Cleansing Indoor RFID Tracking Data

Baba, Asif Iqbal

DOI (link to publication from Publisher):
10.5278/vbn.phd.engsci.00160

Publication date:
2016

Document Version
Publisher's PDF, also known as Version of record

CLEANSING INDOOR RFID TRACKING DATA

BY
ASIF IQBAL BABA

DISSERTATION SUBMITTED 2016

AALBORG UNIVERSITY
DENMARK

Cleansing Indoor RFID Tracking Data

Ph.D. Dissertation

Asif Iqbal Baba

Aalborg University
Department of Computer Science
Dissertation submitted June, 2016

Dissertation submitted: August 2016

PhD supervisor: Associate Prof. Hua Lu
Aalborg University, Aalborg, Denmark

Assistant PhD supervisor: Prof. Torben Bach Pedersen
Aalborg University, Aalborg, Denmark

PhD committee: Associate Professor Bin Yang (chairman)
Aalborg University, Denmark
Professor Mario Nascimento
University of Alberta, Canada
Professor Yan Huang
University of North Texas, USA

PhD Series: Faculty of Engineering and Science, Aalborg University

ISSN (online): 2246-1248
ISBN (online): 978-87-7112-737-9

Published by:
Aalborg University Press
Skjernvej 4A, 2nd floor
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Rosendahls, 2016

Abstract

Radio Frequency Identification (RFID) is a most flexible automatic identification technology that uses radio frequencies to identify and track objects. RFID is taking over the traditional barcode systems and providing the ability to identify individual objects uniquely and without line-of-sight. An RFID system has two main components: readers and tags. RFID is a proximity-based technology in that a reader detects a tagged object only when the object is present within the reader's detection range. The commonly used passive RFID tags do not require battery to store information and exchange data with readers. Due to the flexibility and small size of RFID tags, a vast number of applications are using RFID tags for identifying and tracking objects. Such RFID applications generate massive amounts of raw RFID data. However, the generated data is not reliable enough for high-level analysis and processing. The raw RFID data contains errors such as redundant readings, cross readings and missing readings. These errors are mainly caused by the hardware limits, inconsistent radio signals, and the dynamic indoor environments. Therefore, it is paramount to make the raw RFID data reliable, which is aim of this PhD thesis. The PhD thesis is done as a part of projects NILTEK and BagTrack (http://daisy.aau.dk/bagtrack), which aim to develop an IT solution to improve aviation baggage handling quality. The projects introduce the auto-identification tagging for check-in bags at airports using RFID tags. The projects have a number of data management research challenges, one of which is cleansing indoor RFID data.

The PhD thesis provides several novel data cleansing solutions applicable for indoor RFID data. First, the thesis provides a cleansing solution to reduce the redundant and cross readings in raw RFID data. The solution is based on a graph model that captures the information about indoor constraints and the deployed reader characteristics. Second, the graph model is further enhanced with the information such as transition probabilities from reader to reader such that it can be used to handle missing readings in raw RFID data. Third, the thesis presents a learning based solution that manages to reduce cross readings and recover missing readings in the raw RFID data. The approach introduces a Hidden Markov Model (HMM) models the errors

present in raw RFID data and three different state space designs are proposed that can be used to learn the probabilistic parameters from indoor raw RFID data. After computing the most probable hidden state sequence, the learned model is able to determine the most probable observation sequence and use it as the cleansed data. Fourth, the thesis presents an approach to map raw RFID data to most probable indoor paths. The approach uses regular expressions to model RFID data and possible indoor paths, and constructs an automaton to capture all possible indoor paths in regular expressions. A probabilistic error model is designed to represent the errors between raw RFID data and semantic symbols in indoor paths. The automaton is used to find the most probable indoor paths according to the error model. All the proposed solutions have been validated through experiments using real airport baggage RFID data and synthetic datasets. The experimental results show that our proposals are efficient and effective. The techniques proposed in the PhD thesis can be employed in real systems to improve the aviation baggage handling quality; they also make advancements in the research area of spatio-temporal data management.

Resumé

Radio Frequency Identification (RFID) er en fleksibel auto identifikationsteknologi, som anvender radiofrekvenser til at identificere og spore objekter. RFID er ved at erstatte de traditionelle streg-kode systemer, og giver muligheden for at identificere enkelte objekter entydigt og uden line-of-sight. Et RFID system har to hovedelementer: læsere og tags. RFID er en afstandsbaseret teknologi i den forstand, at en læser kun detekterer et objekt hvis det er indenfor læserens rækkevide. De ofte anvendte passive RFID-tags, kræver ikke batteri hverken til at lagre information, eller for at udveksle data med læseren. Grundet fleksibiliteten og den relativ lille størrelse af RFID-tags, anvendes de i en lang række forskellige applikationer til at identificere og spore objekter. Sådanne RFID applikationer genere store mængder af rå RFID data, men det generede data er ikke pålideligt nok til høj-niveau analyse og processering. Den rå RFID data indeholder fejl, såsom redundante aflæsninger, kryds aflæsninger, og manglende aflæsninger. Disse fejl er primært forårsaget af hardware begrænsninger, inkonsistente radio signaler, og de dynamiske indendørs miljøer. Derfor er det afgørende at pålideliggøre den rå RFID data, hvilket er formålet med denne PhD afhandling. Denne PhD afhandling udgøre en del af projekterne NILTEK og BagTrack (http://daisy.aau.dk/bagtrack), der har til formål at udvikle en IT løsning til at forbedre kvaliteten af bagagehåndtering i lufthavne. Disse projekter udvikler et system til auto-identificerings tagging af check-in bagage i lufthavne, ved brug af RFID tags. Disse projekter har en række forskningsmæssige udfordringer indenfor datastyring, bl.a. udrensning af indendørs RFID data.

PhD afhandlingen introducere flere nye data udrensningsmetoder, med anvendelse til indendørs RFID data. Først introduceres en udrensningsmetode til at reducere antallet af redundante og kryds-aflæsninger i det rå RFID data. Løsningen er baseret på en grafisk model som udvinder informationer vedrørende indendørs begrænsninger og karakteristikken af læseren. For det andet udvides den grafiske model med yderligere informationer, såsom overgangs-sandsynligheder mellem læsererne, således at modellen kan håndtere manglende læsninger i rå RFID data. For det tredje præsenteres en læringsbaseret løsning, som er i stand til at reducere antallet af kryds-

aflæsninger og gendanne manglende aflæsninger i den rå RFID data. Denne metode gør brug af en hidden Markov model (HMM), til modellere usikkerheden i indendørs RFID data og et antal tilstandsrum designs som kan anvendes til at lære parametrene ud fra den rå RFID data. Efter at have udregnet den mest sandsynlige sekvens af hidden states, anvendes den indlærte model til at bestemme den mest sandsynlige observationssekvens, der anvendes som en repræsentation for det rensede data. For det fjerde præsenterer afhandlingen en metode til at transformere den rå RFID, til de mest sandsynlige indendørs stier. Metoden anvender regulære udtryk til at modellere RFID data og mulige indendørs stier, og konstruere en automat til at udtrykke alle mulige indendørs stier i regulære udtryk. En probabilistisk fejlmodel er designet til at repræsentere fejl mellem RFID rådata og semantiske symboler i indendørs stier. Automaten bruges til at finde de mest sandsynlige indendørs stier i henhold til fejl-modellen. Alle de foreslåede løsninger er blevet valideret gennem eksperimenter ved brug af både reelle og syntetiske datasæt. Resultaterne af disse eksperimenter viser, at vores forslag er effektive og virksomme. De teknikker der er foreslået i afhandlingen, kan anvendes i rigtige systemer til at forbedre kvaliteten af bagagehåndtering indenfor luftfart. De foreslåede teknikker repræsentere ligeledes fremskridt i forskningsområdet indenfor spatiotemporal datatyring.

Acknowledgments

First and foremost I praise and thank Allah, the Almighty for showering his blessings on me and giving me perseverance to continue, and for leading me to finish my PhD thesis successfully. This journey would not have been complete without the support of my supervisor Hua Lu. He has been guiding force through good times and bad. He has been patient with me though out the PhD period, I keep missing deadlines, he keep setting new ones for me. Torben Bach Pedersen, my co-supervisor has been an inspiration and a through professional to learn from. His ability to provide concrete ideas and detailed comments were unmatched. During my PhD I also worked with Associate professor Manfred Jaeger, very detailed and helpful. Learned lot from him, he spent hours teaching me machine learning stuff. I am extremely thankful to him for giving me equal opportunity and time as his own students. I went to Auburn University, USA for my abroad study, where I worked with Associate professor Wei-Shinn (Jeff) Ku. I want to thank Jeff for accepting me as visiting research student in his group and giving me opportunity guide some of his master students.

I would like to thank all the people from Daisy who were there to help. My appreciation and gratitude to the supporting staff, particularly to two H's, Helle Scroll and Helle Westmark who were always there to help and guide. Would like to thank Lon Nguyen, an IT support guy, always provide me an extra laptop for my experiments with special hard disk.

I also would like to thank all my friends and colleagues in AAU (Aalborg University, Denmark) and AU (Auburn University, USA). Special thanks to Syed Umair, who every time give me ride to Friday prayer and helped me translating my thesis abstract.

My Special thank you to my family back in Kashmir (Dady, Baji, Aafi and others) who tirelessly pray for my well-being and for my success. My brother Jav, who is my go to person whenever I am in trouble and I thank him for making it worse every time. My family in Denmark, (wife Sabeeha, two sons Imaad and Muaad, family friends Tanvir, Nadia and Manha). I cannot thank enough my wife for being patient, caring and understanding.

Contents

Thesis Details

Thesis Title: Cleansing Indoor RFID Tracking Data
Ph.D. Student: Asif Iqbal Baba
Supervisors: Assoc. Prof. Hua Lu, Aalborg University
Co-Supervisor: Prof. Torben Bach Pedersen, Aalborg University

The main body of this thesis consist of the following papers.

[A] Asif Iqbal Baba, Hua Lu, Xike Xie and Torben Bach Pedersen, *Spatiotemporal Data Cleansing for Indoor RFID Tracking Data*, ISBN: 978-0-7695-4973-6, In Proceedings of the 14th IEEE International Conference on Mobile Data Management (MDM 2013), pp. 187–196, 2013.

[B] Asif Iqbal Baba, Hua Lu, Torben Bach Pedersen and Xike Xie, *Handling False Negative in Indoor RFID Data*, ISBN: 978-1-4799-5705-7, In Proceedings of the 15th IEEE International Conference on Mobile Data Management (MDM 2014), pp. 117–126, 2014.

[C] Asif Iqbal Baba, Manfred Jaeger, Hua Lu, Torben Bach Pedersen, Wei-Shinn Ku and Xike Xie, *Learning-Based cleansing for indoor RFID data*, ISBN:978-1-4503-3531-7, In Proceedings of the 2016 International Conference on Management of Data (ACM SIGMOD 2016), pp. 925-936, 2016.

[D] Asif Iqbal Baba, Hua Lu, Wei-Shinn Ku and Torben Bach Pedersen, *From Raw RFID Data to Most Probable Indoor Paths: An Approach Based on Regular Expression Matching*, submitted to ACM SIGSPATIAL GIS 2016.

In addition to the main papers, Paper E has also been included which is shorter version of Paper B.

[E] Asif Iqbal Baba, Hua Lu, Torben Bach Pedersen and Xike Xie, *A Graph Model for False Negative Handling in Indoor RFID Tracking Data*, ISBN: 978-1-4503-2521-9, In Proceedings of the 21st International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2013) , pp. 464–467, 2013.

This thesis has been submitted for assessment in partial fulfillment of the Ph.D. degree at Aalborg University, Denmark. The thesis is based on the submitted or published scientific papers which are listed above. Parts of the papers are used directly or indirectly in the extended summary of the thesis. As part of the assessment, co-author statements have been made available to the assessment committee and are also available at the Faculty. The permission for using the published and accepted articles in the thesis has been obtained from the corresponding publishers with the conditions that they are cited, DOI pointers and/or copyright/credits are placed prominently in the references.

Part I

Thesis Summary

1 Introduction

1.1 Background

The PhD work is accomplished as a part of the projects NILTEK and BagTrack, which aims to build a global IT solution based on RFID technology to improve the baggage handling quality world wide. The projects have several industrial partners which include IATA, SAS airlines, Aalborg Airport, Lyngsoe systems.Further details about the project can be found at this link: http://daisy.aau.dk/bagtrack.

In particular, NILTEK project aims to build to RFID technology based network in selected airports in Scandinavia to measure baggage handling . The baggage handling automation is achieved using the RFID technology in which a selected passenger check-in bag is tagged with an RFID tag. During the baggage handling process the bag goes through several RFID readers deployed at different semantic locations such as check-in desks, sorters, belt loaders etc. Example deployment of RFID readers are shown in Figure 1.

RFID is slowly but surely becoming the preferred choice for object tracking and monitoring systems, particularly in indoor environments like airports, museums, subway stations and so on. RFID technology is a key part of most of the modern applications, such as baggage tracking at airports, animal tracking, health-care and many more. The technology is regarded as a successor of traditional barcode system in retail marketing. One of the key differences between the two is that bar-codes is a line-of-sight technology where as RFID does not require line-of-sight and can detect objects within the range of reader. A typical RFID system have two main components: 1) a transponder which is also known as RFID tag,these tags are generally attached to items to be tracked or monitored. 2) an interrogator commonly known as RFID reader, which can be static or mobile depending on the application. RFID technology uses electromagnetic fields to detect or track tags attached to objects. There are two kinds of tags: passive tags and active tags. Passive tags do not have any energy source and use the energy from the nearby reader while as active tags have their energy source such as batteries and can be detected from hundreds of meters from the reader. However, with

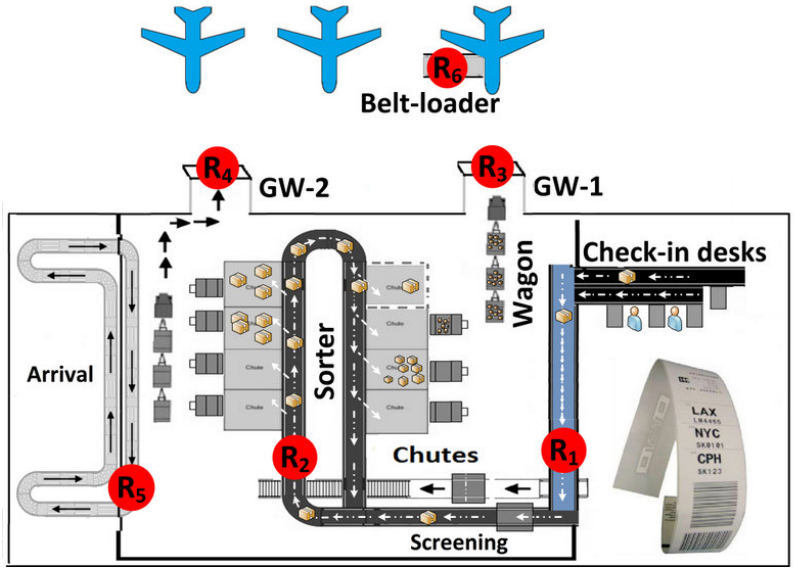


Fig. 1: Aalborg Airport baggage hall

the feasibility of working for longer period of time and cost, passive tags are widely been used by RFID based applications. The tags typically store more than 2 kilobytes of data in the user memory, which is sufficient to store the data about the object it is attached to. The applications like airport baggage handling, postal services etc., use a simple tag which is easy to manufacture and is disposed after its use. A typical numbering scheme used by these simple tags to identify the objects is Electronic Product Code (EPC). A 96 bit string is written into a tag by RFID printer and is generally referred as "license plate". An example of airline baggage tag is shown in Figure 1. The standards to use RFID on baggage in airport industry is controlled by IATA. The specification is called "Recommended Practice 1740C, Radio Frequency Identification (RFID) Specifications for Interline Baggage" [6].

As mentioned earlier, most of the tracking and monitoring applications use passive tags, therefore requires to have an active RFID reader, which transmits the RF signals and receives the reply from tags with the data on it. RFID based systems along with other tracking and monitoring technologies such as Bluetooth, WiFi, etc. generate huge amount of data by tracking the object continuously. RFID based applications are generally composed of RFID tags attached to different objects (e.g., a passenger bag, an item in a retail store, etc.) and the RFID readers which detects the objects within it proximity range and generates a report about object's presence. A typical format of a raw RFID reading is <*deviceID, tagID, t*>, which means an RFID

reader with identity *deviceID* detected an object with identity *tagID* at *t* [1–3]. The data gives easy access to the spatial and temporal information of a moving object. RFID based applications used in many retailer stores and others can generate 1,000 times more data than a conventional bar code systems. One of the leading retailers, Walmart recently started using RFID for item-level tracking are predicted to generate more than 7 terabytes of raw RFID data per day [12].

RFID baggage data cleansing being one of the three main area of research of project NILTEK/Bagtrack project and the focus of this PhD thesis. Others correlated ares of the NILTEK/Bagtrack are, continuous queries on RFID baggage data, and warehousing as well as analyzing huge amounts of RFID baggage data.

1.2 Motivation

Baggage handling being one of the major concerns of the aviation industry, several projects world wide are concentrated to provide some efficient solution to the long lasting problem of baggage mishandling. The passenger baggage are handled at possible three points during the journey: 1) When the passenger checks-in the bag at the departure airport, 2) During the transfers, i.e., changing a flight in the middle of the journey and 3) at the arrivals where the bag is handed back to the passenger (see Figure 2). During the process of baggage handling the bag is handled by different parties responsible such as airports, airlines, baggage handling agencies, etc. It is important to develop the techniques to do the analysis of handling quality of each partner involved in the process and gain insights into the causes of mishandling.

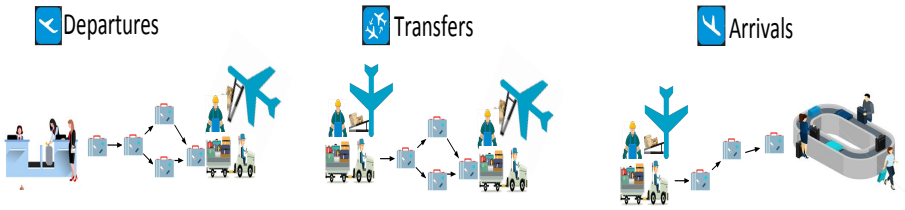


Fig. 2: Baggage handling process

According to latest SITA reports [18], the largest amount of bags are mishandled during the transfers. In 2014 more than 34M bags are mishandled annually, costing the industry over 3 billion USDs per year. Moreover, it takes more than 1.5 days on average for a passenger to receive a mishandled bag, and the delay further frustrates the passenger. The report says that the baggage mishandling has improver drastically over the years. Compared to 18 bags per 1000 in 2007, it came down to 7.3 bags per 1000 in year 2014. The

improvement over the years are largely due to to the introduction of modern technology and baggage system automation and processing as reported by the report. NILTEK/BagTrack project is one of several cases where baggage handling automation and intelligent tagging is introduced.

With the introduction of RFID technology in the baggage tracking process to automate the process, a huge amount of RFID tracking data is generated every day. The raw RFID data generated with such an unprecedented rate need to be handled and used effectively. With the inconsistent nature of RF signals, and the dynamic indoor environments, data generated is bound with the noises and errors. There are three main kind of errors which are found in raw RFID data: namely, *redundant readings* also known as duplicate readings, *false positives* also known as cross readings and *false negatives* also known as missing readings [1–4].

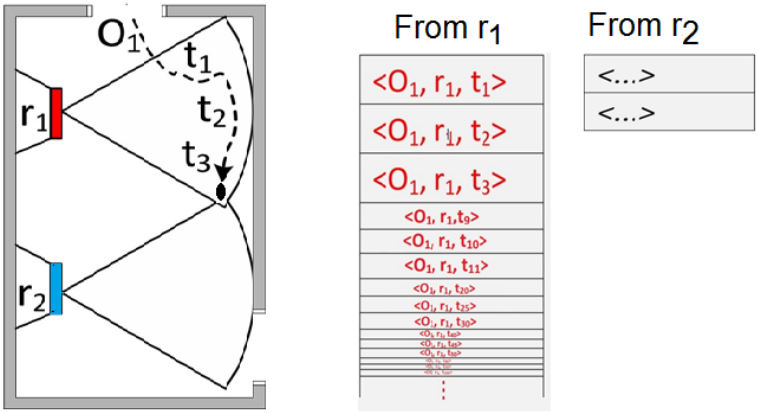


Fig. 3: Redundant readings example

Redundant readings are generated when an object is read by the reader for longer period of time, with a millisecond read rates reader can generate a large amount of duplicate data. As illustrated in Figure 3, object O_1 remained in reader r_1 range for unexpectedly longer period, generating redundant readings.

False positives are created in the data when the object is read by one or more readers unexpectedly, i.e., the object is detected by the readers also which were not intended to read the object. Reasons for false positives can be reflection of signals from surrounding metal items or unexpected re-direction of reader antenna(s). For an illustration, an example in Figure 4 shows the moving path of object O_1 [1]. The moving object enters into Hall-1 where it was first detected by readerr_1 at t_1. At time time point t_3, object O_1 was detected by two readers r_1 and r_2 placed in two different locations (Hall-1 and

Hall-2) with non-overlapping detection ranges. Due to spatial constraints of two halls it is not possible for an object to be present at two locations at the same time. Therefore, giving rise to false positives .

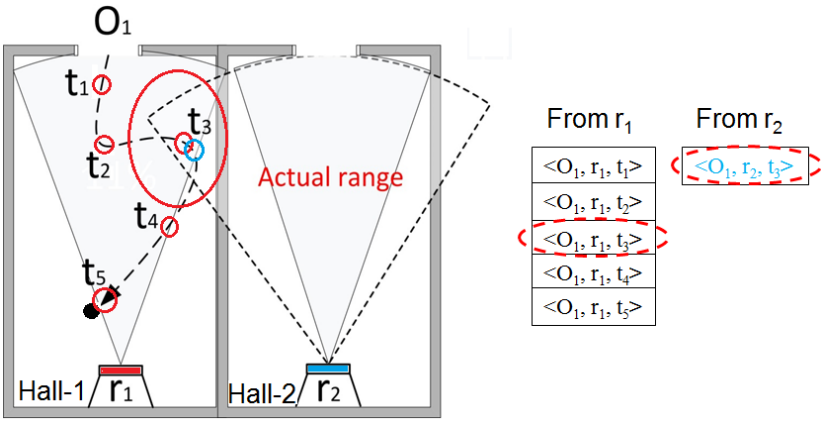


Fig. 4: False positives example [1]

False negatives happens in the data when the intended reader could not detect the object in its proximity range. Therefore, the reader failed to generate any data about the object's presence. An example in Figure 5 shows an object O_1 goes through three readers r_1, r_2 and r_3 [2]. At first object O_1 entered Hall-1 where reader r_1 detected O_1 from t_1 until t_3. After that, O_1 was detected by reader r_3 placed in Hall-2 from t_7 until t_9. However, it is not possible for O_1 to be detected by r_3 unless it passes through reader r_2 placed in Hall-1, because to enter into the Hall-2 O_1 must go through the detection range of r_1 and r_2 in Hall-1. As a result, false negatives are generated in the data.

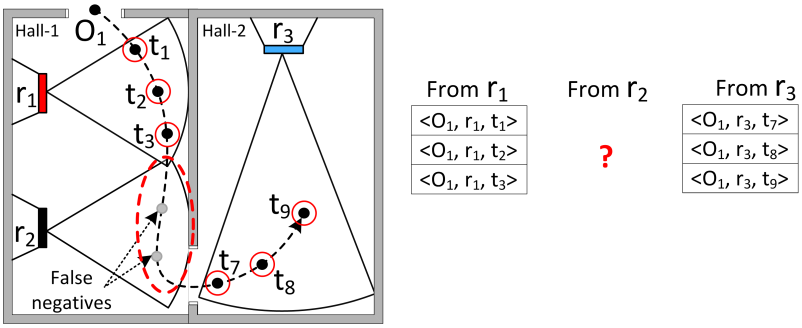


Fig. 5: False negatives example [2]

With the presence of errors in raw RFID data, performing query analysis

and high level business processing will lead to inconsistent business decisions. Therefore, the raw RFID data must be cleansed before any high level query or analysis is done over it.

To improve the quality and reliability of the raw RFID data, several solutions are proposed in existing work. Trotter et al. have classified these solutions into three main categories: Physical solutions, middle-ware solutions and deferred solutions [20]. Physical solutions are related to the solutions which use extra hardware to increase the quality of raw data as described in [20]. For example, multiple RFID tags are attached to the same object or multiple readers are deployed overlapping to each to reduce the chances of missing the object [7, 20]. The middleware solutions include algorithms which handle the data online and correct the data stream before the data is passed to some storage [13, 14]. The deferred solutions handle the offline data stored in the database [8, 17]. There are many approaches already used in RFID data cleansing mainly focused on false negatives and false positives. Most of the ongoing work is focused on middleware base solutions. However, working with continuous data stream is important but more challenging than if the data is first stored in the database.

This thesis proposes several cleansing solutions using different approaches. Different cleansing techniques focus on different kinds of errors present in raw RFID data. The thesis proposes solutions based on graph which captures information about indoor space settings and reader deployment [1, 2], a learning based approach which learns the parameters of proposed HMM model from raw RFID data [3], and a regular expression matching based approach which utilizes the error model devised from the domain knowledge [4]. All the proposed solutions in this thesis have been validated using a real airport baggage RFID data and synthetic raw RFID datasets. The real dataset is collected by the RFID-based baggage handling system used at Aalborg Airport.

2 Symbolic Indoor Tracking

The indoor space are generally divided into different partitions and each of these partitions are symbolized by certain name or number. To describe the symbolic indoor tracking, an example floor plan of a university office building is presented in Figure 6 [3]. The floor plan have several partitions including rooms, halls, corridors and stairs. Each partition is identified with a unique symbolic number [1–3] and may have one or more entrances (doors). As shown in Figure 6, each entrance of the partition is deployed with an RFID readers represented by a solid circles in red color and a dashed circle around each red solid circle indicates the range of each reader.

As mentioned earlier, RFID technology is based on proximity analysis, i.e,

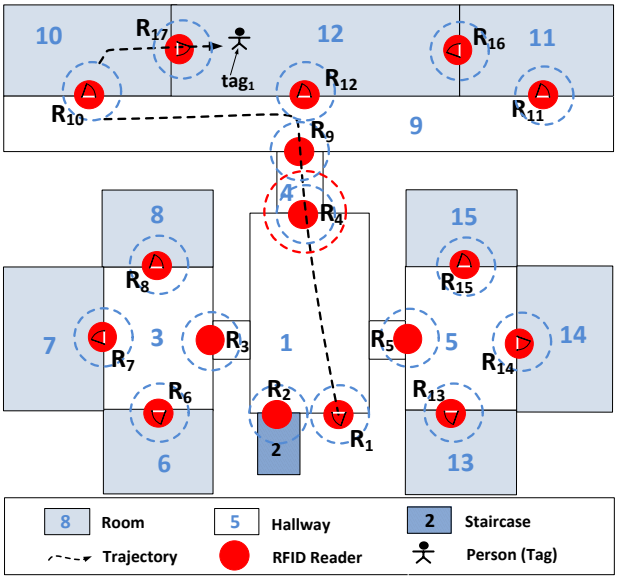


Fig. 6: Symbolic indoor space with deployed RFID readers [3]

object with an RFID tag can only be detected when it present inside the range of an RFID reader. The location where the object is detected is extracted by mapping the reader tuple present in each reading to locations. The deployed RFID reader may have different detection ranges and sampling frequency. The object is continuously detected by the RFID reader within it detection range and based on the sampling frequency of each RFID reader, the number of reports about object's presence are generated.

Typically, the format of each generated raw RFID reading is (*deviceID, objectID, t*), which is tuple containing a reader identifier, object identifier and time. The detection range of any positioning device is usually assumed to be round with a configurable radius. Each reading gives accessibility to both spatial (symbolic location) and temporal (time) information about the object. A massive amount of raw RFID data is generated by the readers deployed in indoor spaces.

For an illustrate, an example taken from [3] showing a movement history of a person attached with an RFID tag tag_1 shown in Figure 6. The raw data generated about tag_1 is presented in Table 1.

At first, a person with tag_1 is detected by reader R_1 at the entrance of the building, generating four readings as shown in Table 1. The four readings give information about same person and reader, therefore giving rise to redundant readings. After leaving reader R_1, tag_1 is detected by several other readers deployed inside the building. At time points t_{13} and t_{14}, a person

deviceID	objectID	t	deviceID	objectID	t	deviceID	objectID	t
R_1	tag_1	t_0	R_4	tag_1	t_{10}	R_9	tag_1	t_{15}
R_1	tag_1	t_1	R_4	tag_1	t_{11}	R_9	tag_1	t_{16}
R_1	tag_1	t_2	R_4	tag_1	t_{12}	R_{17}	tag_1	t_{26}
R_1	tag_1	t_3	R_9	tag_1	t_{13}	R_{17}	tag_1	t_{27}
R_4	tag_1	t_7	R_4	tag_1	t_{13}	R_{17}	tag_1	t_{28}
R_4	tag_1	t_8	R_9	tag_1	t_{14}	R_{17}	tag_1	t_{29}
R_4	tag_1	t_9	R_4	tag_1	t_{14}			

Table 1: Example raw RFID data [3]

with tag_1 is detected by readers R_4 and R_9 simultaneously, due to the unexpected detection range expansion of reader R_4. Therefore making object whereabouts unclear, thus giving rise to spatial ambiguities (false positives). A moving person can appear in both locations, since two readers R_4 and R_9 detection range cover two separate locations.

After t_{16}, a person with tag_1 was next detected by reader R_{17} from time point t_{26} until t_{29}. However, looking at the movement path a person has taken, it is clear that a person has passed through reader R_{10} before going through R_{17}, but somehow reader R_{10} completely missed tag_1, therefore failed to create any data.

3 Thesis Overview

This section gives an overview of the papers included in this thesis.

3.1 Spatiotemporal Data Cleansing for Indoor RFID Tracking data

This section gives an overview of Paper A [1].

Motivation and Problem Statement

RFID data cleansing is paramount to the RFID based applications, since the data generated by these applications are inherently unclean. Therefore it is a norm to cleanse the RFID data before any top level processing and analysis.

The work presented in [1] addresses the two main types of errors, redundant readings which are referred as *Temporal Redundancy* and false positives which are referred as *Spatial ambiguity*. Temporal redundancy occurs when a moving object with an RFID tag is continuously read by a same reader for more than allocated time. Spatial ambiguity occurs when a moving object is detected by more than one reader simultaneously. As a result objects

whereabouts are reported by several readers at different locations, therefore, causing spatial ambiguities. To cater the problem of redundant readings and spatial ambiguities present in the raw RFID data, following two tasks are intended to accomplish it.

Temporal Redundancy Elimination. In this task raw RFID readings are aggregate into tracking records. The data reduction is lossless i.e., size of the raw data is significantly reduced without any loss of information.

Spatial Ambiguity Reduction. After getting data from first phase. The spatial ambiguities are first identified in a given large set of tracking records and later reduced by utilizing the information captured by distance-aware graph model [1].

Solution Overview

To cater these errors a two phase solution shown in Figure 7 is presented in paper [1].

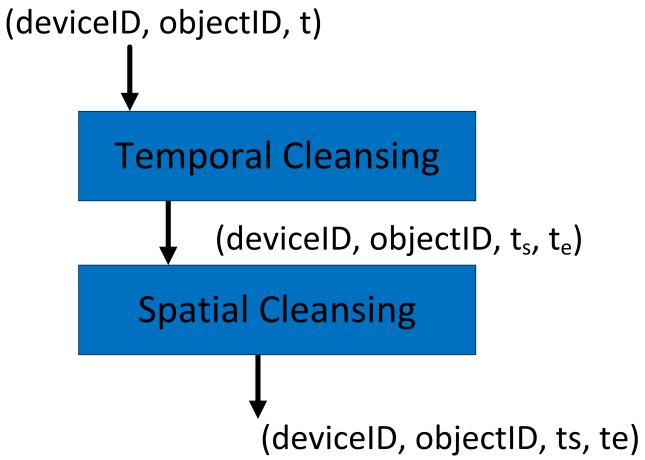
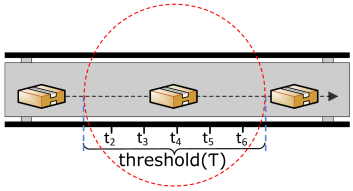


Fig. 7: Two-phase spatio-temporal cleansing [1]

The *Temporal Cleansing* takes raw RFID readings $(deviceID, objectID, t)$ as an input. The raw readings are sequentially scanned and subsequently turned tracking records of format $(deviceID, objectID, t_s, t_e)$.The aggregation process is controlled by a threshold parameter τ. The raw RFID readings of same object are combined together if the difference between the detection time of two consecutive readings are less than threshold value; otherwise, two different tracking records are created for the two readings. A new table called Aggregate Tracking Table (ATT) is created which stores all generated tracking records.

The threshold τ for each reader is estimated using the detection range of that reader and the speed with which the object is moving, see Figure 8



$$threshold = \frac{detection\ range\ diameter}{object\ moving\ speed}$$

Fig. 8: Threshold setting example and formula [1]

The *Spatial cleansing* phase takes the data from first phase as an input. The cleansing process starts first with identifying cross readings in the data, and then remove them by using a *distance-aware graph model* [1]. The distance-aware graph models the deployed RFID readers as the graph vertices and paths connection them together as edges. Each edge captures the minimum travel time require by the object to move from one reader to other. To keep it flexible, minimum travel time is not captured directly on the edges, rather the distance between the two readers and the speed of moving object between the two readers, which are then used to calculate the exact travel time a moving object may take to reach from one reader to another. The vertices representing each deployed reader captures the minimum dwell time which can be modified by tuning the sampling frequency of a reader.

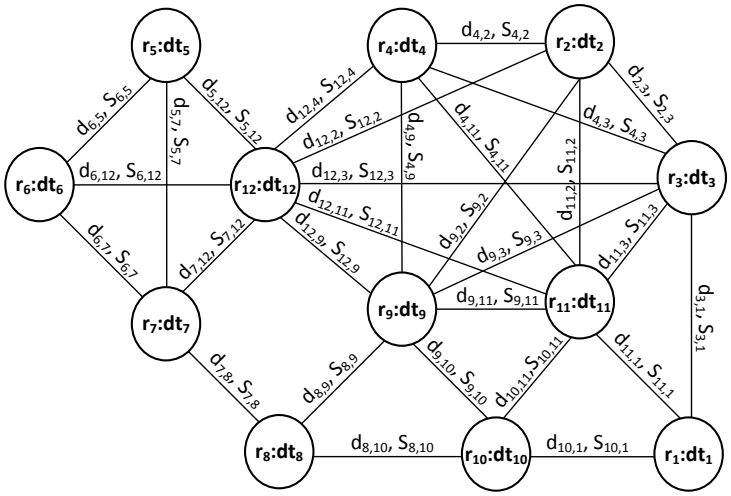


Fig. 9: Example distance aware graph model [1]

For illustration, a distance-aware graph model example is shown in Figure

9. Each vertex represents a a deployed reader and captures the dwell time e.g, reader r_1 and its dwell time is shown as $r_1 : dt_1$, where, dt_1 is the dwell time of reader r_1. The paths between readers are represented as edges and each path captures the parameters true for the path connecting two particular readers e.g., an edge between reader r_1 and r_{10} captures the distance ($d_{10,1}$) between r_1 and r_{10} and the speed ($S_{10,1}$) with which an object can move between these two readers.

The proposed algorithms for temporal redundancy elimination and spatial ambiguity reduction were validated using both real airport baggage RFID dataset and synthetic RFID datasets. The real dataset where obtained form the automatic baggage handling system at Aalborg airport. The data about 20,000 bags were recorded continuously for two month period. The result of the experiment performed in [1] to validate the threshold formula shown in Figure 8 indicates how the data reduction ratio remain unaffected with the threshold values which are very small and very large. The data reduction ratio used to measure the effectiveness is defined as defined as $\frac{\text{\# of records before cleansing}}{\text{\# of records after cleansing}}$. The results are shown in Figure 10.

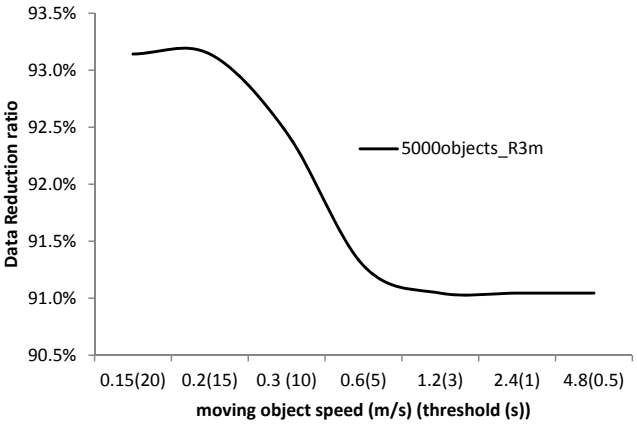


Fig. 10: Effectiveness plot of the threshold formula [1]

Over all experiments show that the proposed algorithms both effectiveness and efficiency.

3.2 Handling False Negatives in Indoor RFID Data

This section gives an overview of Paper B [2].

Motivation and Problem Statement

In large indoor space like an airport, shopping malls etc., number of readers are deployed. A commonly used passive RFID tags The massive amount of data generated by RFID based applications need to be stored efficiently and cleansed before using it for any high level business processing. One of the key error present in raw RFID data is false negatives. False negatives, aka., missing readings occur in the data when one or more readers somehow fail to generate the data about the object presence when present in its detection range. False negatives can happen in the data for various possible factors such as reader hardware malfunctioning, tag orientation, and other factors. Even in the most ideal circumstances, it is very often that the missing rate is between 30-40% [11]. With such a high error rate, the inconsistencies are bound to happen when relying directly on the raw RFID data. The false negatives can happen when a particular reader misses the object during some epochs (reading cycles) and detects during some, therefore creating possible gaps in between the data generated by that particular reader, such false negative are refereed as *partial false negative* while as when a reader completely misses the object, a *full false negatives* are created in the data. This paper specifically focuses on the full false negatives.

In [2] the problem recovering the missing information is formalized as two parts. First, the places in the data where data is missing are located, and second the missing information is inserted in the data.

Solution Overview

A new solution for handling false negatives raw RFID tracking data is presented in [2]. The cleansing process starts with the detection of false negatives and then inserting the recovered data in the tracking data.The missing reading recovery problem is presented as to find the most most likely path between the last reader (source) where object was last detected and the next reader (destination) where the object was next detected. Some of the possible false negatives cases are shown in Figure 11.

Some of the possible false negatives are shown in Figure 11. A simplest case of full false negatives is shown in Figure 11(a). In this case the RFID readings from both previous (R_s) and next (R_d) reader are present and the one reader(R_{miss}) in between have completely failed to detected the object. The second case in Figure 11(b) is the case where more than one reader in between the R_s and R_d have not been able to detect the object. In this case it is hard to recover the information, since there is a possibility of having more than one path between R_s and R_d and object might have taken one of such paths, which one is that is hard to find from given data. In Figure 11(c) a case is presented, where there is no further evidence of any reading which might

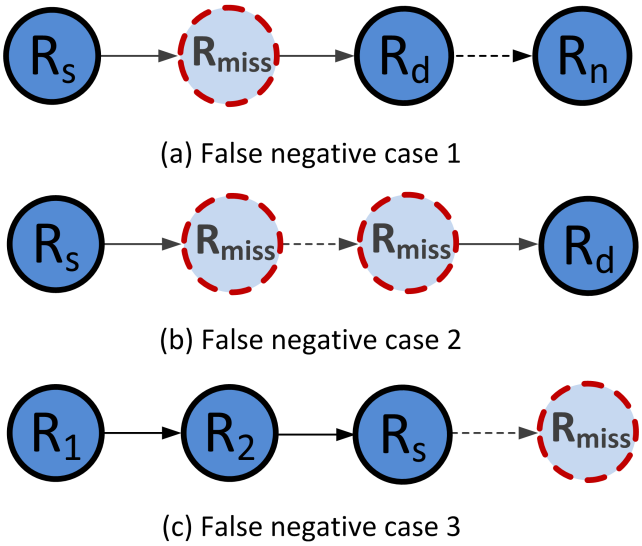


Fig. 11: False negative cases [2]

have indicated whether an object have moved forward or not, thus, any false negative recovery may result in creating false positives in data.

