A Highly Accurate and Scalable Approach for Addressing Location Uncertainty in Asset Tracking Applications

Rengamathi Sankarkumar, Damith C. Ranasinghe
Auto-ID Labs, The University of Adelaide,
Adelaide SA 5005, Australia
{rengamathi.sankarkumar,damith.ranasinghe}@adelaide.edu.au

Thuraiappah Sathyan
The School of Computer Science, The University of Adelaide,
Adelaide SA 5005, Australia
thuraiappah.sathyan@adelaide.edu.au

***Abstract*—Tracking systems that use RFID are increasingly being used for monitoring the movement of goods in supply chains. While these systems are effective, they still have to overcome significant challenges, such as missing reads, to improve their performance further. In this paper, we describe an optimised tracking algorithm to predict the locations of objects in the presence of missed reads using particle filters. To achieve high location accuracy we develop a model that characterises the motion of objects in a supply chain. The model is also adaptable to the changing nature of a business such as flow of goods, path taken by goods through the supply chain, and sales volumes. A scalable tracking algorithm is achieved by an object compression technique, which also leads to a significant improvement in accuracy. The results of a detailed simulation study shows that our object compression technique yields high location accuracy (above 98% at 0.95 read rate) with significant reductions in execution time and memory usage.**

I. Introduction

RFID enabled tracking of objects, such as their condition and location, improves the visibility of objects travelling through a supply chain. Improving the visibility of objects is the key to resolving several problems that arise in supply chains such as counterfeiting, cold chain monitoring, and inventory management [1]–[3]. According to [4] one of the most important benefits of improved information visibility is realized in inventory management and asset utilization. However, RFID technology cannot be directly utilized, because to obtain an accurate location tracking system, first, we must overcome uncertainty in RFID based tracking networks [5].

Uncertainty in RFID data can be caused by false negatives and false positives [6]. Among them RFID based asset tracking is particularly prone to false negatives, which is also known as missed reads. Missed reads refer to objects that exist in a readable area, but may not have their tag detected by readers due to environmental factors such as interference, noise, malfunction of RFID components and distance between a reader and a tag. Missed reads make RFID data incomplete [7] and using such RFID data in object tracking lead to ambiguity in the locations of objects. Hence, managing missed reads in raw RFID data is important for developing effective tracking applications.

In this paper, we present an approach to address location uncertainty caused by missed reads. Here, we model objects travelling through a supply chain using an object flow graph to capture possible movements of objects in a two dimensional state space. Our approach overcomes location uncertainty in a continuous state space, even though raw RFID data are incomplete. Finally, we evaluate the performance of our approach to demonstrate its accuracy and scalability in an RFID enabled returnable asset tracking application. This paper is an extension of the work presented in [8]. The summary of our contributions are:

- We introduce an object flow graph, which models the possible moving paths of objects and supports the creation of a dynamic motion model that is capable of adapting to changes in object flow as a result of evolving Business to Business (B2B) and Business to Client (B2C) relationships.
- We propose an approach that exploits business related contextual information to aggregate objects that are travelling together to develop an optimised Particle Filter (PF) based tracking algorithm. This algorithm is highly scalable and capable of improving the accuracy of estimating the most likely location of assets under uncertainty.
- Finally, we implement the algorithm in a returnable asset management scenario and conduct extensive simulated experiments to evaluate: i) the accuracy at locations where missing reads are high in practice as well as the overall accuracy; ii) the accuracy with increasing number of customers; iii) the effectiveness of the proposed motion model; and iv) scalability (processing time and memory usage) of our algorithm.

The rest of the paper is organised as follows. Section II discusses related work and Section III presents the problem statement. Sections IV, V and VI explain the main contributions of the paper and Section VII concludes the paper.

II. Related Work

Previous research on managing uncertainty in RFID systems [9], [10] proposed an adaptive temporal smoothing filter for cleaning raw RFID data. To address uncertainty, this approach decides whether a tag is still within the read range or moved away from the read range of a given reader. However, these techniques are not applicable to estimate the location of

978-1-4799-3587-1/14/$31.00 ©2014 IEEE

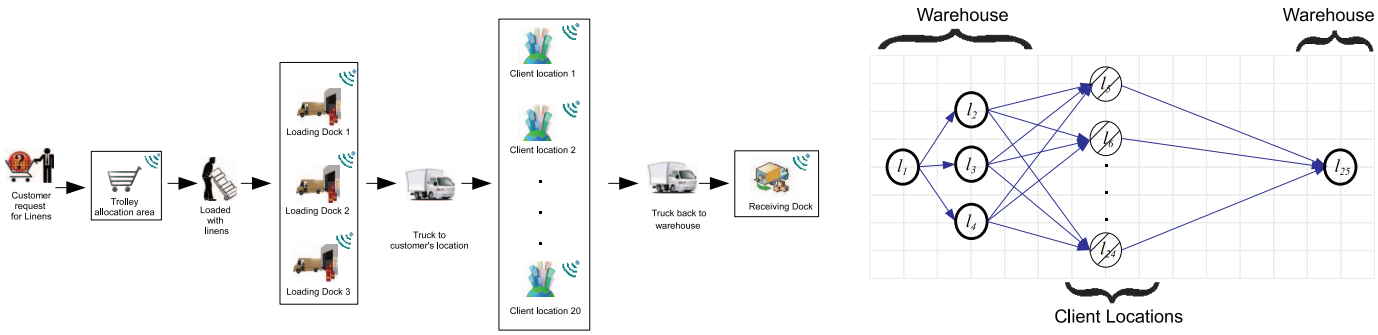

Fig. 1: (a) Business process (b) Object flow graph

objects in a supply chain because they cannot predict a missed object's likely location.

