel time between two readers monitoring the belt is not merely determined by the distance but also by the current moving speed of the belt. Furthermore, in a large conveyor belt system or multiple belt systems, the moving speed is not always the same among different parts. Therefore, our design of both distance and speed weights on each edge is necessary to support such needs in reality.

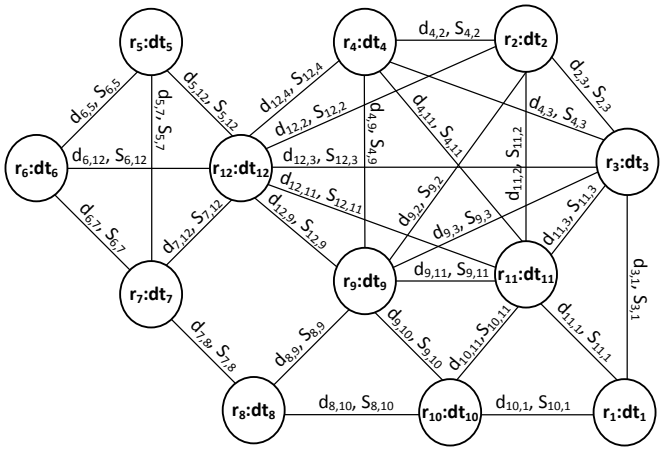


Fig. 7. Example of distance-aware deployment graph

B. Distance-Aware Deployment Graph Construction

The distance-aware graph is constructed by Algorithm 2. It explicitly takes the set of devices and their weights as input. For each pair of devices r_i and r_j (lines 2–3), the indoor shortest path P is found if the edge (r_i, r_j) is not in the graph yet (lines 4–6). If P only contains devices r_i and r_j, a new edge is created with the corresponding weights (line 7–8). Otherwise, each pair of consecutive devices on P are processed likewise (line 10–14).

Algorithm 2 DistanceGraphConstruction(Devices D, Device weights W)

1: $G_{dd}(V, E, \mathcal{L}_V, \mathcal{L}_E) \leftarrow (D, \emptyset, W, \emptyset)$
2: **for** each device $r_i \in D$ **do**
3: **for** each device $r_j \in D$ and $r_j \neq r_i$ **do**
4: **if** $(r_i, r_j) \in G_{dd}.E$ **then**
5: **continue**
6: find the indoor shortest path P from r_i to r_j
7: **if** there is no other device on P **then**
8: add edge (r_i, r_j) to $G_{dd}.E$ with the weights between them
9: **else**
10: **for** each pair of consecutive devices r_k and r_l on P **do**
11: **if** $(r_k, r_l) \in G_{dd}.E$ **then**
12: **continue**
13: **else**
14: add edge (r_k, r_l) to $G_{dd}.E$ with the weights between them
15: **return** G_{dd}

Inside Algorithm 2, the indoor shortest pathes are computed according to the algorithms proposed elsewhere [15]. For the sake of simplicity, the input for those algorithms are implicitly passed to Algorithm 2.

C. Spatial Cleansing Algorithm

We use the information captured by the distance-aware deployment graph (G_{dd}) to conduct the spatial cleansing for the indoor RFID tracking data. To identify and reduce the possible spatial ambiguity involving two RFID readers r_s and r_t, we first compute the minimum traveling time $(min\_tt(r_s, r_t))$ that a moving object needs to reach from r_s to r_t. Specifically, we apply the Dijkstra's algorithm to graph (G_{dd}), expanding the search from r_s until r_t is reached. In the process, we also take into account the minimum dwell time of a device r_i, which is captured by the corresponding vertex v_i's weight $G_{dd}.\mathcal{L}_V(v_i)$, in prioritizing the visiting order of unvisited vertices (devices). Due to the page limit, we omit the low level details of $min\_tt(r_s, r_t)$ computation.