A *probabilistic distance-aware graph* [2] based solution is presented in this paper. The model is utilized first to locate the places where data has been missed and then used to inset the right information. The proposed graph shown in Figure 12 is an enhancement of a graph proposed in [1]. The graph models the deployed RFID readers as vertices and captures the relevant information of each reader such detection range and sampling rate e.g., a vertex $(R_1 : d_t, S_f)$ represents the reader R_1 with its detection time (d_t) and sampling frequency (S_f). The paths between readers represent edges which captures the parameters like minimum travel time and transition probability; The minimum travel time is the amount of time require to travel from one reader to another and the transition probability is likelihood that a moving objects moves form one reader to other reader e.g., the edge between the readers R_1 and R_2 captures $tt_{1,2}$ as minimum travel time and $p_{1,2}$ as a transition probability with which objects move between R_1 and R_2. The transition probabilities are learned from a historical data.

The devised graph is used first to detect errors by sequentially scanning the raw data. Starting form first reader, every time a readings from new reader appears algorithm checks if the readings are from one of the neighbors of previous reader, the process is simply repeated if readings are from one of the neighbor, otherwise false negative occurred in the data. The processes is simply stopped if there is no neighbor found. Once the false negative is

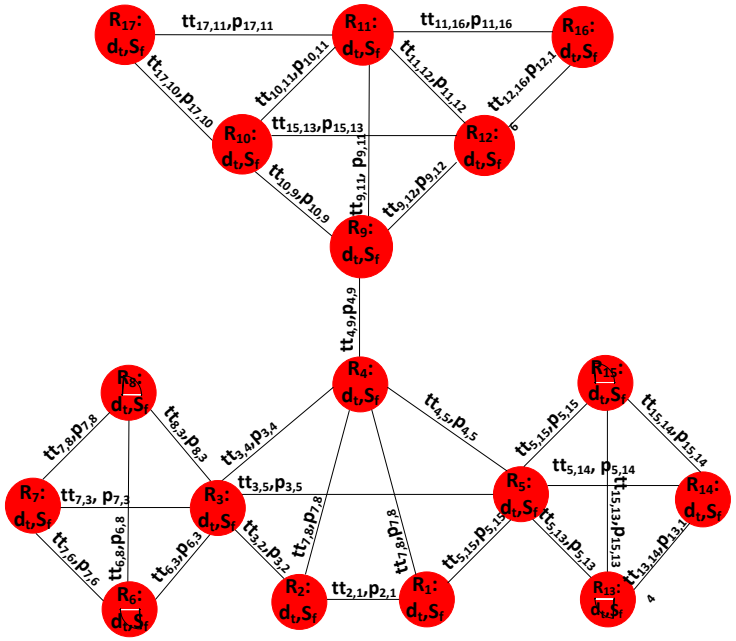


Fig. 12: Example Probabilistic distance-aware graph model [2]

detected, the recovery process of false negative starts by finding a set of candidate paths paths between the previous reader and next reader using the probabilistic distance-aware graph. Next a most likely path is chosen from a candidate set of paths using the transition probabilities captured by the graph model. The information of each reader in the most likely path is then filled using the information like detection range, sampling rate and minimum travel time captured by the graph. The proposed algorithms were tested using a real airport passenger baggage RFID dataset and a synthetically generated RFID dataset. The experiments conducted indicate the effectiveness and efficiency of proposed algorithms.

3.3 Learning-Based Cleansing for Indoor RFID Data

This section gives an overview of Paper C [3].

Motivation and Problem Statement

A learning-based RFID cleansing solution is proposed in [3]. The work is aimed to address the challenges posed by the uncertainties (specifically, false negatives and false positives) present in raw data. Many existing raw RFID data cleansing techniques need detailed prior knowledge for cleansing op-

erations. For offline indoor RFID data cleansing, the number of proposed cleansing approaches [1, 2, 10] depend on graph models that capture the indoor space topology, the reader deployment information, and relevant information such as the minimum travel time and the minimum detection time. In online RFID data case, the data is continuously streamed. In this case models which are probabilistic in nature are built for cleansing raw RFID data [19]. Another approach [15] presented a model to handle online data by using the correlated between the locations of neighboring objects. In contrast to most of the existing works, the work presented in [3] overcomes such assumptions and requires only some basic domain information about the indoor space settings (more details are given in [3]). In this work the RFID data cleansing problem is addressed using a machine learning approach. The RFID based systems are generally non-deterministic and probabilistic in nature, therefore to model such a data a Hidden Markov Model (HMM) [16] is a suitable choice. A more formal problem definition is given below:

Definition 1
(**RFID Data Cleansing Method**) [3] Given the raw RFID data and the HMM model λ. Using HMM model λ, the correct locations of a moving objects are inferred by filtering out the false positives and fill in the inscribed false negatives from raw RFID data.

Solution Overview

In Paper C [3], a HMM-based solution is provided to deal with the erroneous raw RFID data. The probabilistic parameters of model are learned from raw RFID data. In particular, a variant of the traditional HMM [16] an *Indoor RFID Multi-variate Hidden Markov Model (IR-MHMM)* is proposed used in inferring the exact reader locations of moving objects.

The RFID data acquired needs to be transformed into a set of multivariate observations suitable for a IR-MHMM. The RFID data is first transformed into binary multivariate time series, where the timestamped data represents the status of deployed readers. The status of the reader is set to 1, if reader detects the object otherwise set to 0.

To define the number of intervals to be used in the time series, a new parameter α is introduced. Given the RFID data of a specific object, we can get the number of intervals as $T = \frac{(t_e - t_s)}{\alpha}$, where t_s and t_e are the first and last reading reported about the specific moving object. For example we choose a time granularity $\alpha = 0.5$ seconds and total travel time ($t_e - t_s$) is 30 seconds, we then have $T = \frac{30}{0.5} = 60$ intervals.

For a trajectory of a person with tag_1 shown in Figure 6, we define a binary multivariate *(BS)* time series as:

$$\mathcal{TS}_i = \langle BS_i^{(1)}, BS_i^{(2)}, ..., BS_i^{(T)} \rangle,$$

$\mathcal{TS}_i$ is a sequence of T binary vectors, each BS_i^t represents the status of readers detecting a moving object tag_i at some time interval t. Each vector is of the size equal to the number of deployed readers, and each data point (bit) takes a binary value based on whether a corresponding reader detects an object or not at time t. If a moving object is detected by one or more readers at time t, the corresponding bits are set to 1, otherwise all the bits are set to 0.

1000000000000000	0000000100000000
1000000000000000	0000000100000000
1000000000000000	0000000000000000
1000000000000000	0000000000000000
0000000000000000	0000000000000000
0000000000000000	0000000000000000
0000000000000000	0000000000000000
0001000000000000	0000000000000000
0001000000000000	0000000000000000
0001000000000000	0000000000000000
0001000000000000	0000000000000000
0001000000000000	0000000000000001
0001000000000000	0000000000000001
0001000010000000	0000000000000001
0001000010000000	0000000000000001

Fig. 13: Binary vector representation [3]

To illustrate the data transformation of raw RFID data into multivariate binary reader observation sequences, the raw RFID data shown in Table 1 [3] and the time granularity parameter α is set to 1s. A multivariate binary reader observation sequences representing the raw RFID data from Table 1 are shown in Figure 13. In the example, the readers do not seem to have detected tag_1 in many intervals, and in the 16-th and the 17-th interval tag_1 is detected by two readers. A proposed IR-MHMM in [3] is able to predict the status of a reader, i.e., whether a given reader will produce a reading or not.

The tracking of moving objects in an indoor space can be characterized by two sets of random variables: *visible variables* are the observations in the RFID data, and the *invisible variables* (hidden) represent the true unobserved

locations of tracked objects. The two random variables are outcomes of two stochastic processes which can be well modeled by *Indoor RFID Multi-variate HMM (IR-MHMM)*.

Definition 2 (IR-MHMM)
An IR-MHMM proposed in [3] is a Multi-variate Hidden Markov Model $\lambda = (S, V, A, B, \pi)$, where:

1. $S = \{s_1, \ldots, s_N\}$ is a set of states which are hidden.
2. $V = (V_1, \ldots, V_M)$ is the multi-variate observation space, with $V_m = \{0, 1\}$ for $m = 1, \ldots, M$.
3. A is a transition probability distribution of states in S, $A = (a_{ij})_{i,j=1,\ldots,N}$.
4. B is a probability distribution of observations, given the states in S. $B = (b_{ik})_{i=1,\ldots,N;k=1,\ldots,M}$.
5. $\pi = (\pi_i)_{i=1,\ldots,N}$ is the initial state probability distribution.

For $t = 0, 1, 2, \ldots$ we denote with $S^{(t)}$ a random variable with values in the state space S, and $\mathbf{V}^{(t)}$ a multi-variate random variable with values in the observation space $\mathbf{V}$. The IR-MHMM defines a joint probability distribution of these variables according to the graphical model shown in Figure 14, and the conditional probability distributions $P(S^{(1)} = s_i) = \pi_i$ $(i = 1, \ldots, N)$, $P(S^{(t+1)} = s_j | S^{(t)} = s_i) = a_{i,j}$ $(i, j = 1, \ldots N; t = 0, 1, 2, \ldots)$, $P(V_k^{(t)} = 1 | S^{(t)} = s_i) = b_{ik}$ $(k = 1, \ldots, M; i = 1, \ldots, N; t = 0, 1, 2, \ldots)$. Note that apart from the standard Markov assumption, the model assumes that the components of the observation vector are independent given the current hidden state. Concretely, at a given point in time, whether a reader produces a false negative or false positive reading only depends on the current hidden state of the tracked object but not on the output of any other readers [3].

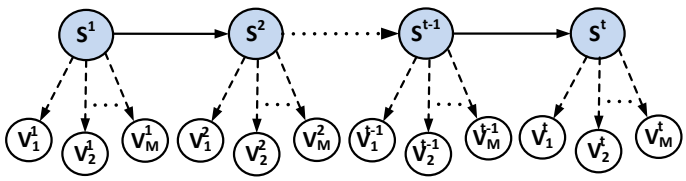


Fig. 14: IR-MHMM for RFID raw data [3]

Consider the example of the moving object tag_1 from Figure 6. A subsequence of tag_1 is taken to illustrate the state model. A state set $S = \{S_1, S_2, ..., S_N\}$ models the covered and un-covered states through which tag_1 passes. For example, state S_1 represents the reader R_1. We use a state sequence $Q = \{q^{(t)}, q^{(t+1)}, ..., q^{(T)}\}$ to model the transitions of object locations

of tag_1 from one state to another state over the T intervals. A state $q^{(t)}$ models the object locations in the t-th interval which depends only on its previous state $q^{(t-1)}$. In Figure 15, a state $q^{(t)} = S_1$ indicating that tag_1 is in reader R_1 at $t = t_1$ and is dependent upon a state in the *(t-1)*-th interval $q^{(t-1)} = S_1$, which is also in R_1. A binary RFID data time series $\mathcal{T}S_i$ models the variation of observation status of each deployed reader during the T intervals. From Figure 15, the binary multivariate observations variables v_{t_0} at time t_0 is 1000000000000000000, which indicates that in t_0-th interval, tag_1 is tracked by reader R_1 only as shown in Table 1.

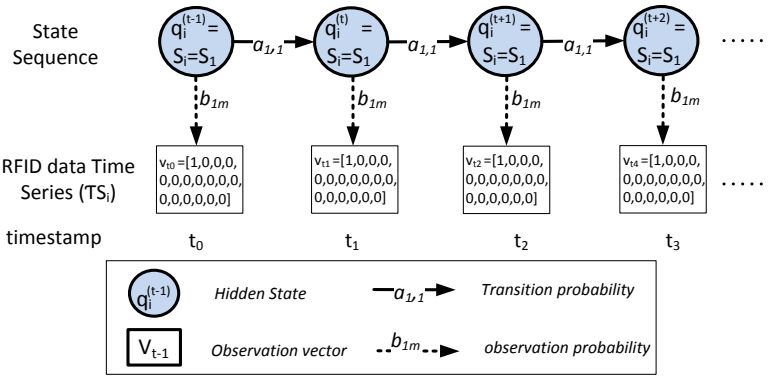


Fig. 15: Movement history of a moving object tag_1.

The topology of an HMM model is chosen in advance (prior), then the estimation of fixed parameters are learned from a sample data. However choosing the best HMM topology is not trivial. Two key components of an HMM topology is the number of states and the connectivity. Typically when a model is selected, several HMM's with different number of states and connectivity are trained and the one with maximum posterior probability is selected. However, this approach have two main problems. 1) training an HMM with each feasible topology is very costly and 2) learning parameters is prone to get stuck into local minima. To overcome this problem, the best way out is to learn the HMM parameters is by sampling from initial conditions, which can be derived from the background information of the application for which the HMM is modeled for. The similar approach is followed in this work and derived the initial conditions from the domain knowledge. Three different state space models are designed [3], namely, *minimum-state model (MSM)*, *last-state model (LSM)* and *in-between-state model (ISM)*. The three models are designed from indoor topology with different number of states and with different connectivity.

Data Cleansing

Once the models are ready i.e., the state set S is known, the probabilistic parameters of the IR-MHMM are learned from the observed historical RFID data using a general method EM (Expectation Maximization) [5]. After learning the model parameters, the model is used to cleanse the raw RFID data. First the the most probably hidden state sequence is computed given the observation using standard Viterbi algorithm [21]. Further on the most probably observation sequence is computed from the most probable hidden state sequence pointwise for each timepoint t.

The work [3] also presented the evaluation framework which is used to evaluate the effectiveness of proposed models. The inferred observation trajectory is compared with the ground truth trajectory to find the similarity between the two trajectories. A variant of traditional edit distance known as weighted edit distance is used to quantify the dissimilarity by the number of operations (insertion, deletion and substitution) needed to transform one trajectory into the other. In weighted edit distance the cost of substituting any symbol depends on which symbol in the string is deleted or inserted [3]. For evaluation, weighted edit distance computation methods is applied in two different ways. In first way, two original sequences are used, an observation sequence inferred by a model and its corresponding ground truth sequence. In this way both spatial and temporal aspects of the inferred observation sequences are computed.The edit distance score in our case quantifies the accuracy the inference accuracy of learned models. Following the second way, the two sequences, real observation sequence and the actual sequence are first transformed into block sequences. The block sequences are formed by merging together the same observation symbols into a single symbol. The inferred block sequence length is directly dependent upon the accuracy of the inferred results. Once the block sequences are ready, an edit distance is measured on two. For an illustration, an example is presented in Figure 16. Assuming a cost for insertion, deletion and substitution operation as 1, the edit distance between original sequences and block sequences are measured.

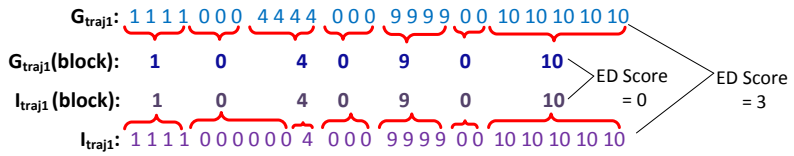


Fig. 16: Edit distance example

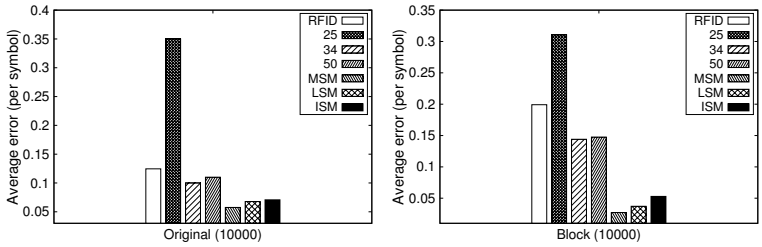
An extensive experiments studies are done with both real airport passenger baggage dataset and synthetic dataset [3]. The performance evaluation of learned models are done with following two scenarios. 1) **Online Scenario:**

In this scenario, two entirely different datasets were used, one for learning and the other for testing the effectiveness of the learned models. 2) **Offline Scenario:** In this scenario, a single dataset is used for both training (learning) and testing.

A metric average error [3] is used to measure the effectiveness of proposed approaches.

$$Average\ error = \frac{sum\ of\ edit\ distance\ scores\ of\ all\ the\ sequences}{total\ \#\ of\ inferred\ symbols\ of\ each\ sequence}$$

The effectiveness of three proposed structured models MSM, LSM and ISM were highlighted by comparing the results of three with three un-structured models with a state space of 25, 34 and 50 states, respectively [3]. The result shown in Figure 17 clearly demonstrates the significance of structured models.



(a) Original sequences (b) Block sequences

Fig. 17: Structured Vs unstructured models [3]

The paper [3] presents the experiments to check the effect of increase training data were done in both online and offline case. Also experiments to check scalability by increasing state space size, versatility check by using different indoor environment like airport, office building, warehouse etc., were done. A comparison with the state of art [1, 2, 10] is also done using query analysis by issuing two type of queries, namely, *stay queries* and *trajectory queries*. Overall results indicate our proposed models are effective, scalable and versatile.

3.4 Mapping Raw RFID Data to Most Probable Indoor Paths

This section gives an overview of Paper D [4].

Motivation and Problem Statement

A key problem with raw RFID data is its quality. The raw RFID data is generated by unreliable devices and communicated through inconsistent radio

frequency channels, and the indoor environments in which the devices operate are quite dynamic. These along with many other factors produce errors in raw RFID data. With the presence of errors in the raw RFID data, any kind of an high level processing will produce inconsistent results. It is therefore required either to cleanse the RFID data physically, i.e., reduce the false positives and insert the false negatives into the data, or have technique to extract some useful information from the raw data.

In [4], a new approach based on regular expression matching is proposed to find the most probable indoor paths a moving object may have taken from raw RFID data. Some of the exiting work in data mining, knowledge discovery and molecular biology [22–24] follow similar approach, but not in the context of handling raw indoor RFID data. In RFID-based setting, the raw RFID data consists of readings whcih are generally in the form $\langle r, o, t \rangle$, which means that reader r detected an object o in its detection range at time t. On the other hand, indoor spaces are generally identified and represented as symbols in a given indoor context, e.g., security check, tax-free shops, passport checks, and boarding gates in an airport setting. In this work the goal is to map an object's raw RFID data to possible sequences of symbols that are likely to be the object's movement paths of semantic locations. In case there are errors such as cross readings and missing readings present in the raw data, mapping the raw data into symbol sequences that are perfectly matching or sufficiently close to the actual paths is not easy. Therefore, the major challenge in this work is to achieve accurate mapping from raw data to indoor paths in the presence of errors in the data.

For illustration, an example of an airport baggage handling system with multiple paths is presented in Figure 18 [4].

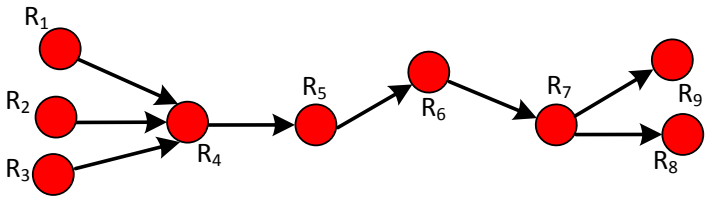


Fig. 18: Example indoor route and layout [4]

A passenger may check-in his bag at any of the three check-in desks with three readers (R_1, R_2 and R_3). From there a bag continues with a route through readers R_4, R_5, R_6 and R_7 until it takes one of the two belt-loader readers (R_8 or R_9). The example set of routes is represented by regular expression pattern: $p_1 = (R_1|R_2|R_3)0R_40R_50R_60R_70(R_8|R_9)$. The regular expression p_1 have three sub patterns, first one is $(R_1|R_2|R_3)$, which indicates that a bag can start from either R_1, R_2 or R_3, followed by value 0, indicating

that the bag is seen by no reader. Second, a single occurrence of sub-pattern $R_4 0 R_5 0 R_6 0 R_7 0$ followed by third and last sub-pattern $(R_8|R_9)$, which indicates that bag can go either to R_8 or R_9. Suppose that a bag's raw RFID data is captured as an input string $s = R_1 0 R_4 0 R_5 0 R_6 0 R_7 0 R_8$, then we can say there is a match of pattern p_1 in s. In particular, R_1 of s matches the sub-pattern $(R_1|R_2|R_3)$, R_8 at the end of s matches the sub-pattern $(R_8|R_9)$, and in the middle there is a single sub-pattern $R_4 0 R_5 0 R_6 0 R_7 0$ in s. Therefore, there is a match of p_1 in s.

Solution Overview

The paper [4] presents a matching algorithm to map raw RFID data to most probable indoor path moving object may have taken. The approach is composed of three main components: an automaton that accepts regular expressions as input, an error model that captures the distribution of possible errors in the regular expression matching, and relevant matching algorithms that map raw RFID data to probable indoor paths as sequences of symbols.

In the context of indoor RFID tracking, the regular expressions is set of symbols which represent the deployed RFID readers and are simply referred as RFID regular expressions (RRE). On the other hand, the ground truth regular expressions represent the true traces of movement patterns of moving object in a given floor plan. The true traces of object movements in a given indoor space are generated first using the indoor space constraints and the deployed reader characteristics, then specified as RREs. The indoor topology is used to derive the connectivity between a pair of indoor locations where RFID readers are deployed. Also the travel time (TT) constraint is used, which is the time required to reach from one location to another. In order to recognize RFID observation sequences of indoor moving objects, these regular expressions are presented in an automaton.

For an illustration, an example RRE is shown in Equation 1. Each reader in the path generates n readings about the moving object, depending on reader's sampling frequency. For ease of understanding, the travel time between any two directly connected readers is assumed to be 2 seconds and sampling frequency of each reader is assumed to be same, each generating 4 readings. However, in reality travel time between readers may vary and the sampling frequency of readers may be different. The true trace (ground truth) is represented by following RRE.

$$GT = [a]\{4\}[00][b]\{4\}[00][c]\{4\}[00][e]\{4\} \tag{1}$$

Here, $[p]n$ represents that the pattern p occurs n times.

Other important component of the proposed approach is an error model which captures the possible errors between the observation values and true values. The error model is assumed to be an independent error distribution

for each character in a string. Each error value e_i is a function of an observation value r_i and the true value x_i, i.e., $e_i = (r_i, x_i)$. The error model is designed based on the indoor distance between the readers and the detection range of each reader. The design principle is justified by reality, i.e., farther the two readers less like it is to have an observation from one to be true for other. For any true value there is a possibility of having multiple observed values. In general each error value is function of observed value and true value. The error distribution for each reader is represented in matrix called certainty matrix.

Finally, the matching algorithm utilizes both the automaton representing the RREs and the error model to find the most probable indoor paths from the raw RFID data. The matching algorithm uses a minimum threshold value find the set of candidate paths, then depending upon the k parameter returns the top-k indoor paths.

4 Summary of Contributions

The major portion of this thesis are the four papers, each of which devises new techniques to cleanse indoor RFID data.The four papers make the following contributions.

- Paper A [1] formalizes the problem of cleansing of redundant readings and cross readings for indoor raw RFID tracking data. First, a threshold-based cleansing algorithm is proposed to eliminate the redundant readings in indoor RFID data. Second, an algorithm to reduce the spatial ambiguities (cross readings) is proposed. The algorithm uses a distance-aware deployment graph [1], which captures the spatiotemporal constraints of an indoor space and the deployed reader characteristics. A spatial cleansing algorithm uses the proposed graph first to identify the cross readings in raw RFID data, then to reduce the identified spatial ambiguities. The paper also presents a through experimental studies to show that the algorithms proposed are effective, efficient and scalable.

- Paper B [2] formalizes the problem of cleansing missing readings in raw indoor RFID tracking data. A new weighted graph is devised, which is the enhancement of the graph model proposed in [1]. The proposed graph model is used to design an algorithms to single out the places where information is missing in indoor processed raw RFID tracking data. Subsequently, the graph model is used to design an algorithm to recover the identified false negatives. The exact number of missing readings are filled using the information captured by each graph vertex. A thorough experimental study is done to evaluate the proposed

false negative handling algorithms. The experimental results show the effectiveness, efficiency and scalability of the proposed cleansing techniques.

- In Paper C [3], a new machine learning approach to handle the uncertainties in indoor raw RFID tracking data is proposed. A new Multivariate Hidden Markov Model (IR-MHMM) is proposed, which models the errors present in indoor RFID data. A non-trivial problem of choosing the state space and the structure of the model is catered by proposing three different state space design principles to design IR-MHMM for a given RFID reader deployment in an indoor floor plan. Paper also presents an evaluation framework based on weighted edit distance to find the correctness of results inferred by the learned models [3]. A detailed experimental study is conducted using both real RFID baggage data and synthetic data. The result not only demonstrate the efficiency, scalability, versatility and effectiveness of the proposed learning-based cleansing approach but also give insights about design principles of proposed IR-MHMM.
- Paper D [4] proposes an approach based on regular expression matching to find the most likely paths for object movements captured in raw indoor RFID data. We design an automaton that captures all possible indoor path in regular expressions, together with the algorithm to construct the automaton from a given indoor settings. An error model is designed that describes possible errors between raw RFID data and semantic symbols probabilistically. A new matching algorithms is devised that uses the automaton and the error model to find the most probable indoor path for given raw RFID data. Number of experiments are performed using both synthetic and real data sets to justify the efficiency and effectiveness of proposed algorithms.

References

[1] A. I. Baba, H. Lu, X. Xie, and T. B. Pedersen, "Spatiotemporal data cleansing for indoor RFID tracking data," in *Mobile Data Management*, 2013, pp. 187–196.

[2] A. I. Baba, H. Lu, T. B. Pedersen, and X. Xie, "Handling false negatives in indoor RFID data," in *MDM*, 2014, pp. 117–126.

[3] A I. Baba, M. Jaeger, H. Lu, T. B. Pedersen, W. S. Ku, and X. Xie. Learning-based cleansing for indoor RFID data. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 925–936, New York, NY, USA, 2016. ACM.

[4] A. I. Baba, H. Lu, W. S. Ku, and T. B. Pedersen, "Handling false negatives in indoor RFID data," in *GIS*, 2017, (submitted).

[5] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *Annals of Mathematical Statistics*, vol. 37, pp. 1554–1563, 1966.

[6] IATA BAGGAGE SYSTEMS INFRASTRUCTURE WORKGROUP, " IATA RP 1740c/ATA 30.39.," in *https://www.iata.org/about/sp/areas/Documents/baggage-systems-infrastructure.pdf*,.

[7] H. Chen, W.-S. Ku, H. Wang, and M.-T. Sun, "Leveraging spatio-temporal redundancy for RFID data cleansing," in *SIGMOD Conference*, 2010, pp. 51–62.

[8] P. J. Darcy, B. Stantic, and A. Sattar, "A fusion of data analysis and non-monotonic reasoning to restore missed RFID readings," in *ISSNIP*, 2009.

[9] G.D. Trotter, M.S.and Durgin. Survey of range improvement of commercial rfid tags with power optimized waveforms. In *IEEE International Conference on RFID*, pages 557–575, 2010.

[10] B. Fazzinga, S. Flesca, F. Furfaro, and F. Parisi, "Cleaning trajectory data of RFID-monitored objects through conditioning under integrity constraints," in *EDBT*, 2014, pp. 379–390.

[11] C. Floerkemeier and M. Lampe, "Issues with RFID usage in ubiquitous computing applications," in *Pervasive*, 2004, pp. 188–193.

[12] H. Gonzalez, J. Han, X. Li, and D. Klabjan, "Warehousing and analyzing massive RFID data sets," in *ICDE*, 2006, p. 83.

[13] ——, "Warehousing and mining massive RFID data sets," in *ADMA*, 2006, pp. 1–18.

[14] S. R. Jeffery, M. N. Garofalakis, and M. J. Franklin, "Adaptive cleaning for RFID data streams," in *VLDB*, 2006, pp. 163–174.

[15] Y. Nie, Z. Li, S. Peng, and Q. Chen, "Probabilistic modeling of streaming RFID data by using correlated variable-duration hmms," in *SERA*, 2009, pp. 72–77.

[16] L. R. Rabiner and B. H. Juang, "An introduction to hidden markov models," *IEEE ASSp Magazine*, 1986.

[17] J. Rao, S. Doraiswamy, H. Thakkar, and L. S. Colby, "A deferred cleansing method for RFID data analytics," in *VLDB*. VLDB Endowment, 2006, pp. 175–186. [Online]. Available: http://dl.acm.org/citation.cfm?id=1182635.1164144

[18] SITA, "The annual sita baggage report," Specialists in air transport communications and IT solutions, Tech. Rep., 2014.

[19] T. T. L. Tran, C. A. Sutton, R. Cocci, Y. Nie, Y. Diao, and P. J. Shenoy, "Probabilistic inference over RFID streams in mobile environments," in *ICDE*, 2009, pp. 1096–1107.

[20] G. Trotter, M.S.and Durgin, "Survey of range improvement of commercial RFID tags with power optimized waveforms," in *IEEE International Conference on RFID*, 2010, pp. 557–575.

[21] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 2, pp. 260–269, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1109/TIT.1967.1054010

[22] G. Kucherov and M. Rusinowitch. Matching a set of strings with variable length don't cares. *Theoretical Computer Science*, 178:129–154, 1997.

[23] H. Mannila. Methods and problems in data mining. In *ICDT*, pages 41–55, 1997.

[24] G. Navarro and M. Raffinot. *Flexible Pattern Matching in Strings: Practical On-line Search Algorithms for Texts and Biological Sequences*. Cambridge University Press, New York, NY, USA, 2002.

Part II

Papers

Paper A

Spatiotemporal Data Cleansing for Indoor RFID Tracking Data

Asif Iqbal Baba, Hua Lu, Xike Xie, Torben Bach Pedersen

The paper has been published in the
Proceedings of 14th IEEE MDM, pp. 187–196, 2013.

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.

The layout has been revised.

1 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 A.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

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

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 $\langle r_1, O_1, [t_1, t_5] \rangle$.

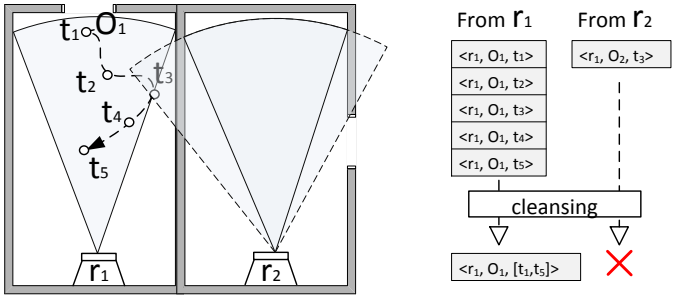


Fig. A.1: An example of RFID data cleansing

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 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 2 reviews the related works on RFID data management/cleansing and symbolic indoor tracking. Section 2.2 formulates the indoor RFID data cleansing problem that we tackle in this paper. Section 4 details the temporal cleansing algorithm, followed by the spatial cleansing techniques in Section 5. Section 6 presents extensive experiments on both synthetic and real data sets. Finally, Section 7 concludes the paper and discusses the directions for future research.

2 Related Work

In this section we review related work. We cover RFID data management and cleansing in Section 2.1, and symbolic indoor tracking in Section 2.2.

2.1 RFID Data Management and Cleansing

In recent years RFID technology has been widely used in many scenarios such as supply chain management [2–4], health care [5], people and object tracking [6–10]. Massive amounts of RFID data is generated by RFID-based applications, e.g., Wal-Mart produces about 7 terabytes of RFID data per day [3]. Roozbeh et al. [11] discuss the general challenges for RFID data management.

As raw RFID data are not clean, cleansing is needed. To handle missing readings, smoothing filters [12, 13] are employed in the steaming context. Mylyy [12] 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 [13] 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. [14] 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. [15] employs a counting based smoothing filter to remove cross reads in RFID streams. Liao et al. [16] proposes a probability model to estimate the tag density for each RFID reader, and utilize the model to remove cross reads. Tinggaard et al [17] 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 [12–16] 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.

2.2 Symbolic Indoor Tracking

We briefly review symbolic indoor positioning and tracking using the example floor plan shown in Figure A.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.

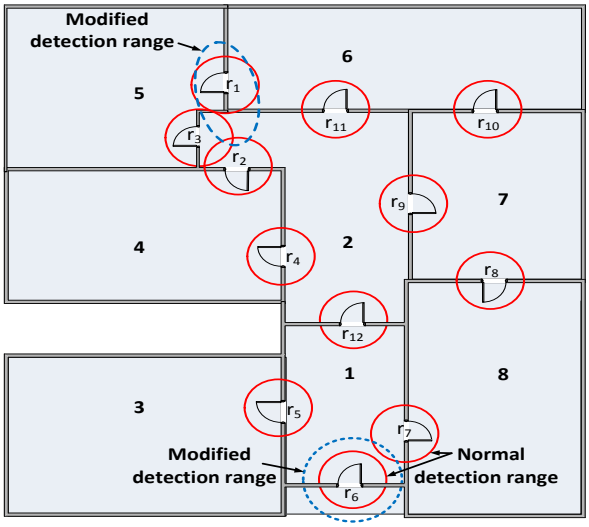


Fig. A.2: Positioning device deployment

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 A.2, an example table of raw readings is shown in Table A.1.

Table A.1: 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}
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}

Graph Based Indoor Tracking

Jensen et al. [19] 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 $G_{dev} = (V, E, D, \mathcal{L}_E)$, 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 A.3 presents the deployment graph for the indoor setting shown in Figure A.2. Note that vertex 9 in the graph represents the outdoor space.

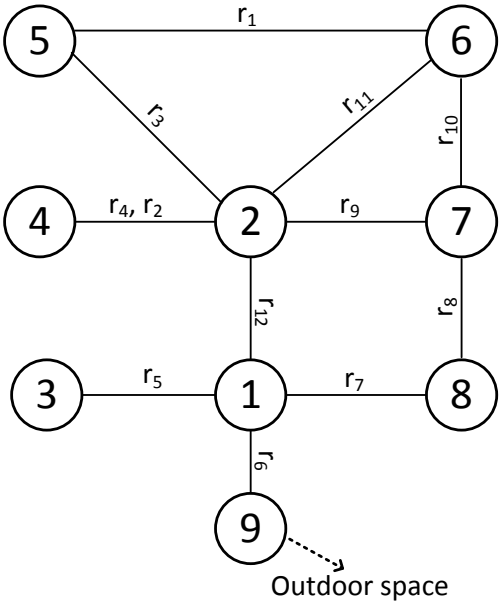


Fig. A.3: Deployment graph for the deployment in Figure A.2

In addition to the deployment graph, two mapping structures [19] 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 A.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 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 A.3, $V2D(1)$ returns $\{r_6, r_5, r_7, r_{12}\}$.

This paper distinguishes itself from the previous work [19] with several important characteristics. First, this paper considers the overlaping between different positioning devices, whereas the previous work [19] 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 [19] focuses on tracking the object when it is not seen by any device. This important difference is illustrated in Figure A.4. Third, in order to support data cleansing, this paper proposes a graph (Section 5.1) different from the deployment graph [19].

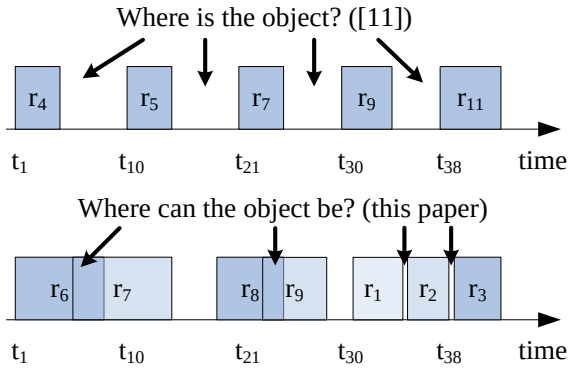


Fig. A.4: Difference from the previous work

3 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 3.1. We give an overview of our solution in Section 3.2. Table A.2 lists the notation used throughout the paper.

3.1 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 3
(**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 $rr_i.objectID = rr_j.objectID$.

Definition 4
(**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 $[t_s, t_e]$.

[2] τ is an application-specific threshold [15, 20].

Table A.2: 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 5
(**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.

3.2 Overview of Spatiotemporal Data Cleansing

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

[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 5.3.

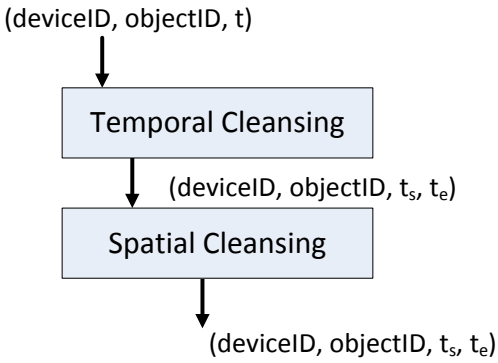


Fig. A.5: Two phase spatiotemporal data cleansing

otherwise, they will be aggregated into the same tracking record. We store all generated tracking records in the *Aggregate Tracking Table* (ATT).

Consider all the raw readings about $object_1$ from Table A.1, 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 A.3. 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 A.3: Aggregate Tracking Table (ATT)

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_{17}
r_9	$object_1$	t_{19}	t_{21}
r_1	$object_1$	t_{25}	t_{34}
r_2	$object_1$	t_{26}	t_{32}
r_3	$object_1$	t_{31}	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 an 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 5.

Refer to the running example whose ATT is shown in Table A.3. 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 A.4.

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.

Table A.4: 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_{17}
r_9	$object_1$	t_{19}	t_{21}
r_1	$object_1$	t_{25}	t_{34}
r_2	$object_1$	t_{26}	t_{32}
r_3	$object_1$	t_{31}	t_{38}

4 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 [19] assumes data are online in a stream and it does not use a time threshold.

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

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

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 A.6. In Section 6, we experimentally test the effect of this method for setting τ.

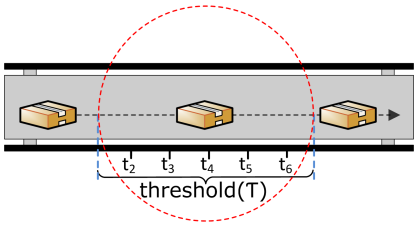


Fig. A.6: An example of threshold setting

5 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 5.1, we describe a graph that captures those indoor spatiotemporal constraints. In Section 2.3, we detail

how to construct such a graph given the RFID reader deployment in an indoor space. In Section 5.3, we propose the spatial cleansing algorithm using the graph.

5.1 Distance-Aware Deployment Graph

As described in Section 2.2, 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 [19] as shown in Figure A.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 A.2 is shown in Figure A.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.

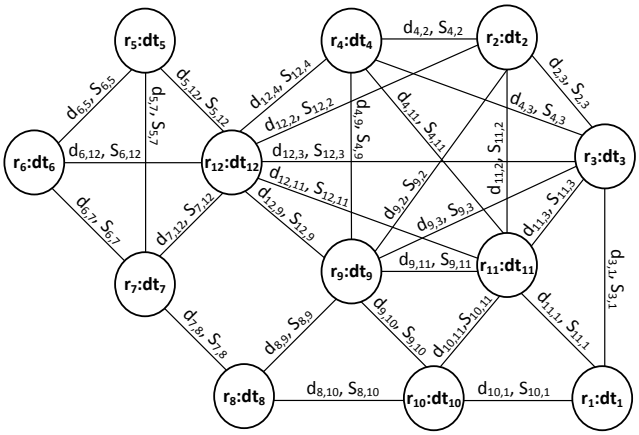


Fig. A.7: Example of distance-aware deployment graph

5.2 Distance-Aware Deployment Graph Construction

The distance-aware graph is constructed by Algorithm 13. 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).