A number of existing publications [6], [8], [11]–[13] have used Bayesian techniques to address the location uncertainty problem. In [6] authors have only addressed uncertainty caused by false positives. In [11] authors' aim is not addressing missed reads, nevertheless they reduce the effect of missed reads by aggregating an object's readings to a single read during a pre-defined time period. However, they need at least one reading during that time period to avoid missing an object. Thus, the accuracy of their technique reduces when an object is completely missing for a period of time.

In [8] and [12], the authors designed a transition model that depicts the probable flow of objects and serves as a base for their prediction of past, current and future locations of RFID tagged objects, even in the case of missed reads. However, these methods need a detailed transaction history of a business to develop the transition model. Also the approach in [12] cannot be used for continuous object tracking. The work in [13] aims to precisely locate an object placed on a shelf using a mobile reader. However, the accuracy of their approach to find the precise location of static objects relies on the accurate measurement of the sensor model obtained from training data. In widely distributed supply chains, obtaining training data for each location is a tedious process. In addition, their algorithm does not predict the motion of the object in the case of missed reads and thus their approach cannot predict the possible location of a moving object.

In [7], the authors demonstrate how containment relationships of objects enable object localization. Their proposed method relies on packaging level information and forms coloured time varying graphs that depict inter-object containment relationships. The data inference technique estimates the most likely location of an object, if there is a missed read. The inference techniques infer edges and nodes (objects) in the graph, building a probabilistic distribution over all possible locations for each node. Iterative inference combines both edge and node inference estimates to find the most probable location of an object. However, the ability to find missed reads is highly dependent on the inter-object containment relationship such as cases on a pallet.

We propose a tracking algorithm that is capable of estimating an object's location in the case of missed reads. In contrast to [12], we continuously track objects in a large scale supply chain and our dynamic motion model is flexible to adapt to changes in object flow and can be used in any widely distributed supply chains with large volumes of complex transactions. Unlike [11], once an object is detected by a reader, our approach is capable of estimating an object's location throughout the supply chain even if the object is missed by all other readers. In contrast to [13], we are not interested in a precise location estimation and so we do not need training data to build an accurate sensor model that predicts the location of missed objects. Although we could have used a measured sensor model at each location, this requires extra effort. Finally, the ability of the proposed algorithm to predict the location of missed objects is independent of inter-object relationships. In addition, our approach can also be applied to contained objects, as in [7].

III. Problem Statement

In returnable asset management, objects such as trolleys and pallets are attached with RFID tags and not the individual entities inside these objects. Increasing the visibility of the returnable assets is of significant interest to effectively manage inventories, prevent shrinkage and to ensure timely delivery. If the asset tags are not read by readers (missed reads) then the location of the asset in the supply network is ambiguous. In this paper, we focus on developing a real-time tracking algorithm, which is capable of estimating the most likely location of assets in RFID enabled returnable asset tracking.

We have derived our returnable asset management scenario requirements from the International Linen Services company (ILS), where trolleys (assets) are reusable containers for linens and they are attached with RFID tags. The company's business scenario is depicted in Fig. 1(a), where RFID readers are installed in the trolley allocation area, the loading docks, the customer locations and the receiving dock. On the receipt of a customer request, trolleys are loaded with the requested items (linens in this case). On the exit of the trolleys from the allocation area, the reader in the allocation area reports all the RFID tags in read successfully. Then, the trolleys are scanned in one of three loading docks through which the linen is loaded into trucks that depart the warehouse and deliver the linen to various customer locations. Trolleys are scanned next, after reaching the customer location, to acknowledge the delivery of the linens. Finally, the trolleys are collected from the customer

locations with or without soiled linens and are delivered and read at the receiving dock. If a trolley is missed (not detected by a reader) at any of these locations, then the location of the trolley is unknown until it is seen again, possibly at the next location.

We now define the notations that are used in this paper: i) L = $\{l_p|p = 1, .., s\}$ denotes a set of locations where the RFID readers are deployed; ii) T = $\{t_i|i = 1, .., m\}$ denotes a set of discrete timestamps; iii) O = $\{o_j|j = 1, .., q\}$ is a set of tagged objects; and iv) *transit* = $\{transit_{l_p}|l_p \in L\}$ denotes the maximum transit time from a given location l_p to the next possible set of locations.

Raw RFID reads from RFID readers can be represented by the schema {time (t_i), objectID (o_j), location (l_p)}. When an object o_j exists at location l_p (say in allocation area), it is next expected to be at a defined set of possible locations (in loading docks 1, 2 or 3) within a transit time period $transit_{l_p}$. If an object o_j is not seen in any of the predefined locations within the specified time period $transit_{l_p}$(due to various reasons such as a missed read, the object is still in transit between two locations or the object is misplaced/miss directed or stolen), then the location of that object is unknown. We construct a PF based tracking algorithm to predict the most likely location of an object when the state of the object is unknown, using the observations obtained so far. Here we assume that the most likely reason for an unknown state is a missed read.

