An Indoor Spatial Query Evaluation System based on RFID and Particle Filters

by

Jiao Yu

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
May 4, 2013

Keywords: Indoor spatial query, RFID, Particle filter

Approved by

Wei-Shinn Ku, Associate Professor of Computer Science and Software Engineering
David Umphress, Associate Professor of Computer Science and Software Engineering
Xiao Qin, Associate Professor of Computer Science and Software Engineering

Abstract

People spend a significant amount of time in indoor spaces (e.g., office buildings, subway systems, etc.) in their daily lives. Therefore, it is important to develop efficient indoor spatial query algorithms for supporting various location-based applications. However, indoor spaces differ from outdoor spaces because users have to follow the indoor floor plan for their movements. In addition, positioning in indoor environments is mainly based on sensing devices (e.g., RFID readers) rather than GPS devices. Consequently, we cannot apply existing spatial query evaluation techniques devised for outdoor environments for this new challenge. Because particle filters can be employed to estimate the state of a system that changes over time using a sequence of noisy measurements made on the system, in this research, we propose the particle filter-based location inference method as the basis for evaluating indoor spatial queries with noisy RFID raw data. Furthermore, two novel models, indoor walking graph model and anchor point indexing model, are created for tracking object locations in indoor environments. Based on the inference method and tracking models, we develop innovative indoor range and k nearest neighbor (kNN) query algorithms. We validate our solution through extensive simulations with real-world parameters. Our experimental results show that the proposed algorithms can evaluate indoor spatial queries effectively and efficiently.

Acknowledgments

This research has been funded in part by the National Science Foundation (NSF) Grants CNS-0831502 (CT) and CNS-0855251 (CRI).

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

Today most people spend a significant portion of their time daily in indoor spaces such as subway systems, office buildings, shopping malls, convention centers, and many other structures. In addition, indoor spaces are becoming increasingly large and complex. For instance, the New York City Subway has 468 stations and contains 209 miles (337 km) of routes [28]. In 2011, the subway system delivered over 1.64 billion rides, averaging approximately 5.3 million rides on weekdays [15]. Therefore, users will have more and more demand for launching spatial queries for finding friends or Points Of Interest (POI) in indoor places. However, existing spatial query evaluation techniques for outdoor environments (either based on Euclidean distance or network distance) [18, 6, 16, 19, 12] cannot be applied in indoor spaces because these techniques assume that user locations can be acquired from GPS signals or cellular positioning, but the assumption does not hold in covered indoor spaces. Furthermore, indoor spaces are usually modelled differently from outdoor spaces. In indoor environments, user movements are enabled or constrained by entities and topologies such as doors, walls, and hallways.

Radio Frequency Identification (RFID) technologies have become increasingly popular over the last decade with applications in areas such as supply chain management [20], health care, and transportation. In indoor environments, RFID is mainly employed to support track and trace applications. Generally, RFID readers are deployed in critical locations while objects carry RFID tags. When a tag passes the detection range of a reader, the reader recognizes the presence of the tag and generates a record in the back end database. However, the raw data collected by RFID readers is inherently unreliable [21, 8], with false negatives as a result of RF interference, limited detection range, tag orientation, and other environmental phenomena [26]. In addition, readers cannot cover all areas of interest because of their high cost or privacy concerns [24]. Therefore, we cannot directly utilize RFID raw data to evaluate commonly used spatial query types (e.g., range and kNN) for achieving high accuracy results in indoor environments.

In this research, we consider the setting of an indoor environment where a number of RFID readers are deployed in hallways. Each user is attached with an RFID tag, which can be identified by a reader when the user is within the detection range of the reader. Given the history of RFID raw readings from all the readers, we are in the position to design a system that can efficiently answer indoor spatial queries.

Particle filters are sequential Monte Carlo methods based on point mass representations of probability densities, which can be applied to any state-space model [1]. Particle filters can be employed to estimate the state of a system that changes over time using a sequence of noisy measurements made on the system. In this paper we propose the particle filter-based location inference method, the indoor walking graph model, and the anchor point indexing model for inferring object locations from noisy RFID raw data. On top of the location inference, indoor range and kNN queries can be evaluated efficiently by our algorithms with high accuracy. The contributions of this study are as follows:

- We design the particle filter-based location inference method as the basis for evaluating indoor spatial queries.

- We propose two novel models, the indoor walking graph model and the anchor point indexing model, and an RFID-based system for tracking object locations in indoor environments.