Inside Algorithm 13, the indoor shortest pathes are computed according to the algorithms proposed elsewhere [21]. For the sake of simplicity, the input for those algorithms are implicitly passed to Algorithm 13.

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}

5.3 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).

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

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

```
1:  for each tr in ATT do
2:      mdt ← G_dd.L_V(tr.deviceID)
3:      tr' ← the previous record in ATT such that (tr.objectID = tr'.objectID) ∧
        (tr.deviceID ≠ tr'.deviceID)
4:      if tr' = null then
5:          continue
6:      mtt ← min_tt(tr'.deviceID, tr.deviceID)
7:      if (tr.t_s − tr'.t_e < mtt + mdt) then
8:          tr.t_s ← tr'.t_e + mdt + mtt
9:          if (tr.t_e − tr.t_s ≤ 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).

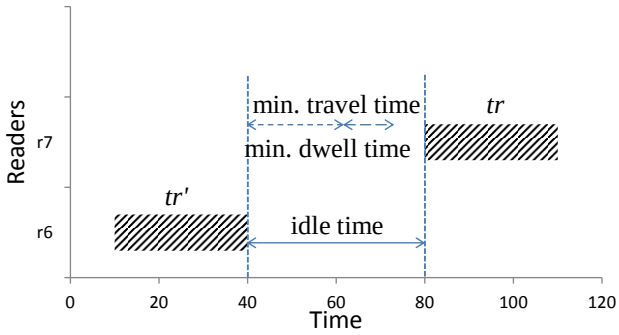


Fig. A.8: Spatial Cleansing Case 1

We illustrate spatial cleansing cases in Figure A.8, A.9 and A.10. The case shown in Figure A.8 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.

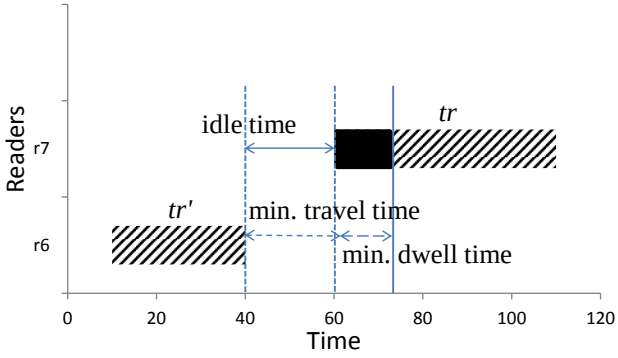


Fig. A.9: Spatial Cleansing Case 2

The case shown in Figure A.9 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.

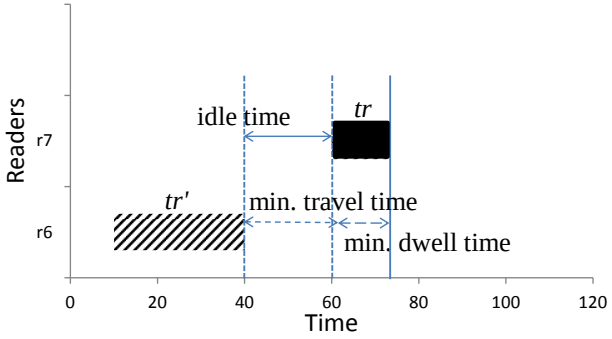


Fig. A.10: Spatial Cleansing Case 3

The case shown in Figure A.10 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.

6 Experimental Studies

In this section, we conduct experiments to evaluate our devised indoor RFID data cleansing techniques. Section 6.1 describes the experimental settings. Section 6.2 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

Table A.5: 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

memory.

6.1 Experimental Setup

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

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:

$$\text{data reduction ratio} = \frac{\text{\# of records before cleansing}}{\text{\# of records after cleansing}}$$

. 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.

Datasets

We used both synthetic and real data in our experimental studies. The parameter settings are summarized in Table A.5. The default values are bolded.

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 [22] 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 A.11, 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.

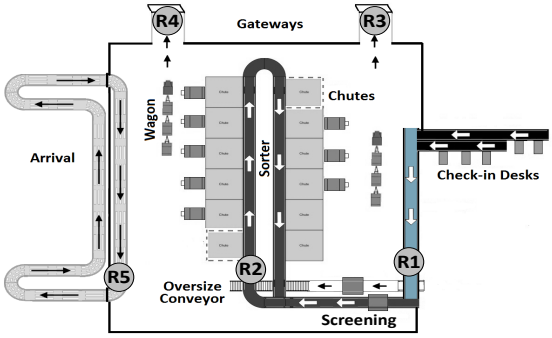


Fig. A.11: Baggage hall at Aalborg Airport

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.

6.2 Experimental Results

We start with investigating the effect of the formula (equation A.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 A.12. 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.

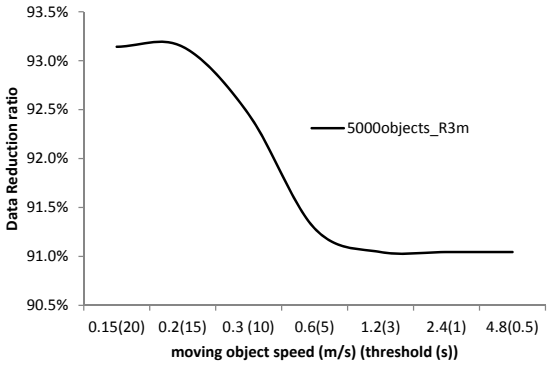


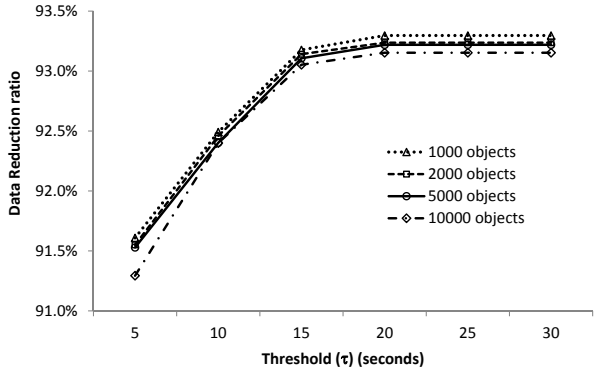
Fig. A.12: Effectiveness of the threshold formula (Eq. A.1)

Sections 6.2 and 6.2 report the spatiotemporal cleansing results on synthetic and real data, respectively. Section 6.2 briefly shows the experimental results on the distance-aware deployment graph construction.

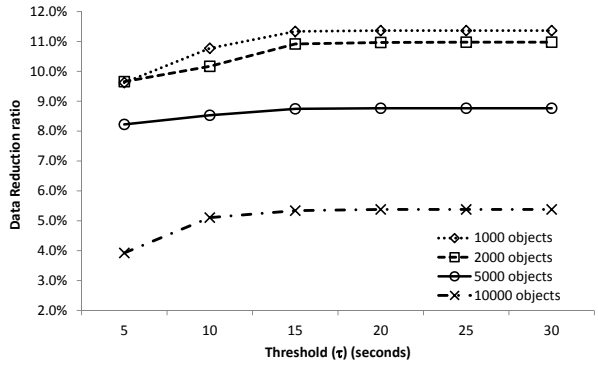
Spatiotemporal Cleansing Results on Synthetic Data

The results on the temporal cleansing effectiveness are reported in Figure A.13.

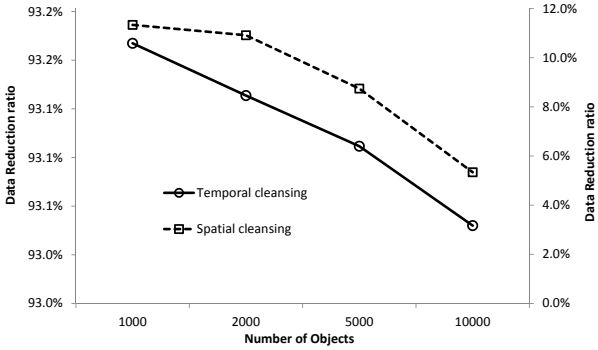
By transferring raw readings into aggregated tracking records, a significantly large portion of redundant records can be eliminated, as shown in Figure A.13(a). Specially, for the dataset having 10K objects, the reduction ratio is as high as over 90% in all tested cases. Also, the data reduction ratio 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



(a) Temporal cleansing effectiveness

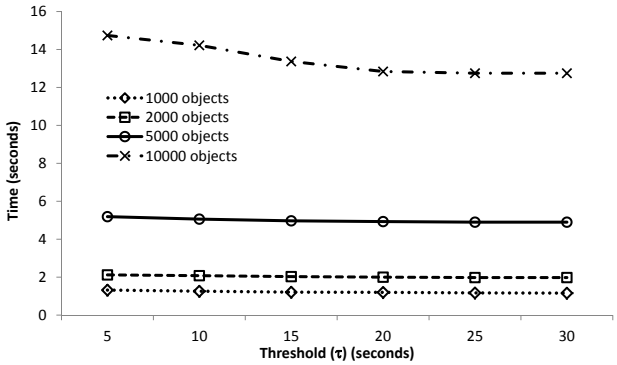


(b) Spatial cleansing effectiveness

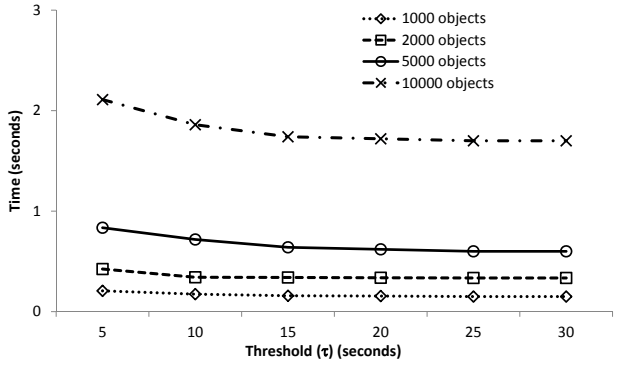


(c) Spatiotemporal cleansing effectiveness

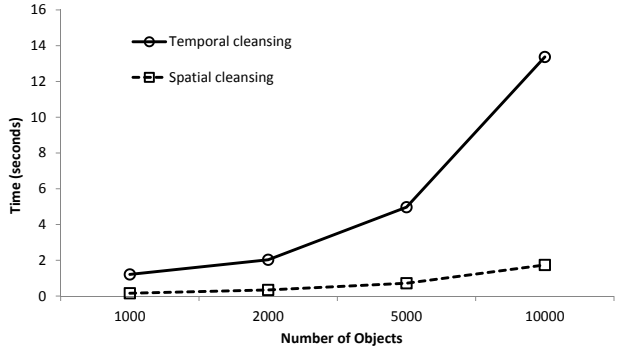
Fig. A.13: Cleansing effectiveness on synthetic data



(a) Temporal cleansing efficiency



(b) Spatial cleansing efficiency



(c) Spatiotemporal cleansing efficiency

Fig. A.14: Cleansing efficiency on synthetic data

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 A.13(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.

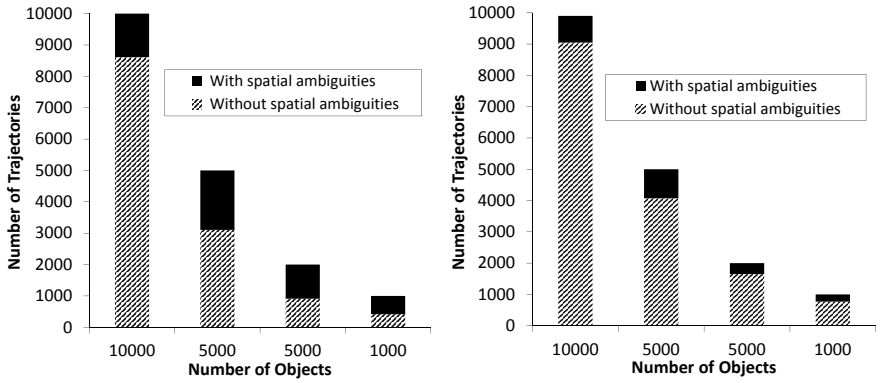
We also test the spatiotemporal cleansing effectiveness with respect to the number of objects and report the results in Figure A.13(c). The effectiveness of temporal cleansing does change much with increasing number of objects, as can be seen from the left Y-axis. More than 90% of raw data is reduced 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 A.14. Referring to the results shown in Figure A.14(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 A.14(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 A.14(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 the spatial cleansing. The results are shown in Figure A.15. 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. A.15: Valid path evaluation on synthetic data

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 A.16.

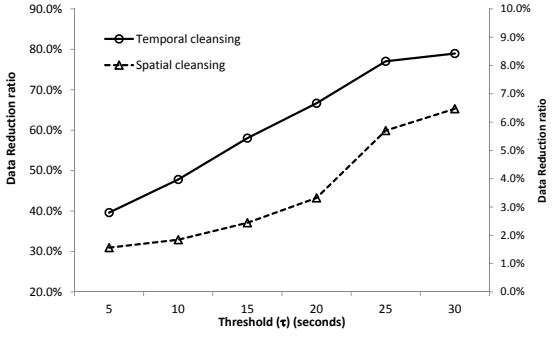


Fig. A.16: Cleansing effectiveness on real data

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.

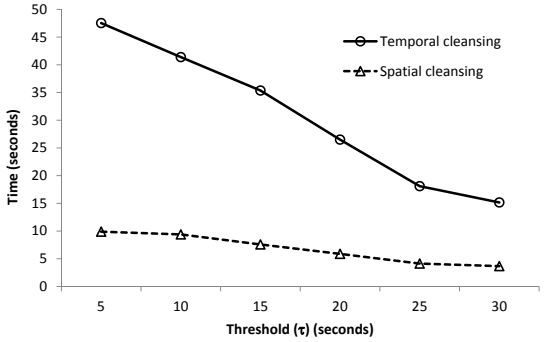
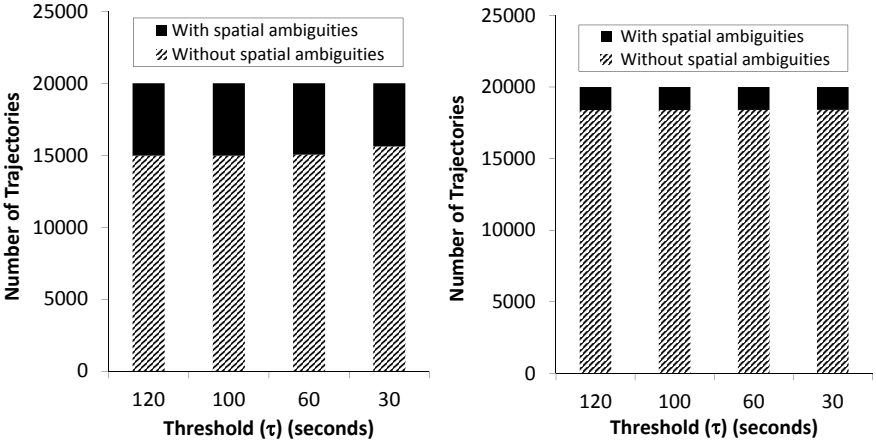


Fig. A.17: Cleansing efficiency on real data

We measure the efficiency of both algorithms over the real dataset and report the results in Figure A.17. The processing time of both algorithms decreases with the increase of the threshold. The trend is opposite to the observations in Figure A.16, 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 A.18 reports the relevant results.



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

Fig. A.18: Valid path evaluation on real data

Figure A.18(a) shows that before cleansing more than 25% trajectories have spatial ambiguities. In contrast, Figure A.18(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.

Distance-Aware Deployment Graph Construction Results

Finally, we evaluate the efficiency of the distance-aware deployment graph construction (Algorithm 13). 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 A.19.

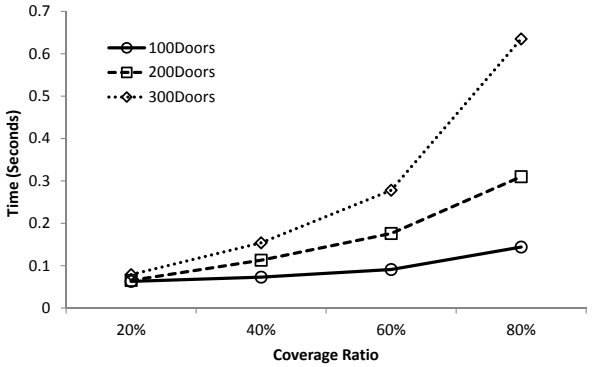


Fig. A.19: Distance-aware graph construction

In Figure A.19, each reported value is the average of 50 runs of Algorithm 13. 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 13, 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 A.11, it takes less than 50 milliseconds, which is very efficient. Due to the space limit, we omit the further details.

7 Conclusion

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. Here is the conclusion.

References

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

[2] H. Gonzalez, J. Han, and X. Li, "Flowcube: Constructuing RFID flowcubes for multi-dimensional analysis of commodity flows," in *VLDB*, 2006, pp. 834–845.

[3] H. Gonzalez, J. Han, X. Li, and D. Klabjan, "Warehousing and analyzing massive RFID data sets," in *ICDE*, 2006, p. 83.

[4] C.-H. Lee and C.-W. Chung, "Efficient storage scheme and query processing for supply chain management using RFID," in *SIGMOD Conference*, 2008, pp. 291–302.

[5] P. Fuhrer and D. Guinard, "Building a smart hospital using RFID technologies," in *ECEH*, 2006, pp. 131–142.

[6] C. Ban, B. Hong, and D. Kim, "Time parameterized interval r-tree for tracing tags in RFID systems." in *DEXA'05*, 2005, pp. 503–513.

[7] 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*, 2005, pp. 1140–1151.

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

[9] K. E. Bite, "Improving on passenger and baggage processes at airports with RFID," pp. 121–136, 2010.

[10] Z. Cao, C. Sutton, Y. Diao, and P. J. Shenoy, "Distributed inference and query processing for RFID tracking and monitoring," *PVLDB*, vol. 4, no. 5, pp. 326–337, 2011.

[11] D. Roozbeh, O. Maria, and L. Xue, "RFID data management: Challenges and opportunities," in *IEEE International Conference on RFID 2007*, 2007, pp. 175–182.

[12] O. Mylyy, "RFID data management, aggregation and filtering," 2007.

[13] S. R. Jeffery, M. N. Garofalakis, and M. J. Franklin, "Adaptive cleaning for RFID data streams," in *VLDB*, 2006, pp. 163–174.

[14] H. Mahdin and J. H. Abawajy, "An approach to filtering duplicate RFID data streams," in *FGIT-UNESST*, 2010, pp. 125–133.

[15] Y. Bai, F. Wang, and P. Liu, "Efficiently filtering RFID data streams," in *CleanDB*, 2006.

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

[17] S. Tinggaard and R. L. Wejdling, "A data warehouse solution for flow analysis utilising sequential pattern mining," *Master Thesis, Aalborg University*, June 2009. [Online]. Available: http://projekter.aau.dk/projekter/files/17697709/d621a.pdf

[18] J. Rao, S. Doraiswamy, H. Thakkar, and L. S. Colby, "A deferred cleansing method for RFID data analytics," in *VLDB*. VLDB Endowment, 2006, pp. 175–186. [Online]. Available: http://dl.acm.org/citation.cfm?id=1182635.1164144

[19] C. S. Jensen, H. Lu, and B. Yang, "Graph model based indoor tracking," in *Mobile Data Management*, 2009, pp. 122–131.

[20] F. Wang and P. Liu, "Temporal management of RFID data," in *VLDB*, 2005, pp. 1128–1139.

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

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

Paper B

Handling False Negative in Indoor RFID Data

Asif Iqbal Baba, Hua Lu, Torben Bach Pedersen, Xike Xie

The paper has been published in the
Proceedings of15th IEEE MDM, pp. 117–126, 2014.

Abstract

The Radio-Frequency Identification (RFID) is a useful technology for object tracking and monitoring systems in indoor environments, e.g., airport baggage tracking. Nevertheless, the data produced by RFID tracking is inherently uncertain and contains errors. In order to support meaningful high-level applications including queries and analyses over RFID data, it is necessary to cleanse raw RFID data. In this paper, we focus on false negatives in raw indoor RFID tracking data. False negatives occur when a moving object passes the detection range of an RFID reader but the reader fails to produce any readings. We investigate the topology of indoor spaces as well as the deployment of RFID readers, and propose the transition probabilities that capture how likely objects move from one RFID reader to another. We organize such probabilities, together with the characteristics of indoor topology and RFID readers, into a probabilistic distance-aware graph. With the aid of this graph, we design algorithms to identify false negatives and recover missing information in indoor RFID tracking data. We evaluate the proposed cleansing approach using both real and synthetic datasets. The experimental results show that the approach is effective, efficient and scalable.

The layout has been revised.

1 Introduction

The Radio Frequency Identification (RFID) technology modernizes object tracking and monitoring systems in indoor environments, e.g., airport baggage tracking. Most RFID applications today use passive RFID tags that are preferred for their small size, cheap price, and low energy need. A typical RFID based system consists of tags each attached to an object (e.g., a bag in an airport) and readers each being able to detect the tags within a distance through RF communication. As a result, an RFID reader reports an 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 management of indoor RFID tracking data opens new doors for various applications that range from monitoring to analysis of indoor moving objects. However, the unreliable nature of raw data captured by readers is a major factor hindering the development of such applications. Under normal circumstances, it is quite often that the loss and error rate is between 30-40% [1]. RFID data errors come from different sources. The read events are frequently missed due to the detection ability of a reader, the quality of an RFID tag, and constraints of the environment [2]. Radio signals are not consistent and tend to change time to time, especially in indoor environments where signals get reflected and/or blocked by different entities and therefore readers sometimes may miss the tags nearby which are supposed to be detected.

Cleansing raw RFID data is necessary in order to support high-level business logic processing, e.g., queries and analyses. In this paper, we focus on *false negatives* in indoor RFID tracking data. False negatives occur when a reader fails to read out a tag in its detection range. A large number of tags within a reader's detection range are not read consistently due to either their distance from the reader, orientation with respect to reader, presence of metal, dielectric or water close to the tag and other factors. We will exploit spatio-temporal constraints of indoor spaces as well as the properties of deployed RFID readers to design a data cleansing technique to identify and recover such false negatives.

An example is shown in Figure B.1. There are two readers r_1 and r_2 in one hall and r_3 in another hall. An object o_1 enters the hall at time point t_1 and is continuously tracked by r_1 until time point t_3. After that, object o_1 is detected by r_3 at time point t_7 and r_3 keeps tracking o_1 until time point t_9. However, object o_1 is not supposed to be detected by r_3 before it is detected by reader r_2 on its way, because to enter into the hall where r_3 is, object o_1 must pass the detection range of r_1 and r_2 and cannot remain undetected for time interval $[t_3, t_7]$. By considering such indoor spatio-temporal constraints,

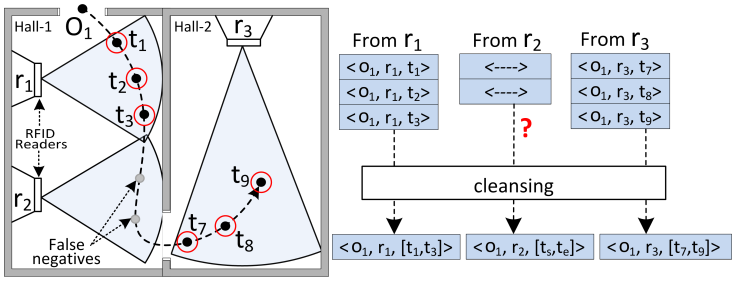


Fig. B.1: An example of RFID data cleansing

our cleansing technique will detect and fill in the missing information about the whereabouts of the object in the format of $(o_1, r_2, [t_s, t_e])$. Time points t_s and t_e will be determined by using indoor spatio-temporal aspects of the indoor space and the properties of deployed RFID readers.

We make the following contributions in this paper.

- We formulate the false negative handling problem for RFID data obtained in indoor spaces.
- With the aid of a probabilistic distance-aware graph model[1], we design algorithms to detect the occurrences of false negatives in indoor RFID data.
- Furthermore, we design an algorithm to recover tracking information, whereabouts of an moving object, for the identified false negatives in indoor RFID data.
- We conduct extensive experimental studies to evaluate our false negative cleansing technique. The experimental results demonstrate that our cleansing techniques are effective, efficient and scalable.

The rest of this paper is organized as follows. Section 2 presents the preliminaries. Section 2 details the probabilistic distance-aware graph model. Section 3 proposes the algorithms for false negative handling. Section 4 reports on extensive experiments. Section 6 reviews the related works. Finally, Section 7 concludes the paper.

[1]The graph model was briefly introduce in our previous work [**?**]. We include the definitions, and give detailed examples in Section 2.

2 Preliminaries

2.1 Symbolic Indoor Tracking

We describe symbolic indoor positioning and tracking using the example floor plan shown in Figure B.2. The indoor space is partitioned into several rooms, corridors and a staircase, each having its own number. Each room has one or more doors. A number of RFID readers are placed at the doors, where the numbered circles represent the deployed RFID readers and their detection range. Each indoor partition semantically represents an independent piece of space that is connected with each other through one or several doors.

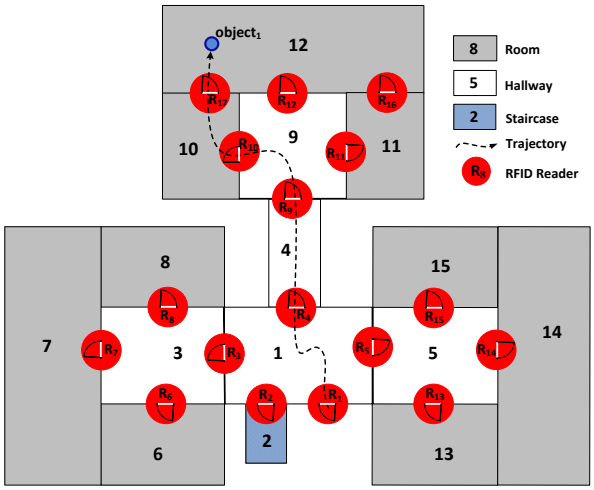


Fig. B.2: Example Floor Plan

Generally most of the indoor tracking technologies employ the proximity analysis, i.e, object is only seen when it is within the detection range of a tracking 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 RFID reader continuously detects object in its range and generate reports with the frequency determined by its sampling rate. 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. All the positioning devices in an indoor space generate large amount of raw readings. To illustrate, a movement of an object $object_1$ through the floor plan example shown in Figure B.2, a set of raw readings generated about $object_1$ is shown in Table B.1.

We used a mechanism proposed in [3] to merge the raw readings into

Table B.1: Raw Readings Table

deviceID	objectID	t
r_1	$object_1$	t_0
r_1	$object_1$	t_1
r_1	$object_1$	t_2
r_1	$object_1$	t_3
r_1	$object_1$	t_7
r_1	$object_1$	t_8
r_4	$object_1$	t_{12}
r_4	$object_1$	t_{13}
r_4	$object_1$	t_{15}
r_4	$object_1$	t_{16}
r_9	$object_1$	t_{20}
r_9	$object_1$	t_{21}
r_9	$object_1$	t_{22}
r_9	$object_1$	t_{23}
r_{17}	$object_1$	t_{31}
r_{17}	$object_1$	t_{32}
r_{17}	$object_1$	t_{33}
r_{17}	$object_1$	t_{37}

more meaningful tracking records. The raw reading are sequentially scanned from the raw reading table to generate tracking records by aggregating on the time. A threshold τ is used to control the aggregation results. Two consecutive readings of an object are merged together if they are apart from each other for a period less than τ, otherwise, they will be put into two different tracking records. We store all generated tracking records in *Aggregate Tracking Table* (ATT). Each generated tracking record is in the format of (*deviceID, objectID,* t_s*,* t_e*, r_Count*), which means object identified by *objectID* is detected by device identified by *deviceID* during time interval [t_s,t_e] *r_Count* times.

Consider all the raw readings about $object_1$ from Table B.1, and suppose that threshold τ is 3 time units. The temporal cleansing [3] on all those readings will generate the aggregate tracking table as shown in Table B.2. For example, tracking record $(r_1, object_1, t_0, t_3, 4)$ means that $object_1$ is reported 4 times by a reader r_1 from time t_0 to t_3. Without loosing any information we also keep the count of the readings by each reader. The count will be used to fill in the missing readings.

Table B.2: Aggregate Tracking Table (ATT)

deviceID	objectID	t_s	t_e	r_Count
r_1	$object_1$	t_0	t_3	4
r_1	$object_1$	t_7	t_8	2
r_4	$object_1$	t_{12}	t_{16}	4
r_9	$object_1$	t_{20}	t_{23}	4
r_{17}	$object_1$	t_{31}	t_{33}	3
r_{17}	$object_1$	t_{37}	t_{37}	1

2.2 Problem Formulation

In this section we formulate the problem of false negatives in indoor RFID tracking data. Table B.3 lists the notation used throughout the paper.

Table B.3: Notation

Notation	Meaning
r_i, r_j, R_i, R_j	RFID readers
R_s, R_d	Source reader and destination reader
rr, rr_i, rr_j	Raw RFID readings
tr, tr'	tracking records
RRT	Raw reading table
ATT	Aggregate tracking table
G_{pdm}	Probabilistic distance-aware graph model
dt_i	The minimum dwell time of reader r_i
S_{f_i}	a sampling frequency of reader r_i
p_{ij}	a transition probability between r_i and r_j

Without loss of generality, we assume that all raw readings are ordered by their detection times and are aggregated into tracking records stored in Aggregate Tracking Table (ATT). We formally define other concepts as follows:

Definition 6
(**False Negatives**) Given a reader r_i and a moving object O, if object O goes through the detection range of reader r_i during time interval $[t, t']$, but r_i does not generate any reading about O's presence during time interval $[t, t']$, a false negative occurs in the data.

Definition 7
(**False Negative Handling**) Given an Aggregated Tracking Table (ATT), the false negative handling process detects false negatives and inserts recovered tracking record into the ATT.

In this paper our main task has two parts, one is to detect the occurrences of the false negatives in the data, and secondly to recover false negatives (missing information).

3 Probabilistic Distance-Aware Graph Model

We propose a probabilistic distance-aware graph model G_{pdm} in this section. The graph model constructed captures the deployed reader information and

enable deriving the minimum travel time and most probable path from one reader to another. The basic idea is to model the readers as graph vertices and assign to each vertex a corresponding reader's information such as detection range and sampling rate. The minimum travel times required to move from one reader to another is captured in the graph edge between them, the edge also captures the transitional probability, which captures how likely a moving objects moves from one reader to another.

3.1 Transition Probabilities

At each step a move to next reader (vertice) only depends on current reader and not on a number of readers that precede it, thus holding a Markov property of "memorylessness". A moving object at reader r_i, moves to a neighboring reader r_j with a probability proportional to the probability weight of the edge (r_i,r_j), that is, the probability of a transition from r_i to neighboring r_j is:

$$\frac{p(r_i, r_j)}{\sum_{k \in N(r_i)} p(r_i, k)} \tag{B.1}$$

where, $N(r_i)$ is the set of neighbors of reader r_i. The transitional probability $p(r_i, r_j)$ denotes the probability that a moving object is detected by r_j at some time later after t given that it was detected by r_i at time t without any third reader involved in-between.

Definition 8
(**Transition**) A transition takes place when a moving object moves from a reader r_i to another reader r_j without passing through any intermediate readers.

When a finite number of readers, $r_1, r_2, r_3 ... r_n$ are deployed, a probabilistic matrix is used to model them probabilistically:

$$P = \begin{bmatrix} p(r_1, r_1) & p(r_1, r_2) & \cdots & p(r_1, r_n) \\ p(r_2, r_1) & p(r_2, r_2) & \cdots & p(r_2, r_n) \\ \vdots & \vdots & \vdots & \vdots \\ p(r_n, r_1) & p(r_n, r_2) & \cdots & p(r_n, r_n) \end{bmatrix} \tag{B.2}$$

For each deployed reader r_i, $\sum_{j=1}^{n} p(r_i, r_j) = 1$. The transitional probabilities are recorded on a historical data generated by deployed readers. The elements of the transition matrix P are then obtained as follows:

$$p(r_i, r_j) = \begin{cases} \frac{N_{ij}}{N_i}, & \text{if } (r_i, r_j) \in G_{pdm}.\text{E}; \\ 0, & \text{otherwise.} \end{cases} \tag{B.3}$$

Above,

- N_{ij} is the number of objects moving from r_i to r_j

- N_i is the total number of objects moving from r_i.

The generated path with a random walk consists of edges which correspond to first entrance to an arbitrary vertex in a graph G_{pdm}, other than starting vertex.

3.2 Graph Model

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

1. V is a set of vertices. Each vertex $v_i \in V$ represents a deployed reader r_i.

2. E is the set of edges, where $E = \{(r_i, r_j) \mid r_i, r_j \in V \wedge r_i \neq r_j\}$. One can move from r_i to r_j without being detected by any other reader in between.

3. $\mathcal{L}_V : V \rightarrow \mathcal{R} \times \mathcal{R}$ assigns two values to each vertex v_i , a corresponding reader r_i's minimum dwell time and sampling frequency. Specifically, $\mathcal{L}_V(r_i) = (d_t, S_f)$.

4. $\mathcal{L}_E : E \rightarrow \mathcal{R} \times \mathcal{R}$ assigns two values to an edge (r_i, r_j) the minimum travel time between two readers r_i and r_j and the transition probability. Specifically, $\mathcal{L}_E(r_i, r_j) = (tt_{i,j}, p_{r_i,r_j})$.

A probabilistic distance-aware graph model G_{pdm} is enhanced from a distance deployment graph proposed in [3]. An example probabilistic distance-aware graph of corresponding floor plan in Figure B.2 is shown in Figure B.3. The choice of weights captured by the graph is justified by practical needs. The deployed RFID readers (or any other positioning device) usually imply different minimum dwell times and sampling rate. However, it is also possible that a minimum dwell time or sampling rate of a particular reader may be changed due either by tuning its parameters or by surroundings. Our design of vertex weights are flexible and can support such differences and possible changes.

The travel time between two readers is not solely depend on the distance between them. In order to determine the travel time exactly, the moving speed with which objects are moving. For example, the speed of the conveyor belt system in an airport keep changing with the baggage traffic loads. Therefore, the travel time between readers monitoring the belt is not merely

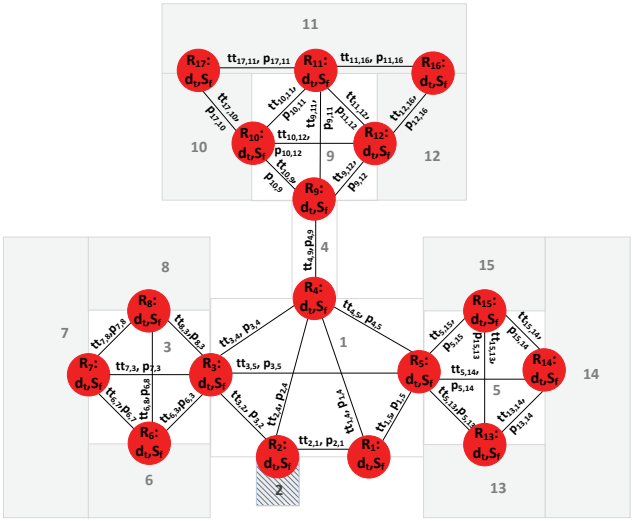


Fig. B.3: Example of probabilistic distance-aware graph model

determined by the distance but also by the speed of the conveyor 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 having travel time weight (determined by both distance and speed) on each edge is necessary to support such applications. A moving object can move to any of the available connected paths.The traffic load of each of the outgoing paths from a particular location is presented as transition probabilities, we capture such probability in a transition matrix. The probability weight captured by each edge will be used to predict the most likely path an object may have taken. We also capture the read rate (sampling frequency) of each reader along with minimum dwell time on each vertex. This information will be used to find the exact count of readings.

4 Handling False Negatives

The false negative handling process takes aggregated raw data, detects the occurrences of false negatives in it and then subsequently recovers them by filling the missed information one by one. Once the false negative(s) are detected, we look to find the paths between source reader R_s and destination reader R_d. We require paths to be simple (loop-free). To further understand the process we need following definitions:

Definition 9
(**Path**) A path δ is a sequence of readers r_1, r_2, r_3, ..., $r_{|\delta|}$, where there is an

edge connecting r_i and r_{i+1} for $i = 1, 2, ..., n$. A path is simple if all r_i are all distinct.

Definition 10
(**Candidate Path**) Given a source reader R_s and a destination reader R_d, a path from R_s to R_d, represented as $R_s \overset{\delta}{\rightsquigarrow} R_d$ is a candidate path if it satisfies the spatio-temporal constraints captured by a graph.

Definition 11
(**Most Likely Path**) Given a set of candidate paths $\{\delta_m\}$, a most likely path is:

$$\delta^* = \underset{m}{\text{argmax}} \prod_{E_{ij} \in \delta_m} p_{i,j}$$

4.1 Topological Scenarios

Before going directly into false negative cleansing it is important to understand different underlying deployed reader topological scenarios. To identify false negatives in indoor RFID tracking data, we use a probabilistic distance-aware graph model which captures the reader deployment information. Some of the possible topological scenarios are shown in Figure B.4. Each scenario represent a sub-graph of a deployed graph, where R_s, R_d, R_i, R_j and R_n are the deployed readers. More specifically, we discuss several scenarios here to better explain how our proposed probabilistic distance-aware graph model is used to find a path between two readers:

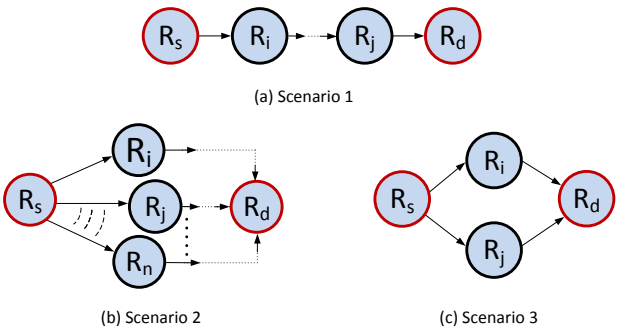


Fig. B.4: Possible topological scenarios

Scenario 1: Figure B.4(a) represents a simple case, where a single path from source reader R_s goes to a destination reader R_d. Source reader R_s has only one out going edge to R_i and destination reader R_d has one incoming

edge from R_j. Between R_i and R_j there can be none or many readers deployed all connected in linear fashion. Therefore, it is to easy to find that the next expected reader from source reader R_s, then subsequently from each successor readers until destination reader R_d is reached. Here the probabilities captured by the edges are not relevant. If from the data we have a tracking record generated by R_s and next tracking record is not of a reader R_i, we simply conclude that R_i has failed to detect the object, thus forming a false negative. We can fill the missing information with 100% certainty, as the probability of the edge (R_s, R_i) is 1. Similarly, we can detect if any other reader failed to detect an object and fill the missing readings. It is also possible that we only have tracking record of R_s and then a tracking record from R_d, which means all the deployed readers between R_s and R_d have failed to detect a moving object. In this case we can find whole path between R_s and R_d and fill the missing readings of each reader using a graph model G_{pdm}. Both these cases are depicted in Figure B.5(a) and Figure B.5(b) respectively.

Scenario 2: Figure B.4(b) is a case where source and destination reader have several paths in between them. Here source reader R_s have many outgoing edges to R_i, R_j to R_n. The outgoing paths seems to go independent until they reach R_d. In this case object can move to any of the connected paths to reach R_d from R_s. Here, we can find the all candidate paths using the spatio-temporal constraints captured by a graph. Once the candidates paths are found, transitions probabilities captured by each edge of graph can be used to choose the most probable path to reach destination reader R_d. If from the data a next tracking record after a tracking record of R_s is not from any of the neighbors $N(R_s)$, it is difficult to say that which reader on which outgoing path have failed to detect the moving object without traversing the each path until the R_d is reached. Again both cases depicted in Figure B.5(a) and Figure B.5(b) are possible, it could be just one missing reader or many in between R_s and R_d.

Scenario 3: Figure B.4(c) is a sub-case of previous case where source reader R_s and destination reader R_d have two paths between them. Both paths satisfies the spatio-temporal constraints equally, making it difficult to directly find a first candidate path. A most likely path will finally be decided on the transition probabilities the edges of a graph capture. If from the data the next tracking record after tracking record of reader R_s is not from either R_i or R_j, we use a transition probability weights on the edges to choose a next node. However, R_d can be reached through both reader R_i and R_j, an edge with highest probability will always be chosen.

4.2 False Negatives Cleansing

We used the information captured by a probabilistic distance-aware graph model (G_{pdm}) to handle the false negatives in the indoor RFID tracking data.

We will present detecting false negative algorithm in Section 4.2, followed by the false negative recovery algorithm in Section 4.2.

Detecting False Negatives

The false negative detection procedure is shown in Algorithm 4. The algorithm take the Aggregated Tracking Table (ATT) and a probabilistic distance-aware graph model (G_{pdm}) as an input.

Algorithm 4 DetectingFalseNegatives (Aggregate tracking table ATT, Probabilistic distance-aware graph model G_{pdm})