IV. Particle Filtering Based Tracking Algorithm

In this section, first, we present the object flow graph, which serves as a state space for applying PF based tracking algorithms. Then, we briefly review the bootstrap particle filter [14], which is used in the proposed algorithm. Finally, we present the proposed tracking algorithm that utilises the object flow graph.

A. Object Flow Graph

In PF, a workspace under investigation is modelled as a state space. Here, we model the state space with a graphical model called the object flow graph, which also helps define the motion model of the objects to be tracked. The object flow graph *G(L,E)* is derived from considering all the possible movement paths of the objects. The graph *G* has a set of nodes *L* and a set of edges *E*. Each RFID reader deployed in the business process is represented as a node. We use the *(x,y)* co-ordinates to represent a reader location in the graph. Edges are the possible moving path between locations (nodes). Then, we restrict the movement of the particles to be along the edges of the graph to reduce the complexity of the state space and to constrain the tracking region since object movement is possible over a large geographical zone and is not limited to a warehouse.

Fig. 1(b) depicts the object flow graph of the linen services business process. Each reader is denoted by a node in the object flow graph: i) one node in the allocation area; ii) three nodes at the loading docks; iii) 20 nodes for the twenty customer locations (the most important 20 customer locations); and iv) one node at the receiving dock. The possible moving paths are denoted by the edges represented by blue arrows connecting the nodes. The trolley travels in and out of the warehouse as depicted in the object flow graph in Fig. 1(b).

Algorithm 1 *particle_filtering_based_tracking_algorithm*

Require: $raw\_reads^r = (T, O, L)$ where r = 1,2,... // (time, object ID, location) where T =$\{t_i|i = 1, .., m\}$, O =$\{o_j|j = 1, .., q\}$, L = $\{l_p|p = 1, .., s\}$
Require: N = state particles, where N = $\{p_n|n = 1, 2, ..\}$
while $\forall$ r $\in raw\_reads$ **do**
  if new_object_found **then**
    initialize the state particles N
    $object\_list.add(r.t_i, r.o_j, r.l_p, probability \leftarrow 1)$
  else
    object_list.update(r) //update t_i and l_p of $r.o_j$
  end if
  for every particle p_n of o_j **do**
    let p_n move along the edges at a constant speed
    number of particles and direction is determined by motion model
  end for
  predict, update, normalize and resample
  $r.l_p \leftarrow$ estimate the most likely location of o_j
  $probability \leftarrow$estimated probability of o_j being in the location l_p
  $object\_list.update(r, probability)$
end while
managing_missing_objects(object_list)

B. Particle Filters

PF is sequential Monte Carlo method that is an effective technique for state estimation in nonlinear/non-Gaussian state space models [14]. The state space model is defined using a motion model f_t and measurement model h_t and the PF operates on the state space model in a recursive fashion to estimate the unknown state.

Motion model: It describes how the system evolves (system dynamics) from one time step $t-1$ to another *t*. $x_t = f_t(x_{t-1}, v_{t-1})$, where f_t is a possible non-linear function, x_t and x_{t-1} is the state of the object at current and previous time steps and v_{t-1} is independent and identically distributed (i.i.d.) process noise.

Measurement model: The measurement model describes how an observation y_t relates to the true state x_t of the system. $y_t = h_t(x_t, u_t)$, where h_t is a possible non-linear function, y_t is the observation, and u_t is i.i.d. measurement noise. We now describe the steps involved in one iteration of the PF.

Initialise: Particles corresponding to the state of an object are initialised according to $x_0^i \sim p(x_0)$, *i = 1,...,N*, where N is the number of particles used to represent the posterior state distribution of the object.

Predict: Predict the location of an object using the motion model at each time step. At *t* the particles *x* are predicted to be in a location considering the state at $t-1$ and the observations (*raw_reads*) obtained so far. For *i = 1,...,N*, predict particles, $x_t^i \sim q(x_t|x_{t-1}^i, raw\_reads_{1:t})$, where q(.) is an importance function [14].

Update: On receiving an observation the location of predicted particles are updated by weighting the particles using the measurement model to obtain importance weights w_t,

Algorithm 2 *managing_missing_objects*

Require: object_list
Require: transit = $\{transit_{l_p} | l_p \in L\}$ //mean time of an object to move from the given location to the next location (Section III)
1: current_time ← current time from the system
2: **while** not_end_of(*object_list*) **do**
3: *obj* ← $object\_list.get\_next$
4: $max\_time \leftarrow obj.t_i + transit_{(obj.l_p)}$
5: **if** $current\_time > max\_time$ **then**
6: predict
7: *pred_loc* ← estimate the most likely location of o_j
8: $probability$ ←estimated probability of o_j being in the location l_p
9: *pred_time* ← $current\_time$
10: $obj.t_i \leftarrow pred\_time$, $obj.l_p \leftarrow pred\_loc$
11: *object_list.update(obj)*
12: **end if**
13: **end while**

$w_t^i = p(y_t|x_t^i)$, where high weights are given to the particles nearer to the measurement.