- Indoor spatial query evaluation algorithms for range and kNN queries are developed based on the proposed system.

- We demonstrate the efficiency and effectiveness of our approach by comparing the performance of our system with the symbolic model-based solution [30] through extensive simulations using real-world parameters.

The rest of this paper is organized as follows. In chapter 2, we survey previous works for indoor object monitoring and spatial queries. Background knowledge of particle filters and the symbolic model-based location inference is provided in chapter 3. In chapter 4 we introduce our particle filter-based indoor spatial query evaluation system. The experimental validation of our design is presented in chapter 5. Chapter 6 proposes tentative solutions for more complicated continuous spatial query types. And Chapter 7 concludes this paper with a discussion of future work.

Chapter 2
Related Work

In this section, we review previous work related to indoor spatial queries and RFID data cleansing.

2.1 Indoor Spatial Queries

Outdoor spatial queries, e.g., range and kNN queries, have been extensively studied both for Euclidean space [18, 6] and road networks [16, 19, 12]. However, due to the inherent differences in spatial characteristics, indoor spatial queries need different models and cannot directly apply mature techniques from their outdoor counterparts. Therefore, indoor spatial queries are drawing more and more research attentions from industry and academia. For answering continuous range queries in indoor environments, Jensen *et al.* [9] proposed using the *positioning device deployment graph* to represent the connectivity of rooms and hallways from the perspective of positioning devices. Basically, entities that can be accessed without having to be detected by any positioning device are represented by one cell in the graph, and edges connecting two cells in the graph represent the positioning device(s) which separate them. Based on the graph, initial query results can be easily processed with the help of an indexing scheme also proposed by the authors [29]. Query results are returned in two forms: certain results and uncertain results. To reduce the workload of maintaining and updating the query results, Yang *et al.* further proposed the concept of *critical devices*. Only from the ENTER and LEAVE observations of its critical devices can a query's results be affected. However, the probability model utilized in Yang's work is very simple: a moving object is uniformly distributed over all the reachable locations constrained by its maximum speed in a given indoor space. This simple probability model is incapable of taking advantage of the moving object's previous moving patterns, such as direction and speed, which would make the location prediction more reasonable and precise. In addition, Yang *et al.* [30] also addressed the problem of kNN queries over moving objects in indoor spaces. Unlike another previous work [14] which defines nearest neighbors by the minimal number of doors to go through, they proposed a novel distance metric, minimum indoor walking distance, as the underlying metric for indoor kNN queries. Moreover, Yang *et al.* provided the formal definition for Indoor Probabilistic Threshold kNN Query (PTkNN) as finding a result set with k objects which have a higher probability than the threshold probability T. Indoor distance-based pruning and probability threshold-based pruning are proposed in Yang's work to speed up PTkNN query processing. Similarly, the paper employs the same simple probabilistic model as in [29], and therefore has the same deficiencies in probability evaluation.

2.2 RFID-Based Track and Trace

RFID is a very popular electronic tagging technology that allows objects to be automatically identified at a distance using an electromagnetic challenge-and-response exchange of data [23]. An RFID-based system consists of a large number of low-cost tags that are attached to objects, and readers which can identify tags without a direct line-of-sight through RF communications. RFID technologies enable exceptional visibility to support numerous track and trace applications in different fields [31]. However, the raw data collected by RFID readers is inherently noisy and inconsistent [21, 8]. Therefore, middleware systems are required to correct readings and provide cleansed data [7]. In addition to the unreliable nature of RFID data streams, another limitation is that due to the high cost of RFID readers, RFID readers are mostly deployed such that they have disjoint activation ranges in the settings of indoor tracking. Furthermore, privacy (i.e., readers are deployed in hallways rather than rooms in office buildings) is also an important concern [26].

To overcome the above limitations, RFID data cleansing is a necessary step to produce consistent data to be utilized by high-level applications. Tran *et al.* [22] used a sampling-based method called particle filtering to infer clean and precise event streams from noisy raw data produced by mobile RFID readers. Three enhancements are proposed in their work to make traditional particle filter techniques scalable. However, their work is mainly designed for warehouse settings where objects remain static on shelves, which is quite different from our setting where objects move around in a building. Therefore, Tran's approach of adapting and applying particle filters cannot be directly applied to our settings. Another limitation of [22] is that they did not explore further utilization of the output event streams for high-level applications. Chen *et al.* [3, 10] employed a different sampling method called Markov Chain Monte Carlo (MCMC) to infer objects' locations on shelves in warehouses. Their method takes advantage of the spatial and temporal redundancy of raw RFID readings, and also considers environmental constraints such as the capacity of shelves, to make the sampling process more precise. Their work also focuses on warehouse settings; thus is not suitable for our problem of general indoor settings. The works in [17, 25, 13] target settings such as office buildings, which are similar to our problem. They use particle filters in their preprocessing module to generate probabilistic streams, on which complex event queries such as "Is Joe meeting with Mary in Room 203?" can be processed. However, their goal is to answer event queries instead of spatial queries, which is different from the goal of this research. Furthermore, a hot research topic of the robotics research community, simultaneous localization and mapping (SLAM), also makes extensive utilization of particle filters [27, 2].