```
1:  path ← ∅; neighbors ← ∅;
2:  for each tr in ATT do
3:      neighbors ← getNeighbors(tr.deviceID , G_pdm)
4:      if (neighbors.length > 0) then
5:          tr' ← the next record in ATT such that (tr.objectID
                = tr'.objectID)
6:          if (tr' = null) then
7:              continue
8:          if (tr'.deviceID ∈ getNeighbors(tr.deviceID, G_pdm)) then
9:              continue
10:         else
11:             path←findPath(G_pdm, tr, tr')
12:             prev_t_e ← tr.t_e
13:             FalseNegativeRecovery(prev_t_e, path)
```

Considering the first reader always reads the object, therefore we always have first tracking record of a moving object. For each tracking record tr in ATT, the false negative cleansing works as follows. We first find out all the outgoing neighbors of the first tracking device ($tr.deviceID$) (line 3). At line 4, we check if the current device $tr.deviceID$ is not the last device. Next we get tr's next tracking record tr' from ATT that involves the same object as tr (line 5). We assume that $tr.deviceID$ is current valid reader since it appears before tr'. We first check if the next tracking record (tr') is $null$ (line 6). If next tracking record is null, that is, we do not have any notion of destination reader, we simply continue with another object. Next we check, if there is a next tracking record (tr'). If there is and $tr'.deviceID$ is equal to one of the neighbors of current tracking reader $tr.deviceID$, we simply continue the process (lines 8-9). Otherwise there is false negative(s), we will use Algorithm 5 to find a most likely path in a graph G_{dpm} between current tracking reader $tr.deviceID$ and next tracking reader $tr'.deviceID$ (line 10). Once the path is retrieved, we will used Algorithm 7 to fill in the missing readings of each reader in the path.

Referring to the example aggregate tacking table shown in Table B.2, we detected false negative in the data using a graph model G_{pdm}. Result of this phase is shown in Table B.4.

Table B.4: Result of False Negative Detection Phase

deviceID	objectID	t_s	t_e	r_Count
r_1	$object_1$	t_0	t_3	4
r_1	$object_1$	t_7	t_8	2
r_4	$object_1$	t_{13}	t_{16}	4
r_9	$object_1$	t_{20}	t_{23}	4
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
r_{17}	$object_1$	t_{31}	t_{33}	3
r_{17}	$object_1$	t_{37}	t_{37}	1

The Algorithm 5 presents the procedure $findpath()$, which takes a probabilistic distance-aware graph model G_{pdm} and two tracking records tr and tr' as an input. The algorithm calls the *findAllPaths* procedure shown in Algorithm 10 to get all the possible paths between source reader R_s and destination reader R_d using a graph G_{pdm}. Both R_s and R_d are extracted from tracking records tr and tr' respectively at line 2 and 3 of Algorithm 5. To check if the path satisfies the temporal constraint, a total traverse time (T_{cal}) is calculated (lines 5-15). A total traverse time T_{cal} is a time a moving object takes to traverse a path from R_s to R_d. A minimum travel time mtt between the readers (line 12) and the minimum dwell time mdt (line 11) of each corresponding reader is used to calculate the T_{cal}. If the total traverse time of a path is larger than the time object was first tracked the destination reader R_d, the path is treated as invalid and is deleted from the set of candidate paths (line 17). A most likely path is then estimated based on the maximum likelihood decision rule, Here transition probabilities are used as maximum likelihoods. A most likely path is returned at line 20.

Algorithm 5 findPath(Probabilistic distance-aware graph model G_{pdm}, Tracking record tr, Tracking record tr')

$neighbors \leftarrow \emptyset$; $path \leftarrow 0$; $paths[] \leftarrow \emptyset$;
$R_s \leftarrow$ tr.deviceID
$R_d \leftarrow$ tr$'$.deviceID
$findAllPaths(G_{pdm}, R_s, R_d, paths)$
for each path p in $paths$ **do**
    $T_{cal} = 0$
    for each reader R in path p **do**
        $R' \leftarrow$ the next reader in $path$
        if $(R' = R_d)$ **then**
            $mtt \leftarrow G_{edd}.\mathcal{L}_E(R, R').tt$
            $mdt \leftarrow G_{edd}.\mathcal{L}_V(R').dt$
            $T_{cal} = T_{cal} + mtt + mdt$
            break
        $mtt \leftarrow G_{edd}.\mathcal{L}_E(R, R').tt$
        $T_{cal} = T_{cal} + mtt$
    if ($tr.t_e + T_{cal} > tr'.t_s$) **then**
        delete p from $paths$
for (all paths $\{\delta_m\}_{m=1}^{n}$ in $paths$) **do**
    $path = \underset{m}{\text{argmax}} \prod_{E_{ij} \in \delta_m} p_{i,j}$
return $path$

In Algorithm 10 the procedure *findAllPaths()* is shown, which takes a graph G_{pdm}, source reader R_s and destination R_d as an input and returns all the paths between R_s and R_d. The algorithm follows an improved depth-first search paradigm, which estimates all the possible paths between two readers. The algorithm is typical backtracking algorithm.

Algorithm 6 findAllPaths(Probabilistic distance-aware graph model G_{pdm}, Source reader R_s, Destination reader R_d)

stack $S \leftarrow \emptyset$; $neighbors \leftarrow \emptyset$; $paths \leftarrow \emptyset$;
$S.push(R_s)$
if $(curr = R_d)$ **then**
    $paths[] \leftarrow S$
$neighbors \leftarrow getNeighbors(R_s, G_{pdm})$
for each n in $neighbors$ **do**
    if (n not in stack S) **then**
        $findAllPaths(G_{pdm}, n, R_d)$
$S.pop()$

Taking data shown in Table B.4 as an example, suppose we choose r_9 as

source reader R_s and r_{17} as destination reader R_d. Looking at at the floor plan shown in Figure B.2, we found following set of possible paths between source reader r_9 and destination reader r_{17}, which are further further pruned using a spatio-temporal constraints imposed by a graph.

$$r_9 \rightsquigarrow r_{17} = \left\{ \begin{array}{c} r_9 \rightarrow r_{10} \rightarrow r_{17}, \\ r_9 \rightarrow r_{12} \rightarrow r_{17}, \\ r_9 \rightarrow r_{11} \rightarrow r_{16} \rightarrow r_{17}, \\ \vdots \end{array} \right\}$$

After getting a set of candidate paths between r_9 to r_{17}, path $r_9 \rightarrow r_{10} \rightarrow r_{17}$ is assumed to a most likely path.

Recovering False Negatives

The false negative recovery procedure is shown in Algorithm 7. It takes a most likely path *path* and a last timestamp ($prev\_t_e$) a moving object was detected before false negative occurred as input. A *path* contains all the readers between source reader R_s and destination reader R_d.

Algorithm 7 FalseNegativeRecovery(Previous Timestamp $prev\_t_e$, Most Likely path *path*)

1: $mtt \leftarrow 0$; $mdt \leftarrow 0$; $samp\_rate \leftarrow 0$;
2: **for** each N in *path* **do**
3: $N' \leftarrow$ the next element in *path*
4: **if** $(N' \neq tr'.deviceID)$ **then**
5: $mtt \leftarrow G_{edd}.\mathcal{L}_E(N, N').tt$
6: $mdt \leftarrow G_{edd}.\mathcal{L}_V(N').dt$
7: $samp\_rate \leftarrow 1/G_{edd}.\mathcal{L}_V(N').S_f$
8: **insert**$(N'$, $tr.objectID$, $prev\_t_e + mtt$, $prev\_t_e + mtt + mdt$, $\frac{mdt}{samp\_rate})$
9: $prev\_t_e \leftarrow prev\_t_e + mtt+mdt$

We fill the missing information of each reader in the path one by one by retrieved a minimum traveling time between the current readers and next reader in the path (line 5). We also retrieved the minimum dwell time of a corresponding reader (line 6). This information is then used to fill in exact starting and end time of newly filled tracking records. We fill the exact number of readings a reader can generate using a sampling rate and the minimum dwell time of a corresponding reader.

Referring to the running example shown in Table B.4, we assume after calculating the probabilities, the path $r_9 \rightarrow r_{10} \rightarrow r_{17}$ is most likely path. Assuming minimum traveling time between device r_9 and r_{10} is 2s, a minimum

dwell time (d_t) of device r_{10} is 4s and a sampling frequency (S_f) is 1/s, i.e., one reading cycle per second. A new tracking record (r_{10}, $object_1$, t_{26}, t_{29}, 4) is added to set of tracking records of $object_1$. The reading count is calculated as $\frac{\text{min. dwell time}}{\text{sampling interval}} = \frac{4s}{1s} = 4$. Here, sampling interval = $\frac{1}{S_f}$. The result is shown in Table B.5.

Table B.5: Result of False negative cleansing

deviceID	objectID	t_s	t_e	r_Count
r_1	$object_1$	t_0	t_3	4
r_1	$object_1$	t_7	t_8	2
r_4	$object_1$	t_{13}	t_{16}	4
r_9	$object_1$	t_{20}	t_{23}	4
r_{10}	$object_1$	t_{26}	t_{29}	4
r_{17}	$object_1$	t_{31}	t_{33}	5
r_{17}	$object_1$	t_{37}	t_{37}	1

In the example above, we assume that the first tracking record in ATT is always recorded and we make the false negative cleansing check on each subsequent tracking record in ATT with respect to previous tracking records. 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 missed. As a matter of fact, our false negative 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 false negative 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"* with high precision rate may be placed at an airplane and they thus yield known correct ending records.

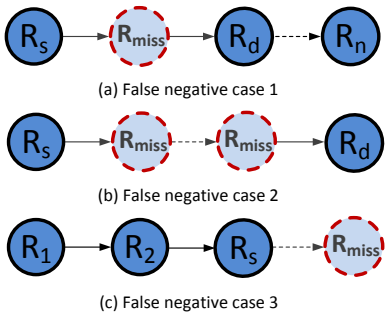


Fig. B.5: False negative cases

We illustrate false negative cases in Figure B.5. The case shown in Figure B.5(a) is a simple case where the information of both readers, a source reader (R_s) and destination reader (R_d) is present and there is only one reader (R_{miss}) which fails to detect a moving object. We use the information of R_s and R_d captured by a graph G_{pdm} to fill in the missing information of reader R_{miss}. The case shown in Figure B.5(b) illustrates the case when more than one readers in between R_s and R_d fail to detect a moving object. We start from R_s and finds the most likely path a moving object may have taken to reach destination reader R_d and fill in the missing information by using the information captured by a graph G_{pdm}. The case shown in Figure B.5(c) illustrates a case where a moving object is tracked by all the reader from first reader R_1 till R_{n-1}, R_{n-1} is represented by R_s in Figure B.5(c). The last reader(s) R_{miss} somehow fails to detect the object. Without any further evidence of object movement, it is hard to determine the final destination of a moving object. Therefore, filling any new data by any workaround solution may give rise to false positives. All the three cases may possibly occur several times during a travel from source to destination.

5 Experimental Study

In this section, we report the results from the experiments we conducted to evaluate our devised false negative cleansing techniques for indoor RFID data. A computer with a 64-bit Windows 7, a 2.8GHz core i7 processor, and 8GB main memory is used to run all the experiments, which were implemented in C++.

5.1 Experimental Setup

We describe the performance evaluation metrics used to evaluate our experiments followed by the description of the datasets used.

Metrics

We consider two performance metrics: efficiency and effectiveness. The efficiency is measured by counting clock time.

The effectiveness is measured by recovery rate, which is defined as:

$$\text{recovery rate} = \frac{\text{\# of recovered records}}{\text{\# of missing records}}$$

The number of recovered records in the numerator represents the number of tracking records recovered by our cleansing process. Specifically, the recovered records for false negative cleansing is $|ATT'| - |ATT|$ where ATT' refers to the output of false negative recovery method shown in (Algorithm 7).

We further measure the effectiveness of our false negative cleansing algorithm by counting the number of trajectories free of false negatives before and after cleansing.

Datasets

For our experimental evaluation we used both synthetic and real data. The parameter settings are shown in Table D.4. The default values are shown in bold.

Synthetic Data. We generated the synthetic data for a floor plan with 85 rooms. For simplicity we regard hallways and staircases as rooms, and staircase entrances as doors. We vary the number of parameters like number of objects, detection range etc. By default, there are 5000 RFID tagged-objects moving inside the building. Each object movement follows a random waypoint model [4] with a constant speed of 4km/h. To generate object movements, an object in a room can move inside the room or move to any other room chosen randomly and then move along a shortest path from the source location to the destination location. While moving from current room to destination room, an object can be detected by one or more RFID readers deployed at each door with a default detection range of 3m. The false negatives are introduced by partially or fully muting a randomly selected reader(s) for a randomly selected set of objects depending on the false negative rate, which is defined as $\frac{\text{\# of false negative records}}{\text{total \# of records}}$. While generating the synthetic RFID data, we also generated ground truth by recording all the readings of the moving objects with 100% accuracy i.e., each deployed reader detects an object on each reading cycle and generates a reading.

Table B.6: Experimental parameters

Parameters	**Settings**
Detection range:	1m, **3m**, 5m
Object number:	1000, 2000, **5000**, 10000
False negative rate:	50%, 40%, **30%**, 20%, 10%
Threshold (Syn.):	5s, 10s, **15s**, 20s
Threshold (Real):	5s, 10s, 15s, 20s, **25s**, 30s, 35s, 40s, 45s, 50s

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 B.6, the baggage handling system features a number of specific semantic locations (e.g., check-in desks) and RFID readers. During the check-in phase, each (randomly selected) bag is attached with a passive RFID tag which is to be detected by the deployed RFID readers. From check-in

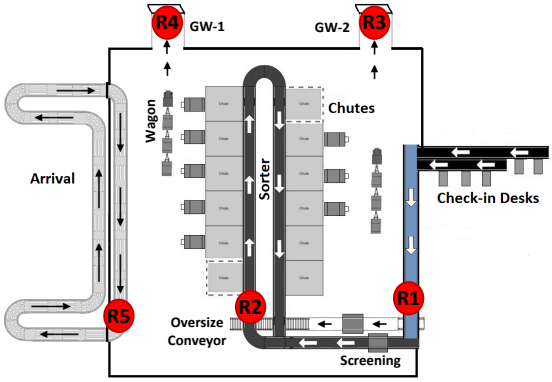


Fig. B.6: RFID-based baggage handling system at Aalborg Airport

desks, bags pass the screening area through conveyor belts. After a successful security screening, the bags enter the main sorting area where the sorting system ensures that the bag is pushed into a designated chute. The bags 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 through gateway GW-1 where baggage handling staff unload the bags from wagons on to the arrival conveyor belt.

We have continuously recorded the data for two consecutive months. We collect more than half a million raw reading records for about 20000 RFID tagged bags going from Aalborg to Copenhagen airport. The real data used for our experiments were obtained through the baggage handling system installed by Lyngsoe Systems.

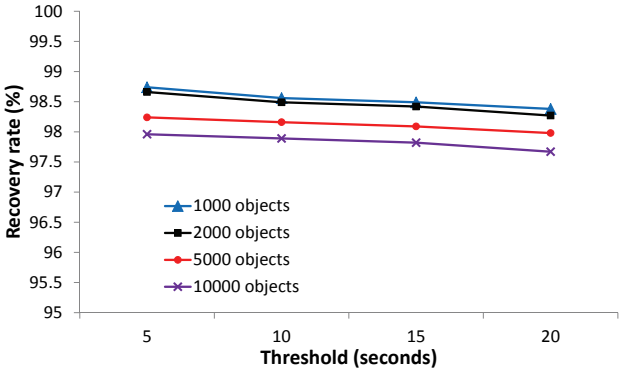
5.2 Experimental Results

In this section we present the experimental results on synthetic data followed by the results on real data. In the end we also present the results of probability distance-aware graph construction.

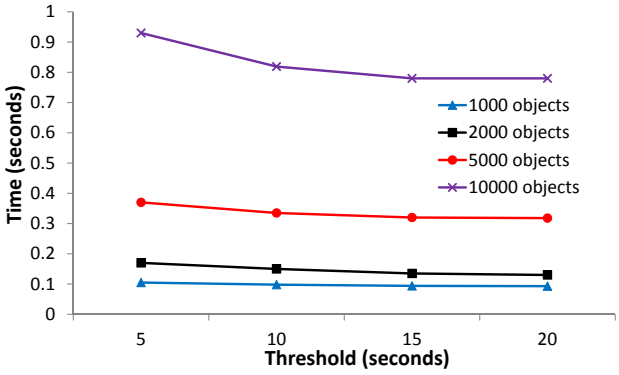
We do not provide comparisons of our proposal with existing, related proposals (see Section 6) because these are all focused on improving the tracking accuracy at either the hardware or the middleware level. This is orthogonal to our focus that simply assumes a given set of tracking data.

Results on Synthetic Data

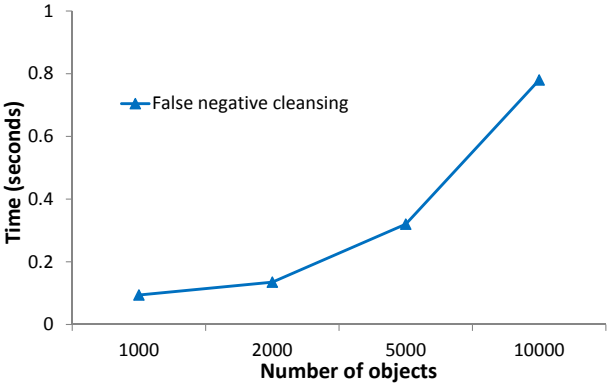
In the following experiment, we use the data about different set of objects 1K, 2K, 5K and 10K. The raw data of each set is aggregated into tracking records using different threshold 5, 10, 15 and 20 seconds. The results on the false



(a) Cleansing effectiveness vs. Threshold



(b) Cleansing efficiency vs. Threshold



(c) Cleansing efficiency vs. Number of objects

Fig. B.7: False negative cleansing on synthetic data

negative cleansing effectiveness are reported in Figure B.7(a). For each set of data, the recovery rate is quite significant on almost each threshold value. The recovery rate here is based on the total number of recovered during the cleansing process. Result shows that around 97-99% of false negatives were recovered. The recovery rate seems to slightly decrease with larger datasets and with large threshold values. A larger datasets tends to have more more missing records, therefore the probability of having higher number of missing records which can not be recovered. However, the slight decrease with higher threshold values does not have any apparent reason.

Results about false negative cleansing efficiency are reported in Figure B.7(b). The algorithm achieves better efficiency at a higher threshold. The reason is that fewer tracking records are generated by aggregation using larger thresholds. The cleaning algorithm takes less than 1s for handling a data of 10000 moving objects (about 1M records), also scales well with respect to the varied thresholds.

Figure B.7(c) reports the results on the efficiency with respect to the number of objects for false negative cleansing algorithm. In this experiment we fixed the threshold value to 15s to aggregate the data. Algorithm take less than 1 second for the largest testing dataset. The running time for false negative cleansing algorithm do not increase too fast. It is efficient because the complexity is lower. Aggregated data tend to get much smaller in size than that of a raw data.

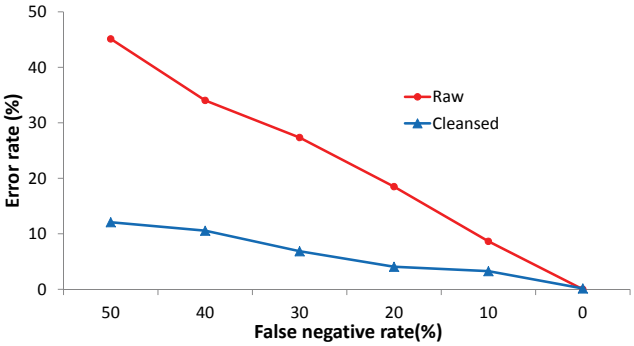
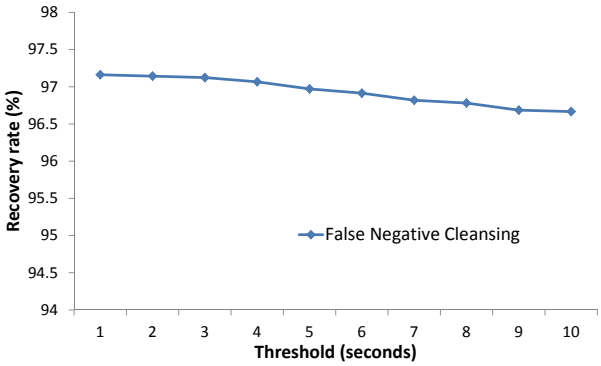
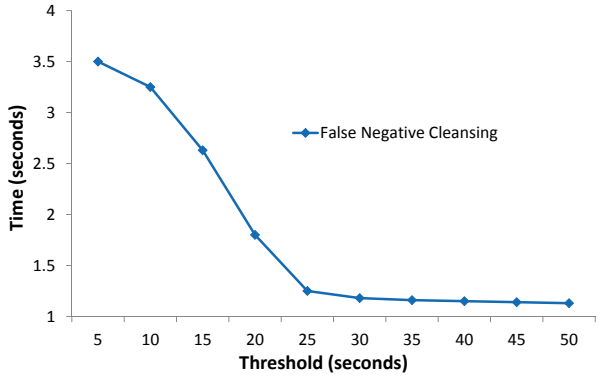


Fig. B.8: Error rate vs. False negative rates

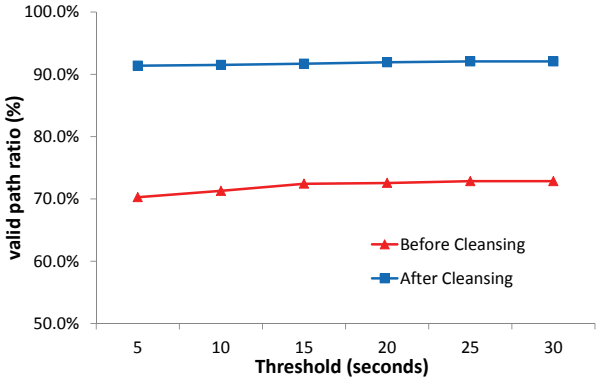
In Figure B.8, we present the result on the effectiveness of our false negative cleansing algorithm with respect to the false negative rate. The false negative rate represents the amount of false negatives present in the generated RFID data. All the missing readings are counted as false negative. The error rate is define as $\frac{\text{\# of false negatives}}{\text{\# of total readings}}$. The error rate is directly dependent on the amount of false negatives in the data, higher false negative rate on the other hand directly reflects to the reading accuracy of the readers. Our false



(a) Cleansing effectiveness vs. Threshold



(b) Cleansing efficiency vs. Threshold



(c) Valid path evaluation on real data

Fig. B.9: False negative cleansing on real data

negative cleansing algorithm however managed to decrease the error rate substantially even when the reading accuracy is very low i.e., when the false negative rate is high. With the false negative rate high as 50%, our algorithm still manages to bring error rate down to 10%, thus signifies the effectiveness our false negative cleansing algorithm.

Results on Real Data

We also use the real dataset from Aalborg Airport to test both the effectiveness and efficiency of our proposed false negative cleansing algorithms. The results in Figure B.9(a) reports effectiveness of false negative cleansing on real data. The recovery rate in real data is also significantly on higher side. The recovery rate is almost 97% on each threshold value, which is relatively slightly lower than that of synthetic data. The reason behind this is the large amount of moving objects (bags in this case) are not detected by the last deployed reader i.e., $R4$ and $R3$ in Figure C.23. As mentioned earlier the bags are loaded into a wagons which then passes through the gateway readers $R3$ and $R4$. The probability of not detecting a bag is higher when they are moved together than moving individually.

We measure the efficiency of false negative cleansing on real data. In Figure B.9(b), we report the result of false negative cleansing. With the increase of threshold the processing time of false negative cleansing algorithm decreases. The trend directs reflects that more data to be cleansed, more time is required. However, it may be noticed that our algorithm is still faster even when the data is least aggregated (smaller threshold).

We also test the effectiveness of false negative cleansing on real data by estimating the valid path ratio which is $\frac{\text{(total \# of trajectories) - (\# of invalid trajectories)}}{\text{total \# of trajectories}}$. The results are reported in Figure B.9(c). The result shows that before cleansing more than 30% paths have false negatives, thus are counted as invlaid. However, after false negative cleansing, more than 93% of trajectories are valid. This again demonstrates the effectiveness of our false negative cleansing technique.

We investigate the mechanism behind our proposed false negative cleansing, we used a raw RFID data trace of a single moving object with RFID tag attached. The readings are generated with a simulator with realistic parameter. The cleansing of these readings are challenging as the data is highly unreliable.

In Figure B.10, a time line of a tracking history of a moving object is shown. The reality shown in the figure represents the ground truth (e.g, the readings produced by 100% perfect reader). The raw trace shown in figure shows some of the readers failed to detected the object. During time interval $[t_3, t_4]$ and $[t_{11}, t_{12}]$, moving object remain completely undetected, and partially detected during time interval $[t_5, t_6]$, $[t_7, t_8]$ and t_9, t_{10}. A false

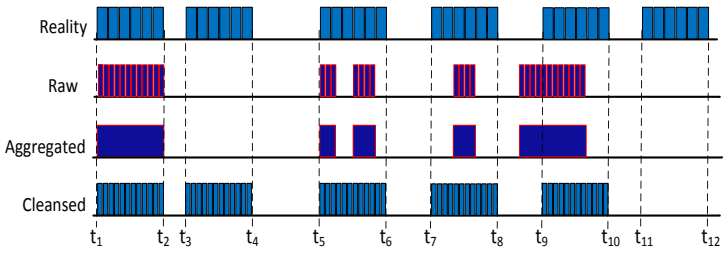


Fig. B.10: False negative cleansing of a single tag data trace

cleansing trace shows how our false negative cleansing algorithm fills in all the missing readings of using a spatio-temporal constraints and the transition probabilities captured by a distance probabilistic graph G_{pdm}. In cleansed trace we can see our algorithm could not recovered the readings of last reader because it is hard to detect the false negative if there is no future evidence that the object might have actually passed through the last reader, therefore can not be recovered back. On the whole our algorithm recovers the false negatives in all the cases except a case when there is no future evidence.

6 Related Work

Data cleansing is a key requirement of RFID middleware. Existing proposals in RFID data cleansing target the filtering of false negatives. To reduce such errors, basic filter are incorporated in RFID middleware solutions [1] [**?**]. The common solution applied to suppress these errors are low pass filters commonly called as smoothing filters.

In [6], Mylyy 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 false negatives from the data stream. Jeffery et al. [7] propose smooth RFID data in temporal sliding window and also group several readers in a spatial granule to correct false negatives and remove outliers. However, it is difficult to decide the best window size for varying RFID data especially in the mobile environments. In such environments it is important to ensure balance between two application requirements, *Completeness*: to ensure that all tags are reader in the reader's detection range and *Tag Dynamics*: to capture the dynamics of tag movement in and our from the reader's detection range. The completeness is ensured by setting large window sizes to smooth out the false negatives from a data, but they are not efficient in detecting tag transitions and also introduces false positives. On the other hand small window sizes are able to detect transitions, but they are not capable of compensating for the false negatives.

Jeffery et al. [8] propose adaptive SMURF (*Statistical sMoothing for Unreliable RFid data*), which uses sampling estimators to automatically adjust the filter window size. Valentine et al. [9] presented an adaptive sliding-window based approach for reducing false negative reads in RFID data streams. However due to smaller window size used, it produces more false negatives.

Xie et al. [10] propose a sampling based method for information recovery in RFID data. Chen et al. [11] propose a Bayesian inference based approach, which takes full advantage of data redundancy, for cleaning RFID raw data. The likelihood is captured by designed a state detection model. Yan et al. [12] propose a Kalman Filter based cleaning method, which solves false negative and false positive from single readers.

Gu et al. [13] propose a data imputation model for RFID by efficiently maintaining and analyzing the correlations of the monitored objects. The spatio-temporal correlation between monitored objects are used to fill in the false negatives. In [3], we propose a distance-aware deployment graph based on RFID reader deployment graph [14] to handle spatial ambiguities in indoor RFID data.

This work is different from the related works in several aspects. First, this work has a unique setting, namely indoor space where spatio-temporal constraints can be exploited for data cleansing. Second, this work utilizes a probabilistic distance-aware graph model to identify and recover false negatives. Third, the proposals in this work can be applied to cleanse other kinds of symbolic indoor tracking data, e.g., Bluetooth tracking data.

7 Conclusion and Future Work

In this paper we study indoor RFID tracking data cleansing. We focus on false negatives in indoor raw RFID tracking data. To handle false negatives in the indoor RFID tracking data, we designed a false negative detection and recovery algorithm which utilizes a probabilistic distance-aware graph model which capture transition probabilities together with the spatio-temporal constraints implied by the deployment of RFID readers as well as the indoor topology. We conduct extensive experimental studies using both synthetic data and real data. The results demonstrate that the proposed techniques are effective and efficient.The techniques proposed is applicable to indoor tracking data obtained by other symbolic positioning technologies, e.g., Bluetooth.

For future work there are several directions to consider for cleansing indoor tracking data. First, it is interesting to consider the Radio Signal Strength Information (RSSI) parameter of indoor tracking devices, it can be exploited to enhance the data cleansing. RSSI can be used to derive the distance between an object and the relevant positioning device(s).

Second, it will be interesting to verify the possibility to conduct data

cleansing for each objects individually, given that their individual features (e.g., moving speed) are known.

Third, it is relevant to conduct high level queries on cleansed indoor tracking data to obtain more interesting, more informative and more precise results with out physically cleansing the data.

8 Acknowledgement

This work is supported in part 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] C. Floerkemeier and M. Lampe, "Issues with RFID usage in ubiquitous computing applications," in *Pervasive*, 2004, pp. 188–193.

[2] D. Roozbeh, O. Maria, and L. Xue, "RFID data management: Challenges and opportunities," in *IEEE International Conference on RFID 2007*, 2007, pp. 175–182.

[3] A. I. Baba, H. Lu, X. Xie, and T. B. Pedersen, "Spatiotemporal data cleansing for indoor RFID tracking data," in *Mobile Data Management*, 2013, pp. 187–196.

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

[5] S. S. Chawathe, V. Krishnamurthy, S. Ramachandran, and S. E. Sarma, "Managing rdfi data," in *VLDB*, 2004, pp. 1189–1195.

[6] O. Mylyy, "RFID data management, aggregation and filtering," in *R. Meersman and Z. Tari (Eds.): CoopIS/DOA/ODBASE 2005, LNCS 3760*, 2007, pp. 557–575.

[7] S. R. Jeffery, M. J. Franklin, and M. N. Garofalakis, "An adaptive RFID middleware for supporting metaphysical data independence," *VLDB J.*, vol. 17, no. 2, pp. 265–289, 2008.

[8] S. R. Jeffery, M. N. Garofalakis, and M. J. Franklin, "Adaptive cleaning for RFID data streams," in *VLDB*, 2006, pp. 163–174.

[9] L. V. Massawe, J. D. M. Kinyua, and H. Vermaak, "Reducing false negative reads in RFID data streams using an adaptive sliding-window approach," 2012.

[10] J. Xie, J. Yang, Y. Chen, H. Wang, and P. S. Yu, "A sampling-based approach to information recovery," in *ICDE*, 2008, pp. 476–485.

[11] H. Chen, W.-S. Ku, H. Wang, and M.-T. Sun, "Leveraging spatio-temporal redundancy for RFID data cleansing," in *SIGMOD Conference*, 2010, pp. 51–62.

[12] W. Yan and S. B. Yan., "Cleaning method of RFID data stream based on kalman filter." Journal of Chinese Computer Systems, 2011, pp. 1794–1799.

[13] Y. Gu, G. Yu, Y. Chen, and B. C. Ooi, "Efficient RFID data imputation by analyzing the correlations of monitored objects," in *DASFAA*, 2009, pp. 186–200.

[14] C. S. Jensen, H. Lu, and B. Yang, "Graph model based indoor tracking," in *Mobile Data Management*, 2009, pp. 122–131.

Paper C

Learning-Based Cleansing for Indoor RFID Data

Asif Iqbal Baba, Manfred Jaeger, Hua Lu, Torben Bach Pedersen, Wei-Shinn Ku and Xike Xie

The paper has been published in the
Proceedings of ACM SIGMOD, pp. 925–936, 2016.

Abstract

RFID is widely used for object tracking in indoor environments, e.g., airport baggage tracking. Analyzing RFID data offers insight into the underlying tracking systems as well as the associated business processes. However, the inherent uncertainty in RFID data, including noise (cross readings) and incompleteness (missing readings), pose challenges to high-level RFID data querying and analysis. In this paper, we address these challenges by proposing a learning-based data cleansing approach that, unlike existing approaches, requires no detailed prior knowledge about the spatio-temporal properties of the indoor space and the RFID reader deployment. Requiring only minimal information about RFID deployment, the approach learns relevant knowledge from raw RFID data and uses it to cleanse the data. In particular, we model raw RFID readings as time series that are sparse because the indoor space is only partly covered by a limited number of RFID readers. We propose the Indoor RFID Multivariate Hidden Markov Model (IR-MHMM) to capture the uncertainties of indoor RFID data as well as the correlation of moving object locations and object RFID readings. We propose three state space design methods for IR-MHMM that enable the learning of parameters while contending with raw RFID data time series. We solely use raw uncleansed RFID data for the learning of model parameters, requiring no special labeled data or ground truth. The resulting IR-MHMM based RFID data cleansing approach is able to recover missing readings and reduce cross readings with high effectiveness and efficiency, as demonstrated by extensive experimental studies with both synthetic and real data. Given enough indoor RFID data for learning, the proposed approach achieves a data cleansing accuracy comparable to or even better than state-of-the-art techniques requiring very detailed prior knowledge, making our solution superior in terms of both effectiveness and employability.

The layout has been revised.

1 Introduction

Recently there has been a remarkable proliferation of RFID in indoor tracking and monitoring systems, e.g., airport baggage monitoring. Most RFID based applications today use low-cost passive RFID tags that are very small and require no source of power. These tags can be attached to a person or an item. Deployed RFID readers detect such tags and thus the moving objects that they are attached to when the objects appear in the readers detection ranges. Typically, a raw RFID detection reading $\langle o, r, t \rangle$ means that object o is detected by RFID reader r at time point t.

However, the dirtiness of RFID data poses challenges to high-level RFID data querying and analysis. In this paper, we focus on cleansing *false negatives* and *false positives* in indoor RFID data.

False negatives (missing readings) occur when a reader fails to read a tag in its detection range. They are caused by power failures on a reader, tag orientation with respect to the reader, presence of metal, dielectric or water close to the tag, and other factors.

False positives (cross readings) occur when a tagged object is unexpectedly read by multiple readers simultaneously. They 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 placed at different positions, and thus spatial ambiguities are caused.

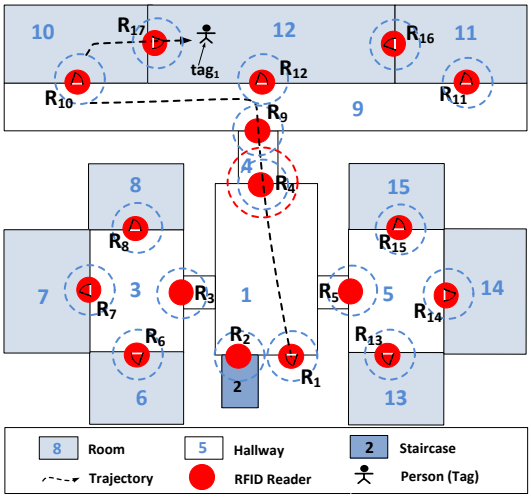


Fig. C.1: Example of indoor space and RFID reader deployment

For illustration, we consider an example floor plan with 17 readers shown

in Figure C.1. A moving object with tag tag_1 moved into the building and was first detected by reader R_1 from time point t_0 to time point t_3, yielding four observations by reader R_1. The raw RFID data about tag_1 is shown in Table C.1. Note that a tagged object is not continuously detected by RFID readers, which introduces uncertainties in the raw RFID data: for many time points the data does not tell where the object is.