Normalize: The weights are normalised according to $w_t^i = w_t^i / \sum_{j=1}^{N} w_t^j$.

Resample: Increasing number of prediction and update iterations leads to a situation where the weights of only a few of the particles are non-negligible. This is referred to as sample degeneracy [14]. Resampling step eliminates the particles that have lower weights and replicates the particles that have higher weight within a probabilistic framework. This results in a new set of i.i.d. particles [11].

C. PF based Tracking Algorithm

The PF algorithm estimates the location of an object with *N* state particles, at every timestep t_i, as a probability distribution over L possible locations at t_i.

Motion Model: We explore two kinds of motion models in this paper: i) a static model; and ii) a dynamic model. Both models assume objects are moving with a nearly constant velocity and we use a transition matrix to represent transition probabilities of objects from one location to the other. Hence, the transition probability matrix defines the possible moving paths of an object [12]. The matrix is built using conditional probabilities, $p(L_{t_i}|L_{t_{i-1}})$ between locations L_{t_i} and $L_{t_{i-1}}$ at current and previous time steps t_i and t_{i-1}, respectively. Assume that set *L* has *p* locations, then *M* can be represented by a $p \times p$ matrix.

$$M = \begin{bmatrix} p(l_1|l_1) & ... & p(l_p|l_1) \\ ... & p(l_f|l_g) & ... \\ p(l_1|l_p) & ... & p(l_p|l_p) \end{bmatrix}$$

However, the static model can only be constructed after studying the transaction history of a business to derive the full set of transition probabilities because obtaining an accurate prediction requires having a motion model that captures the underlying movements of objects accurately. Therefore, this model can only be used in a well-structured business process with access to the complete transaction history.

Algorithm 3 *object_compression_based_tracking_algorithm*

Require: $raw\_reads^r = (T, O, L)$ where r = 1,2,... // (time, object ID, location)
Require: δt
1: $t \leftarrow r.t_1$ //get the initial time from raw_reads
2: **while** ∀ r ∈ raw_reads **do**
3: **if** $r.t_i >= t$ **and** $r.t_i <= t + \delta t$ **then**
4: **if** new_object_found **then**
5: $object\_list.add(r.t_i, r.o_j, r.l_p, probability \leftarrow 1, new\_object \leftarrow true)$
6: **else**
7: $object\_list.update(r, probability, new\_object \leftarrow false)$
8: **end if**
9: **else**
10: **while** $\forall o_j \in object\_list$ **do**
11: $collect\_object$ ← collect objects within the time window [t and $t + \delta t$]
12: $group\_object^n$ ← grouped objects by their delivery location obtained from the contextual information
13: **end while**
14: **for** k = 1 **to** n **do**
15: $hashtable.put(ID^k, group\_object^k)$
16: $group$ ←*(t, ID^k, location of k^{th} group_object)*
17: **if** $group\_object^k$ contains $new\_object(true)$ **then**
18: initialize group
19: **end if**
20: predict, update, normalize and resample group
21: $object^d \leftarrow hashtable.get(ID^k)$ //get back the compressed objects
22: $object\_list.update(\forall d \in object, probability)$
23: **end for**
24: $t \leftarrow r.t_i$ //next t from r
25: **end if**
26: **end while**
27: *managing_missing_objects(object_list)*

In contrast, the dynamic model is based only on the object flow graph and initially we assume every transition is equally likely. After each step of a transaction, the model gains some knowledge of the object flow and updates the model. The dynamic model is more suitable for large scale businesses where it is hard to analyse large volumes of complex transaction records, often distributed across third parties.

Measurement Model: The measurement model is defined such that the particles that are in the vicinity (i.e., within the active range) of a reader antenna are assigned a higher weight compared to particles that are farther from the reader antenna. The default sensor model used in this paper to assign weights is a Gaussian distribution, as it models the decreasing probability of the tag being read as it moves farther from the reader. Here, we consider the reader antenna to be at the centre of a node and the active range to be the radius of a node.

We designed a PF based tracking algorithm with prior knowledge of the object flow graph. In Algorithm 1, lines 1 to 7 examine *raw_reads* to identify if objects are seen for the first time to initialize the state particles for new objects. The new

objects are added to the *object_list*, while the existing objects are updated. In lines 8 to 15, the algorithm predicts and updates the locations of objects based on their most recent readings.

A missed reading event is identified if there is no read for an object in the *object_list* within the expected transit time. In Algorithm 2 we explicitly manage possible missed readings of objects. First line of Algorithm 2, retrieves the *current_time* from the system and evaluates if the transit time of all the objects are within their maximum thresholds. If the current time is greater than the maximum transit time threshold for a given object, then, only the prediction step of the PF is carried out to estimate the approximate location of the object. Next, we estimate the probability of each missed object being in the active range of every location. The probability of an object o_j being at each possible location l_p is $p(o_j) = n/N$ where *n* is the number of particles that is present within the active range of that reader and *N* is the total number of particles. Thus, the location l_p having the maximum probability $p(o_j)$ is assumed as the location of the missed object.