Chapter 3
Preliminary

In this section, brief introductions to the mathematical background of particle filters, particle filter-based location inference, and symbolic model-based location inference [29] are provided. Particle filters are the main technique utilized in this paper to infer the posterior probability distributions of objects' locations. We first introduce the mathematical derivation of particle filters. Then, we present particle filter-based location inference for supporting indoor spatial queries. To the best of our knowledge, symbolic model-based location inference is the only method of drawing the probability distribution of an object's location for the purpose of indoor spatial queries in the literature. Therefore, we describe it here in order to compare it with our methods. Table 1 summarizes the notations used in this paper.

3.1 Particle Filters

In this section, we describe the formal mathematical statements of the Sampling Importance Resampling (SIR) filter (the original particle filtering algorithm) [5], which provides a technical context for later chapters.

A particle filter is a method that can be applied to nonlinear recursive Bayesian filtering problems [1]. The system under investigation is often modeled as a state vector, which contains all relevant information about the system. At least two models are critical in analyzing and making inferences about a dynamic system: the system model and the measurement model, which are given by Equations (3.1) and (3.2), respectively.

$$x_k = f_k(x_{k-1}, v_{k-1}) \tag{3.1}$$

Equation (3.1) describes how the system evolves from the state vector x_{k-1} at time $k-1$ to state vector x_k at time k. f_k is a possible nonlinear function, and v_{k-1} is an independently identically distributed (i.i.d.) process noise sequence.

$$z_k = h_k(x_k, u_k) \tag{3.2}$$

Equation (3.2) describes how observation z_k relates to the true state x_k of the system, where h_k is a possible nonlinear function and u_k is an i.i.d. measurement noise sequence.

The objective of the particle filter method is to construct a discrete approximation to the probability density function (pdf) $p(x_k|z_{1:k})$, that is, the probability of the system true state at time k given previous observations from time 1 to k. Particle filters approximate this pdf function by a set of random samples with associated weights. Simply put, every

Symbol	Meaning
q	An indoor query point
o_i	The object with ID i
C	A set of candidate objects
D	A set of sensing devices
G	The indoor walking graph
p_i	A probability distribution function for o_i in terms of all possible locations
ap_i	An anchor point with ID i
N_s	The total number of particles for an object
u_{max}	The maximum walking speed of a person
l_{max}	The maximum walking distance of a person during a certain period of time
$UR(o_i)$	The uncertain region of object o_i
s_i	The minimum shortest network distance
l_i	The maximum shortest network distance
$Area_i$	The size of a given region i

Table 3.1: Symbolic notations.

particle represents a hypothesis (sample) of the system true state; According to whether the hypothesis is consistent with the observation, each particle is assigned a different weight. The weights are determined by the principle of importance sampling with the importance density to be $p(x_k|x_{k-1})$ for the SIR filter [1]. Given $\{x_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_s}$ where $\{x_{k-1}^i, i = 1, \ldots, N_s\}$ is a set of support points (particles) with associated weights $\{w_{k-1}^i, i = 1, \ldots, N_s\}$, the support points update formula and weight update formula for the SIR filter are:

$$x_k^i \sim p(x_k|x_{k-1}^i) \tag{3.3}$$

$$w_k^i \propto w_{k-1}^i p(z_k|x_k^i) \tag{3.4}$$

From Equation (3.3), we can see that the particle x_k^i at time k is sampled from the conditional pdf $p(x_k|x_{k-1}^i)$ with x_{k-1}^i being its parent particle from time $k-1$. Theoretically, $p(x_k|x_{k-1}^i)$ is related to and can be inferred from the system model (3.1). Equation (3.4) means that the new weight w_k^i is proportional to the old weight w_{k-1}^i augmented by the observation likelihood $p(z_k|x_k^i)$, which can be inferred from the measurement model (3.2). Thus, particles which are more likely to cause an observation consistent with the true observation result z_k will gain higher weights than others.