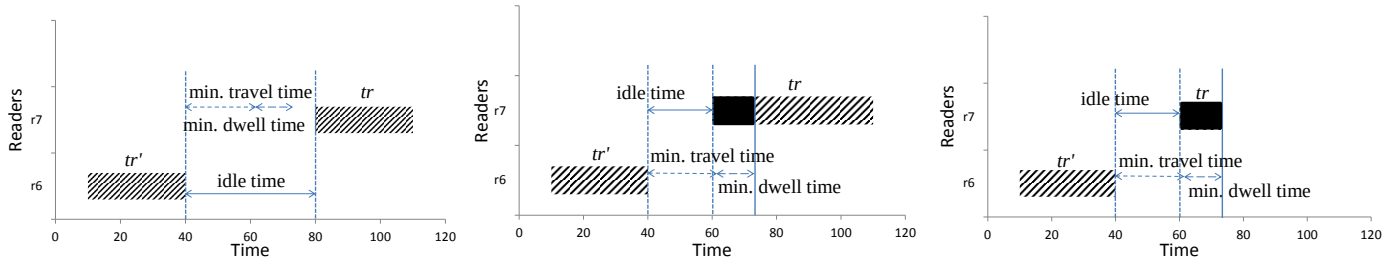
The spatial cleansing procedure is shown in Algorithm 3. It takes the aggregate tracking table (ATT) and the distance-aware deployment graph G_{dd} as input. For each tracking record tr in ATT, the spatial cleansing works as follows. We first check its dwell time[4] (line 2).

Algorithm 3 SpatialCleansing(Aggregate tracking table ATT, Distance-aware deployment graph G_{dd})

1: **for** each tr in ATT **do**
2: $mdt \leftarrow G_{dd}.\mathcal{L}_V(tr.deviceID)$
3: $tr' \leftarrow$ the previous record in ATT such that $(tr.objectID = tr'.objectID) \wedge (tr.deviceID \neq tr'.deviceID)$
4: **if** $tr' =$ null **then**
5: **continue**
6: $mtt \leftarrow min\_tt(tr'.deviceID, tr.deviceID)$
7: **if** $(tr.t_s - tr'.t_e < mtt + mdt)$ **then**
8: $tr.t_s \leftarrow tr'.t_e + mdt + mtt$
9: **if** $(tr.t_e - tr.t_s \leq 0)$ **then**
10: delete tr from ATT

Next, we get tr's previous tracking record tr' from ATT that involves the same object and device as tr (line 3). We assume that tr' is cleaned since it appears before tr', and subsequently clean tr with respect to tr' (lines 6–10). Specifically, we get the minimum traveling time from $tr'.deviceID$ to $tr.deviceID$ (line 6) according to the procedure described above. If the idle time between tr' and tr, i.e., $tr.t_s - tr'.t_e$, is too short compared to the minimum traveling time plus the minimum dwell time for $tr.deviceID$ (line 7 tr appears too early as a tracking record. Therefore, we truncated $tr.t_s$ accordingly (line 8) to make sure the idle time is sufficient with respect to the spatiotemporal constraint. Further, we delete tr from ATT if its updated dwell time is too short (lines 9–10).

[4]Given a tracking record tr, its dwell time is $tr.t_e - tr.t_s$.



(a) Spatial Cleansing Case 1 (b) Spatial Cleansing Case 2 (c) Spatial Cleansing Case 3

Fig. 8. Spatial cleansing cases

We illustrate spatial cleansing cases in Figure 8. The case shown in Figure 8(a) is a clean case where the idle time between two consecutive tracking records is sufficiently long, i.e., longer than the minimum traveling time between devices r_6 and r_7. We need to do nothing for such a case in the spatial cleansing. The case shown in Figure 8(b) illustrates that we need to truncate $tr.[t_s, te]$, cutting tr's part shown in black, because the idle time between tr' and tr is too short, i.e., shorter than the minimum traveling time between devices r_6 and r_7. The case shown in Figure 8(c) takes place as a subcase of the previous case. Here, tr is deleted after the spatial cleansing because its remaining dwell time is too short.

VI. Experimental Studies

In this section, we conduct experiments to evaluate our devised indoor RFID data cleansing techniques. Section VI-A describes the experimental settings. Section VI-B presents the experimental results.

All the experiments were implemented in C++. They were run on a 64-bit Windows 7 enabled PC with 2.8GHz core i7 processor and 7.89GB main memory.

A. Experimental Setup

We describe the performance metrics used in our evaluation, followed by the descriptions of the datasets used.

1) Metrics: The performance of our proposals is evaluated by two metrics: efficiency and effectiveness. The efficiency is measured by counting the clock time.