In spite of PFs effectiveness in improving location tracking accuracy, high computational cost is an important drawback when implementing PFs in large scale tracking applications. For example, as the object flow graph becomes more complex, more particles are required to achieve greater accuracy. Further, tracking a large number of objects also increases the memory requirement. In the following section we show how the scalability of the PF based tracking algorithm can be improved.

V. Optimization of the Tracking Algorithm

We propose an object compression technique for things travelling together to the same location to reduce the computational complexity of the proposed algorithm. Our approach exploits contextual information (customer order information) to aggregate objects that are travelling together from one location to another into a single aggregated object. This compression can be achieved in three steps:

Collect objects: The first step is to create a time window of small duration δt and collect all the objects that are observed within this time duration. A time window is used to segment the incoming data from RFID readers and limit delays because all the objects related to a single order may not be loaded onto a truck immediately after orders are processed. *Group objects*: Collected objects are grouped, according to the delivery location obtained from customer orders and objects allocated during order processing (contextual information). *Compressing objects*: Finally, compress the grouped objects travelling to the same location to a single aggregated object. A hash table maintains the mapping between a grouped object and the objects contained within a group.

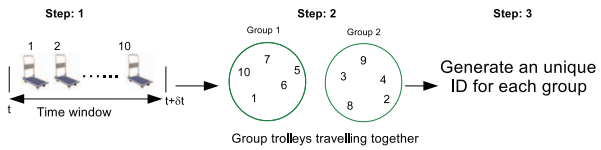

Fig. 2: Dynamic grouping

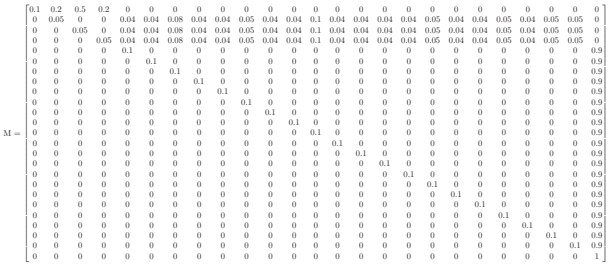

Fig. 3: Static model

The PF algorithm developed in the previous section is then applied to the aggregated object. The aim of aggregating the objects is to eliminate the need for using the PF algorithm for every single object. Algorithm 3, explains the object compression based tracking algorithm.

VI. Simulations and Results

A. Simulation Scenario

We implemented a prototype of the linen company's business process (see Fig. 1) in Matlab and conducted an extensive simulation study to evaluate the performance of the proposed algorithm. The simulated trolleys' operational period (time spent in the supply chain) was 2 - 6 hours. The number of trolleys per transaction (order) was randomly selected to be either 5 or 10. Approximate time between customer requests (transactions) was randomly selected to be within 2 - 15 minutes. The transit times between the allocation area to loading dock, loading dock to customer locations and customer locations to the receiving dock were also randomly selected to be between 20 - 40, 60 - 120 and 180 - 240 minutes, respectively. Number of orders sent to each loading dock and the number of orders generated for each customer, was based on the probabilities generated from historical transactions captured from the company and presented in the static model given in Fig. 3.

We have defined read rate as the probability that a tag is read successfully by the reader and error rate as the percentage of incorrect location predictions. For simulating the trolley movements, we set the speed of the particles to be a Gaussian distribution with μ = 0.42 m/s and σ = 0.01. The readable range is set as 10 m in the object flow graph.

B. Illustration of the Tracking Algorithm

Fig. 4 illustrates the implementation of the PF based tracking algorithm using the business process of the linen company. In Fig. 4, we tracked a trolley with ID 356 from the allocation area to its return at the receiving dock. Trolley 356, had missed a read at the loading dock at 10:42. The PF based tracking algorithm initialised the object particles at the first time step, i.e., at 10:32, in the allocation area. Then, the particles started moving according to our static motion model until the transit time expired. After the transit time expired our tracking algorithm identified a missed reading in the loading dock. Then, the object location is predicted using the particle filter based tracking algorithm to manage the missed read. Our algorithm managed to correctly predict the location of the