The posterior filtered density $p(x_k|z_{1:k})$ can be approximated as

$$p(x_k|z_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(x_k - x_k^i) \tag{3.5}$$

Equations (3.3) and (3.4) are conceptual processes of how to iteratively calculate particles and their weights; however, in real world applications it is hard to derive analytical forms of $p(x_k|x_{k-1}^i)$ and $p(z_k|x_k^i)$. In our application, particles update their locations according to the object motion model employed in our work. Simply put, the object motion model assumes objects move forward with constant speeds, and can either enter rooms or continue

Algorithm 1 Resampling Algorithm

```
1. $\{\{x_k^{j*}, w_k^j\}_{j=1}^{N_s}$=RESAMPLE$[\{x_k^i, w_k^i\}_{i=1}^{N_s}]\}$
2. Initialize the CDF: $c_1 = 0$
3. for $i = 2$ to $N_s$ do
4.     Construct CDF: $c_i = c_{i-1} + w_k^i$
5. end for
6. Start at the bottom of the CDF: $i = 1$
7. Draw a starting point: $u_1 \sim \bigcup[0, N_s^{-1}]$
8. for $j = 1$ to $N_s$ do
9.     Move along the CDF: $u_j = u_1 + N_s^{-1}(j-1)$
10.    while $u_j > c_i$ do
11.        $i = i + 1$
12.    end while
13.    Assign sample: $x_k^{j*} = x_k^i$
14.    Assign weight: $w_k^j = N_s^{-1}$
15. end for
```

to move along hallways. Once an object is inside a room, it will continue to reside in the room with probability 0.9, or start to move out of the room with probability 0.1. Weights of particles are updated according to the device sensing model [3] used in this research.

Resampling is a method to solve the degeneration problem in particle filters. Degeneration means that with more iterations only a few particles would have dominant weights while the majority of others would have near-zero weights. The basic idea of resampling is to eliminate low weight particles, replicate high weight particles, and generate a new set of particles $\{x_k^{i*}\}_{i=1}^{N_s}$ with equal weights. In SIR filters, the resampling step is performed at every time index. The algorithm of resampling is shown in Algorithm 1.

3.2 Particle Filter-Based Location Inference

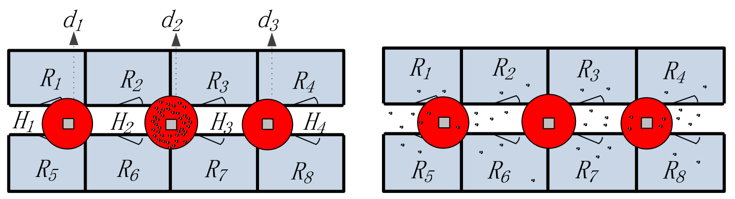
Now we are ready to explain how we apply particle filters to the problem of RFID-based indoor location inferences. We will use Figure 3.1 as an example.

In Figure 3.1, d_1, d_2 and d_3 are RFID readers which partition the hallway into four different sections labelled H_1, H_2, H_3, and H_4, respectively. Suppose from raw readings we know that a tag is first seen at d_2 at time t_0, then later is seen at d_3 at time t_1. We want to predict its location in a probabilistic form after the tag leaves the activation range of d_3.

After leaving d_3, the person carrying the tag is more likely to keep his/her original moving direction and move towards H_4 rather than backward to H_3. By their very nature, particle filters will produce filtered results consistent with our expectation. The rest of this section will explain why particle filters are able to predict this trend.

We assume particle filters start running at t_0. At first, particles represent samples drawn from the initial pdf $p(x_0)$ of the person's location. In other words, each particle represents a hypothesis of the person's state with its own location, moving direction, and speed. At t_0 the person's tag is detected by d_2, which means that the person must be somewhere within the detection range of d_2. Initially, particles are distributed randomly within the detection range of d_2 as shown in Figure 3.1(a). Every particle randomly picks its moving direction and speed. For simulating people's indoor movements, we set particles' speed to be a Gaussian distribution of $\mu = 1$ m/s and $\sigma = 0.1$.