The effectiveness is measured by the *data reduction ratio*, defined as $\frac{\text{\# of records before cleansing}}{\text{\# of records after cleansing}}$. Specifically, the data reduction ratio for temporal cleansing is $\frac{|RTT|-|ATT|}{|RTT|}$, where RTT and ATT are respectively input and output of the temporal cleansing (Algorithm 1). The data reduction ratio for spatial cleansing is $\frac{|ATT|-|ATT'|}{|ATT|}$, where ATT' refers to the spatial cleansing method (Algorithm 3)'s output. We also test the trends of the effectiveness by varying the thresholds of both algorithms.

We further measure the effectiveness of our spatial cleansing algorithm by counting the number of trajectories free of spatial ambiguity before and after cleansing.

2) Datasets: We used both synthetic and real data in our experimental studies. The parameter settings are summarized in Table V. The default values are bolded.

TABLE V
Experiment Parameters

Parameters	Settings
Detection range	1m, **3m**, 5m
Number of objects	1000, 2000, **5000**, 10000
Number of floors	1, **2**, 3
Number of doors	100, **200**, 300
Threshold (τ)	5, 10, **15**, 20. 25, 30

Synthetic Data. For simplicity, we regard hallways and staircases as rooms, and staircase entrances as doors. We adopt a floor plan with 85 rooms, which are connected by 100 doors. We vary the number of floors, and also the number of rooms and doors. Each pair of adjacent floors are connected by 2 staircases.

By default, there are 5000 RFID tagged objects moving inside a 2-floor building. The movement of an object follows the random waypoint model [12] with a constant speed of 1.1m/s . More specifically, an object in a room can move inside the room, or move to another room that is chosen at random. On the way to the destination, an object can be detected by one to ten RFID readers. The RFID readers are deployed at the doors that connect different rooms. The detection range of each reader is randomly chosen from 1m-5m.

Real Data. We use the real data set collected from Aalborg Airport that operates with an automatic RFID-based baggage handling system. As shown in Figure 9, the baggage handling system features a number of specific semantic locations (e.g., check-in desks, sorter) and RFID readers. A total of 5 RFID readers are deployed in different semantic locations.

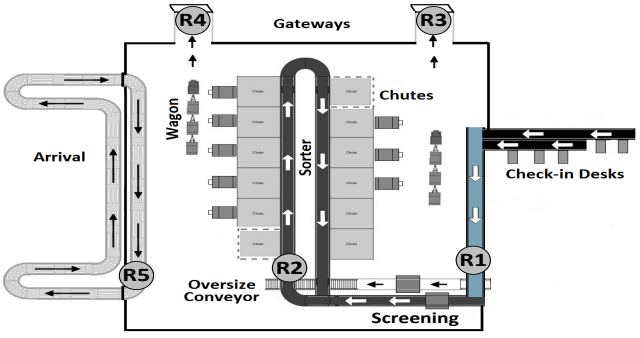


Fig. 9. Baggage hall at Aalborg Airport

During the check-in phase, each bag is attached with a

passive RFID tag which is to be detected by the deployed RFID readers. From check-in desks (CD), bags pass the screening area through conveyor belts. After a successful security screening, the bags enter the main sorting area (SO) where the sorting system ensures that the bag is pushed into a designated chute. Note the bags move in a constrained manner on different conveyors belts before reaching into chutes. The length and the speed of each conveyor are different. Such constraints are captured in the proposed distance-aware deployment graph, and subsequently utilized in spatial cleansing.

The bags in chutes are then loaded into wagons by the baggage handling staff before they are transported to a designated aircraft through one of the gateways (GW-1 or GW-2). On the other hand, arrival bags are carried by wagons from an aircraft to the arrival hall (AR) through gateway GW-1 where baggage handling staff unloads the bags from wagons on to the arrival conveyor belt.

We have continuously recorded the data for two consecutive months. Then, we collect more than half a million raw reading records for about 20000 RFID tagged bags. The real data used for our experiments were obtained through the system installed by Lyngsoe Systems.

B. Experimental Results

We start with investigating the effect of the formula (equation 1) for setting threshold. We fixed the default detection range as 3m and compared the effectiveness of temporal cleansing (data reduction ratio) using different threshold values. The result is shown in Figure 10. The data reduction ratio seems to flatten at very small and very large threshold values. With a larger threshold, more raw readings are aggregated together and with very small thresholds not many raw readings are aggregated together. The size of threshold is bounded at the low end by the sampling frequency and at the higher end by the detection range of the reader. The results show how application parameters can be effectively used to set the threshold.