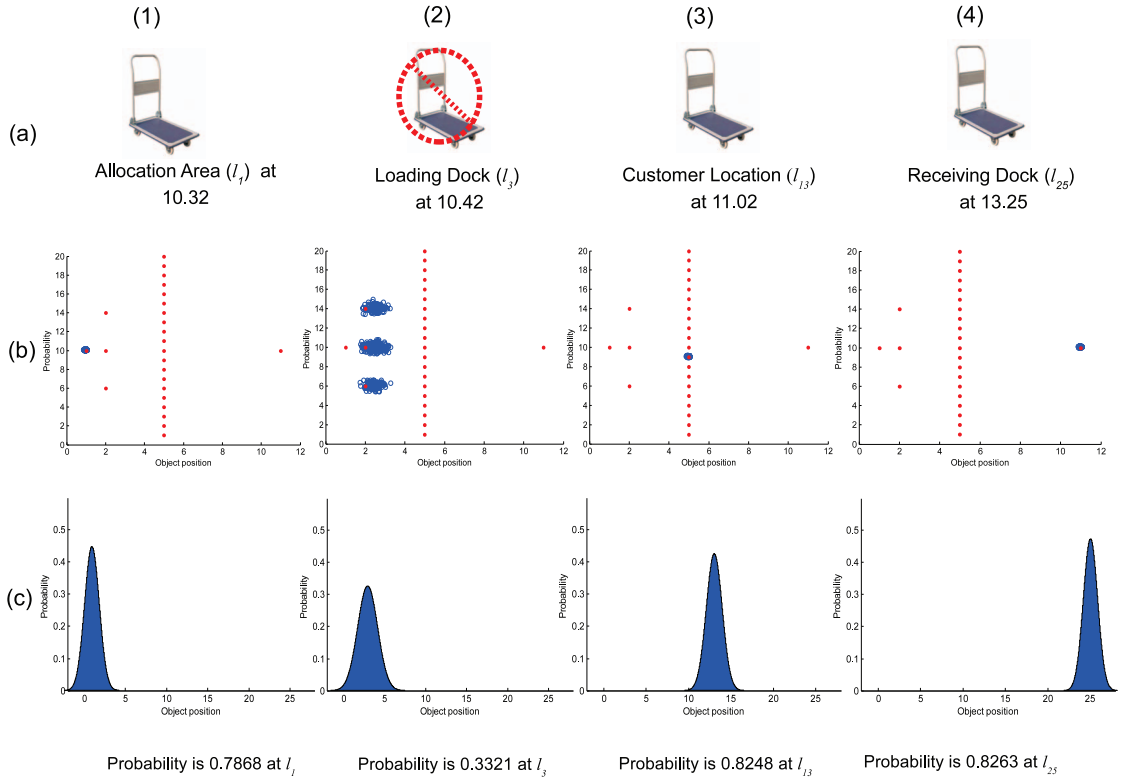

Fig. 4: (a) Location of trolleys at different time steps (b) PF distribution in the object flow graph (c) Most likely location based on the highest probability estimated from the PDF

trolley to be in l_3 as shown in the Fig. 4(c)(2). Later, at 11:02, the trolley was located at the customer location and at 13:25, at the receiving dock. Fig. 4(c) shows the probability distribution over the state space and the most likely location identified by our tracking algorithm. In Fig. 4(c)(2), the probability of the trolley being at l_3 is lower because of the uncertainty caused by the missed read.

C. Simulation Study

In this section, we discuss the experiments conducted to evaluate the accuracy and scalability of the proposed tracking algorithm and the utility of the static and dynamic motion models. First we investigate the effect of the number of particles on the performance of the tracking algorithm. Then, in order to determine the effectiveness of our algorithms in the ILS scenario, compared to not employing them, we conducted three simulations to evaluate the accuracy of location estimates at: a) the loading dock; b) receiving dock; and c) over all locations. This is because in practice, handheld RFID readers are used to scan the trolley at the allocation area (where trolleys are checked manually) and at customer locations. Therefore, we assume that the probability of missing reads is higher at the loading and receiving docks, and negligible at the other locations. The results reported for accuracy were generated by averaging 10 repeated simulation runs using randomly generated trolley movement data. Next, to study the effects of an increase in the number of locations and the corresponding location accuracy, we expand the business process by increasing the number of customer locations as shown in Fig 7. The new business process has i customer locations, where i = 40, 60, 80 and 100, instead of 20 (considered before). We also evaluate the processing time and memory requirement of our tracking algorithm with up to 10,000 trolleys to investigate its scalability. Finally, we investigate the dynamic motion model's ability to adapt to changes in the object flow. As already discussed, object compression with the dynamic model is an adaptive tracking algorithm. Therefore we considered 5 different business transaction scenarios where 100, 200, 300, 400 and 500 trolleys per day, respectively, were processed and investigate the agility of the proposed tracking algorithm.

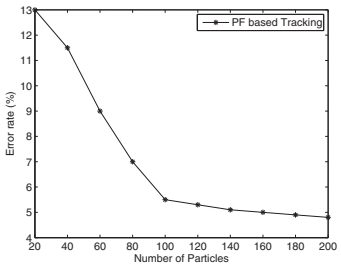

Fig. 5: Effects of number of particles used in the PF

D. Results

In this section, we discuss the results of our approach in terms of accuracy and scalability. We first discuss the effects