At time points t_{12} and t_{13}, the moving object with tag_1 was detected by readers R_4 and R_9, due to the unexpected expansion of R_4's detection range. As a result, tag_1 seems to be present in both locations at time points t_{12} and t_{13}, thus giving rise to a false positives. It is, however, impossible that tag_1 can appear in both locations at the same time as the two readers R_4 and R_9 cover two separate locations and their detection ranges are not deployed to overlap.

After time point t_{16}, object with tag_1 was detected by reader R_{17} that kept detecting tag_1 until time point t_{29}. However, tag_1 is not supposed to be detected by R_{17} before it is detected by reader R_{10} on its way as shown in Figure C.1. To enter room 12 from room 9 via room 10, tag_1 must pass R_{10}'s detection range and yield the observation. But this is not the case here: tag_1 passed through R_{10} but failed to generate any information, thus giving rise to false negatives.

tagID	readerID	t	tagID	readerID	t	tagID	readerID	t
tag_1	R_1	t_0	tag_1	R_4	t_{10}	tag_1	R_9	t_{15}
tag_1	R_1	t_1	tag_1	R_4	t_{11}	tag_1	R_9	t_{16}
tag_1	R_1	t_2	tag_1	R_4	t_{12}	tag_1	R_1	t_{26}
tag_1	R_1	t_3	tag_1	R_9	t_{13}	tag_1	R_{17}	t_{27}
tag_1	R_4	t_7	tag_1	R_4	t_{13}	tag_1	R_{17}	t_{28}
tag_1	R_4	t_8	tag_1	R_9	t_{14}	tag_1	R_{17}	t_{29}
tag_1	R_4	t_9	tag_1	R_4	t_{14}			

Table C.1: Raw readings table

To support high-level RFID business logic processing, it is necessary to perform data cleansing to remove false positives and recover false negatives. Existing RFID data cleansing techniques require considerable specific prior knowledge for cleansing operations. Aiming at cleansing historical indoor RFID data, the graph model based cleansing approaches [1–3] rely on graphs that capture the indoor topology, the deployment of RFID readers, and multiple pertinent spatio-temporal properties such as the distances between readers and the minimum dwell time for an object to be detected by a reader. In the context of streaming RFID data, a probabilistic approach [4] demands to build four domain-specific probabilistic models before any data cleansing.

In a similar online setting, another approach [5] assumes that the locations of neighboring objects are correlated to each other. In contrast, this work lifts such assumptions and needs only minimal prior knowledge about the indoor settings for which the data cleansing is to be conducted (details given in Section 3).

In this paper, we model the uncertainties in indoor RFID data by a hidden Markov model (HMM) and design methods to learn the model parameters from raw RFID data without requiring detailed prior knowledge about the indoor setting. In particular, we use a variant of the HMM [6] in finding out the exact whereabouts of a moving object from the raw indoor RFID positioning data. We transform raw RFID data into sparse time series, as objects are not under surveillance at each time point. Each moving object moves differently and may take different paths. We model the movement of an individual object as a series of state transitions outputted by an HMM that operates on a set of states. We model the indoor RFID data uncertainties and the generated object RFID readings using an *Indoor RFID Multi-variate Hidden Markov Model (IR-MHMM)*.

We model the RFID tracking and data cleansing problem by means of three connected stochastic processes. The underlying object location transition from one location to another location is a stochastic process which is not directly observable (*hidden*). A second stochastic process is the observation process, which generates (unclean) RFID observations dependent on the current object's location. A third, again unobserved, process represents the clean RFID data that ideally should have been observed given the object's location. Under suitable assumptions on the relationship between the observable unclean observation process, and the ideal clean data process, we model all three processes by means of an IR-MHMM. Using a learned IR-MHMM, we infer the clean RFID data process from the observed raw data process, and in that manner we filter out the false positives and insert the correct readings omitted as false negatives.

The paper makes the following contributions. First, we propose a general framework that models the uncertainties in indoor RFID data, which enables a cleansing service by inferring the observation states. Second, we formalize an indoor RFID multi-variate hidden Markov model (IR-MHMM) as the major building block of the cleansing service. Third, we present three different state space designs for IR-MHMM from a given indoor floor plan. Fourth, we conduct comprehensive experimental studies using both real and synthetic data. The results give insight into the design properties of the proposed IR-MHMM and demonstrate the efficiency and effectiveness of the proposed cleansing approach on indoor RFID data.

The rest of this paper is organized as follows. Section 2 covers preliminaries. Section 3 details the proposed method. Section 4 presents the evaluation approach for the proposed method. Section 5 reports the experimental re-

sults. Section 6 reviews related work, followed by the conclusion in Section 7.

2 Preliminaries

We briefly recall the fundamental concepts and definitions of *Hidden Markov Models (HMMs)*. An HMM models two connected (discrete time) stochastic processes: an un-observed state transition process, and an observation process consisting of observable signals generated at each state. The underlying (hidden) state transition process is assumed to be Markovian and stationary. Formally, an HMM is a tuple $\lambda = (\mathcal{S}, \mathcal{O}, A, B, \pi)$, where:

1. $\mathcal{S} = \{s_1, \ldots, s_N\}$ is a set of (hidden) states.
2. $\mathcal{O} = \{o_1, \ldots, o_K\}$ is a set of possible observations.
3. A is an $N \times N$ transition probability matrix:
   $A = (a_{ij})_{i,j=1,\ldots,N}$, where a_{ij} represents the transition probability from hidden state s_i to hidden state s_j.
4. B is an $N \times K$ observation probability matrix:
   $B = (b_{ih})_{i=1,\ldots,N;h=1,\ldots,K}$, where b_{ih} is the probability of observing o_h when the hidden process is in state s_i.
5. π is a N-dimensional initial state probability vector:
   $\pi = (\pi_i)_{i=1,\ldots,N}$, where π_i is the probability that the hidden state process starts in state s_i.

An HMM defines a discrete time stochastic process over the combined state and observation space $\mathcal{S} \times \mathcal{O}$. The state of the process at time t is described by random variables $S^{(t)}$ with values in $\mathcal{S}$, and $O^{(t)}$ with values in $\mathcal{O}$, and the joint distribution of the $S^{(t)}$ and $O^{(t)}$ is defined by $P(S^{(0)} = s_i) = \pi_i$, $P(S^{(t+1)} = s_j \mid S^{(t)} = s_i) = a_{ij}$, and $P(O^{(t)} = o_h \mid S^{(t)} = s_i) = b_{ih}$.

For modeling processes where at each point in time observations of multiple variables are made, the basic HMM model has been generalized to *Multi-variate Hidden Markov Models (MHMMs)* [7]. In this model the simple observation space $\mathcal{O}$ is replaced by a multivariate observation space $\mathcal{O}_1 \times \cdots \times \mathcal{O}_M$, and at each time t one observes variables $O_1^{(t)}, \ldots, O_M^{(t)}$. In this model one can furthermore introduce the assumption that the different observations are independent given the hidden state, i.e., $P(O_1^{(t)}, \ldots, O_M^{(t)} \mid S^{(t)}) = \prod_{i=1}^{M} P(O_i^{(t)} \mid S^{(t)})$, and then parameterize the observation probability distribution given the hidden states by M separate matrices for the distributions $P(O_i^{(t)} \mid S^{(t)})$. A customized version of this MHMM model will be the basis for our RFID data cleansing model.

3 The IR-MHMM Approach

3.1 Data Transformation

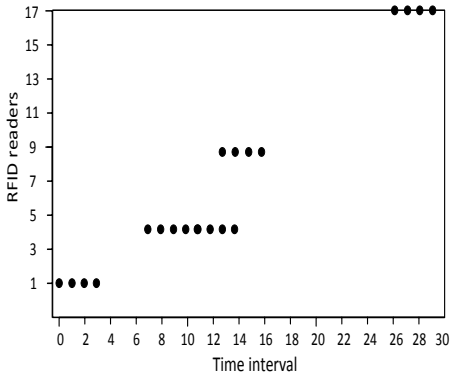
To prepare the application of MHMMs to our problem, we have to transform the raw readings data as illustrated in Table C.1 into a format suitable for processing by MHMMs. RFID readers continuously detect objects in their range and report in the frequency determined by their sampling rate. They thereby generate raw data entries of the form $\langle tagID, readerID, t\rangle$, which means that the object with *tagID* is detected by the reader *readerID* at time t.

In this paper we are concerned with cleaning the tracking data for one object at a time, so that from the raw reader data we first select all records relating to a specific tag (as already shown in Table C.1). The *tagID* then becomes a constant, and the relevant part of the records are just the pairs $\langle readerID, t\rangle$.

As a first step, we discretize the continuous timestamp data. For this, a time granularity parameter α is chosen. For an object whose tracking data spans an interval of duration $(t_e - t_s)$, where t_s and t_e are the first and last readings reported about the object, we introduce $T = \frac{(t_e - t_s)}{\alpha}$ discrete time points $0, \ldots, (T-1)$. A continuous time record $\langle readerID, t\rangle$ then is replaced by $\langle readerID, i\rangle$, where i is such that $t \in [t_s + i \cdot \alpha, t_s + (i+1) \cdot \alpha]$. The choice of α should be based on two considerations: First, it should be chosen small enough, so that the discretization does not introduce any spurious appearances of cross readings. This can be ensured when $\alpha < \Delta_{min}/2$, where Δ_{min} is the minimum absolute difference between continuous time-stamps of records with different *readerID*s. Secondly, the time granularity must be small enough, so that clean data of this granularity is sufficient for subsequent tracking and analysis purposes. Under these two constraints, α should be chosen as large as possible, to minimize the computational complexity of the cleaning process.

For illustration purposes, assume that the data in Table C.1 is already discretized, i.e. t_i in this table stands for the discrete time point i. Figure C.2 (a) shows an alternative representation of this data, where we plot the readings in a coordinate system with the time points on the x-axis, and the possible reader IDs on the y-axis. This time series can equivalently be encoded as a sequence of T binary vectors of length M, where M is the number of readers (Figure C.2 (b)).

At the end of the transformation process, the data now consists of a sequence $\boldsymbol{V}_r^{(0)}, \ldots, \boldsymbol{V}_r^{(T-1)}$, where each $\boldsymbol{V}_r^{(t)} \in \{0,1\}^M$. The subscript r stands for *raw*, to emphasize that this still is raw data with possible false positives and false negatives. The kth component of $\boldsymbol{V}_r^{(t)}$ is denoted $V_{r,k}^{(t)}$. In the following, we will consider the $\boldsymbol{V}_r^{(t)}$ as the observations in a MHMM process, and



(a) Plotted

10000000000000000	00000000100000000
10000000000000000	00000000100000000
10000000000000000	00000000000000000
10000000000000000	00000000000000000
00000000000000000	00000000000000000
00000000000000000	00000000000000000
00000000000000000	00000000000000000
00010000000000000	00000000000000000
00010000000000000	00000000000000000
00010000000000000	00000000000000000
00010000000000000	00000000000000000
00010000000000000	00000000000000001
00010000000000000	00000000000000001
00010000100000000	00000000000000001
00010000100000000	00000000000000001

(b) Binary vector representation

Fig. C.2: Raw data of Table C.1 as discrete time-series.

use standard HMM learning and inference techniques to transform the $\boldsymbol{V}_r^{(t)}$ into cleaned versions $\boldsymbol{V}_c^{(t)}$. The cleaned data still takes values in $\{0,1\}^M$, but now with the condition that each $\boldsymbol{V}_c^{(t)}$ contains at most one non-zero entry (i.e., no cross readings). This cleaning process will be based on certain probabilistic modeling assumptions and computational approximations. In the following section we clarify these underlying assumptions. We do this on a more general basis than our specific data model $\boldsymbol{V}_r^{(t)}$, so that our analysis also provides a basis on which we can compare approaches taken in related work. The next section may also be skipped at a first reading, and returned to when a deeper understanding of the proposed method is desired. The concrete description of our cleaning method for the transformed data $\boldsymbol{V}_r$ continues in Section 3.3.

3.2 Probabilistic Inference for RFID Streams

In a broader sense than data cleansing alone, probabilistic methods for processing raw RFID data streams can be characterized as follows. We are given a sequence of raw readings $\boldsymbol{o}_r = o_r^{(0)}, \ldots, o_r^{(T-1)}$, where each $o_r^{(t)}$ is an observation of a random variable $O_r^{(t)}$ that takes values in a raw observation space $\mathcal{O}_r$. From $\boldsymbol{o}_r$ we want to infer a clean data sequence $\boldsymbol{o}_c = o_c^{(0)}, \ldots, o_c^{(T-1)}$ of observations of random variables $O_c^{(t)}$ that take values in a clean observation space $\mathcal{O}_c$. Based on a probabilistic model for $\boldsymbol{O}_r$ and $\boldsymbol{O}_c$ one can infer the clean from the raw data $\boldsymbol{o}_r$ by selecting the $\boldsymbol{o}_c$ that maximizes the conditional probability $P(\boldsymbol{O}_c = \boldsymbol{o}_c \mid \boldsymbol{O}_r = \boldsymbol{o}_r)$.

A straightforward approach to defining the joint distribution of $\boldsymbol{O}_r, \boldsymbol{O}_c$ is

in the form of an HMM with $O_r^{(t)}$ as the observable variables, and $O_c^{(t)}$ as the hidden state variables $S^{(t)}$. This approach, however, faces two fundamental difficulties. The first difficulty lies in the fact that here the hidden state space must correspond to the clean observation space that is required for the RFID data stream processing problem at hand. On the basis of this particular hidden state space, and under the limitation of the Markov assumption for the state transition model, it may not be possible to obtain an accurate model for the joint distribution $P(\boldsymbol{O}_c, \boldsymbol{O}_r)$. To illustrate this point, consider the concrete raw and clean data models $\boldsymbol{V}_r^{(t)}, \boldsymbol{V}_c^{(t)} \in \{0,1\}^M$ we have introduced in the previous section. The clean state 0^M (no reading) would then be a hidden HMM state. Under the Markov assumption, the model could then not represent that given that currently there is no reading, the probability distribution over the next reading depends on what the most recent available reading has been. In our example of Figure C.1, when currently there is no reading, then the information that the last available reading has come from reader R_1 should increase the probability that at the next point in time we observe a reading from one of R_2, R_3, R_4 or R_5.

The second difficulty arising from equating $\mathcal{S} = \mathcal{O}_c$ lies in constructing or learning an adequate model. Since now inferred hidden state sequences are the target of inference, one has to ensure that inferred hidden state values actually correspond to the relevant properties of the underlying tracking process that we want to capture with the clean data $\boldsymbol{O}_c$. When hidden HMM states are endowed in this manner with an intended meaning, we also call them *semantic states*. The construction of HMMs whose hidden states adhere to an intended semantics usually requires either manual design, or learning from labeled training data in which besides the observation variables $\boldsymbol{O}$ also the hidden states $\boldsymbol{S}$ are given.

In spite of these difficulties, most previous work that employs HMMs for RFID data stream analysis follows the $\mathcal{S} = \mathcal{O}_c$ approach [4, 5]. The first difficulty can be dealt with by a careful design of $\mathcal{S}$, whose elements must simultaneously represent the relevant information for the clean data, and some additional information to support an effective Markov model (e.g. some form of limited memory about past states). It is usually not possible to circumvent the second difficulty without taking recourse to some manual model specification, or learning from data that carries additional annotations beyond pure raw readings.

It is our goal to construct HMMs for $P(\boldsymbol{O}_c \mid \boldsymbol{O}_r)$ in a completely unsupervised manner, i.e., by learning from training data that consists of the raw observations $\boldsymbol{O}_r$ only. The hidden state space $\mathcal{S}$ then consists of *latent states*: states that have no pre-defined interpretation, and that are only introduced to support an adequate Markov model for the variables of interest $\boldsymbol{O}_c, \boldsymbol{O}_r$. Thus, our model will have to define a joint distribution $P(\boldsymbol{S}, \boldsymbol{O}_r, \boldsymbol{O}_c)$. Making

a natural conditional independence assumption between $\boldsymbol{O}_r$ and $\boldsymbol{O}_c$ given $\boldsymbol{S}$, we can write:

$$P(\boldsymbol{S}, \boldsymbol{O}_r, \boldsymbol{O}_c) = P(\boldsymbol{S})P(\boldsymbol{O}_r \mid \boldsymbol{S})P(\boldsymbol{O}_c \mid \boldsymbol{S}). \tag{C.1}$$

Given the model $P(\boldsymbol{S}, \boldsymbol{O}_r, \boldsymbol{O}_c)$, the data cleaning inference task then is to compute for a given observed raw data sequence $\boldsymbol{o}_r$:

$$\arg\max_{\boldsymbol{o}_c} P(\boldsymbol{O}_c = \boldsymbol{o}_c \mid \boldsymbol{O}_r = \boldsymbol{o}_r) = \sum_{\boldsymbol{s}} P(\boldsymbol{S} = \boldsymbol{s} \mid \boldsymbol{O}_r = \boldsymbol{o}_r)P(\boldsymbol{O}_c = \boldsymbol{o}_c \mid \boldsymbol{S} = \boldsymbol{s}). \tag{C.2}$$

Here the sum is over all hidden state sequences $\boldsymbol{s}$ of the same length as $\boldsymbol{o}_r$. To implement this approach, two main challenges have to be met. First, one has to learn the model (C.1), which includes the conditional distribution $P(\boldsymbol{O}_c \mid \boldsymbol{S})$ containing only variables for which we have no observations at all. Second, the maximization problem (C.2) has to be solved efficiently (which precludes actual summation over all $\boldsymbol{s}$).

We tackle the first challenge by making some strong assumptions on the relationship between $\boldsymbol{O}_c$ and $\boldsymbol{O}_r$, the result of which will be that we implicitly also learn $P(\boldsymbol{O}_c \mid \boldsymbol{S})$ by learning $P(\boldsymbol{O}_r \mid \boldsymbol{S})$. Our first assumption is that the clean data space is a subset of the raw data space:

$$\mathcal{O}_c \subseteq \mathcal{O}_r. \tag{C.3}$$

This assumption is true in our specific data model, where $\mathcal{O}_r = \{0, 1\}^M$, and $\mathcal{O}_c \subseteq \mathcal{O}_r$ consists of the vectors with at most one non-zero component.

We now make a crucial assumption on the relationship between $\boldsymbol{O}_r$ and $\boldsymbol{O}_c$. Intuitively we assume that $\boldsymbol{O}_r$ and $\boldsymbol{O}_c$ are generated by the same random process given $\boldsymbol{S}$, only that $\boldsymbol{O}_c$ is conditioned on the subset $\mathcal{O}_c$. Formally, this can be expressed by the assumption that for sequences $\boldsymbol{o}_c, \boldsymbol{s}$ (of a common length T):

$$P(\boldsymbol{O}_c = \boldsymbol{o}_c \mid \boldsymbol{S} = \boldsymbol{s}) = P(\boldsymbol{O}_r = \boldsymbol{o}_c \mid \boldsymbol{S} = \boldsymbol{s}) / P(\boldsymbol{O}_r \in \mathcal{O}_c^T \mid \boldsymbol{S} = \boldsymbol{s}). \tag{C.4}$$

Under assumptions (C.1), (C.3) and (C.4), our inference problem (C.2) becomes to compute

$$\arg\max_{\boldsymbol{o}_c \in \mathcal{O}_c^T} \sum_{\boldsymbol{s}} \frac{P(\boldsymbol{S} = \boldsymbol{s} \mid \boldsymbol{O}_r = \boldsymbol{o}_r)P(\boldsymbol{O}_r = \boldsymbol{o}_c \mid \boldsymbol{S} = \boldsymbol{s})}{P(\boldsymbol{O}_r \in \mathcal{O}_c^T \mid \boldsymbol{S} = \boldsymbol{s})} \tag{C.5}$$

We note that in the case where $\boldsymbol{o}_r$ already is a clean sequence, i.e., $\boldsymbol{o}_r \in \mathcal{O}_c^T$, the $\boldsymbol{o}_c$ maximizing (C.5) need not be equal to $\boldsymbol{o}_r$. Thus, the inference (C.5) can change a formally clean sequence (a sequence not containing any cross

readings) by a more likely clean sequence (e.g. adding additional readings to correct likely false negatives).

The summation over all hidden state sequences $\boldsymbol{s}$ in (C.5) will usually be computationally infeasible. We therefore approximate the posterior distribution $P(\boldsymbol{S} = \boldsymbol{s} \mid \boldsymbol{O}_r = \boldsymbol{o}_r)$ by only considering the most probable state $\hat{\mathbf{s}} = arg\,max_{\boldsymbol{s}}\, P(\boldsymbol{S} = \boldsymbol{s} \mid \boldsymbol{O}_r = \boldsymbol{o}_r)$, and then computing

$$\hat{\boldsymbol{o}}_c = \underset{\boldsymbol{o}_c \in \mathcal{O}_c^T}{arg\,max}\, P(\boldsymbol{O}_r = \boldsymbol{o}_c \mid \boldsymbol{S} = \hat{\mathbf{s}}) \tag{C.6}$$

Note that the term $P(\boldsymbol{O}_r \in \mathcal{O}_c^T \mid \boldsymbol{S} = \hat{\mathbf{s}})$ now is a constant that can be omitted. In summary, our approach is characterized by making the probabilistic modeling assumptions (C.1), (C.3) and (C.4), and using the approximation (C.6) for (C.5).

Like most statistical assumptions, (C.1) and (C.4) are idealizations that typically will not be fully satisfied by the actual distribution $P(\boldsymbol{S}, \boldsymbol{O}_r, \boldsymbol{O}_c)$. Inferences obtained by an approach that implicitly makes these assumptions can still be accurate when the assumptions are not strictly valid (compare, e.g., the Naive Bayes classifier, whose underlying independence assumptions are hardly ever fully justified for the data that the classifier is used on).

3.3 Indoor RFID Multi-variate HMM

Based on the data transformation of Section 3.1 we implement the approach described in more general terms in Section 3.2 by a customized version of MHMMs. We define *Indoor RFID Multi-variate HMM (IR-MHMM)* as Multivariate Hidden Markov Models $\lambda = (\mathcal{S}, \{0,1\}^M, A, B, \pi)$, where $\mathcal{S}, A, \pi$ are as in the generic (M)HMM definition, and the observation space is $\mathcal{O} = \{0,1\}^M$. Assuming that different readers are independent given the hidden state, the observation model is given by the probabilities $b_{ik} := P(V_{r,k}^{(t)} = 1 \mid S^{(t)} = s_i)$ (i.e., the probability of getting in the ith hidden state a reading from the kth reader). The observation model of the IR-MHMM now is represented by a matrix B of dimensions $N \times M$ with entries b_{ik}. Figure C.3 gives a graphical representation of the model.

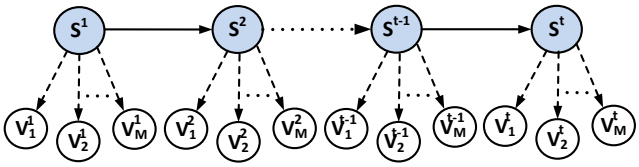


Fig. C.3: IR-MHMM for RFID raw data

To construct an IR-MHMM for an RFID cleaning application the model parameters N, A, B, π have to be determined (the parameter M is given by

the application as the number of distinct readers). The construction is a combination of design and learning. It is our overall objective to minimize the amount of domain specific information required in the design component (such as data on spatial distances, or average travel times between readers), and to base the learning component entirely on learning from the raw data $\boldsymbol{V}_r$, without any further annotations.

In the design step, first the cardinality N of the state space is chosen. This may already complete the design component, and the probabilistic parameters A, B, π can now be learned. In this setting, the states $s_i \in \mathcal{S}$ are purely latent, without any associated semantics a-priori. Only after learning may it be possible to assign a semantic interpretation to some states. For example, if the learned value b_{ik} is much higher than other values b_{jk} ($j \neq i$), then state s_i represents that the tracked object is in the range of reader R_k. Alternatively, one can try to associate a semantic meaning with states a-priori. It must be emphasized, however, that since learning is from raw data that does not contain information about the true nature of the hidden states, there is no guarantee that after learning the states closely adhere to their intended semantics. The best we can do is to bias the learning algorithm towards exploiting the suggested semantics of the hidden states. We do this by imposing prior constraints on the probabilistic components A and B. The following section describes three concrete designs of state spaces whose latent states are endowed with intended semantics.

3.4 State space design

We propose four different state space designs for IR-MHMMs. All our designs require as input at most some qualitative spatial information as expressed by a reader deployment graph. Designs define a set of states with associated intended semantics, such as states representing (approximate) spatial locations, or states representing some history of past transitions. The intended semantics of hidden states lead to constraints of the form $a_{ij} = 0$ (for some combinations i, j of states direct transitions will not be possible).

As a baseline, we consider the design where states are entirely latent, without associated semantics.

Unstructured-N (U-N). This design is given by the number N of hidden states alone. In our experiments for the floor plan in Figure C.1, we consider the choices $N = 25, 34$ and 50, which correspond to the number of states of the three structured designs described next for a running example.

We next describe three different state space designs where states are associated with an intended semantics. We elaborated on the structure of the states spaces, and the constraints imposed on the transition model A. We defer the explanation of how the observation model B is handled to Section 3.5. The following three state space designs are illustrated in Figure C.4, C.4 and

C.4 for the running example in Figure C.1.

Minimum State Model (MSM). This design contains a state for each deployed reader R_i, and a state for each hallway or any indoor partition that is connected with more than one reader. For example, reader R_4 in Figure C.1 is represented as state s_4 in the design shown in Figure C.4. Furthermore, hallway 3 in Figure C.1 is connected with four readers R_3, R_6, R_7, and R_8, and thus it is represented by a state s_{19}. The total number of states for our example scenario is $N = 25$. The transition model is constrained by setting $a_{ij} = 0$ if s_i and s_j correspond to locations that are not directly connected.

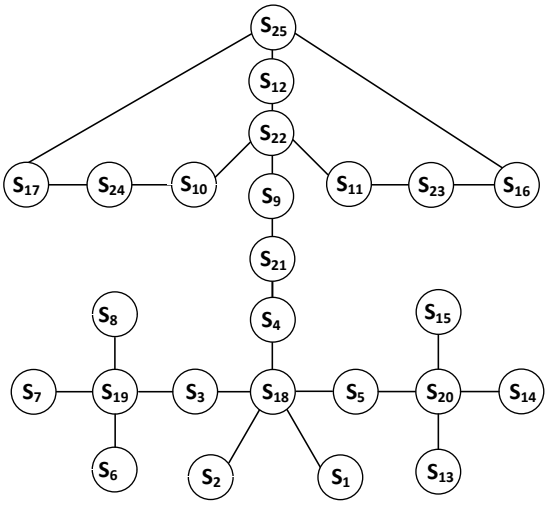


Fig. C.4: Minimum state model example

Last State Model (LSM). This design contains two states s_i and l_{si} for each deployed reader R_i. The intended meaning of s_i is that the object is within reader R_i's detection range, whereas state l_{si} represents that R_i was the last reader in whose range the object has been. The intention of this model is to alleviate the limitations of the Markov assumption by encoding a (very limited) history of previously visited states at the current state.

In this design, the non-zero transition probabilities are from states s_i to the corresponding l_{si}, and from states l_{si} to states s_j when there is a direct path from reader R_i to R_j. The state space size is $N = 2M$, i.e., $N = 34$ for our example domain of Figure C.1.

In-between State Model (ISM). This design contains a state s_i for each deployed reader R_i, and a state s_{ij} for each pair of adjacent readers R_i and R_j. The intended meaning of s_{ij} is that the object is transitioning between R_i and R_j (in either direction). The transition probabilities are non-zero only between states s_i and s_{kj}, where $i \in \{k, j\}$. The design for our example domain as

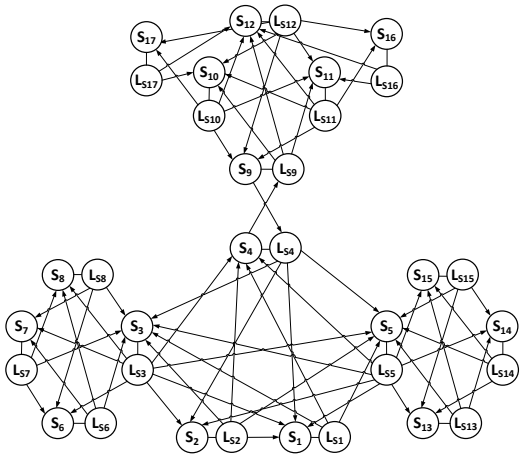


Fig. C.5: Last state model example

shown in Figure C.6 has 50 states.

The three state space designs are applicable to any indoor space with any identification technology such as RFID, Infrared, Bluetooth, etc. The efficiency and effectiveness of the final design will, however, be dependent upon the indoor topology, target application, and the kind of data which is used to learn the parameters of the model.

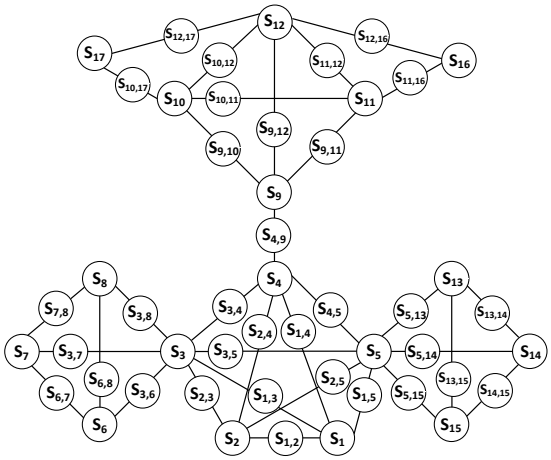


Fig. C.6: In-between state model example

3.5 Learning Parameters of IR-MHMM

Given a state space design, the unconstrained parameters (i.e., those not set to zero by the design) have to be estimated. For this we are using the standard EM algorithm (a.k.a. Baum-Welch algorithm in the HMM context), which operates on the full transition probability matrices A. Parameter constraints of the form $a_{ij} = 0$ are easy to integrate into this learning procedure: one only needs to initialize these parameters with zero values. The iterations of the EM algorithms will never turn zero values into non-zero values, so that the constraints will also be satisfied by the final solution, and the unconstrained parameters are fitted to the data.

In practice, in order to avoid division by zero errors, it is useful to initialize the constrained parameters with small values greater than 0. We therefore initialize those a_{ij} that are zero according to our state space design with values 0.01. For a fixed i, the initial values of the remaining transition probabilities a_{ij} then are set uniformly, so that $\sum_{j=1}^{N} a_{ij} = 1$. In our experiments we observe that imposing only the "soft" initial constraints $a_{ij} = 0.01$ does not lead to final solutions where the eventually learned a_{ij} parameter is significantly larger than 0. On the contrary, the parameters initialized with 0.01 typically become still much smaller during the EM iterations (usually ending with values $< 10^{-5}$ at termination) .

In all our state space designs, we have for each reader R_i a designated state s_i corresponding to locations within R_i's range. We still have to explain how we bias the learning procedure towards solutions that respect the intended semantics of these states. One could try to strictly impose the intended semantics by parameter constraints $b_{ii} = 1$ and $b_{ik} = 0$ for $k \neq i$. However, these hard constraints would preclude the possibility of cross readings. We therefore again use an initialization with a soft version of these constraints, setting $b_{ik} = 0.01$ ($k \neq i$) and $b_{ii} = 1 - 0.01(M - 1)$. Due to the presence of cross readings in the data, it may easily happen that in the final solution some b_{ik} with $k \neq i$ become significantly larger than their initial 0.01 values.

3.6 Data Cleaning

After an IR-MHMM λ has been constructed and its parameters have been learned, we use it to clean the raw RFID data as follows (cf. Algorithm 8). The input for the data cleansing procedure is raw RFID data transformed into a multi-variate binary observation sequence $\boldsymbol{v}_r = (v_{r,1}^{(t)}, \ldots, v_{r,M}^{(t)})_{t=0\ldots T-1}$, and an IR-MHMM λ.

First the standard Viterbi algorithm [8] is used to compute the most probable hidden state sequence $\hat{\mathbf{s}} = (\hat{s}^{(t)})_{t=0\ldots T-1}$ given $\boldsymbol{v}_r$. We then compute the most probable observation sequence $\hat{\boldsymbol{v}}_c = (\hat{v}_{c,1}^{(t)}, \ldots, \hat{v}_{c,M}^{(t)})_{t=0\ldots T-1}$ given $\hat{\mathbf{s}}$, under the constraint that at each t $(\hat{v}_{c,1}^{(t)}, \ldots, \hat{v}_{c,M}^{(t)})$ contains at most one non-zero

entry. We denote by $\boldsymbol{e}_k \in \{0,1\}^M$ the binary vector that has value 1 at position k, and value 0 in all other components; $\mathbf{0}$ denotes the vector with value 0 in all components.

Due to the HMM structure, the most probable observation sequence given $\hat{\mathbf{s}}$ can be determined pointwise for each t by maximizing $P(\boldsymbol{V}_r^{(t)} = \boldsymbol{v}_r^{(t)} | S^{(t)} = \hat{s}^{(t)})$. If $\hat{s}^{(t)} = s_i$, then this probability is maximized by setting $\hat{v}_{r,k}^{(t)} = 1$ if $b_{ik} > 0.5$, and $\hat{v}_{r,k}^{(t)} = 0$ otherwise. Under the constraint that $\hat{v}_{r,k}^{(t)}$ can be nonzero for at most one k, i.e., we want to find the most probable clean observation vector $\hat{\boldsymbol{v}}_c$, this is modified to setting $\hat{v}_c^{(t)} = \boldsymbol{e}_k$ if $b_{ik} > 0.5$, and $b_{ik} > b_{ik'}$ for all $k' \neq k$ (lines 5-14). This implements (C.6) for our specific application.

Algorithm 8 Data Cleansing(IR-MHMM λ for M readers, Raw transformed RFID sequence $\boldsymbol{v}_r$ of length T)