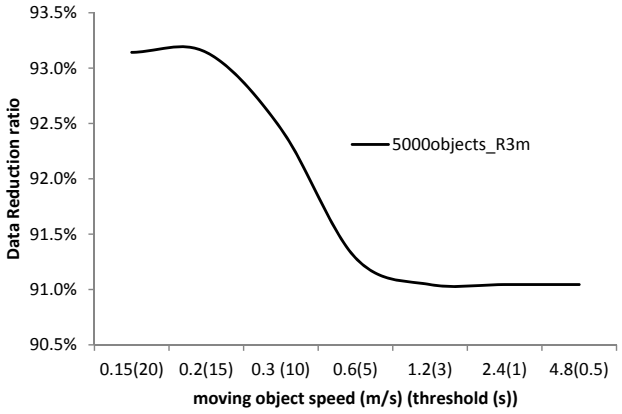


Fig. 10. Effectiveness of the threshold formula (Eq. 1)

Sections VI-B1 and VI-B2 report the spatiotemporal cleansing results on synthetic and real data, respectively. Section VI-B3 briefly shows the experimental results on the distance-aware deployment graph construction.

1) Spatiotemporal Cleansing Results on Synthetic Data: The results on the temporal cleansing effectiveness are reported in Figure 11.

By transferring raw readings into aggregated tracking records, a significantly large portion of redundant records can be eliminated, as shown in Figure 11(a). Specially, for the dataset having 10K objects, the reduction ratio is as high as over 90% in all tested cases. Also, the data reduction ratio increases while enlarging the value of the threshold. With a larger threshold, the chance is higher for raw readings being aggregated into a single tracking record. After the threshold exceeds some value, say 15, the increasing trend flattens. It implies that most raw readings could be aggregated by using a reasonably large threshold. Meanwhile, the ratios slightly vary for different numbers of objects, since the trajectories for different objects are generated independently.

The aggregate tracking table returned by the temporal cleansing is passed to Algorithm 3 to do the spatial cleansing. We also test the data reduction ratio for spatial cleansing. According the results reported in Figure 11(b), the data could be further reduced by removing spatial ambiguities. Over 10% of records with spatial ambiguities can be reduced for the default dataset. Again, the data reduction ratio increases slightly after the threshold reaches 15. Larger threshold values correspond to stricter spatial distance constrains between readers, and thus more records can be disqualified.

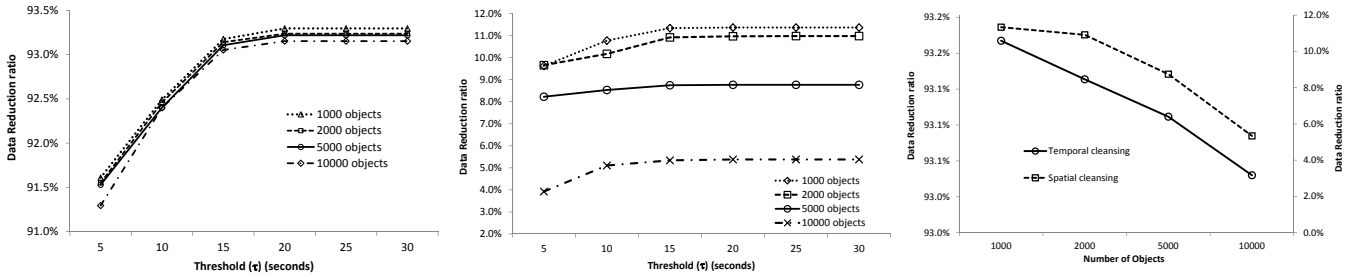
We also test the spatiotemporal cleansing effectiveness with respect to the number of objects and report the results in Figure 11(c). The effectiveness of temporal cleansing does change much with increasing number of objects, as can be seen from the left Y-axis. More than 90% of raw data is reduced after temporal cleansing that loses no information. Regarding spatial cleansing effectiveness, slight decrease is noticed when number of the object increases, see from the right Y-axis. The overall data size is further reduced by 6-10%. These findings suggest that proposed spatiotemporal cleansing is effective.