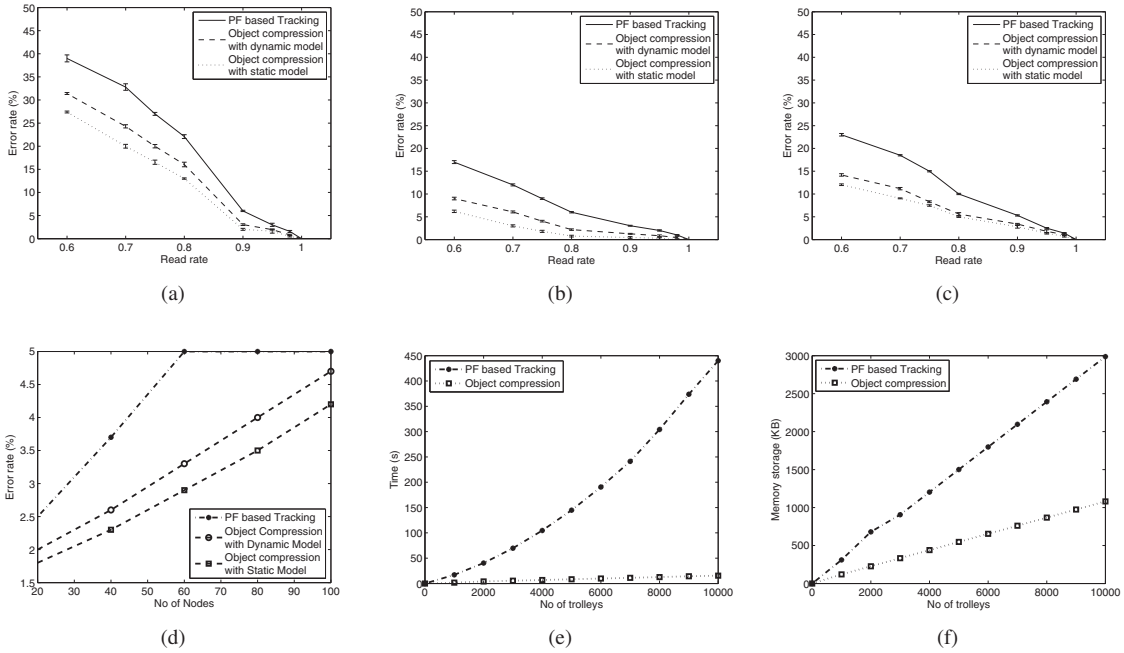

Fig. 6: (a) Accuracy at the loading docks (b) Accuracy at the receiving dock (c) Overall accuracy (d) Overall accuracy with increasing nodes (customer locations) (e) Execution time (f) Memory usage

of the number of particles used for tracking each object in the PF based tracking algorithm. In order to avoid the degeneracy problem, we need a sufficient number of particles. On the other hand, keeping a large number of particles will significantly increase computational costs. We perform simulations to explore the effects of the number of particles on location estimation accuracy to determine the appropriate size of particles. As shown in the Fig. 5, the accuracy of the algorithm increases with higher numbers of particles. Another observation is that the accuracy changes slowly when the number of particles is above 100. Therefore, we conclude that 100 particles are adequate for our algorithm to guarantee good enough accuracy without a high computational cost.

Accuracy is measured by error rate with respect to the read rate of the simulated readers. In Fig. 6(a), object compression with the static model outperformed the other two techniques, because, the PF based tracking algorithm tracks the individual object and does not have any knowledge of other trolleys travelling with the trolley whose tag reading is missed. The dynamic model requires time to learn the flow of trolleys from executed transactions. Any missed reads occurring during the learning period have a higher chance of an incorrect location estimation. Therefore tracking algorithm with dynamic model did not outperform the tracking algorithm with the static motion model.

Fig. 6(b) shows the accuracy in case of missed reads at the receiving dock. The error rates in all three techniques are significantly less than at the loading dock (Fig. 6(a)) because even with missed reads the possible location estimations are limited to a single active range as opposed to three at the loading dock. Fig. 6(c) shows the overall accuracy at all locations obtained after having varied read rates at all 25 locations. At 90% read rate, the accuracy of all three algorithms was above 95%. Therefore using a particle filter has halved the error rate from 10% to 5%. Again, object compression with dynamic model has slightly lower accuracy than the static model.

Fig. 6(d) shows the overall accuracy (where the read rate is set to 95%) with increasing number of customer locations. It can be seen that increasing the number of locations has an adverse effect on the accuracy of the system. Object compression with the static model achieved the highest accuracy when compared to others. When the number of nodes reaches 100, even the accuracy of the static model falls to 95.8%. Consequently, the PF based tracking algorithm cannot provide a performance improvement in relation to read rate (set at 95%) beyond 60 customer locations. This is because the probable set of locations for the missed object increases as the number of customer locations increase.

We have only considered the algorithms with the static model to investigate execution time and memory consumption because the motion model does not have an impact on the execution time and memory used. Fig. 6(e) shows the total execution times taken by the returnable asset management

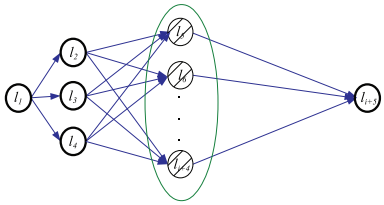

Fig. 7: Object flow graph with increased nodes

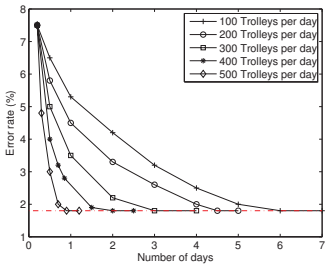

Fig. 8: Investigate the accuracy of the dynamic motion model

scenario simulated. In Fig. 6(e) it is clear that the object compression technique with the static model outperformed the PF based tracking technique because the PF based tracking had to track individual trolleys as opposed to groups.