```
1:  ŝ ← Viterbi (v_r, λ)
2:  v̂_c = ∅
3:  for t = 0 ... T − 1 do
4:      i = state index of ŝ^(t)
5:      p ← 0.5
6:      k_max = 0
7:      for k = 0 ... M do
8:          if (b_ik > p) then
9:              k_max ← k
10:             p ← b_ik
11:     if k_max > 0 then
12:         Concat(v̂_c, e_{k_max})        ▷ Append e_{k_max} as tth component to output sequence
13:     else
14:         Concat(v̂_c, 0)                ▷ Append 0 as tth component to output sequence
15: return v̂_c
```

4 Edit Distance Based Evaluation

To evaluate the accuracy of inferred results, we check the similarity between the inferred observation trajectories and the corresponding ground truth trajectories. The two trajectories of the same object can be of different lengths. Since our data is not geometric but symbolic, we use a variant of edit dis-

tance [9]. It quantifies the dissimilarity between two strings by the number of operations (insert, delete, or substitute) required to transform one string into the other. When computing the edit distances, we use a slightly modified representation of the binary observation vectors. Basically, we use a sequence of reader IDs to represent an object's trajectories. If there is no reading at a time point, symbol 0 is used; if there are more than one reading at a time point, a set of reader IDs are used. Referring to the example data of object tag_1 shown in Figure C.2(b), its raw trajectory (R_{traj1}) and ground truth trajectory (G_{traj1}) are shown in Figure C.7.

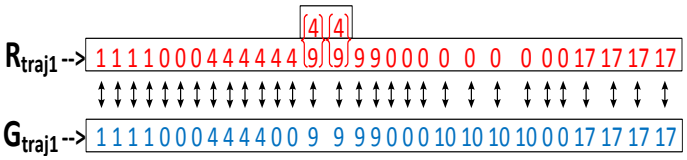


Fig. C.7: Example raw RFID and ground truth sequences

We use a weighted edit distance method, which allows the cost of a substitution to depend upon the symbols that are deleted or inserted. For a substitution of a symbol set $\{i, j\}$ with i or j we assign a cost of 1, whereas a substitution with $k \neq i, j$ incurs a cost of 2. Any insertion or deletion has a cost of 2. For example, in Figure C.7, while comparing R_{traj1} and G_{traj1}, we compare "$\{9, 4\}$" with "9" and do the edit operation with an edit cost of 1. This way conducts a much fairer evaluation, considering that one correct observation along with one wrong observation is still better than having two completely wrong observations.

We use two methods for edit distance computation. In the first method, we apply a weighted edit distance measurement to the original inferred observation sequence and the ground truth seque- nce. We quantify how accurate the inferred sequence is both spatially and temporally with respect to the ground truth. We not only check if the correct observation symbol is inferred but we also check if it is correctly inferred at each time point. The edit distance score depends on how accurately our learned models infer the correct observation and how many times they infer correctly. The length of the inferred trajectory (I_{traj1}) is always equal to the length of R_{traj1} but not necessarily equal to G_{traj1}. The edit distance between G_{traj1} and R_{traj1} quantifies the amount of inaccuracies in the raw RFID sequence, and that between G_{traj1} and I_{traj1} determines how accurate the inferred results are. The difference between these two edit distances determines a learned model's effectiveness at cleansing the raw RFID data.

The ground truth trajectory G_{traj1} and inferred trajectory I_{traj1} are shown in Figure C.8 for the running example. Here, R_{traj1} contains false negatives

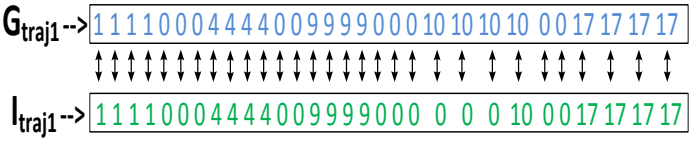


Fig. C.8: Edit distance on original sequences

at time points 20 to 23 (reader 10 has missing readings), whereas I_{traj1} only recovers the missing reading at time point 23. This is a typical result obtained by cleaning with IR-MHMMs, due to the underlying Markov assumptions. Since there is still a discrepancy between true and inferred sequences at several time intervals, the edit distance here is still quite large at a value of 6. Measuring the error of I_{traj1} by the edit distance on the full sequences is appropriate when exact timepoints and durations of an object's presence in a reader's range are of importance. However, if the application scenario only demands to reconstruct the qualitative, spatial aspect of the trajectory, we recommend using the edit distance method on block sequences as described as follows.

In the second method, we treat an object's sequence of symbols as a sequence of blocks (of symbols), and we use the edit distance on the two block sequences, i.e., G_{traj1}(block) and I_{traj1}(block). The block sequences are formed from the original sequences by merging together continuous blocks of the same observation symbols. The length of the inferred block sequence depends on the accuracy of the inferred results. To illustrate, we modify the example I_{traj1} shown in Figure C.8 by assuming that our model does not infer any reading at interval 23. The two corresponding block sequences are shown in Figure C.9. The two block trajectories $\mathcal{G}_{traj1}$(block) and $\mathcal{I}_{traj1}$(block) are of length 9 and 7, respectively. Their edit distance 2+2=4, much less than the edit distance on original sequences.

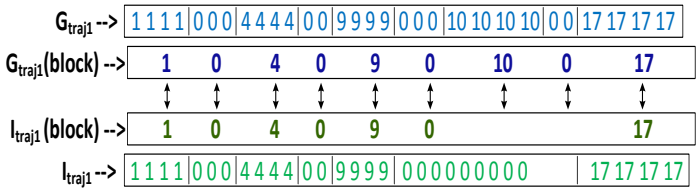


Fig. C.9: Edit distance on block sequences

5 Experimental Studies

In this section we report on the experimental studies that evaluate our proposed HMM-based approach for indoor RFID data cleansing. We evaluate the effectiveness, efficiency, scalability, and versatility of the approach, study the effect of reader accuracy, and compare with state-of-the-art methods. Our approach was implemented using MVNHMM [7], an open source implementation of HMM in C++. All experiments were conducted on a Linux machine with a 2.8GHz Core i7 processor and 8GB of main memory. We used both real and synthetic RFID data, as detailed below.

5.1 Settings and Performance Metrics

We performed the evaluation of learned models in two ways. 1) **Online data case:** In this case, we use two separate sets of data, one for learning the parameters of the models and the other as test data to evaluate the effectiveness of the learned models. This case is suitable for application scenarios where RFID data is continuously streamed, and incoming data is to be cleaned based on a model learned from historical data. 2) **Offline data case:** In this case, we used only one data set. We first use raw data for learning the model parameters and then use the model to clean the same data set. This case is applicable in application scenarios where a given batch of data is available off-line and needs to be cleaned. We note that we do not "overfit" by using the true results (ground truth) during training: We use only the raw data for training and the ground truth only later during testing.

To evaluate our learning and inference process, we consider two metrics: efficiency and effectiveness. The efficiency of inferring correct observations is measured using wall clock time and the effectiveness is measured by two metrics. First, the *average error* of a trajectory is defined as

$$Average\ error = \frac{sum\ of\ edit\ distance\ scores\ for\ all\ sequences}{total\ \#\ of\ inferred\ symbols}$$

The edit distance scores are calculated using the procedures described in Section 4. Second, the *error reduction ratio* between uncleaned and cleaned data is defined as

$$Error\ reduction\ ratio = \frac{errors\ before\ cleansing\ -\ errors\ after\ cleansing}{errors\ before\ cleansing}$$

Table C.2 summarizes the parameter settings in our experiments, with the default values in boldface. The meaning of the movement patterns is given below.

5.2 Experiments on Synthetic Data

Data Generation

We generated synthetic RFID data using a data generator that simulates object movements in an indoor space. The underlying indoor floor plan mainly used to generate the data is shown in Figure C.1 (other layouts are considered later). The process of data generation to evaluation (experimental validation) is shown in Figure C.10.

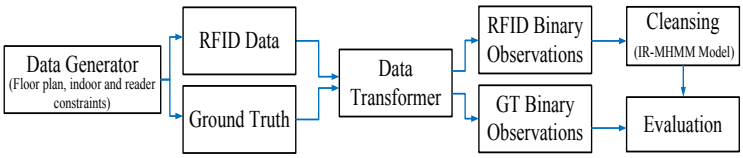


Fig. C.10: Simulation process

The data generator module takes an indoor RFID deployment graph and generates both raw RFID data and ground truth about the moving objects inside the given indoor space. The moving objects follow a random waypoint model [10] with a constant speed of 4km/h (1.1m/s). To generate object movements, an object in a room can move inside a room or move to any other room chosen randomly and then move along the shortest path from the source to the destination location. While moving from the current room to the destination room, an object can be detected by one or more deployed RFID readers. We generate the ground truth by recording the readings generated by the readers with 100% accuracy, by assuming that each deployed reader detects a present object for each reading cycle (epoch), which means no false negatives. Similarly, readings are generated only by the intended reader (no false positives). Later, we add inaccuracies to the RFID data. False negatives are introduced by "muting" randomly selected readers in a randomly selected path, while false positives are introduced by generating additional readings for the one or two nearest neighbors in addition to the actual reader. The data transformer module transforms the generated RFID data into multi-variate binary observation sequences, which are used to learn the model parameters. The learned model, along with the raw RFID data, is then passed to the cleansing module which is basically an inference model used to infer the true locations of the moving objects.

When generating synthetic data, we consider three typical object movement patterns *Directional-P(arallel), Directional-C(rossing) and Random*. For Directional-P and Directional-C we fix a set of source and destination locations, and objects always move along the shortest path between them. The difference is that in Directional-P, paths are *parallel*, i.e., do not overlap, while

Parameters	Settings
Number of sequences	50, 100, 500, 1000, 1500, **3000**, 10000
State models	MSM, LSM, ISM
Reader accuracy	**70**%, 80%, 90%
Movement patterns	Directional-P, Directional-C, Random

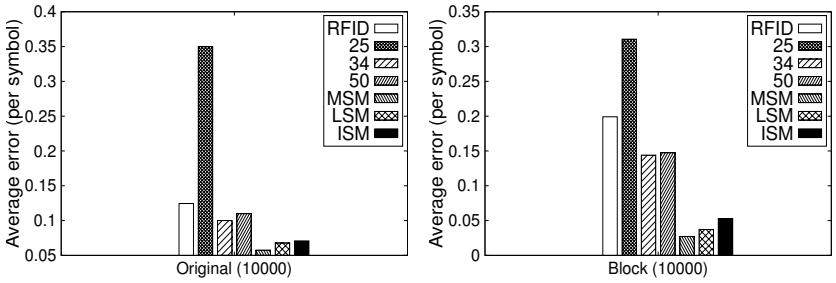
Table C.2: Experiment parameters

in Directional-C, they can *cross* (overlap). In Random paths, any location can be source or destination as long as they are not the same, and paths can cross/overlap.

Results on Synthetic Data

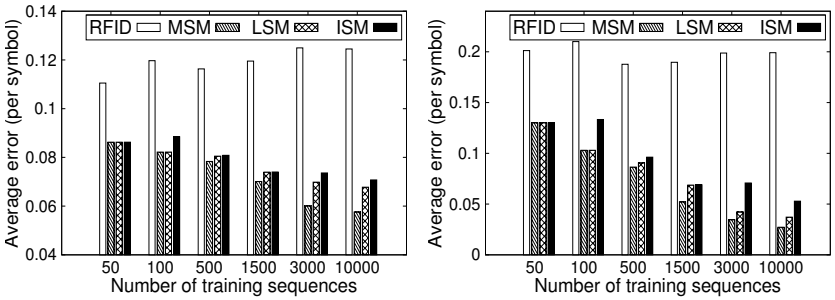
The data sets used to learn the models are generated with a default accuracy level of 70%. As an example, suppose that the fully correct raw readings table contains 100 readings. In the default case, we remove 15 readings as false negatives and add 15 readings as false positives. False negatives and false positives are equally distributed. We compute the edit distance between each pair of an inferred observation sequence and its corresponding ground truth observation sequence. We also compute the edit distance between raw RFID data (RFID) and ground truth to measure the effectiveness of the learned models.

Effectiveness: We start with evaluating the three structured models namely, *Minimum State Model* (MSM), *Last State Model* (LSM), and *In-between State Model* (ISM), and unstructured models with state spaces of 25, 34 and 50 states, respectively. We train all models using 10,000 training sequences and evaluate in the online case.



(a) Original sequences (b) Block sequences

Fig. C.11: Effectiveness of structured and unstructured models



(a) Original sequences (b) Original sequences

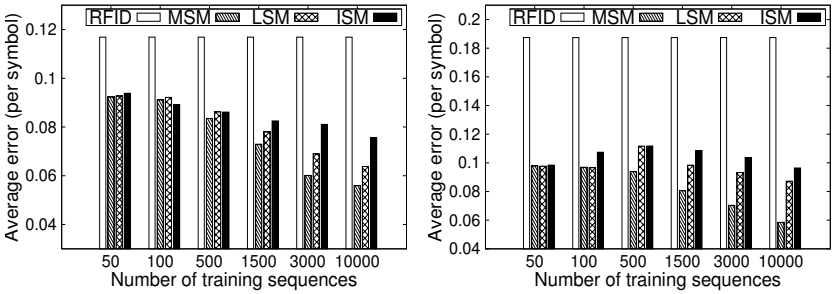
Fig. C.12: Results on Directional-C offline data

Figure C.11 shows that the unstructured models with 34 and 50 states manage to clean the data well, while the unstructured model with 25 states is too simple to be effective, it actually introduces more errors into the data to be cleansed. However, MSM, LSM, and ISM are significantly better than the unstructured models. Thus, it is evident that incorporating minimum knowledge about the reader deployment into the models can considerably improve the cleansing effectiveness.

Figure C.12 shows results for the Directional-C data offline case. Specifically, Figure C.12(a) reports results about the original inferred sequences. We see that MSM is more effective than LSM and ISM, and gets more effective when given more training sequences. With 10,000 training sequences, MSM reduces the average error by 60% over raw RFID data. Figure C.12(b) show the average error estimated on the block sequences, which gives a more qualitative measure of the effectiveness of the three models. MSM is the most effective as the effectiveness increases from 60% to 90% when increasing the number of the training sequences to 10,000. LSM and ISM show a similar trend, but at a lower overall level of effectiveness.

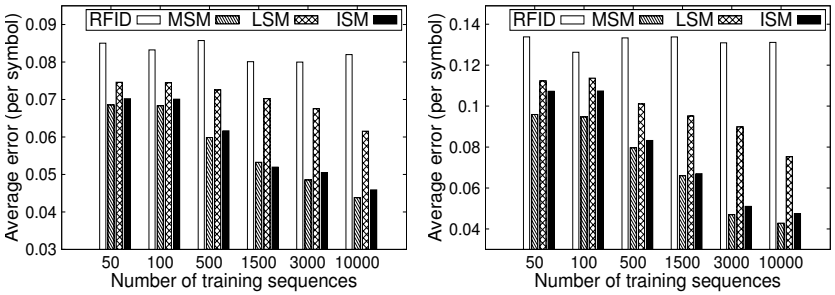
Figure C.13 shows results for the Directional-C data online case. We fixed a set of 1000 sequences as test data and learned the three models with different sets of training sequences, as shown in Table C.2. In Figure C.13(a), the results for original inferred sequences indicate that the learned models can reduce the average error in all cases. Again, MSM is more effective even in the online data case and the effectiveness increases with more training data. Figure C.13(b) shows the results for block sequences, indicating that MSM learned with a larger number of sequences is more effective as the average error is reduced by 70-75% for 10,000 training sequences.

The results in Figure C.14 are based on the Random movement offline case. Figure C.14(a) gives the results for original inferred sequences. Both



(a) Original sequences (b) Original sequences

Fig. C.13: Results on Directional-C online data

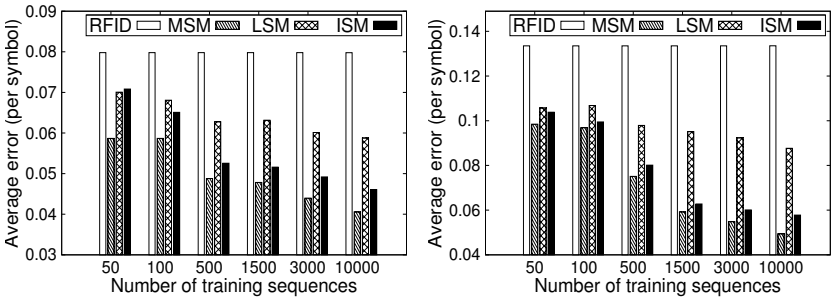


(a) Original sequences (b) Original sequences

Fig. C.14: Results on Random offline data

MSM and ISM perform better than LSM and reduce the average error with more training sequences, which implies that the more data there is available for learning, the higher the cleansing effectiveness. Figure C.14(b), with results for block inferred sequences, show a similar trend, i.e., the average error shrinks with larger training sets. The likely reason for MSM to be better is its design, which represents the underlying reader deployment more closely than the other two models (ISM is a low-level abstraction of MSM). Additionally, MSM can do the learning with less states.

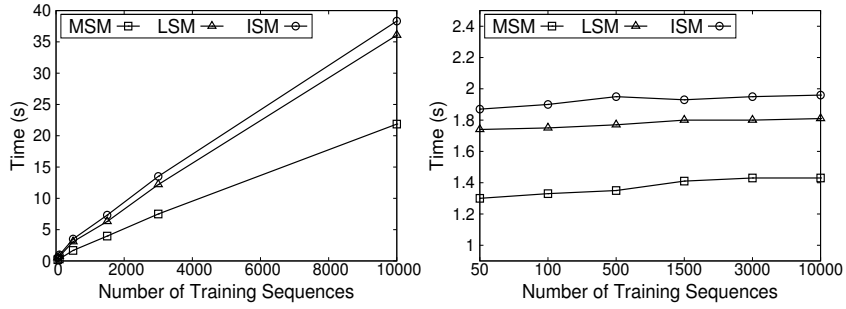
Results for the Random movement online case are shown in Figure C.15. Here, experiments are performed on 1000 test sequences to test the effectiveness of the three models learned with a different set of data sequences. The results for both original inferred sequences and block sequences show a similar trend of increasing effectiveness with more training data. The effectiveness is slightly less than that in the offline case, but still very good. Again, MSM is the best model, achieving 50-60% error reduction.



(a) Original sequences (b) Original sequences

Fig. C.15: Results on Random online data

Efficiency: For the efficiency measure, we present the time that a learned model takes to infer the results for a given data set. We use datasets with Directional-C and Random object movement with variable trajectory lengths. The results report the efficiency (inference time) of the learned models with respect to the number of sequences used to learn them. Figure C.16(a) and C.17(a) show the offline Directional-C and Random cases, respectively. We see that the total inference time grows linearly with number of training sequences for all the models. This is expected, since the Viterbi algorithm takes $O(|V||S|^2)$ time. Specifically, MSM takes at most 22 seconds to learn and infer for both cases, while LSM and ISM use around 38 seconds.

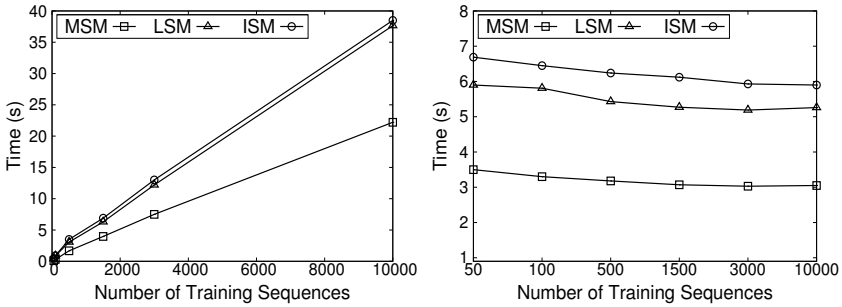


(a) Original sequences (b) Original sequences

Fig. C.16: Efficiency results on Directional-C data

For the online cases, Figure C.16(b) (Directional-C) and Figure C.17(b) (Random) report inference-only time (since the models are already learned) for inferring 1000 sequences. For Directional-C data, the inference time is stable, ranging from 1.3 to 2 seconds, with MSM being the fastest. For Random

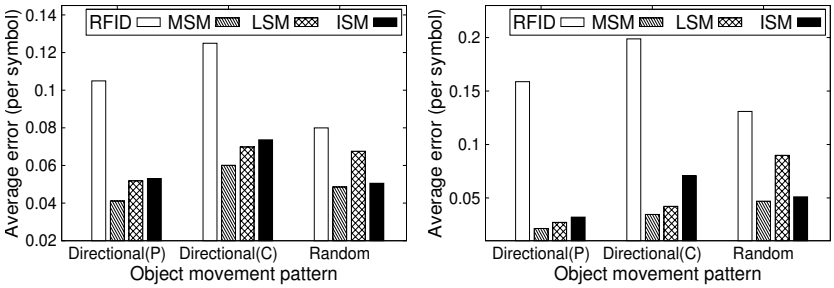
data, inference takes a little longer, from 3–7 seconds, because of the diversity and varying length of the trajectories. We omit the simple Directional-P case for brevity. In summary, learning and inference in the models are very efficient, with MSM being the fastest.



(a) Original sequences (b) Original sequences

Fig. C.17: Efficiency results on Random data

To see the effect of the movement patterns, we compare the models using a dataset of 3000 sequences with a reader accuracy of 70%. The results, in Figure C.18, show that the movement pattern has a significant effect on model effectiveness and that the three models perform differently for different movement patterns. MSM is robust across all three movement patterns and as before outperforms the other two, particularly for directional movement. LSM is more effective in Directional-C, while ISM is more effective for Random movement. All three models perform better on the simple Directional-P paths.



(a) Original sequences (b) Original sequences

Fig. C.18: Effect of object movement patterns

Effect of reader accuracy: In Figure C.19, we show the effectiveness of

the three models with respect to reader accuracy. Reader accuracy is defined as the ability to detect an object correctly when it is inside its intended range. We use 3000 training sequences with the Directional-C movement pattern and accuracy levels of 70% (80%, 90%, resp.), i.e., 30% (20%,10%, resp.) inaccurate readings are introduced into the raw data as equal amounts of false negatives and false positives. These accuracy levels correspond both to typical real-world accuracies and the range of accuracies which is practically relevant for cleansing (close to 100% there is no need for cleansing and below 50-60% it becomes random "coin-flip" scenarios). All three models, MSM, LSM and ISM, show their effectiveness as the error reduction rate only decreases slightly when reader accuracy decreases. We see that MSM is the most effective model, as it reduces the error by more than 80% for 70% accuracy, and 90% for 90% accuracy.

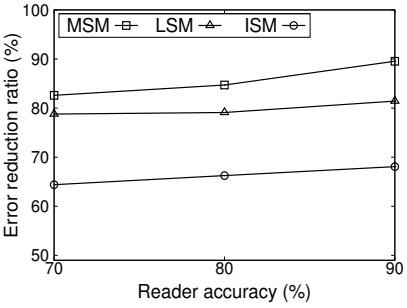


Fig. C.19: Effect of reader accuracy

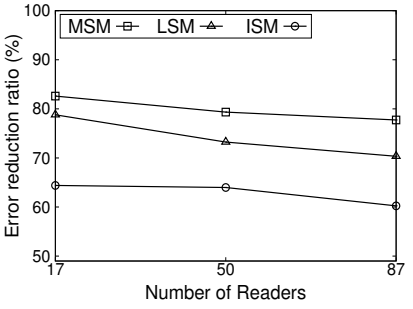
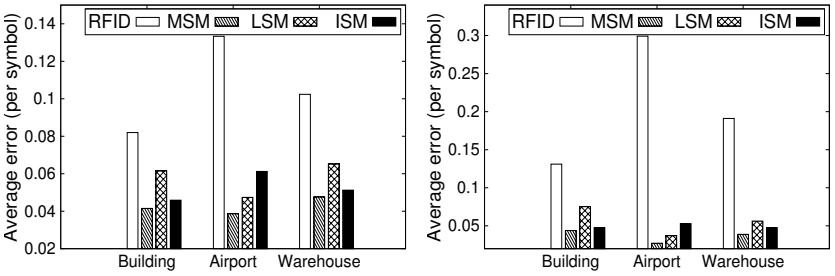


Fig. C.20: Effect of number of readers (state space size)

Scalability: To demonstrate the scalability of the three models, we change the following parameters: 1) total number of deployed readers, 2) connectivity between readers, and 3) the length of the path (the number of readers deployed from source to destination location). We used the floor plan in Figure C.1 for the 17 reader case. For the 50 and 87 reader cases, we adopted a floor plan of a multi story building, where the floors are connected by 4 staircases. For simplicity, we regard hallways and staircases as rooms, and staircase entrances as doors. We used a single floor plan for 50 readers and a two floor plan for 87 readers. We generated data sets for 3000 objects with Random movement in each setting and 70% reader accuracy. We learned the models and evaluated them using the error reduction ratio defined above. The results, in Figure C.20, show that our models can still achieve very good effectiveness even for large state spaces. For a deployment of 87 readers, we have 169 and 262 states for MSM and ISM, respectively. We see that both ISM and MSM achieve good effectiveness even as the number of readers increases, with MSM being the best at around 80% accuracy. Thus, our models scale well with larger reader deployments (larger state spaces). We note that these

numbers of readers correspond to typical real-world deployments. Very few readers will often give good results, but less need for cleansing, while more than 100 readers are not typical even in the largest deployments.

Versatility. To evaluate their versatility, we also test the three models in different indoor environments. We consider environments which differ in aspects like floor maps, reader placements, movement patterns, and connectivity. We changed the setting of these parameters with respect to three real world environments: an office building (Figure C.1), an airport scenario (Figure C.23), and a warehouse scenario [11]. For the building scenario, we used Random movement data with medium connectivity; for the airport scenario, we used Directional-C movement data with low connectivity; and for the warehouse scenario, we used Random movement data with high connectivity. We used data sets of 10,000 sequences with 70% reader accuracy for the three environments, and checked the effectiveness of the models on both original and blocked sequences.



(a) Original sequences

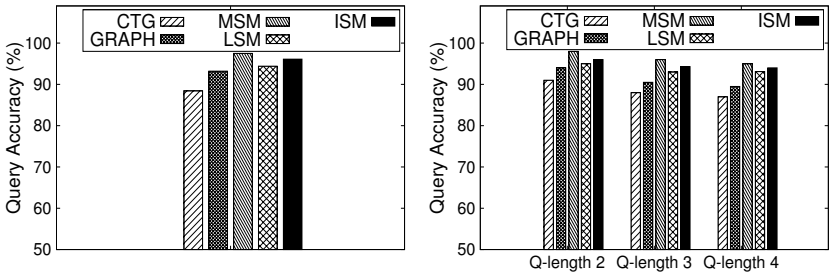
(b) Block sequences

Fig. C.21: Effectiveness of models in different environments

The results, in Figure C.21, show that the models perform differently in the three environments. One notable thing is that all three models are relatively more effective in the airport scenario, probably due to the rather constrained Directional-C movement of the bags. MSM manage to reduce the average error by more than 90% in the airport scenario, and also show significant effectiveness for the building and warehouse scenarios. MSM is the simplest of the three models, which facilitates the learning with a smaller number of states. Further, MSM models the floor map more closely than the other two models. Another effective model is ISM which is almost as effective as MSM in the building and warehouse scenarios, which use Random movement data. ISM has more states, but models the paths at a lower level of abstraction, which can be useful in scenarios where the objects move in a random pattern.

Comparison with state-of-the-art approaches: We now compare our models with the two state-of-the-art cleansing approaches proposed in [1, 2], and [3]. The two approaches are close in terms of their use of constraints of the indoor settings. The proposal in [3] (denoted as CTG) does not correct errors in the raw data (as we do) but only maps the raw readings to semantic locations. Put simply, CTG works at a higher level, above the raw data level which our technique targets. Thus, we cannot use the edit distance measure for comparison with CTG, and instead use a query-based comparison, detailed below. Particularly, CTG exploits the so-called constrained trajectory graph (CTG) that represents the trajectories in presence of integrity constraints (reachability, traveling time, and latency) to distinguish valid and invalid trajectories. In contrast, the GRAPH approach [1, 2] captures the domain knowledge (distance between readers, moving speed of objects, sampling rate, etc.) in a graph and works at the (low) level of the raw RFID data, as in the present paper. Rather than discarding invalid data, false positives are filtered out and false negatives recovered. For comparison, we use the results of our three model (MSM, LSM and ISM) over the Directional-C dataset.

To measure the quality of these approaches of different natures, we use two types of queries [3]: *stay queries* and *trajectory queries*. A stay query answers where an object is at time t, whereas a trajectory query asks if an object's trajectory matches a given query pattern. A trajectory query is defined as a combination of ?*s, l_is, and $l_i[n]s$. Specifically, ?* is a sequence of 0 or more locations, l_i is a sequence of at least one location(s), and $l_i[n]$ is a sequence of at least n l_is. The answer to a trajectory query is "*yes*" if the sequence of locations traveled by the object is the same as in the query pattern, or "*no*" otherwise. We issue the same set of queries on the outputs of CTG, GRAPH, MSM, LSM and ISM. For each, we measure the query accuracy by comparing the results with their counterparts obtained on the ground truth.



(a) Stay queries

(b) Trajectory queries

Fig. C.22: Query quality comparison results

Each trajectory is queried with 100 stay queries, each generated by picking a random time point of the trajectory. The stay query accuracy results in Figure D.11(a) show that the accuracy achieved by our three models is equal to or better than both CTG and GRAPH. A query accuracy of over 97% is achieved by MSM. This is remarkable, given that our models use only limited domain knowledge, unlike CTG and GRAPH. For trajectory queries, we generate 50 queries of length 2, 3, or 4 for each trajectory randomly. The trajectory query accuracy results, in Figure D.11(b), again show that our approach is better even with trajectory queries of different lengths. The accuracy is over 90% for both MSM and ISM, even when the trajectory length is 4. As expected, trajectory query accuracy decreases slightly when the length of query gets longer, since a small mismatch marks a whole query as failed.

The performance of our learning-based models (MSM and ISM) is remarkable, since they achieve similar or even better results without using extensive domain knowledge, unlike GRAPH and CTG. Given more training data, our models can learn their parameters better, and become more effective than the competing approaches which are in some sense based on hard-coded parameters (domain knowledge).

5.3 Experiments on Real Data

Finally, we use a real data set collected from an airport that operates an automatic RFID-based baggage handling system. As shown in Figure C.23, the baggage handling system features a number of specific locations (e.g., check-in desks) and RFID readers (shown with red circles). During the check-in phase, each (randomly selected) bag is attached with a passive RFID tag which is to be detected by the deployed RFID readers. From the 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. The bags 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). Finally, bags pass through a belt-loader reader while loading them into a plane. On the other hand, arrival bags are carried by wagons from an aircraft to the arrival hall (AR) through gateway GW-2 where baggage handling staff unload the bags from the wagons onto the arrival conveyor belt. We have continuously recorded the data for two consecutive months, collecting more than half a million raw reading records for about 20,000 RFID tagged bags going from one airport to another.

We cannot use the real data directly in the evaluation, since we do not have the related ground truth (the true reading values) available. Instead, we learn the characteristics of the real RFID data, such as the time bags use to move from one reader to another, the probability with which the bags

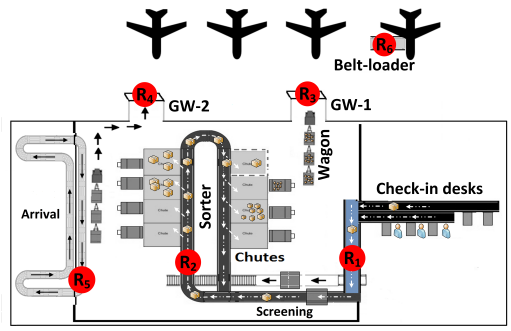


Fig. C.23: Airport baggage hall layout

move from one reader to another, and the error characteristics. We use this information to generate "semi-real" raw data which is equivalent to the real dataset, along with ground truth. Since the data is generated by readers deployed over a baggage handling system which is mostly covered by directional conveyor belts (directional movement), and the airport layout information is available to us, the MSM model is the most appropriate to use. Alternatively, LSM and ISM could have been used if the layout information is either difficult to use (e.g., arena scenario) or is not (readily) available.

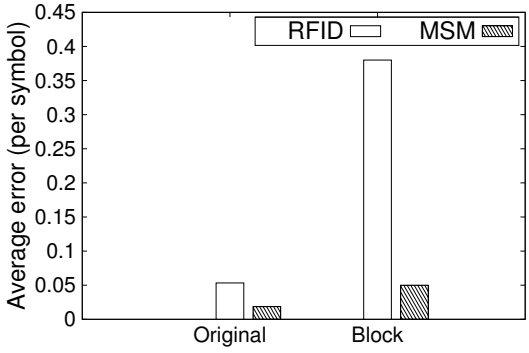
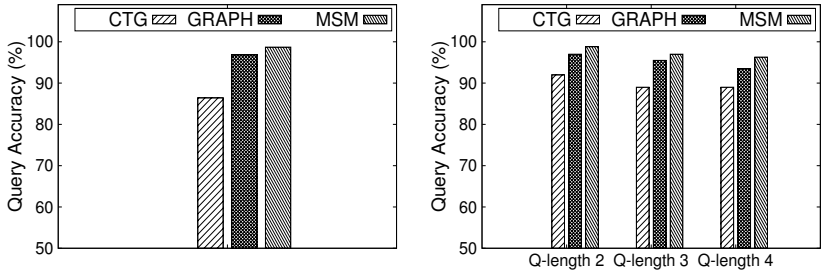


Fig. C.24: Effectiveness results on real data

In Figure C.24 we report the results on the effectiveness of MSM, learned from data generated by 20,000 moving bags with RFID tags. The results, in Figure C.24, show that the average error is reduced by 65–70% when applied to original sequences and more than 90% when applied to block sequences, which validates the effectiveness of the model. Query comparison results for

MSM, CTG and GRAPH, based on the stay and trajectory queries described above, are presented in Figure C.25. The figure shows that MSM achieves the best accuracy results for both stay and trajectory queries. An accuracy of over 98% is achieved for stay queries, and trajectory queries of length 2. The overall results are similar to the results on synthetic data: MSM is particularly effective considered that no detailed domain knowledge is used, unlike CTG and GRAPH. This again shows that learning from larger data sets can outperform the effect of using *hard-coded parameters*.



(a) Stay queries

(b) Trajectory queries

Fig. C.25: Query quality results on real data

6 Related Work

In this section we briefly review the related work. We cover RFID data management/cleansing and relevant learning probabilistic models. Dealing with indoor RFID data is quite challenging. The techniques to represent RFID data along with the methods for accessing the data have been proposed in [13, 14]. In this paper, we focus on the challenge of dealing with the errors present in the RFID data and designing accurate probabilistic models.

In the past decade, considerable research interest has been seen in managing incomplete and uncertain RFID data. Chawathe et al. [15] discussed RFID data management challenges, such as inferences and online warehousing and proposed a system architecture of a distributed RFID system. Wang et al. [16] defined an expressive temporal data model for RFID data to support tracking and monitoring queries. Gonzalez et al. [14] designed a warehousing model that provides RFID data compression and path dependent aggregates. Cabenes et al. [17] used an unsupervised learning approach for mining RFID data, which allows the discovery of a topological space from a set of behavioral observations.

Because RFID data is inherently dirty, many solutions have been proposed

to clean the RFID data. Jeffery et al. [18] proposed the adaptive SMURF (*Statistical sMoothing for Unreliable RFid data*) approach. Modeling the RFID data streams as statistical samples of RFID tags, SMURF uses sampling estimators to automatically adjust the filter window size. Mylyy [19] proposed 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. Chen et al. [20] proposed a Bayesian inference based approach, which takes full advantage of data redundancy, for cleaning RFID raw data. The likelihood is captured by designing a state detection model. Yan et al. [21] proposed a Kalman filter based cleaning method, which solves false negatives and false positives from single readers. Gu et al. [22] proposed a data imputation model for RFID by efficiently maintaining and analyzing the correlations of the monitored objects. The spatio-temporal correlation between monitored objects are used to fill in the false negatives.

Baba et al. [2] [1] proposed an RFID data cleansing solution which utilizes indoor deployment graphs that capture the information about indoor settings and the deployed readers. Different versions of graphs are used to handle either false positives or false negatives. In [1] a cleansing algorithm based on distance-aware graph model is proposed to handle false positives in indoor RFID tracking data and a probabilistic graph is proposed in [2] to handle false negatives. Fazzinga et al. [3, 23] proposed a probabilistic framework and a grid based filtering scheme to clean the RFID data. Their solution uses integrity constraints implied by the map and the mobility characteristics are applied to cleanse the RFID data.

Unlike the work [1–3, 23], the present work does not assume that we know the indoor space spatio-temporal constraints or the detailed characteristics of the deployed readers beforehand. Nevertheless, the proposed approach in this paper achieves essentially the same or even better results without using detailed domain knowledge and thus making it much easier to deploy in practice. The learning with larger data sets tends to improve the accuracy of learned parameters and bring forth some of the intricacies which are hidden in the data in normal views.

We presented a comparison with related HMM-based probabilistic RFID data processing solutions in Table C.3. Nie et al. [5] proposed Correlated Variable Duration Hidden Markov Models to capture uncertainty and correlations of locations of tagged objects. Different from our models proposed in this paper, that probabilistic model design [5] depends on additional information apart from raw RFID data and it assumes one object's locations depend on its neighboring objects. Garcia-Valverde et al. [12] proposed a location prediction of a worker based on its destination. The RFID data used to learn the model is considered as reliable unlike our model which uses raw RFID data to learn the model parameters. Tran et al. [4] proposed a prob-

abilistic model that captures the mobility of a reader, object dynamic, etc., and makes inference based on particle filter to infer object locations from raw RFID streams. In contrast, our model proposed in this paper does not demand such domain-specific prior knowledge about the data and environment.

As a final remark, although the possible world model [24] has been widely used in managing uncertain databases, it is not suitable for our research problem in this paper. Determined by the underlying RFID tracking system, the raw indoor RFID data we need to cleanse cannot be captured as samples with probabilities.

7 Conclusion

In this paper, we presented a learning-based approach to clean raw RFID data. Specifically, we proposed the the Indoor RFID Multi-variate Hidden Markov Model (IR-MHMM) to model the uncertainties present in RFID data as well as the correlation of moving object locations and RFID tagged objects. Compared to related approaches, our approach is almost entirely unsupervised: the only information needed to construct the IR-MHMM model are variants of basic information on the spatial reader deployment, depending on one of the the proposed state space designs. Training of the model is performed using only raw RFID data; no special labeled data or ground truth trajectory data is required. We performed a comprehensive experimental study using both real and synthetic data. Results show that our approach is effective, achieving up to 90% error reduction over raw RFID data, as well as robust, scalable, and efficient. Given enough training data, it outperforms competing state-of-the-art approaches.

In future work, we plan to mitigate the limitations imposed by the Markov assumptions in our model by incorporating a probabilistic timing model to relate the travel time of objects between readers and the time objects spends at each reader. Secondly, we plan to incorporate domain-knowledge into our learning approach, which could possibly improve the effectiveness further.

References

[1] A. I. Baba, H. Lu, X. Xie, and T. B. Pedersen, "Spatiotemporal data cleansing for indoor RFID tracking data," in *Mobile Data Management*, 2013, pp. 187–196.

[2] A. I. Baba, H. Lu, T. B. Pedersen, and X. Xie, "Handling false negatives in indoor RFID data," in *MDM*, 2014, pp. 117–126.

[3] B. Fazzinga, S. Flesca, F. Furfaro, and F. Parisi, "Cleaning trajectory data of RFID-monitored objects through conditioning under integrity constraints," in *EDBT*, 2014, pp. 379–390.

[4] T. T. L. Tran, C. A. Sutton, R. Cocci, Y. Nie, Y. Diao, and P. J. Shenoy, "Probabilistic inference over RFID streams in mobile environments," in *ICDE*, 2009, pp. 1096–1107.

[5] Y. Nie, Z. Li, S. Peng, and Q. Chen, "Probabilistic modeling of streaming RFID data by using correlated variable-duration hmms," in *SERA*, 2009, pp. 72–77.

[6] L. R. Rabiner and B. H. Juang, "An introduction to hidden markov models," *IEEE ASSp Magazine*, 1986.

[7] S. Kirshner, "Modeling of multivariate time series using hidden markov models," Ph.D. dissertation, Long Beach, CA, USA, 2005, aAI3164062.

[8] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 2, pp. 260–269, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1109/TIT.1967.1054010

[9] V. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Soviet Physics Doklady*, vol. 10, p. 707, 1966.

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

[11] J. S. Larson, E. T. Bradlow, and P. S. Fader, "An exploratory look at supermarket shopping paths," *International Journal of Research in Marketing*, vol. 22, no. 4, pp. 395 – 414, 2005.

[12] T. García-Valverde, A. García-Sola, and J. A. Botía, "Improving RFID's location based services by means of hidden markov models," in *ECAI*, 2010, pp. 1045–1046. [Online]. Available: http://dx.doi.org/10.3233/978-1-60750-606-5-1045

[13] 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*, 2005, pp. 1140–1151.

[14] H. Gonzalez, J. Han, X. Li, and D. Klabjan, "Warehousing and analyzing massive RFID data sets," in *ICDE*, 2006, p. 83.

[15] 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*, ser. VLDB '04. VLDB Endowment, 2004, pp. 1189–1195. [Online]. Available: http://dl.acm.org/citation.cfm?id=1316689.1316791

[16] F. Wang and P. Liu, "Temporal management of RFID data," in *VLDB*, 2005, pp. 1128–1139.

[17] G. Cabanes, Y. Bennani, and D. Fresneau, "Mining RFID behavior data using unsupervised learning," *IJAL*, vol. 1, no. 1, pp. 28–47, 2010.

[18] S. R. Jeffery, M. N. Garofalakis, and M. J. Franklin, "Adaptive cleaning for RFID data streams," in *VLDB*, 2006, pp. 163–174.

[19] O. Mylyy, "RFID data management, aggregation and filtering," in *R. Meersman and Z. Tari (Eds.): CoopIS/DOA/ODBASE 2005, LNCS 3760*, 2007, pp. 557–575.

[20] H. Chen, W.-S. Ku, H. Wang, and M.-T. Sun, "Leveraging spatio-temporal redundancy for RFID data cleansing," in *SIGMOD Conference*, 2010, pp. 51–62.

[21] W. Yan and S. B. Yan., "Cleaning method of RFID data stream based on kalman filter." Journal of Chinese Computer Systems, 2011, pp. 1794–1799.

[22] Y. Gu, G. Yu, Y. Chen, and B. C. Ooi, "Efficient RFID data imputation by analyzing the correlations of monitored objects," in *DASFAA*, 2009, pp. 186–200.

[23] B. Fazzinga, S. Flesca, F. Furfaro, and F. Parisi, "Offline cleaning of RFID trajectory data," in *SSDBM*, 2014, p. 5.

[24] N. N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases," in *VLDB*, 2004, pp. 864–875. [Online]. Available: http://www.vldb.org/conf/2004/RS22P1.PDF

	Training Data	Prior Knowledge	Errors Handled	Output
IR-MHMM	Raw RFID data	Indoor topology and device deployment	False positives and false negatives	Clean observation sequences
CVD-HMM [5]	Raw RFID data	Topology, device deployment, flow process of moving objects, and correlation between objects	False positives and false negatives	Inferred locations
García-Valverde et al. [12]	Clean RFID data	Indoor topology, source and destination of trajectories	–	Location predictions for fixed destination
Tran et al. [4]	Raw RFID data	Mobility information of readers, object dynamics	False positives and false negatives	Event streams

Table C.3: Comparison with state-of-the-art HMM-based probabilistic RFID data processing

Paper D

From Raw RFID Data to Most Probable Indoor Paths: An Approach Based on Regular Expression Matching

Asif Iqbal Baba, Hua Lu, Wei-Shinn Ku and Torben Bach Pedersen

The paper is submitted to
ACM SIGSPATIAL 2016.

Abstract

RFID (Radio Frequency Identification) based object tracking is increasingly deployed and used in indoor environments such as airports, shopping malls, etc. However, the inherent noises in the raw RFID data makes it difficult to support queries and analyses on the data. In this paper, we propose an approach to map raw RFID data to the most probable indoor paths, i.e., the most likely paths an object with RFID tag may have taken. We use regular expressions to model RFID data and possible indoor paths, and construct an automaton to capture all possible indoor paths in regular expressions. Furthermore, we design a probabilistic error model to represent the errors between raw RFID data and semantic symbols in indoor paths. Given the raw data of an object, the proposed approach uses the automaton to find the most probable indoor paths according to the error model. We evaluate the proposed approach by conducting experimental studies on both real and synthetic datasets. The results demonstrate the effectiveness of the propose approach.

The layout has been revised.

1 Introduction

In most of the tracking and monitoring application, a passive RFID tags are attached to objects (e.g., check-in bags in an airport or items in a store) and the RFID readers are deployed at several pre-selected places. Any deployed reader detects a moving object when it goes through its detection range. Every time a reader detects an object it reports its presence by generate RFID data and sends it to the database. An RFID based system used by retailers and others can generate 100 to 1,000 times the data of a conventional bar code system. One of the leading retailer, Walmart expects to generate more than 7 terabytes of raw RFID data per day [16].

A key problem with raw RFID data is its quality. The raw RFID data is generated by unreliable devices and communicated through inconsistent radio frequency channels, and the indoor environment in which the devices operate are quite dynamic. These and many other reasons can bring about errors in raw RFID data. Typically, raw RFID data contains missing readings (false negatives) and cross readings (false positives). When any RFID reader does not produce any reading when a moving objects passes through its detection range, we say a missing readings are present in the data. Missing readings are mainly formed because of mal-functioning of reader hardware, noisy RF signals, etc. In contrast, cross readings are created in the data when a tag is detected by a reader which was intended to read it. The possible factors generating cross readings are the reflection of the signals from metal items, inconsistent RF signal, etc.

The errors in the raw data cause problems when the data is used for high level applications like querying and analyzing. For example, it is necessary to find or determine the movement paths of moving objects in many indoor scenarios like airport baggage handling and indoor item monitoring. For a particular object, its raw RFID data may indicate many possible paths due to the errors inherent in the data. For example, check-in bags at an airports are quite often sent to wrong planes particularly when changing planes happens. A passenger may want to know if her/his checked-in bag is loaded in the right plane. Consider the airport baggage handling scenario shown in Figure D.1. There are three check-in desks each covered by an RFID reader R_1, R_2, or R_3, respectively. There are two planes, and RFID readers R_8 and R_9 covers their doors respectively. A check-in bag scheduled to R_8's plane may be sent to R_9's plane due to mishandling.

In this paper, we apply an approach based on regular expression matching to map raw RFID data to most probable indoor paths—the most likely routes an indoor moving object with an RFID tag may have taken. Similar approaches have been used in data mining, knowledge discovery and molecular biology [2, 18], but not in the context of handling indoor RFID data.

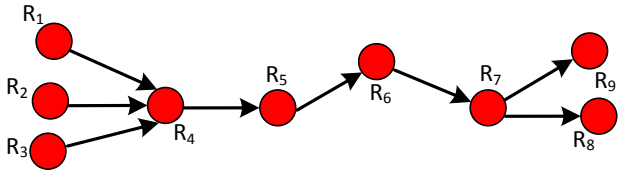


Fig. D.1: Example indoor route and layout

In our setting, the raw RFID data consists of records in the form of a tuple$\langle readerID, objectID, t\rangle$, which have reader identifier, object identifier and time. On the other hand, semantic indoor locations are identified and represented as symbols in a given indoor context, e.g., security check, tax-free shops, passport checks, and boarding gates at an airport. Our goal is to map an object's raw RFID data to possible sequences of symbols that are likely to be the object's movement paths of semantic locations. If there are cross readings and missing readings present in the raw data, it is difficult to map the raw data into symbol sequences that are perfectly matching or sufficiently close to the actual paths an object has moved along. Therefore, the major challenge in our work is to achieve accurate mapping results from raw RFID data to indoor paths in the presence of errors in the data.

To address the challenge, we design an approach based on regular expression matching. The approach is composed of three main components: an automaton that accepts regular expressions as input, and an error model that captures the distribution of possible errors in the regular expression matching, and relevant matching algorithms that map raw RFID data to probable indoor paths as sequences of symbols.

Refer to the example shown in Figure D.1 again. The baggage handling system starts with three readers (R_1, R_2 and R_3) at three check-in desks, and continues with a route through readers R_4, R_5, R_6 and R_7 until it takes one of the two belt-loader readers (R_8 or R_9). The example route is represented by a regular expression pattern $p_1 = (R_1|R_2|R_3)0R_40R_50R_60R_70\ (R_8|R_9)$. The pattern starts with $(R_1|R_2|R_3)$, which indicates that a bag can start from either R_1, R_2 or R_3, followed by value 0, indicating that the bag is seen by no reader. After that, a single occurrence of sub-pattern $R_40R_50R_60R_70$ is before sub-pattern $(R_8|R_9)$. Suppose that a bag's raw RFID data is captured as an input string $s = R_10R_40R_50R_6\ 0R_70R_8$, then we can say there is a match of pattern p_1 in s. In particular, $R1$ of s matches the sub-pattern $(R_1|R_2|R_3)$, R_8 at the end of s matches the sub-pattern $(R_8|R_9)$, and in the middle there is a single sub-pattern $R_40R_50R_6\ 0R_70$ in s. Therefore, there is a match of p_1 in s. Actually, the string s means that the bag was checked in at desk R_1 for a flight with the plane covered by R_8.

The paper make following contributions. First, we propose the design and

construction algorithm for an automaton that captures all possible indoor path in regular expressions. Second, we design an error model that describes possible errors between raw RFID data and semantic symbols probabilistically. Third, we give matching algorithms that use the automaton and the error model to find the most probable indoor paths for given raw RFID data. Fourth, A thorough experimental study is done using synthetic and real data sets to justify that the proposals presented are working.

The paper is organized as follows. Section 2 presents the preliminaries on indoor RFID positioning and problem formulation. Section 3 proposes the automaton design and its construction algorithm. Section 4 details the error model. Section 5 presents the matching algorithm for mapping the raw RFID data to the most probable indoor paths. Section 6 reports on extensive experimental studies. Section 7 reviews related work. Section 8 concludes the paper and discusses future work directions.

2 Preliminaries

2.1 RFID Based Indoor Positioning

To describe the RFID based positioning and tracking in indoor environments, the floorplan present in Figure D.2 is used as an example. The floor plan have several partitions including rooms, halls, corridors and stairs. Each partition is identified with a unique symbolic number [1, 3, 25] and may have one or more entrances (doors). Each entrance of the partition is deployed with an RFID readers represented by a solid circles in red color and a dashed circle around each red solid circle indicates the range of each reader.

RFID based indoor positioning in our setting employs the proximity analysis. An object is only seen when it is within an RFID reader's detection range. The reader identifier in the raw RFID reading refers to the location where object is seen by the reader. Each deployed RFID reader try to detects objects in its range at each epoch and generates report about objects presence. The frequency of epochs are determined by its sampling frequency. Using the information recorded at each epoch (reading cycle) a trajectory $\mathcal{T} = \langle rr_1, rr_2, ..., rr_x \rangle$ is formed, which is a sequence of records of a particular moving object. Each record rr_i specifies both spatial and temporal information of a moving object, in a tuple form $\langle readerID, objectID, t \rangle$. Such a record contains an information about the reader, object and the time information when object passes through the reader. With the help of an indoor map, each RFID reading can be mapped to some location(s) in the underlying floor plan.

For example, consider a moving object $object_1$ that moved from outside to room 10 shown in Figure D.2. The movement generated the raw RFID data shown in Table D.1.

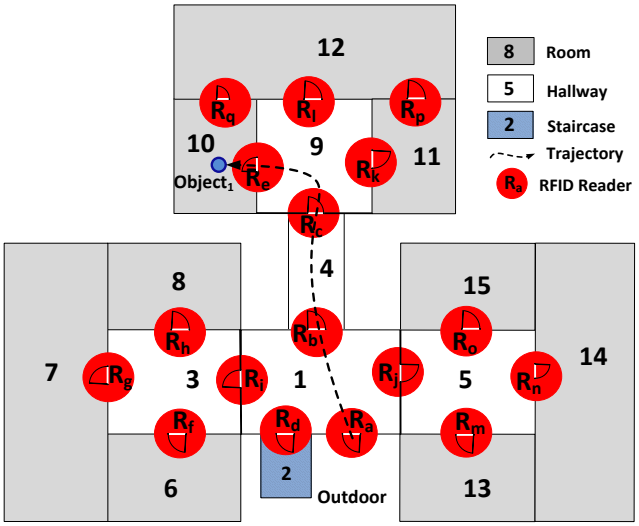


Fig. D.2: Example floor plan

Table D.1: Raw Reading Table

readerID	objectID	t
r_a	$object_1$	t_0
r_a	$object_1$	t_1
r_d	$object_1$	t_1
r_a	$object_1$	t_2
r_d	$object_1$	t_2
r_a	$object_1$	t_3
r_b	$object_1$	t_5
r_b	$object_1$	t_6

readerID	objectID	t
r_b	$object_1$	t_7
r_b	$object_1$	t_8
r_b	$object_1$	t_9
r_b	$object_1$	t_{10}
r_e	$object_1$	t_{19}
r_e	$object_1$	t_{20}
r_e	$object_1$	t_{21}
r_e	$object_1$	t_{22}

2.2 Problem Formulation

Table D.2 lists the notations used throughout the paper.

Given an object o_i, we use $\mathcal{T}_{o_i} = \langle rr_1, rr_2, ... \rangle$ to denote its trajectory captured by the RFID based positioning, where each rr_i is in form $\langle readerID, objectID, t \rangle$ and $\mathcal{T}_{o_i}$ is ordered by time t such that $r_i.t \leq r_{i+1}.t$.

We convert each $\mathcal{T}_{o_i}$ into a character string S_{o_i} by mapping the observed readings to characters in an *alphabet* Σ. Specifically, Σ contains a unique character for each deployed reader, e.g., $R_1 \rightarrow a$, $R_2 \rightarrow b$, and so on. For simplicity of presentation, the deployed reader in Figure **??**, namely $R_a, R_b, ...$, are represented by their subscripts. For example, a simple RFID sequence $(R_a, R_a, R_b ...)$ will be represented as $aab....$ We use the terms sequence and

Table D.2: Notations

Notation	Meaning
R_i, R_j	RFID readers
rr, rr_i, rr_j	Raw RFID readings
RRT	Raw reading table
Σ	the alphabet
R_{o_i}	Raw RFID data sequence of object o_i
S_{o_i}	the string representing R_{o_i}
dt_i	The minimum dwell time of reader r_i
S_{f_i}	a sampling frequency of reader r_i
TT_{ij}	travel time between r_i and r_j
RRE	RFID based regular expression
π, Π	path, set of paths
$\mathcal{E}$	error model

string interchangeably, and use characters and symbols interchangeably in the paper.

Time series data are often discretized for data analysis purpose. Also, some time series data like RFID data may be collected at different sampling rates, depending on the concrete configuration of a deployed reader. To avoid excessive granularity, we introduce a *time granularity parameter* (α) to discretize the continuously observed raw RFID data. We represent a time interval when no reader generates any reading as a special character "0" and adds it to the alphabet Σ.

We refer to the raw RFID data sequence about moving object $object_1$ in Table D.1. Assuming a time granularity α is 1 time unit, $object_1$'s corresponding string is:

$$S_{object_1} = (\mathrm{a}\overset{d\,d}{aa}\mathrm{a0bbbbbb00000000eeee})$$

Here, $\binom{d}{a}$ represents the time point where $object_1$ is detected by two readers R_a and R_d simultaneously.

On the other hand, we need to capture the possible paths that an object can move along in a given indoor environment. Such possible paths are captured as regular expressions. Since we derive such regular expressions from RFID readers in this work, we name them as RFID-based regular expressions or RRE for short. We use the terms regular expressions and patterns interchangeably in the paper. A more formal definition of RRE is given as:

Definition 12
(**RFID Regular Expression**) An RFID Regular Expression (RRE) is a string on symbol set $\Sigma \cup \{\epsilon, |, \cdot, *, (,)\}$. The strings are recursively defined as the

empty string ϵ, a symbol $\alpha \in \Sigma$, (RRE_1), $(RRE_1 \cdot RRE_2)$, $(RRE_1 \mid RRE_2)$, or $(RRE_1)^*$, where RRE_1 and RRE_2 are RREs.

Consider the running example in Figure D.2 for object $object_1$. Each reader in the path generates n readings depending on sampling frequency about the moving object. Assuming that the travel time between any two directly connected readers is 2 time units and sampling frequency of each reader is the same, each generating 4 readings. The actual path (ground truth) by $object_1$ is represented by the following RRE.

$$GT_{object_1} = [a]\{4\}[00][b]\{4\}[00][c]\{4\}[00][e]\{4\}$$

Here, $[p]n$ indicates that the pattern p occurs n times.

Further, we need to model the possible errors between the observations about an object, i.e., the trajectory represented as RFID readings, and its ground truth path. We define the following error model for that purpose.

Definition 13
(**Error model** $\mathcal{E}$) Suppose r_i is an observed value, i.e., a character in an observed trajectory, and x_i is the true value, i.e., the corresponding character in the ground truth path, the error between r_i and x_i is $e_i = (r_i, x_i)$. For any true value x_i, there may be multiple possible observed values. In general, the error e_i is a function of r_i and x_i, and the error model of a sequence is the joint distribution of the e_i's.

A simple error model is to assume an independent error distribution for each character in a string. The error e_i can be easily quantified by the difference between the actual and observed value if the relevant values are numeric. A detailed description of error model is to be given in Section 4.

In this work, we use an automaton of symbols to represent all possible paths in a given indoor environment. Given an object's trajectory observed as a sequence S_{o_i} of symbols, we compare S_{o_i} to all possible paths in the automaton and return the most probable indoor paths for the object. The process is done through RRE matching defined as follows.

Definition 14
(**RRE Matching**) Let a raw RFID data sequence $\mathcal{T}_{o_i} = \langle rr_1, rr_2, ... \rangle$ be represented by a string S_{o_i} according to Definition 12. The RRE matching is to compare S_{o_i} against an automaton representing ground truth sequences $\{p_1, p_2, ...\}$ and to return all the matching paths Π_i with their probabilities under an error model $\mathcal{E}$.

For an Illustration, we give an example, let the ground truth pattern $GT = [a]\{4\}[00][b]\{4\}[00][c]\{4\}[00][e]\{4\}$ and a snippet of the sequence be "$\overset{d}{a}\overset{d}{a}aa00bbbb$", where $(\overset{d}{a})$ implies a noise (false positive). For instance, given

an error mode $\mathcal{E}$, when the true value is 'a', the observation value $\binom{d}{a}$ is indeed 'a' with a certain probability of 0.8 and 'd' with the remaining probability of 0.2. When the whole sequence is traced, we report the two matching paths "*aaaa*00*bbbb*" and "*adda*00*bbbb*" with their matching probabilities. We can then used the probability threshold τ to get a top matching path(s). Our focus in this work is to find the top-k matching paths for a given RFID trajectory with probability at least equal to a given probabilistic threshold τ.

Problem Definition (Top-k Probable Indoor Path Matching) Given a raw RFID data sequence $\mathcal{T}_{o_i} = \langle rr_1, rr_2, ... \rangle$, an error model $\mathcal{E}$, a ground truth regular expression GT, a probability threshold τ, and an integer value k, the top-k matching problem is to report k sequences from GT that form a set of indoor paths Π_i, if all the following conditions are satisfied: 1) Each $\pi \in \Pi_i$ has rr_i as the last character; 2) Each $\pi \in \Pi_i$ has a matching probability at least τ for RRE; 3) $\Pi_i \neq \emptyset$. When $|\Pi_i| > k$, we remove all but the k matching paths with the highest probabilities from Π_i.

3 Automaton for RRE-matching

3.1 Regular Expression Generation

In this section, we generate a set of RFID based regular expressions (RREs) over an alphabet Σ as defined in Definition 12. These regular expressions permit a convenient specification of the true traces of movement patterns of moving objects in a given floor plan. In order to recognize RFID observation sequences of indoor moving objects, we represent these regular expressions in an automaton.

Specifically, we first generate the true traces of object movements in a given indoor space, then specify them as RREs. We use the indoor topology to derive the *connectivity* between a pair of indoor locations where RFID readers are deployed. We also used the *travel time (TT)* constraint, a time required to reached from one location to another. We derive the travel time from the distance between the readers and the speed with which the objects can move between the two readers. The other piece of required information is *sampling rate*, which can be derived from the minimum dwell time and sampling frequency of the deployed readers. It is readily available while deploying the RFID readers. Once all the required information is available, we can generate all the possible true traces of the object movements in the given indoor space, which subsequently takes a form of a set of regular expressions.

For an illustration, we consider three ground truth trajectories a moving object can possibly take form outside to reach room 12 in the floor plan shown in Figure D.2. Assuming that the travel time between any two directly connected readers is 2 time units, a minimum dwell time of a deployed reader is

4 time units, and the sampling frequency is $1s^{-1}$, i.e., one reading cycle per time unit [1]. The three possible ground truth trajectories from outside to room 12 are:

$$aaaa00bbbb00cccc00llll$$
$$aaaa00bbbb00cccc00eeee00qqqq$$
$$aaaa00bbbb00cccc00kkkk00pppp$$

According to Definition 12, these three expressions can be represented in a form of regular expression as follows.

$$[a]\{4\}[00][b]\{4\}[00][c]\{4\}[00]\Big([l]\{4\}|([e]\{4\}[00][q]\{4\}) |([k]\{4\}[0][q]\{4\})\Big) \tag{D.1}$$

The regular expression generation procedure is formalized in Algorithm 9. It explicitly takes as input the set of readers (R) along with their configuration settings, a time granularity parameter (α) and a $|R|\times|R|$ matrix (TT) containing the travel time between readers. The algorithm starts by looking at possible pairs of readers and uses procedure *findAllPaths()* from Algorithm 10 to get all the possible paths between each pair of readers (lines 2–4). After retrieving all the possible paths (*paths*), the algorithm finds a true trace (GT_{trace}) of readings when an object follows any particular path p from *paths*. We check each reader r in the path and use its corresponding parameter settings (minimum dwell time) and time granularity parameter (α) to find the exact number of *samples* of r we will have in GT_{trace} (lines 7–9). Next, we retrieve the travel time tt between current reader r and the next reader r' in the path. We use tt along with time granularity parameter α to add the number of non-reader state "0" to GT_{traces} (line 13–14). Once all the true traces are generated, we call a procedure *Regex()*, which applies a standard syntax of a extended regular expressions defined in literature [22] to represent the true traces as regular expressions (line 16).

Algorithm 10 gives the procedure *findAllPaths()*, which takes as input a source reader R_s, a destination R_d, and an adjacency matrix M that defines the connectivity between the readers. The algorithm returns all the paths between R_s and R_d [3]. The algorithm is typical backtracking algorithm, which follows improved depth-first search paradigm. The algorithm returns all the possible paths between two readers.

[1] For simplicity we consider a uniform travel time between devices and the same sampling rate for deployed devices in the example. Nevertheless, our solution can take flexible travel times and sampling rates.

Algorithm 9 RegularExpressionConstruction(Reader set R, Time granularity α, Travel Time matrix TT)