The results on efficiency are reported in Figure 12. Referring to the results shown in Figure 12(a), the temporal cleansing algorithm is efficient, e.g., it takes less than 15 seconds for handling 10000 objects (about 1M records). It also scales well with respect to the varied thresholds.

The results about spatial cleansing efficiency are reported in Figure 12(b). The algorithm achieves better efficiency at a higher threshold. The reason is that fewer tracking records are generated by temporal cleansing using larger thresholds.

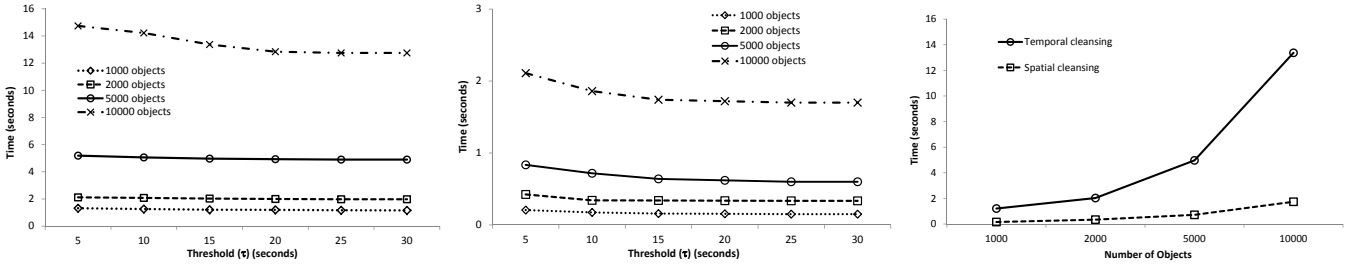
Figure 12(c) reports the results on the efficiency with respect to the number of objects for both spatial and temporal cleansing algorithms. Both algorithms take less than 15 seconds for the largest testing dataset. The processing time of temporal cleansing increases quadratically, which is consistent with Algorithm 1. For spatial cleansing, the running time increases linearly. It is more efficient because its complexity is lower and it has a much smaller input, as the size of an aggregate tracking table is much smaller than that of a raw reading table.

To further study the cleansing effectiveness, we count the trajectories with/without spatial ambiguities before and after



(a) Temporal cleansing effectiveness (b) Spatial cleansing effectiveness (c) Spatiotemporal cleansing effectiveness

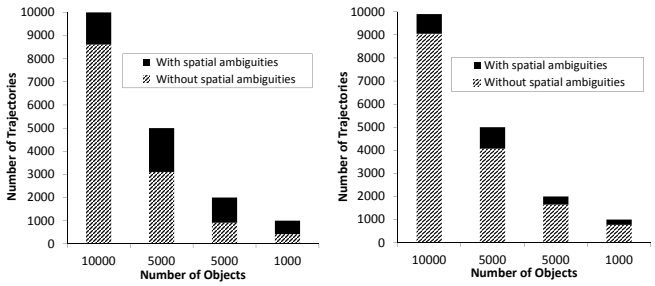
Fig. 11. Cleansing effectiveness on synthetic data



(a) Temporal cleansing efficiency (b) Spatial cleansing efficiency (c) Spatiotemporal cleansing efficiency

Fig. 12. Cleansing efficiency on synthetic data

the spatial cleansing. The results are shown in Figure 13. It can be seen that the portion of trajectories free of spatial ambiguity is increased by 40% in the worst case, and by up to 85% in the best case. These results show that our spatial cleansing technique is effective in ambiguity reduction.



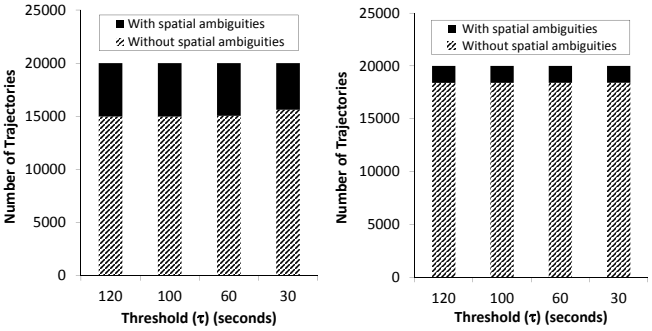
(a) Before Spatial Cleansing (b) After Spatial Cleansing