Memory usage is dominated by the number of trolleys and particles used in the algorithm. Particle count is constant, i.e., 100 for all techniques, therefore the number of trolleys tracked drives the amount of memory consumed. Object compression technique compressed a group of trolleys to a single object and thus, around 50% of the memory storage is saved compared to PF based tracking. The object compression technique considerably reduces memory usage as shown in Fig. 6(f).

Over a period of time we can expect the dynamic model based tracking algorithm to achieve the same accuracy as that of the static model based algorithm. So, we set our target to the highest accuracy (98.2%) reached by the static model at 95% read rate (typically good read rate obtained from practical deployments) in the ILS scenario. This target is represented in Fig. 8 with a red horizontal line. On day one, the simulation ran with the designated number of trolleys and as a result the dynamic model is updated based on the transaction data and the error rate gradually decreased at different rates for different trolley volumes. Fig. 8 shows that as the number of trolleys increases, the target is reached more rapidly. This is because the model adapts and updates quickly when the number of transactions are high. Therefore, if 500 or more transactions are carried out in a day, the dynamic model attains the same accuracy as that of the static model within a day. Hence, as long as business conditions do not change dramatically from day to day the dynamic model will be able to deliver high accuracy while still being able to adapt to respond to change relatively quickly.

VII. Conclusion and Discussion

In this paper, we proposed and evaluated an optimised object tracking algorithm that is capable of addressing missed reads in a returnable asset management scenario. We proposed and evaluated an optimized PF based tracking algorithm that is both scalable and capable of accurately estimating the most likely location of objects. The algorithm with the static motion model was able to minimize the error rate to 1.8% at 95% read rate with considerable improvement in the scalability.

Compared to our previous work in [8] based on using a static motion model for tracking objects, the algorithm that we proposed utilises the dynamic motion model and business context information. By using this dynamic model and contextual information our method can even work in widely distributed supply chains where it is hard to analyse huge volumes of complex transaction data. Furthermore, in contrast to [8] the dynamic motion model based approach is adaptive to changing business conditions requiring only around 500 transactions a day to completely learn the motion model.Next we compared our algorithm with [7]. Although we have not implemented their algorithm in our business process, we compared their accuracy results with ours, because both papers discussed tracking objects in a supply chain scenario. At 80% read rate, with 0% (as in our scenario) and 100% containment relationship their accuracy was approximately 87% and 94%, respectively. In contrast, with a larger supply chain model than in [7] (5 in [7] vs. 25 locations in ours) our algorithm achieved 95% accuracy with the same read rate.

VIII. Acknowledgement

This work was supported by an Australian Research Council Linkage grant (LP100200114) and was partially funded by the International Linen Services (ILS), South Australia.

References

[1] Y. Wu, Q. Z. Sheng, D. Ranasinghe, and L. Yao, "Peertrack: a platform for tracking and tracing objects in large-scale traceability networks," in *15th Int. Conf. on EDBT*, 2012, pp. 586–589.

[2] Y. Wu, D. C. Ranasinghe, Q. Z. Sheng, S. Zeadally, and J. Yu, "RFID enabled traceability networks: a survey," *Distributed and Parallel Databases*, vol. 29, no. 5-6, pp. 397–443, 2011.

[3] H. Gonzalez, J. Han, X. Li, and D. Klabjan, "Warehousing and analyzing massive RFID data sets," in *22nd ICDE*, 2006, pp. 83–83.

[4] Y. V. Joshi, "Information visibility and its effect on supply chain dynamics," Ph.D. dissertation, MIT, 2000.

[5] Y. Diao, B. Li, A. Liu, L. Peng, C. Sutton, T. Tran, and M. Zink, "Capturing data uncertainty in high-volume stream processing," *The Biennial Conf. on Innovative Data Systems Research*, 2009.

[6] M. Goller and M. Brandner, "Probabilistic modeling of rfid business processes," in *IEEE Int. Conf. on RFID-Tech. and Applications (RFID-TA)*, 2011, pp. 432–436.

[7] Y. Nie, R. Cocci, Z. Cao, Y. Diao, and P. Shenoy, "Spire: Efficient data inference and compression over RFID streams," *IEEE TKDE*, vol. 24, no. 1, pp. 141–155, 2012.

[8] R. Sankarkumar, D. C. Ranasinghe, and T. Sathyan, "A highly accurate method for managing missing reads in RFID enabled asset tracking," *10th Int. Conf. on Mobile and Ubiquitous System*, 2013.

[9] S. R. Jeffery, M. Garofalakis, and M. J. Franklin, "Adaptive cleaning for RFID data streams," in *32nd Int. Conf. on VLDB*, 2006, pp. 163–174.

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

[11] J. Yu, W.-S. Ku, M.-T. Sun, and H. Lu, "An RFID and particle filter-based indoor spatial query evaluation system," in *16th Int. Conf. on EDBT*, 2013, pp. 263–274.

[12] T. Kelepouris, M. Harrison, and D. McFarlane, "Bayesian supply chain tracking using serial-level information," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,*, vol. 41, no. 5, pp. 733–742, 2011.

[13] T. Tran, C. Sutton, R. Cocci, Y. Nie, Y. Diao, and P. Shenoy, "Probabilistic inference over RFID streams in mobile environments," in *25th ICDE*, 2009, pp. 1096–1107.

[14] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.