```
1: rrs ← ∅
2: for each reader r_i ∈ R do
3:     for each reader r_j ∈ R and r_j ≠ r_i do
4:         paths ← findAllPaths(r_i, r_j, TT)
5:         for each path p ∈ paths do
6:             GT_trace ← ∅
7:             for each reader r ∈ p do
8:                 samples ← r.dt / α
9:                 for 1 to samples do
10:                    append r's symbol to GT_trace
11:                r' ← the next reader in p
12:                tt ← TT[r][r']
13:                for 1 to (tt / α) do
14:                    append "0" to GT_trace
15:            add GT_trace to rrs
16: Regex(rrs)
```

Algorithm 10 findAllPaths(Source reader R_s, Destination reader R_d, Adjacent matrix M)

```
1: stack S ← ∅ ; neighbors ← ∅ ; paths ← ∅ ;
2: S.push(R_s)
3: if (R_s = R_d) then
4:     paths[] ← S
5: neighbors ← getNeighbors(R_s, M)
6: for each n in neighbors do
7:     if (n not in stack S) then
8:         findAllPaths(G, n, R_d)
9: S.pop()
```

3.2 Automaton Construction

In this section, we present the construction of an automaton, one of the building blocks of our approach. Any automaton can be constructed to represent what a corresponding regular expression recognizes. In our setting, we use regular expressions to represent the true moving patterns of objects in an indoor space where RFID readers are deployed to track the object movements. We use domain knowledge to generate the true movement patterns (without any error) of objects in an indoor space as described in Section 3.1.

For any given pattern, an automaton can be constructed with a Thompson's construction algorithm [27], the most classical construction algorithm

which leads to a non-deterministic finite automaton (NFA) with a linear number of states and the number of transitions. Before diving into the automaton construction, we give a formal definition of an NFA $\mathcal{A}$.

Definition 15
(**RFID Automaton**) An RFID based automaton is 5-tuple $\mathcal{A} = \{S, \Sigma, \delta, s, F\}$, where:

1. S is a finite set of states.
2. Σ is a finite set (alphabet) of input symbols. Our alphabet Σ contains the symbols which represent the deployed RFID readers. We also have unique symbol "0" as a non-reader symbol, which will lead to some non-reader state.
3. δ is a transition function from $S \times \Sigma_\epsilon \rightarrow 2^S$. Here, Σ_ϵ means $\Sigma \cup \{\epsilon\}$ and the result is a subset of all states.
4. $s \in S$ is a starting state.
5. $F \in S$ is a set of final state.

A constructed automaton is a transcription of the expression tree representation [2] of the regular expression. Also, the automation we use simplifies the transcription by using ϵ-transitions, transitions between states which do not absorb any input. Specifically, the regular expression is parsed in a binary expression tree in which each leaf is labeled by a set of characters in alphabet $\Sigma \cup \{\epsilon\}$ and each internal node represents the regular expression operator in the set $\{|, \cdot, *\}$.

For illustration, we present one of the sub-regular expressions of a regular expression shown in Equation D.1 in tree representation in Figure D.3.

A basic idea of the Thompson's algorithm is to construct an automaton for symbol nodes first, and then merge the created automata for each operation inductively. Some of the basic constructs used to build an NFA are shown in Figure D.4. The NFA is constructed by performing the bottom-up tree traversal until the root.

We can recursively construct any complex regular expression using the five constructs shown in Figure D.4 from given automata.

1) A construction of an empty word is shown in Figure D.4(a). The automaton consists of two nodes joined together by ϵ-transition.

2) A construction of a single character "a" is similar, except that the transition is labeled with a character as shown in Figure D.4(b).

3) The construction of two automata of the two children of a tree are merged together using a concatenation construct shown in D.4(c), the final state of the first automaton $\mathcal{A}_1$ becoming the initial state of the second automaton $\mathcal{A}_2$.

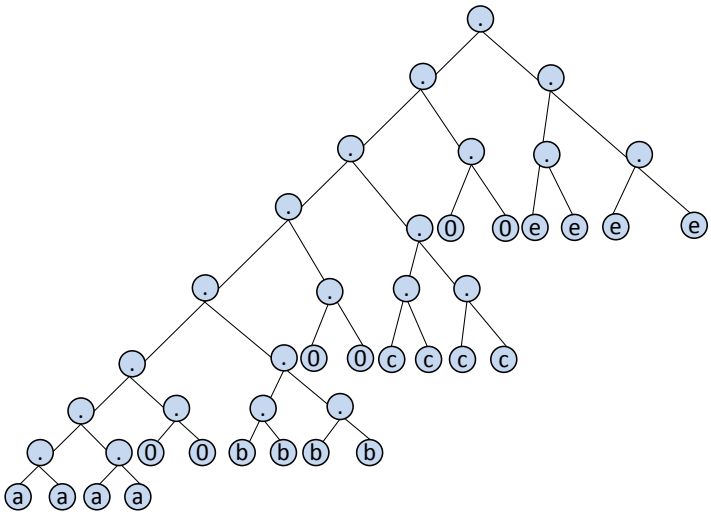


Fig. D.3: Tree representation of a sub-regular expression in Eq. D.1

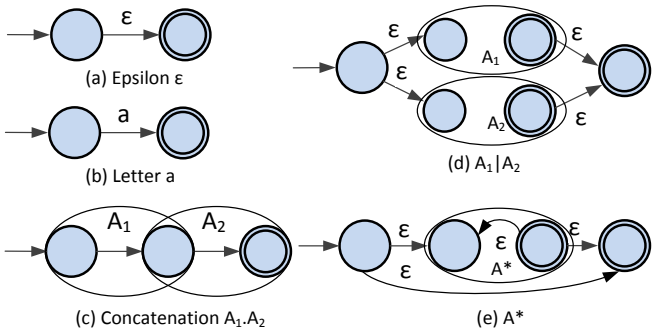


Fig. D.4: Thomson Automaton constructs [27] (a) Automaton accepting a letter 'ϵ' (b) Automaton accepting a letter 'a' (c) Concatenation between Automata A_1 and A_2 (d) A union between Automata A_1 and A_2, i.e., A_1 or A_2 (e) Kleen star (A*)

4) The construction of a union construct is shown in Figure D.4(d) which requires ϵ-transitions. The idea is to enter either of the two automata $\mathcal{A}_1$ or $\mathcal{A}_2$. Two new states are added: One initial state takes ϵ-transitions to the initial states of the two automata, and one final state accepts two ϵ-transitions from the final states of the two automata. A path from the new initial to new final state will always go through one of the automata.

5) The idea of constructing Kleen star shown in Figure D.4(e) is similar to the previous construct, by creating a backward ϵ-transition from the final

state of automaton $\mathcal{A}$ to its initial state. Since Kleen star also means that $\mathcal{A}$ can be ignored, we created two new states, an initial state and an ϵ-transition from it to the initial state of $\mathcal{A}$, and a final state to which we add an ϵ-transition from the final state of $\mathcal{A}$.

For an illustration, we use the aforementioned Thompson's constructs to construct the automaton for the regular expressions in Equation D.1. The full automaton is shown in Figure D.5. In particular, each state represents the location in which object may be present at a particular time point. The object appears to be first detected by reader represented by symbol 'a' and appears to remain there for four consecutive time units.

4 Error model

Error model is the distribution of errors at all characters in an alphabet. Each character represents an RFID reader in our case. Generally speaking, if we have an error at one reader, it is not certain that there must be an error at the next reader. Therefore, modeling the error at each character independently is an intuitive way.

As mentioned earlier about the inconsistent nature of radio frequency signals and the environments in which the RFID readers are deployed, the error sources may change over the time. Therefore, learning a model from raw data is not ideal. Model learned earlier may not be suitable after some time because of different error sources. On the other hand, learning a new model every time will not be feasible.

We utilizes the RFID reader deployment to design our error model $\mathcal{E}$. Specifically, $\mathcal{E}$ has n different error levels, corresponding to the n deployed readers (characters). We use the *distance* between the readers and the *detection range* of each reader to determine the error level independently for each character. We refer to each error level value as a *certainty value*, e.g., $e(r_i, x_i) = 0.1$, which means that with the certainty of 0.1 the observed value x_i is indeed the actual value r_i.

We record all the certainty values in the certainty matrix which is generated using the following formula:

$$e(r_i, x_i) = 1 - log_{10}(distance(d)/detection\ range(dt))$$

Here, d is the distance between the true reader and the reader which generates the observation reading, and dt is the detection range of the observation reader. In the model, we start with the true location (called the main reader), and then allocate the error levels to other readers according to their distances from the main reader. We can define certain threshold distance $\mathcal{D}$ value after which all the readers can be allocated to a fixed error level (very low certainty value or less likely to be an observation reader).

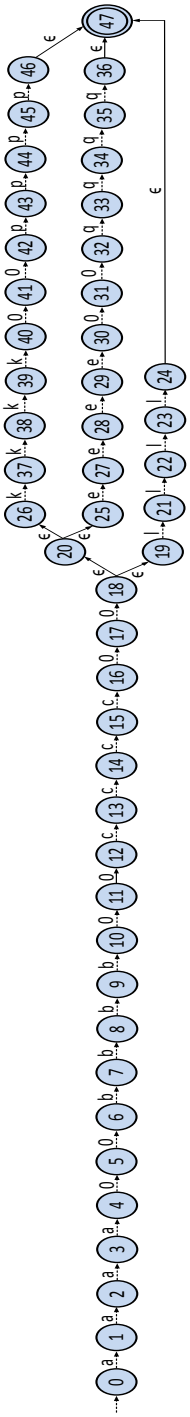


Fig. D.5: Example automaton of Equation D.1

Figure D.6 gives an example of certainty levels of a highlighted area that is from the RFID reader deployment shown in Figure C.1.

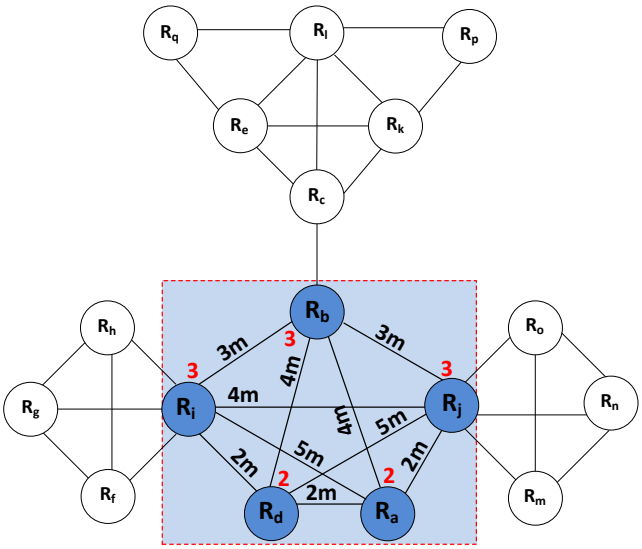


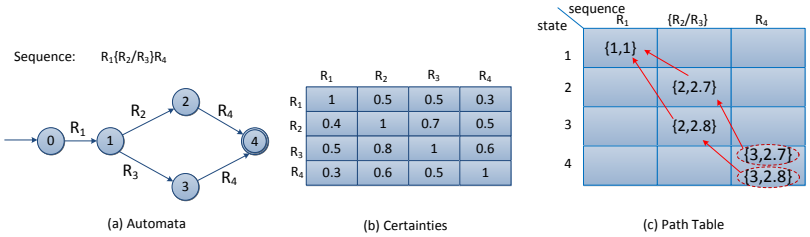
Fig. D.6: Distance graph of RFID data

For the readers R_a, R_b, R_d, R_i and R_j, we use each of them as the main reader one by one and estimate the certainty values based on the distances between the readers and the detection range of readers. Table D.3 shows the estimated certainty values. Here, the rows are true readers and the columns are the observed readers. Each table cell contains an certainty value. The cell value is 1 when the observed reader is the same as true reader. For example, the certainty value in cell (3,2) states that the observed reader R_d does indeed represent in the true reading by R_b with a certainty of 0.43. A moving object can be observed by more than one readers at the same time or in a period of time. Although for simplicity, we handle instances where an object is read by at most two readers at a time, our technique can be easily extended to handle the cases where object are observed by more than two readers. For that purpose, we only need to use a multi-dimensional cube instead of a matrix to store the certainty values.

Table D.3: Certainty matrix

observation \ true	R_a	R_b	R_d	R_i	R_j
R_a	1	0.43	0.52	0.36	0.63
R_b	0.30	1	0.30	0.52	0.52
R_d	0.52	0.43	1	0.63	0.36
R_i	0.22	0.52	0.522	1	0.43
R_j	0.36	0.52	0.52	0.43	1

5 RRE Matching



(a) Automata (b) Certainties (c) Path Table

Fig. D.7: Example RRE-MATCH: a) The sequence and the automaton. b) Table of certainty values c) Table of possible paths.

5.1 True States

We start with an instance of automaton $\mathcal{A}$ that is constructed from the RFID regular expression (RRE) in the query. Since the automaton constructed is Thompson's automaton and the Kleen star (Figure D.4(d)) can be removed without affecting the matching probability [26], the constructed automaton can be considered as DAG (directed acyclic graph). The total number of states of the constructed automaton $\mathcal{A}$ is $O|n|$, where $|n|$ is the number of characters and operators in the RRE. We categorize the states into two categories, *true states* and *empty states*.

Definition 16
(**True State T(s)**) We define a state to be a *true state T(s)* only if it has only one incoming edge with a letter on it.

Definition 17
(**Empty State** $\epsilon(s)$) We define a state to be an *empty state* $\epsilon(s)$, which has one or more incoming edges with epsilon ϵ on them.

As a special case, we consider a start state to be a true state T(s) with incoming edge with empty string ϵ. Each state of an automaton is assigned with a unique identification in a topological sort order, which will be used to maintain the order of the states during the *RRE* matching. We preserve a set of all the previous true states $PT(s)$ of each true state $s \in S$. I.e., for any $s' \in PT(s)$, we have s' as a true state and exactly one character is consumed for the input to move from s' to s. The previous true state list $PT(s)$ at each state will be used to check if the matching can be moved forward when a new character of an observed sequence arrives.

Algorithm 11 shows how to find true states. Taking an automaton $\mathcal{A}$ as input, it starts with initializing $PT(s)$ set to $\emptyset$ for each state s of $\mathcal{A}$ (line 2). To find the next state, it traverses each outgoing edge form s and checks if s is a true state. If so, the true state s is added to $PT(t)$. Otherwise, s is an empty state and $PT(s)$ is passed to $PT(t)$ (lines 4–7).

Algorithm 11 FindTrueStates(Automata $\mathcal{A}$)

```
1: for each state s of A in topological order ID do
2:     PT(s) ← ∅ ;
3:     for each next state t of s do
4:         if (s is a true state) then
5:             PT(t) ← {s} ∪ PT(t)
6:         else
7:             PT(t) ← PT(s) ∪ PT(t)
```

At each true state $T(s)$ of an automaton $\mathcal{A}$, a list of matching path segments are stored during *RRE* matching. After reaching the final state, a simple traceback can give us all the segments of the matching path, and thus there is no need to carry a whole path all along during the matching. Each matching segment of a path is $seg_x = \{j, p_{seg}, j_0, p\}$, where j is the index of an an observed sequence element which matches $T(s)$, p_{seg} is a pointer to the previous matching segment on this path, j_0 is index of the first element on this path, and p is the matching probability of the sub-path up to the current $T(s)$.

5.2 RRE Matching Algorithm

Our *RRE* matching algorithm finds all the possible matching paths incrementally with a character stream representing a raw RFID trajectory. Whenever a character o_j of a sequence o arrives, each true state i of an automaton $\mathcal{A}$ is triggered to check if the matching can be moved forward. However, if a mismatch occurs the path certainty value will soon decrease below the threshold (τ). The algorithm also maintains the list of all the partially matched paths at each true state i, where i is the identifier of a true state of the automaton $\mathcal{A}$.

The *RRE-MATCH* algorithm shown in Algorithm 12 takes as input an observation character sequence o, automaton $\mathcal{A}$, the certainty value table e and the threshold τ. It starts with finding all the true states at each state of $\mathcal{A}$ using a function *FindTrueStates* (line 1). In line 5, we start the matching path by checking if the previous state s is a starting state. We get the certainty value from table e for the first character in the matching path using an error function $e(o_j, \mathcal{A}_i)$. Intuitively, we use the certainty table for remaining matching characters in the matching path.

Further on, we bring the matching forward by triggering at the current true state, and then keep using the certainty values from e to maintain the overall certainty of the path until that true state (lines 11–15). Finally, after landing into the final state, all the paths found are reported. During the matching, each state keeps the segment of the path. When it is needed, a traceback can be used to recover the whole path, and the final certainty of a path remains at the final state.

Algorithm 12 RRE-MATCH(Observation Character sequence o, Automata $\mathcal{A}$, Certainty value table e, Threshold τ)