Fig. 13. Valid path evaluation on synthetic data

2) Spatiotemporal Cleansing Results on Real Data: We also use the real dataset from Aalborg Airport to test the effectiveness of both temporal and spatial cleansing algorithms. The results are reported in Figure 15. For both cleansing algorithms, the effectiveness increases with the increase of thresholds. A larger threshold tends to aggregate more raw readings in temporal cleansing and thus exclude more tracking records to pass to the spatial cleansing. After the threshold reaches 25, data reduction ratios of both algorithms do not show significant changes. The "turning point" of the threshold, which is determined by the objects' moving patterns, can be viewed as a limit of the increasing trend.

We measure the efficiency of both algorithms over the real dataset and report the results in Figure 16. The processing time of both algorithms decreases with the increase of the threshold. The trend is opposite to the observations in Figure 15, since the more data to be cleansed, the more efforts have to be paid.

We also test the effectiveness of spatial temporal cleansing on real data by counting the trajectories with and without spatial ambiguities. Figure 14 reports the relevant results.



(a) Before Spatial Cleansing (b) After Spatial Cleansing

Fig. 14. Valid path evaluation on real data

Figure 14(a) shows that before cleansing more than 25% trajectories have spatial ambiguities. In contrast, Figure 14(b) shows that after our spatial cleansing only around 8% trajectories still have spatial ambiguities. This again demonstrates the effectiveness of our spatial cleansing technique.

3) Distance-Aware Deployment Graph Construction Results: Finally, we evaluate the efficiency of the distance-aware deployment graph construction (Algorithm 2). We use three buildings with 100, 200, and 300 doors. Since we deploy RFID readers at doors, we define the *coverage ratio* as $\frac{\#\text{ of readers}}{\#\text{ of doors}}$. By varying the coverage ratio of a given building, we tune the number of vertices therefore the size of the deployment graph. The relevant results are shown in Figure 17.

In Figure 17, each reported value is the average of 50 runs of Algorithm 2. The construction time increases super linearly with the increase of the coverage ratio. Also, the construction efficiency scales well with the increase of the number of doors. In all tested cases, the construction time is below 0.6 second. We can conclude that by using Algorithm 2, the deployment graph can be efficiently constructed.

We also study the graph construction time cost for real data. To construct a distance-aware deployment graph for a real reader deployment in Figure 9, it takes less than 50 milliseconds, which is very efficient. Due to the space limit, we omit the further details.

VII. Conclusion and Future Work

In this paper we study data cleansing for indoor RFID tracking data. We focus on two relevant tasks: temporal redundancy elimination and spatial ambiguity reduction. For the former, we design a temporal cleansing algorithm to aggregate raw RFID readings temporally such that the data size is compressed without information loss. For the latter, we design a spatial cleansing technique. We propose a distance-aware deployment graph to capture the spatiotemporal constraints implied by the deployment of RFID readers as well as the indoor topology. By exploiting the spatiotemporal constraints captured in the graph, we design a spatial cleansing algorithm to reduce the spatial ambiguity in RFID data. We conduct extensive experimental studies using both synthetic data and real data. The results demonstrate that the proposed techniques are effective and efficient in fulfilling the data cleansing tasks for indoor RFID data. The techniques proposed in this paper also apply to indoor tracking data obtained by other symbolic positioning technologies, e.g., Bluetooth.

There are several directions for future work on cleansing indoor tracking data. First, it is interesting to make use of the Radio Signal Strength Information (RSSI) if such information is available in the raw data. RSSI may indicate the distance between an object and the relevant positioning device(s), which can be exploited to enhance the data cleansing.

Second, it is possible to conduct individualized data cleansing for different objects if their individual features (e.g., moving speed and pattern) are known. In this paper, we use the maximum speed of all objects in the distance-aware deployment graph. To support individualized cleansing, the graph and the device-to-device minimum traveling time computation should be adapted accordingly for individual objects.

Third, it is relevant to conduct queries and/or mining tasks on cleansed indoor tracking data in order to obtain more interesting and more informative results.

Acknowledgments

This work is partially supported by the NILTEK and Daisy Innovation projects funded by European Regional Development Fund. The work is also supported in part by Lyngsoe Systems A/S and Aalborg Airport (AAL).