After the initial distribution, particles update their locations according to their own speeds and directions. Some particles may move right to H_3 or possibly enter rooms R_3 and R_7; some particles may move left to H_2, R_2, and R_6. Up to time t_1, particles already become dispersed as shown in Figure 3.1(b). At t_1 a new reading is generated by d_3, when the person entered d_3's activation range. At every new observation, particle filters are going to perform the steps of reweighting and resampling. For readings from d_3, particles that are within the detection range of d_3 are assigned high weights, while particles elsewhere are assigned a very low weight. Next in the resampling step, particles are sampled with a probability proportional to their weights. Thus after resampling, most particles are replicates of previous highly-weighted particles; that is, the ones within the detection range of d_3. The



(a) Particles Distribution at t_0 (b) Particles Distribution at t_1

Figure 3.1: An example of particle filtering.

newly generated particles maintain the moving direction of those highly weighted particles, which is from left to right. Therefore, at this step after analyzing two devices' readings, particle filters already gain some knowledge of the true moving direction and speed of the person. After the person leaves d_3's activation range but before any new observation, particle filters are going to predict the person's location to be more likely in H_4 rather than in H_3 or H_2, because most particles now are moving in the direction of the hallway from left to right.

3.3 Symbolic Model-Based Location Inference

In this section, we introduce symbolic model based location inference and the corresponding range/kNN query evaluation algorithms proposed in [29] and [30]. In chapter 5, we did extensive experiments to compare the performance of our particle filter based query evaluation algorithms with the state-of-art symbolic model based algorithms, and the results reveal that our algorithm can achieve a much higher accuracy.

3.3.1 Symbolic Model and Deployment Graph

Symbolic models are different from traditional geometric coordinate models, the advantage of symbolic models lies in its ability to capture the semantics associated with indoor entities [9]. In symbolic models, a base graph describes the topology of an indoor space in which each separate partition such as a room, a staircase, or a hallway is represented as a vertex. All the space outside of the whole indoor space is also represented as one vertex, while edges capture the connectivity (undirected, such as a door connecting two rooms) or accessibility (directed, such as a one way entrance/exit) between two vertices. Figure 3.2 and 3.3 show an example of a floor plan and its symbolic model: the connectivity graph or the accessibility graph.

On the foundation of a base graph model for an indoor space, a deployment graph can be constructed according to the deployment of a particular positioning technology [9]. Basically, entities that can be accessed without having to be detected by any device are

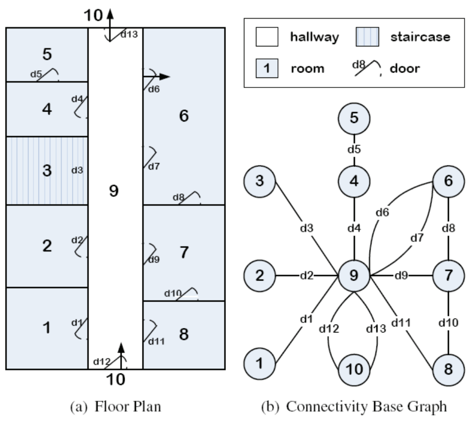


Figure 3.2: Floor plan and connectivity graph.

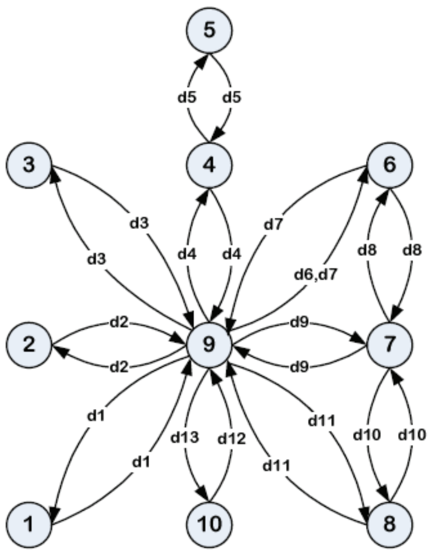
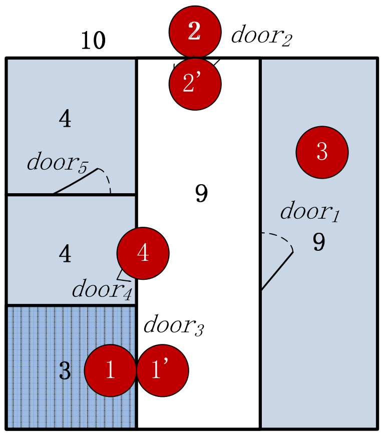
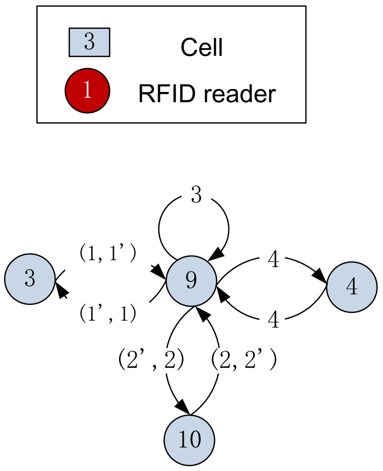