```
1:  FindTrueStates(A)
2:  for each character j in observation sequence o do
3:      for each true state i of A in topological order ID do
4:          for each state s ∈ PT(i) do
5:              if s is a starting state then
6:                  p ← e(o_j, A_i)
7:                  if (p < τ) then continue
8:                  p_seg ← (s, o_j, i)
9:                  path_i ← (j, p_seg, j, p)
10:                 continue
11:             for each path π = (j', p_seg, j_0, p) do
12:                 p ← e(o_j', A_s) + e(o_j, A_i)
13:                 if (p < τ) then continue
14:                 p_seg ← (s, o_j', i)
15:                 path_i ← (j, p_seg, j_0, p)
16:         if i is a final state then
17:             report path_i
```

In Figure D.7, We present a simple example to illustrate how RRE-MATCH algorithm works. An example pattern $R_1(R_2|R_3)R_4$ represents an automaton and an observed sequence is $R_1\{R_2/R_3\}R_4$, as shown in Figure D.7(a). In the observed sequence $R_1\{R_2/R_3\}R_4$, the noise is represented by $\{R_2/R_3\}$, which means readers R_2 and R_3 detected an object at the same time. As a result, it is unknown whether the object actually passes reader R_2, R_3, or some other reader. To find out the possible paths we utilize the certainty values

in the table shown in Figure D.7(b). We have three error levels for each true symbol and the possible observed values. Figure D.7(c) shows the path table that contains all the possible paths along with their certainty values. The rows represent states and the columns correspond to observed symbols. Each table cell contains the path segment(s). To ease the understanding, in each table cell we only show j (the index of the observed sequence element) and the certainty value until that element. We use dashed arrows to point to each previous segment and the red dashed circles to show the complete path. The certainty values are estimated as described in Algorithm 12. Taking the first segment at row 4 and column 3 in Figure D.7(c) as an example. The segment is created when the character at index 3 in the sequences arrives and we end up in state 4 of the automaton. Looking at previous true states of 4, i.e., $PT(4) = S\{2,3\}$. Taking $s = 2$ as an example, 2 is not a starting state, and we add up the certainty values together. State 2 has only one previous true state, i.e., $PT(2) = S\{1\}$, so there is only one path segment at the top-left of the table.

In this work we look for probable paths with probabilities. Without knowing the true values of an observed character, our matching algorithm uses certainty values, i.e., the certainty with which the observed characters is actually the true character. The algorithm chooses the certainty value. By adding up these certainty values at each true state, *RRE-MATCH* efficiently manages to calculate the certainty value of matched paths. The higher the certainty of the path is, the more likely it is a true path.

6 Experimental Studies

All algorithms are implemented in C++. The experiments are run on 2.6GHz Ubuntu with a main memory of size 8GB.

6.1 Performance Metrics

We use two metrics for our experiments. We measure effectiveness by *average error* [25] of a path is defined as

$$Average\ error = \frac{sum\ of\ edit\ distance\ scores\ for\ all\ sequences}{total\ \#\ of\ characters}$$

We use a variant of a traditional edit distance called a weighted edit distance [25] to check the similarity between the path(s) found by the proposed matching algorithm and the actual path. The similarity measure between two strings is calculated based on the number of insertion, deletion or substitution operations. Since we are using weighted edit distance, the cost of substitution depends upon the symbols that are inserted or deleted. Detailed description

about the approach can be found in [25]. We measured by counting clock time for efficiency.

To compare the results of our proposed algorithm with some existing works, we use stay queries and trajectory queries defined d in [4].

A *stay query*, which returns a location of a particular item at a given time-point t.

A *trajectory query* returns "yes" or "no" depending on if the trajectory query pattern matches the locations traveled by the object. A trajectory query is like regular expression with following sub-patterns, $?^*s$, l_is, and $l_i[n]s$. Pattern $?^*$ represents a set consecutive of zero or more locations, l_i represents a set of one or more location(s), and a pattern $li[n]$ represents a set of n location of l_i.

6.2 Synthetic Data Results

The synthetic data is generated using the parameters shown in Table D.4 where the defaults are in bold. The floor plan and the reader deployment shown in Figure D.2 is used as a base for data generation.The number of parameters including the number of objects, detection range, uncertainty ratio (i.e., fraction of uncertain readings or characters in an object's data), length of paths in the number of readers (i.e., pattern length), and the connectivity between the readers, etc, are varied during the data generation. By default, we generate RFID data about 5000 objects. Each object's moves with a constant speed of 1.1 m/s and follows a random waypoint model [13]. Specifically, any object can move within one partition or move to other partition. A shortest-path algorithm is used by moving object to move from one location to other location, while moving object may pass through several deployed readers.

We obtain the real patterns by obtaining the data generated by the readers without any errors in it.Additionally, at the same time the RFID data is generated with uncertainties. The false negatives are introduced fully or partially into randomly chosen objects data, same way false positives are introduced into each randomly chosen object data. All the datasets are generated with the accuracy level of 70%, i.e., 30% uncertainties.

Table D.4: Parameters

Parameters	**Settings**
Reader range:	1m, **3m**, 5m
Uncertainty Ratio:	**30%**, 20%, 10%
Pattern length:	**5**, 10, 15, 20

We use 10000 paths with the uncertainty ratio of 30%, 20% and 10% (i.e., incorrect characters) are introduced into a string (inaccurate readings are introduced into the raw data). We measure the average error defined earlier

to find the top-1 and top-3 probable paths. In a top-3 matching, we take the average accuracy of three paths. The results are shown in Figure D.8. The average error of best path the accuracy (top-1) is very low, i.e., the accuracy of finding the best path is significantly high. The accuracy remains same even when the uncertainties in the data increases. However, the average error increases with looking at the cumulative error of top-3 paths found. The increased average error is expected since paths two and three of top-3 have higher probabilities of having errors and therefore always add to the average error. The best path always has a higher chance that an observed letter may completely match with the required letter. We also vary the path length,

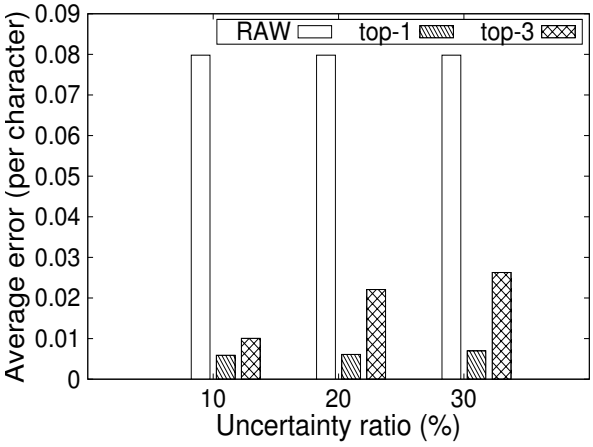


Fig. D.8: Average error Vs Uncertainty ratio

i.e., the number of readers represented by the characters. We use 5, 10, 15 and 20 characters in the paths. The result shown in Figure D.9 indicates that the average error decreases for our matching as a path becomes longer. This indicates that our matching approach is able to make better decisions with more inputs.

In Figure D.10, we present the result about the efficiency of our approach. We consider a fixed number of paths (10000) but vary the length every time. The efficiency moderately decreases (execution time increases) with longer paths, and finding top-3 incurs more time than top-1. When the path length increases, the number of intermediate matching paths also increase. This causes more matching operation to be done by an automaton, which incurs longer total matching time.

We also compare our approach in this paper with the conditioned trajectory graph (CTG) technique proposed in [4]. CTG is close to our approach in that it also works at the semantic level rather than the raw data level and it maps raw RFID readings to indoor locations probabilistically. In CTG, the

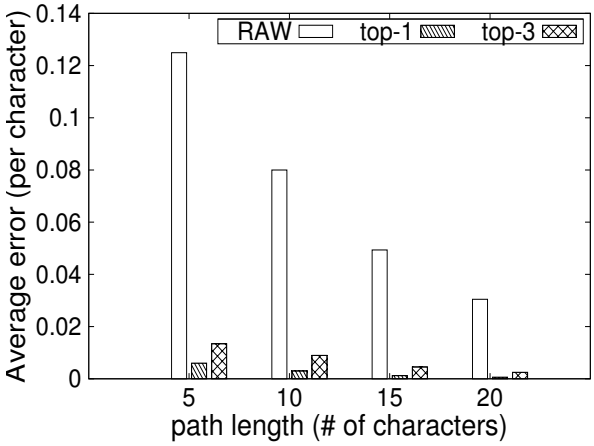


Fig. D.9: Average error Vs path length

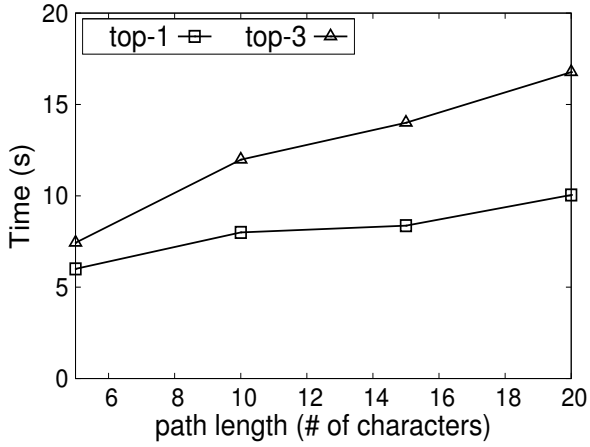
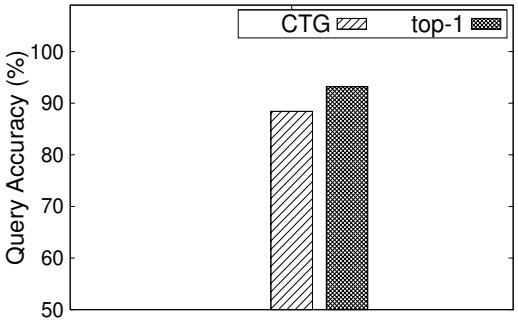


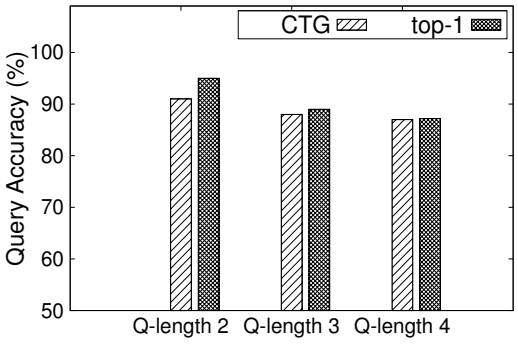
Fig. D.10: Efficiency based on path length

trajectories have to pass several integrity constraints such as reachability, latency, moving time to filter out the invalid trajectories. We use two types of queries *stay* and *trajectory* queries discussed earlier. We compare the result of top-1 with that of CTG, since the latter only returns the best match.

For stay query evaluation, we generated 100 queries for each individual path, each time point is randomly chosen between the first and last reading time of the moving object. The stay query accuracy results are presented in Figure D.11(a), the results indicate our approach outperforms the results of CTG when the top-1 path is returned. Over 95% of query accuracy is achieved on the best paths returned by our approach. The accuracy of many



(a) Stay queries



(b) Trajectory queries

Fig. D.11: Query evaluation results

of the paths are even 100% when the matching characters are the same as the required character.

Next, 50 trajectory queries are generated. The length of each query 2, 3, or 4 is randomly chosen for each selected path. The results are shown in Figure D.11(b), indicate that our approach outperforms the results of CTG. When the trajectory queries of length 4 are used, both approaches achieve an accuracy around 90%. Overall the trajectory query results decreases when queries get longer, since even one character mismatch in a longer query will fail whole query.

6.3 Real Data Results

For experiments on real data, the data about 20,000 passenger bags were collected for a period of two months at Aalborg Airport. The airport operates

with an automatic RFID-based baggage handling system with six RFID readers installed at different geographic locations as shown in Figure D.12. A typically airport handling system features a number of specific locations like, check-in desks, sorter, chutes, etc. The bags checked in by the passengers are tagged during the check-in phase. These bags are then forwarded towards screening area where necessary screening of things are done, before the bags are moved into sortation system which sorts depending upon their destination. From chutes the supporting staff loads the bags into the wagons before leaving to the designated plane through either of the two gates. The bags of arriving passengers are usually carried in mini-wagons from a plane to arrival area, where a support staff put the bags on the designated conveyor. The data about 20,000 bags were collected for a period of two months.

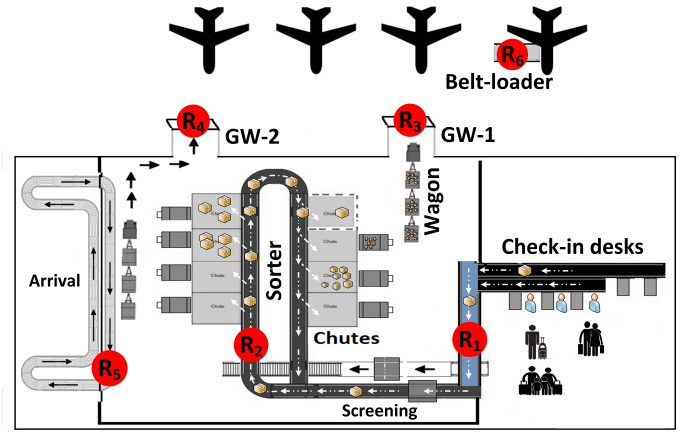


Fig. D.12: Aalborg Airport Baggage Hall

In order to evaluate our approach in the real data, we require ground truth which however is not available. Therefore, we adapt to a work-around solution, which is to generate the semi-real data. We first learn the data dynamics and error ratios from the real data then applied those values to generate a dataset close to the real one. We perform the effectiveness evaluation and the results are shown in Figure D.13. The results indicate that the average error of top-1 and top-3 matching paths are very minimal, meaning the paths returned by our approach are very accurate. Note that the average error is below 1% for both top-1 and top-3 matching results. Since the indoor topology is very simple, the number of possible paths are less (< 3 in many cases). This explains why top-1 and top-3 matching results are close to each other.

We also compare our approach with CTG [4] in terms of query result

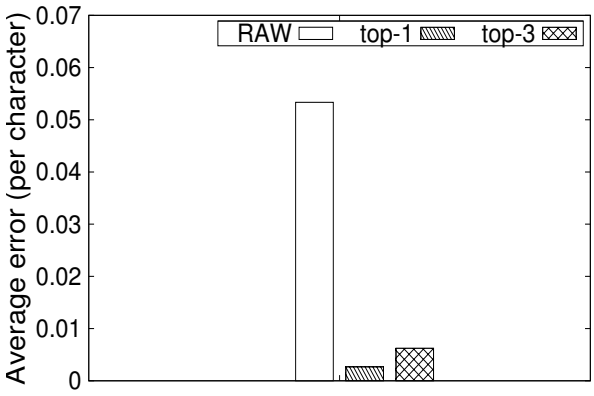


Fig. D.13: Effectiveness over real data

quality. The results are shown in Figures D.14 and D.15 for stay and trajectory queries, respectively. As we can see from Figures D.14, our top-1 matching significantly outperforms CTG in terms of the result quality of stay queries. According to the results shown in Figures D.15, our top-1 matching still clearly outperforms CTG for trajectory queries. Similar to the results over synthetic data, the trajectory query result quality slightly decreases with increasing query length, since more matching operations tend to introduce more matchings inconsistent with the ground truth.

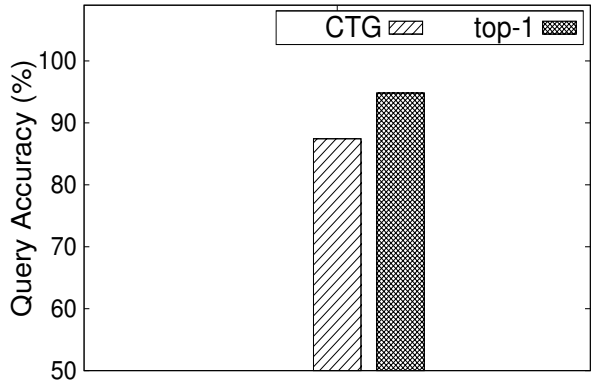


Fig. D.14: Stay query results over real data

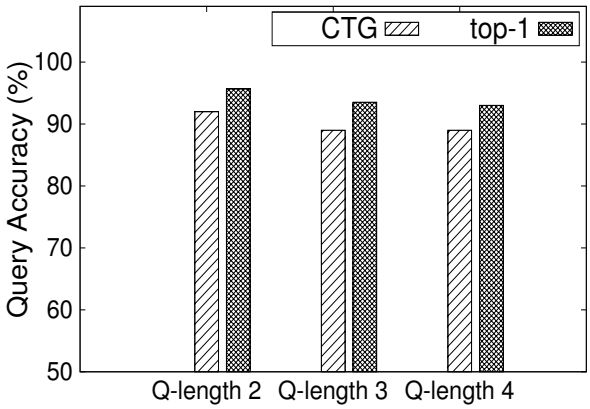


Fig. D.15: Trajectory query results over real data

7 Related Work

Handling indoor raw RFID data is quite challenging [17]. Different RFID data representation techniques and accessing methods have been proposed in [15, 16]. The raw RFID data either need to be cleansed before using it for any processing or techniques are required to extract the useful information from the raw data itself. This paper is about the later, we map error-bound raw RFID data to the most probable semantic indoor paths a moving object has taken. In recent years, lot of attention have been given to techniques which are able to deal with the incomplete and erroneous RFID data. In [17], authors discuss challenges in data management and propose a framework for a distributed RFID system. Gonzalez et al. [16] propose a data warehousing model which is used for RFID data compression based on and path-dependent aggregation. In [31] a RFID data temporal data model for high-level tracking and query monitoring. To cater the dirtiness in the raw RFID data, number of existing works have provided cleansing solutions to improve the reliability of raw RFID data.

In [20] adaptive model called SMURF (*Statistical sMoothing for Unre liable RFid data*) is proposed. In this model a statistical samples of RFID tags are streamed to adjust the filter window size sampling estimators automatically. Chen et al. [24] propose an approach based on Bayesian inference, using data redundancy to its advantage to cleanse raw RFID data. For inserting the false negatives in the data, the correlations between the monitored objects are used to design a data imputation model [29].Tran et al. [5] propose a particle filter based probabilistic inference approach to translate streaming raw RFID

data from mobile readers into location information. Unlike those studies, our work focuses on indoor RFID data of objects moving in indoor environments.

Our previous work [1, 3] uses graph based approaches to cater the problem of cross and missing readings in raw RFID data. In [1], we design a false positive cleansing algorithm for indoor RFID tracking data. It is based on the distance-aware graph model that captures the spatio-temporal information about indoor space and the deployed reader in it. In [3], we augment the graph model with transition probabilities between RFID readers and features of readers, and use the augmented graph to locate and remove false negatives in indoor RFID tracking data. In a recent work [25], we proposed a new cleansing propose based on machine learning approach for indoor RFID data. A new multi-variate Hidden Markov model (IR-MHMM) is proposed handle inaccuracies in raw RFID data. The probabilistic parameters of the proposed model are learned from raw RFID data. The model is used to cleanse RFID data by inferring the most likely observation sequence. Unlike those works on cleansing RFID data, the approach proposed in this paper does not attempt to cleanse the low level raw data; instead, it maps raw data to semantic locations and indoor paths on a higher level.

Bettina et al. [4, 30] design a probabilistic framework to cater the errors present in RFID data. A mechanism based on 2-D grid is used to map raw RFID data to deployed locations. The framework works based on the several constraints like latency, mobility, etc. of the moving objects to rule out impossible indoor paths. The proposed approach in this paper is different in that it employs regular expressions and an automaton based matching algorithm to find the most probable indoor paths for a given set of raw RFID readings. Unlike the techniques in [4, 30], our approach does not require detailed constraints.

8 conclusion and future work

In this paper, we focus on the finding a most probable indoor paths from raw RFID data. Specifically, we use an regular expression based approach to map raw RFID data to a most likely path an indoor moving object with an RFID tag have taken. To perform the regular expression matching, we design an algorithm to formulate and construct an automaton that captures all possible indoor paths in regular expressions. Also, we propose an error model that probabilistically describes all the possible errors between raw RFID data and semantic indoor locations represented in symbols. We evaluated proposed approach with two data sets, a real airport RFID baggage tracking data and a synthetic data set. The results justify that the proposed approach is not only efficient but and effective also. The mapped results are comparable or better than some existing approaches.

Several directions exist for future work. The proposed approach in this paper attempts to map raw RFID data to the semantic level. It is interesting to combine the approach with low level data preprocessing techniques in order to further improve the mapping effectiveness. Also, the error model can be further improved by taking into account the transition probability between RFID readers. Such transition probabilities can be mined from sufficient amounts of historical RFID data. Furthermore, it is also relevant to adapt the proposed approach to handle other types of indoor positioning data, e.g., Wi-Fi positioning data.

References

[1] A. I. Baba, H. Lu, X. Xie, and T. B. Pedersen, "Spatiotemporal data cleansing for indoor RFID tracking data," in *Mobile Data Management*, 2013, pp. 187–196.

[2] G. Navarro, and M. Raffinot, "Flexible Pattern Matching in Strings: Practical On-line Search Algorithms for Texts and Biological Sequences," in *Cambridge University press*, 2002.

[3] A. I. Baba, H. Lu, T. B. Pedersen, and X. Xie, "Handling false negatives in indoor RFID data," in *MDM*, 2014, pp. 117–126.

[4] B. Fazzinga, S. Flesca, F. Furfaro, and F. Parisi, "Cleaning trajectory data of RFID-monitored objects through conditioning under integrity constraints," in *EDBT*, 2014, pp. 379–390.

[5] T. T. L. Tran, C. A. Sutton, R. Cocci, Y. Nie, Y. Diao, and P. J. Shenoy, "Probabilistic inference over RFID streams in mobile environments," in *ICDE*, 2009, pp. 1096–1107.

[6] L. V. Massawe, J. D. M. Kinyua, and H. Vermaak, "Reducing false negative reads in RFID data streams using an adaptive sliding-window approach," 2012.

[7] Y. Nie, Z. Li, S. Peng, and Q. Chen, "Probabilistic modeling of streaming RFID data by using correlated variable-duration hmms," in *SERA*, 2009, pp. 72–77.

[8] L. R. Rabiner and B. H. Juang, "An introduction to hidden markov models," *IEEE ASSp Magazine*, 1986.

[9] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *Annals of Mathematical Statistics*, vol. 37, pp. 1554–1563, 1966.

[10] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 2, pp. 260–269, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1109/TIT.1967.1054010

[11] V. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Soviet Physics Doklady*, vol. 10, p. 707, 1966.

[12] S. Kirshner, "Modeling of multivariate time series using hidden markov models," Ph.D. dissertation, Long Beach, CA, USA, 2005, aAI3164062.

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

[14] J. S. Larson, E. T. Bradlow, and P. S. Fader, "An exploratory look at supermarket shopping paths," *International Journal of Research in Marketing*, vol. 22, no. 4, pp. 395 – 414, 2005.

[15] 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*, 2005, pp. 1140–1151.

[16] H. Gonzalez, J. Han, X. Li, and D. Klabjan, "Warehousing and analyzing massive RFID data sets," in *ICDE*, 2006, p. 83.

[17] S. S. Chawathe, V. Krishnamurthy, S. Ramachandran, and S. Sarma, "Managing RFID data," in *VLDB*, 2004, pp. 1189–1195.

[18] M. Heikki, "Methods and Problems in Data Mining," in *ICDE*, 2005, pp. 41–55.

[19] G. Cabanes, Y. Bennani, and D. Fresneau, "Mining RFID behavior data using unsupervised learning," *IJAL*, vol. 1, no. 1, pp. 28–47, 2010.

[20] S. R. Jeffery, M. N. Garofalakis, and M. J. Franklin, "Adaptive cleaning for RFID data streams," in *VLDB*, 2006, pp. 163–174.

[21] O. Mylyy, "RFID data management, aggregation and filtering," in *R. Meersman and Z. Tari (Eds.): CoopIS/DOA/ODBASE 2005, LNCS 3760*, 2007, pp. 557–575.

[22] IEEE. *Std 1003.1 October 2004 Edition, IEEE Standards Interpretations for IEEE Std 1003.1-2004* . Oct. 2004.

[23] G. Kucherov, and M.Rusinowitch, "Matching a Set of Strings with Variable Length Don't Cares," in *Theoretical Computer Science*, 1997, pp. 129–154.

[24] H. Chen, W.-S. Ku, H. Wang, and M.-T. Sun, "Leveraging spatio-temporal redundancy for RFID data cleansing," in *SIGMOD Conference*, 2010, pp. 51–62.

[25] A. I. Baba, M. Jaeger, H. Lu, T. B. Pedersen, W.-S. Ku, and X. Xie. Learning-based cleansing for indoor RFID data. In *SIGMOD*, pages 925–936, 2016.

[26] Z. Li, T. Ge, and C. X. Chen. Epsilon matching: Event processing over noisy sequences in real time. In *SIGMOD*, pages 601–612, 2013.

[27] K. Thompson. Programming techniques: Regular expression search algorithm. *Commun. ACM*, 11(6):419–422, June 1968.

[28] W. Yan and S. B. Yan., "Cleaning method of RFID data stream based on kalman filter." Journal of Chinese Computer Systems, 2011, pp. 1794–1799.

[29] Y. Gu, G. Yu, Y. Chen, and B. C. Ooi, "Efficient RFID data imputation by analyzing the correlations of monitored objects," in *DASFAA*, 2009, pp. 186–200.

[30] B. Fazzinga, S. Flesca, F. Furfaro, and F. Parisi, "Offline cleaning of RFID trajectory data," in *SSDBM*, 2014, p. 5.

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

Paper E

A Graph Model for False Negative Handling in Indoor RFID Tracking Data

Asif Iqbal Baba, Hua Lu, Torben Bach Pedersen, Xike Xie

The paper has been published in the
Proceedings of 21st ACM SIGSPATIAl GIS, pp. 464–467, 2013.

Abstract

The Radio Frequency Identification (RFID) emerges to be one of the key technologies to modernize object tracking and monitoring systems in indoor environments, e.g., airport baggage tracking. Although RFID has advantages over alternative identification technologies, the raw RFID data produced is inherently uncertain and contains errors. The dirty nature of raw RFID data hinders the progress of applying meaningful high-level applications that range from querying to analyzing. Therefore, cleansing RFID data is a high necessity. In this paper, we focus on handling one of the main aspects of raw RFID data, namely, false negatives*, which occurs when a moving object passes the detection range of an RFID reader but the reader fails to produce any readings. We investigate the topology of indoor spaces as well as the deployment of RFID readers, and propose the transition probabilities that capture how likely objects move from one RFID reader to another. We organize such probabilities, together with the characteristics of indoor topology and RFID readers, into a probabilistic distance-aware graph model. Further, we evaluate the effectiveness and efficiency of devised graph model in recovering the false negatives using real dataset. The experimental results show that the devised graph model is effective and efficient in handling false negatives in indoor RFID tracking data.*

The layout has been revised.

1 Introduction

The Radio Frequency Identification (RFID) emerges to be one of the key technologies to modernize object tracking and monitoring systems in indoor environments, e.g., airport baggage tracking. Most applications based on RFID today use low-cost passive RFID tags. The advantage of passive RFID tags are their size, cost and most importantly they do not require any source of power. RFID based system consists of a tag attached to an object (e.g., a bag in an airport) and a reader, which can detect the tag at a distance through RF communication. As a result, an 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 the deployed readers.

Effective management of indoor RFID tracking data opens new doors for various applications that range from monitoring to analysis of indoor moving objects. However, the unreliable nature of raw data captured by readers is a major factor hindering the development of such applications. Under normal circumstances, it is quite often that the loss and error rate is between 30-40% [1].The read events are frequently missed due to the detection ability of a reader, the quality of an RFID tag, and constraints of the environment [2].

To effectively and efficiently support high-level RFID business logic processing, it is necessary to provide high-quality RFID data. For that case, it is critical to cleanse the RFID raw data , and provide clean data to high level applications to make correct interpretations and analysis of the physical world.

In this paper, we focus on *false negatives* in indoor RFID tracking data. False negatives occur when a reader fails to read out a tag in its detection range. A large number of tags within a reader's detection range are not read consistently due to either their distance from the reader, orientation with respect to reader, presence of metal, dielectric or water close to the tag and other factors. We will exploit spatio-temporal constraints of indoor spaces as well as the properties of deployed RFID readers to devise a graph model to identify and recover such false negatives.

An example in Figure E.1 shows an object O_1 goes through three readers r_1, r_2 and r_3 [**?**]. At first object O_1 entered Hall-1 where reader r_1 detected O_1 from t_1 until t_3. After that, O_1 was detected by reader r_3 placed in Hall-2 from t_7 until t_9. However, it is not possible for O_1 to be detected by r_3 unless it passes through reader r_2 placed in Hall-1, because to enter into the Hall-2 O_1 must go through the detection range of r_1 and r_2 in Hall-1. As a result, false negatives are generated in the data.

Without loss of generality, we assume that all raw readings are ordered by their detection times and are aggregated into tracking records stored in Aggregate Tracking Table (ATT) [4]. Each generated tracking record is in the

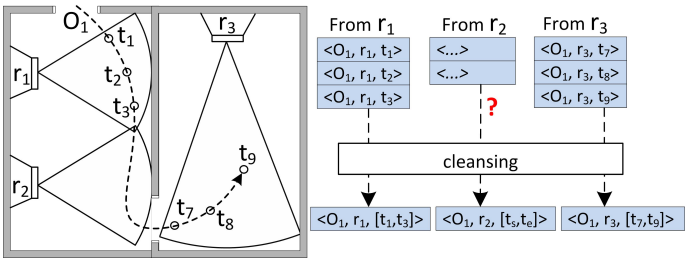


Fig. E.1: An example of RFID data cleansing

format of (*deviceID*, *objectID*, t_s, t_e, *r_Count*), which is tuple with 5 elements, a device identifier, object identifier, starting time (time at which object was first detected), end time (time at which object was last detected) and total number for readings. We formally define other concepts as follows:

Definition 18
(**False Negatives**)
Given a reader r_i and a moving object O, if object O goes through the detection range of reader r_i during time interval $[t, t']$, but r_i does not generate any reading about O's presence during time interval $[t, t']$, a false negative occurs in the data.

Definition 19
(**False Negative Handling**)
Given an Aggregated Tracking Table (ATT), the false negative handling process detects false negatives and inserts recovered tracking record into the ATT.

In this paper our main task has two parts, one is to detect the occurrences of the false negatives in the data, and secondly to recover false negatives (missing information).

The rest of this paper is organized as follows. In section 2 a detailed description of the devised probabilistic distance-aware graph model along with its construction is given. Section 3 details how false negatives handling is performed. Section 4 presents the experiments that evaluate the devised graph effectiveness using real data set. Finally, Section 5 concludes the paper.

2 Probabilistic Distance-Aware Graph Model

We propose a probabilistic distance-aware graph model G_{pdm} in this section. The graph constructed captures the deployed reader information and enable

deriving the minimum travel time and most probable path from one reader to another. The basic idea is to model the readers as graph vertices and assign to each vertex a corresponding reader's properties such as detection range and sampling rate. The minimum travel times required to move from one reader to another is captured in the graph edge between them and the edge also captures the transition probability, which indicates how likely a moving objects move from one reader to another.

2.1 Transition Probabilities

At each step a move to next reader (vertice) only depends on current reader and not the readers that precede it, thus holding a Markov property of "memorylessness". A moving object at reader r_i, moves to a neighboring reader r_j with a probability proportional to the probability weight of the edge (r_i,r_j). The transition probability from r_i to neighboring r_j is defined as:

$$\frac{p(r_i, r_j)}{\sum_{k \in N(r_i)} p(r_i, k)} \tag{E.1}$$

Here, $N(r_i)$ is the set of neighbors of reader r_i. The transitional probability $p(r_i, r_j)$ is the probability that r_j detected object at some time later after t such that same object was detected by r_i at time t.

Definition 20
(**Transition**) A transition takes place when a moving object moves from a reader r_i to another reader r_j without passing through any intermediate readers.

When a finite number of readers, $r_1, r_2, r_3 ... r_n$ are deployed, a probabilistic model representing them can be defined by a transition probability matrix:

$$P = \begin{bmatrix} p(r_1, r_1) & p(r_1, r_2) & \cdots & p(r_1, r_n) \\ p(r_2, r_1) & p(r_2, r_2) & \cdots & p(r_2, r_n) \\ \vdots & \vdots & \vdots & \vdots \\ p(r_n, r_1) & p(r_n, r_2) & \cdots & p(r_n, r_n) \end{bmatrix} \tag{E.2}$$

For each deployed reader r_i, $\sum_{j=1}^{n} p(r_i, r_j) = 1$. We use historical RFID data to learn the transition probabilities. Each transition matrix element P is recorded as:

$$p(r_i, r_j) = \begin{cases} \frac{N_{ij}}{N_i}, & \text{if } (r_i, r_j) \in G_{pdm}.\text{E}; \\ 0, & \text{otherwise.} \end{cases} \tag{E.3}$$

Where, N_{ij} is the number of objects moving from r_i to r_j and N_i is the total number of objects moving from r_i.

2.2 Graph Model

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

1. V is a set of vertices, representing a set of deployed readers.
2. E is the set of edges, where $E = \{(r_i, r_j) \mid r_i, r_j \in V \wedge r_i \neq r_j$. One can move from r_i to r_j without being detected by third reader$\}$.
3. $\mathcal{L}_V : V \rightarrow \mathcal{R} \times \mathcal{R}$ assigns to a vertex v_i a minimum dwell time and sampling frequency of a corresponding reader r_i. Specifically, $\mathcal{L}_V(r_i) = (d_t, S_f)$.
4. $\mathcal{L}_E : E \rightarrow \mathcal{R} \times \mathcal{R}$ assigns to an edge (r_i, r_j) the minimum indoor walking time between two devices r_i and r_j and the probability with which an object can move between them. Specifically, $\mathcal{L}_E(r_i, r_j) = (tt_{i,j}, p_{r_i,r_j})$.

A probabilistic distance-aware graph model G_{pdm} corresponding to floor plan in Figure E.2 is shown in Figure E.3.

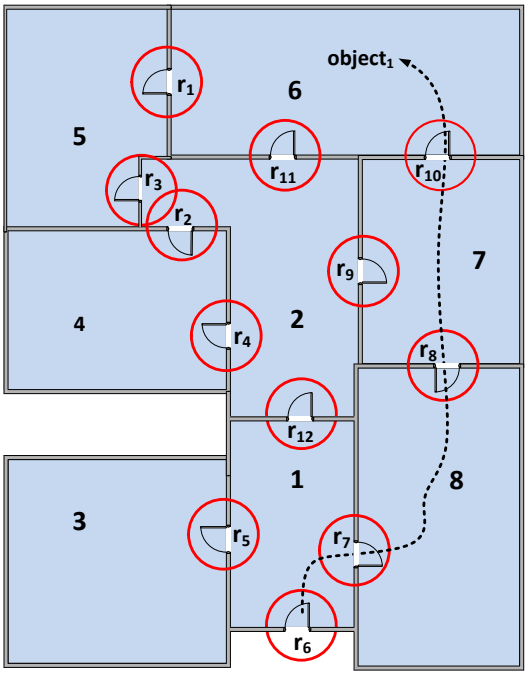


Fig. E.2: Example indoor floorplan

The graph (G_{pdm}) is enhanced from a distance deployment graph proposed in [4]. Our specific design of the weights above is justified by practical needs.

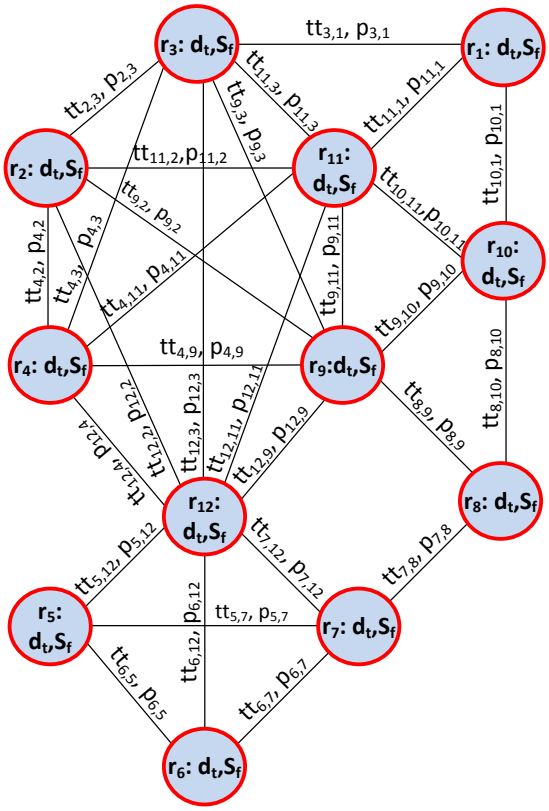


Fig. E.3: Probabilistic distance-aware graph model

Different RFID readers (and other positioning devices) usually imply different minimum dwell times for detection and different sampling rates. Even for one particular reader, its minimum dwell time or sampling frequency may be changed due to the tuning of its physical parameters and surroundings. 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 having minimum travel time weight on each edge is necessary to support such needs in reality. A moving object can move to any of the available connected paths. To determine the traffic load on each of the outgoing paths from a particular location, we capture the probability of each path with which they are used by the moving objects. The probability weight will be used to predict the most likely path an object may have taken.

2.3 Probabilistic Distance-Aware Deployment Graph Construction

The probabilistic distance-aware graph model construction procedure is shown in Algorithm 13. The algorithm takes the set of readers, reader weights, travel

Algorithm 13 Prob-DistanceGraphConstruction(Readers R, Reader weights W, Travel time matrix TT, Transition probability matrix P)

```
1:  G_pdm(V, E, L_V, L_E) ← (R, ∅, W, ∅)
2:  for each reader r_i ∈ R do
3:      for each reader r_j ∈ R and r_j ≠ r_i do
4:          if (r_i, r_j) ∈ G_pdm.E then
5:              continue
6:          find the indoor shortest path sp from r_i to r_j
7:          if there is no other reader on sp then
8:              add edge (r_i, r_j) to G_pdm.E
9:              G_pdm.L_(r_i,r_j) ← (TT[i][j], P[i][j])
10:         else
11:             for each pair of consecutive readers r_k and r_l on sp do
12:                 if (r_k, r_l) ∈ G_pdm.E then
13:                     continue
14:                 else
15:                     add edge (r_k, r_l) to G_pdm.E
16:                     G_pdm.L_(r_k,r_l) ← (TT[k][l], P[k][l])
17: return G_pdm
```

time matrix and the transition probability matrix as input. For each pair of readers r_i and r_j (lines 2–3), the indoor shortest path sp is found if the edge (r_i, r_j) is not in the graph yet (lines 4–6). If sp only contains readers r_i and r_j, a new edge is created with the corresponding weights (lines 7–9). The edge weights contain the minimum travel time (tt_{r_i,r_j}) a moving object takes to move from r_i and r_j and a transition probability (p_{r_i,r_j}) which captures how likely object moves from r_i to r_j. Otherwise, each pair of consecutive readers on sp are processed likewise (lines 11–16).

Inside Algorithm 13, the indoor shortest paths are computed according to the algorithms proposed elsewhere [5]. For the sake of simplicity, the input for those algorithms are implicitly passed to Algorithm 13.

3 Handling False Negatives

The false negative handling process takes aggregated raw data, detects the occurrences of false negatives in it and then subsequently recovers them by filling the missed information one by one. Once the false negative(s) are detected, we look to find the paths between source reader R_s (the last before the false negative) and destination reader R_d (the first after the false negative). We require paths to be simple (loop-free). To further understand the process we need following definitions:

Definition 21
(**Path**) A path is a sequence of readers r_1, r_2, r_3, ..., r_n where there is an edge connecting r_i and r_{i+1} for $i = 1, 2, ..., n$. A path is simple if all r_i are distinct.

Definition 22
(**Candidate Path**) Given a source reader R_s and a destination reader R_d, a path from R_s to R_d, represented as $R_s \overset{\delta}{\rightsquigarrow} R_d$ is a candidate path, if a path δ satisfies the spatio-temporal constraints captured by a graph.

Definition 23
(**Most Likely Path**) Given a set of candidate paths $\{\delta_m\}_{m=1}^{n}$, a most likely path is:

$$\delta^* = \underset{m}{\operatorname{argmax}} \prod_{E_{ij} \in \delta_m} p_{i,j}$$

We design a two-phase solution to handle false negatives in indoor RFID tracking data: *detecting false negatives* and *recovering false negatives*.

Detecting False Negatives: The first phase takes aggregated tracking table ATT as an input, identifies possible false negatives in the data by using the probabilistic distance-aware graph model of all deployed readers in an indoor space. We look at each tracking record tr in ATT and find out the neighbors of $tr.deviceID$, one of which may have detected the moving object next. We take next tracking record tr' and check if $tr'.deviceID$ is in a list of neighbors of $tr.deviceID$. If none of the neighbors matches, we conclude that one or more readers between $tr.deviceID$ and $tr'.deviceID$ have failed to detect a moving object with identity $tr.objectID$. Therefore, false negative(s) are formed in the data. We consider $tr.deviceID$ and $tr'.deviceID$ as source reader R_s and destination reader R_d respectively and find the path between them using a probabilistic distance-aware graph model, described in Section 2.

Recovering False Negatives: The second phase retrieves a path between

source reader R_s (*tr.deviceID*) and destination reader R_d (*tr′.deviceID*) in the graph and fills the missing readings of each reader a path contains between R_s and R_d. The path retrieved is the most likely path, in a set of paths which satisfy the spatio-temporal constraints of subgraph between R_s and R_d. Once the path is retrieved, all the missing readings of each reader in the path between R_s and R_d are filled. To fill the missing readings, the parameters like, minimum traveling time between readers, minimum dwell time and the sampling rate of corresponding reader are used. Such information is captured by the probabilistic distance-aware graph model G_{pdm}. The approximate number of raw readings to be filled for each reader is determined as $\frac{\text{minimum dwell time}}{\text{sampling rate}}$.

4 Experimental Study

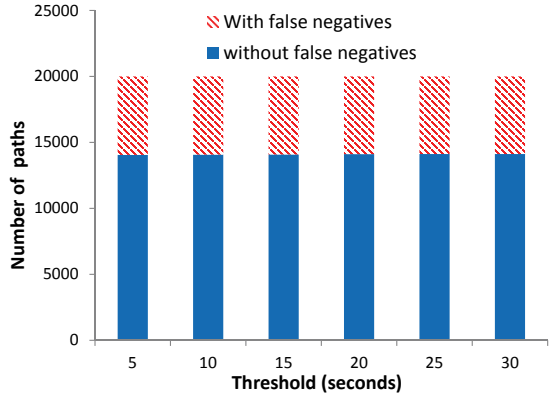
In this section, we report the results from the experiments we conducted to evaluate the performance of our devised graph model for handling false negative(s) in indoor RFID data. A computer with a 64-bit Windows 7, a 2.8GHz core i7 processor, and 8GB main memory was used to run all the experiments, which were implemented in C++.

To evaluate our false negative cleansing effectiveness using a devised graph model, we used the real data set collected from Aalborg airport that operates with an automatic RFID-based baggage handling system. We had continuously recorded the data for two consecutive months. We collected more than half a million raw reading records for about 20000 RFID tagged bags going from Aalborg to Copenhagen airport. The real data used for our experiments were obtained through the baggage handling system installed by Lyngsoe Systems.

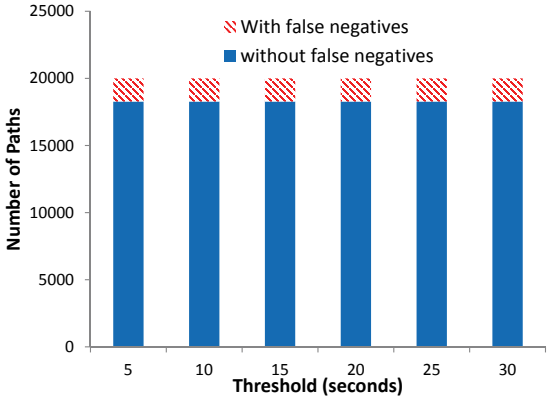
We tested the effectiveness of false negative cleansing on real data by counting the trajectories with and without false negatives. Figure E.4 reports the relevant results. FigureE.4(a) shows that before cleansing more than 30% paths have false negatives. In contrast, Figure E.4(b) shows that after our false negative cleansing only around 7% paths still have have false negatives. This demonstrates the effectiveness of our false negative cleansing technique.

Results on Probabilistic Distance-Aware Deployment Graph model Construction. We also evaluated the efficiency of the probabilistic distance-aware deployment graph construction algorithm shown in Algorithm 13. We used a building of three floors with 100, 200, and 300 doors. Since we deployed RFID readers at doors, we define the *coverage ratio* as $\frac{\text{\# of readers}}{\text{\# of doors}}$. This helped us to tune the number of vertices by varying the coverage ratio of a given building and therefore the size of the deployment graph. The results attained are shown in Figure E.5.

In Figure E.5, each reported value is the average of 50 runs of our graph construction Algorithm 13. The construction time increases super linearly



(a) Before Cleansing



(b) After Cleansing

Fig. E.4: Valid path evaluation on real data

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 conclude that by using our algorithm, the deployment graph can be efficiently constructed. We also studied the graph construction time cost for real data. To construct a probabilistic distance-aware deployment graph model for a real reader deployment, we used the Aalborg airport RFID-base baggage handling system. It takes less than 50 milliseconds, which is very efficient. Due to the space limit, we omit the details.

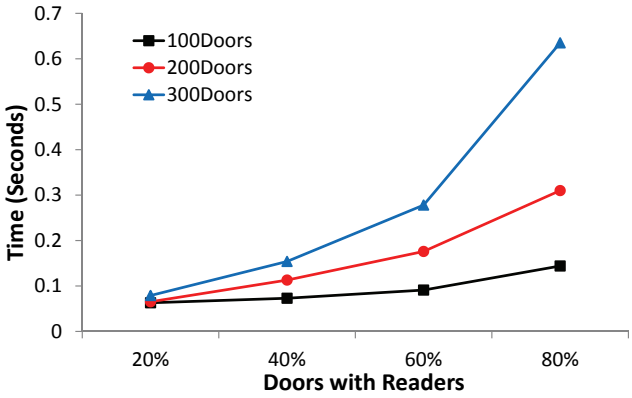


Fig. E.5: Distance-aware graph construction

5 Conclusion

In this paper we study data cleansing for indoor RFID tracking data. we focus on one of the main aspects in raw indoor RFID tracking data, namely, *false negatives*. To handle false negatives in the indoor RFID tracking data, we propose a probabilistic distance-aware graph model to capture probabilities together with the spatio-temporal constraints implied by the deployment of RFID readers as well as the indoor topology. The transition probabilities that capture how likely an object moves from one reader to another are obtained from the existing data. The results demonstrate that the devised probabilistic distance-aware graph model is effective and efficient. The techniques proposed in this paper also apply to indoor tracking data obtained by other symbolic positioning technologies, e.g., Bluetooth.

sectionAcknowledgement

This work is supported in part 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] C. Floerkemeier and M. Lampe, "Issues with RFID usage in ubiquitous computing applications," in *Pervasive*, 2004, pp. 188–193.

[2] D. Roozbeh, O. Maria, and L. Xue, "RFID data management: Challenges and opportunities," in *IEEE International Conference on RFID 2007*, 2007, pp. 175–182.

[3] G. Navarro, and M. Raffinot, "Flexible Pattern Matching in Strings: Practical On-line Search Algorithms for Texts and Biological Sequences," in *Cambridge University press*, 2002.

[4] A. I. Baba, H. Lu, X. Xie, and T. B. Pedersen, "Spatiotemporal data cleansing for indoor RFID tracking data," in *Mobile Data Management*, 2013, pp. 187–196.

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

ISSN (online): 2246-1248
ISBN (online): 978-87-7112-737-9

AALBORG UNIVERSITY PRESS