References

[1] Y. Bai, F. Wang, and P. Liu. Efficiently filtering rfid data streams. In *CleanDB*, 2006.

[2] C. Ban, B. Hong, and D. Kim. Time parameterized interval r-tree for tracing tags in rfid systems. In *DEXA'05*, pages 503–513, 2005.

[3] K. E. Bite. Improving on passenger and baggage processes at airports with rfid, 2010.

[4] Z. Cao, C. Sutton, Y. Diao, and P. J. Shenoy. Distributed inference and query processing for rfid tracking and monitoring. *PVLDB*, 4(5):326–337, 2011.

[5] P. Fuhrer and D. Guinard. Building a smart hospital using rfid technologies. In *ECEH*, pages 131–142, 2006.

[6] H. Gonzalez, J. Han, and X. Li. Flowcube: Constructuing rfid flowcubes for multi-dimensional analysis of commodity flows. In *VLDB*, pages 834–845, 2006.

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

[8] P. Harrop and Holland. Rfid for postal and courier services, 2005.

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

[10] S. R. Jeffery, M. N. Garofalakis, and M. J. Franklin. Adaptive cleaning for rfid data streams. In *VLDB*, pages 163–174, 2006.

[11] C. S. Jensen, H. Lu, and B. Yang. Graph model based indoor tracking. In *Mobile Data Management*, pages 122–131, 2009.

[12] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[13] C.-H. Lee and C.-W. Chung. Efficient storage scheme and query processing for supply chain management using rfid. In *SIGMOD Conference*, pages 291–302, 2008.

[14] G. Liao, J. Li, L. Chen, and C. Wan. Kleap: an efficient cleaning method to remove cross-reads in rfid streams. In *CIKM*, pages 2209–2212, 2011.

[15] H. Lu, X. Cao, and C. S. Jensen. A foundation for efficient indoor distance-aware query processing. In *ICDE*, pages 438–449, 2012.

[16] H. Mahdin and J. H. Abawajy. An approach to filtering duplicate rfid data streams. In *FGIT-UNESST*, pages 125–133, 2010.

[17] O. Mylyy. Rfid data management, aggregation and filtering, 2007.

[18] J. Rao, S. Doraiswamy, H. Thakkar, and L. S. Colby. A deferred cleansing method for rfid data analytics. In *VLDB*, pages 175–186. VLDB Endowment, 2006.

[19] D. Roozbeh, O. Maria, and L. Xue. Rfid data management: Challenges and opportunities. In *IEEE International Conference on RFID 2007*, pages 175–182, 2007.

[20] S. Tinggaard and R. L. Wejdling. A data warehouse solution for flow analysis utilising sequential pattern mining. *Master Thesis, Aalborg University*, June 2009.

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

[22] R. Want. *RFID Explained: A Primer on Radio Frequency Identification Technologies*. Morgan and Claypool, 2006.

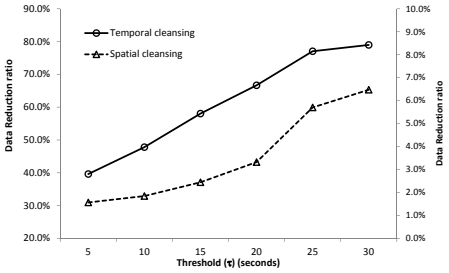


Fig. 15. Cleansing effectiveness on real data

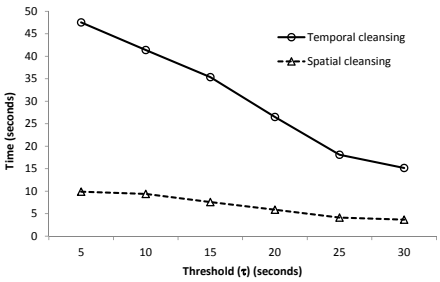


Fig. 16. Cleansing efficiency on real data

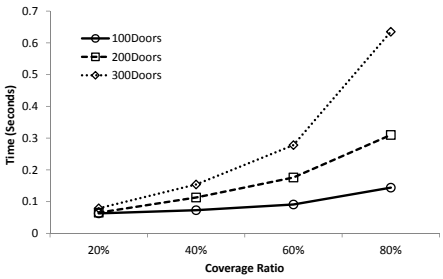


Fig. 17. Distance-aware graph construction