Figure 3.3: Accessibility graph.

represented by one cell in the graph, and edges connecting two cells in the graph represent the device(s) which separate them. We refer readers to [9] to see the detailed algorithm of the RFID reader deployment graph construction. Below, an example of a possible RFID reader deployment in an indoor space and its corresponding deployment graph is shown in Figure 3.4.



(a) Reader deployment in an indoor space.



(b) Reader deployment graph.

Figure 3.4: An example of the symbolic model.

In Figure 3.4(a), since one can enter the hallway in the middle from the room on the right through $door_1$ without being detected by any RFID reader, they are represented as $Cell_9$ in Figure 3.4(b); the same is true for the two rooms on the top left corner of Figure 3.4(a) as $Cell_4$. $Cell_3$ represents the staircase and is separated from other rooms or hallways by a pair of RFID readers ($reader_1$ and $reader_{1'}$). All the outside space is represented by $Cell_{10}$.

The work in [29] defines three types of positioning devices:

- Undirected Partitioning Device: it separates two cells but cannot differentiate the moving directions of objects, such as $reader_4$.

- Directed Partitioning Device: it consists of an entry/exit pair of devices, and is able to not only partition cells but also infer the moving directions of objects by the reading sequence. An example is $reader_1$ and $reader_{1'}$ in Figure 3.4.

- Presence Device: it simply senses objects within its detection range, but does not partition the space into different cells. For example, $reader_3$ is such a device in Figure 3.4.

Symbolic model-based location inference assumes an object's position is uniformly distributed over all possible locations. More specifically, we discuss several cases here to better explain how this probability model works:

Case 1: If an object is currently being observed by an RFID reader, then its possible location is anywhere in the detection range of the reader.

Case 2: If an object leaves a presence device, it must still be in the same cell as the presence device. For example, in Figure 3.4, if an object leaves $reader_3$, it must be inside $cell_9$ before being detected by any other reader.

Case 3: If an object leaves a directed partitioning device pair, the cell the object is entering can be inferred from the reading sequence. For example, if an object is seen at $reader_{1'}$ and then $reader_1$, it must enter $cell_3$.

Case 4: If an object leaves an undirected partitioning device, it can be in either of the cells that the device partitions. For example, if an object leaves $reader_4$, it can either be in $cell_4$ or $cell_9$.

Note that this inference method is very conservative in the sense that it will identify all the possible locations an object can be, but is unable to further differentiate an object's location within all possible cells. Therefore, we choose to apply the more effective particle filter-based location inference technique in our design.

3.3.2 Symbolic Model based Indoor Range Query

To effectively answer range queries with symbolic model and deployment graph, Yang etc. [29] proposed to use an indexing scheme with several hash tables. We are going to introduce the indexing scheme and how to evaluate range queries with the help of the hash tables.

DHT: A device hash table (DHT) maps a *deviceID* to the set of objects active in its reading range.

CDHT: A cell deterministic hash table (CDHT) maps a *cellID* to the set of objects that are known deterministically in it.

CNHT: A cell nondeterministic hash table (CNHT) maps a *cellID* to the set of objects that are possibly in it.

OHT: An object hash table (OHT) maps an *objectID* to a tuple $(STATE, t, IDset)$. STATE is an enumerate type which has three values: active (the object is in the active range of a reader), deterministic (the object is deterministically located in a cell), or nondeterministic (the object is possibly in a set of cells). t is the start time of the current state, and $IDset$ is a device ID, a cell ID, or a set of cell IDs corresponding to the three different states.

It is very obvious that from the four cases of location inference from section 3.3.1, case 1 in which the object enters the range of a reader, should generate an entry in the DHT, case 2 and 3 in which the object leaves a presence device or directed partitioning device, should generate an entry in CDHT, and case 4 in which the object leaves an undirected partitioning device should generate an entry in CNHT.

To answer a snapshot range query, the evaluation algorithm first identifies cells and readers that overlap with the query range, then look up DHT, CDHT, CNHT to get the result set. The algorithm categorizes the results into two types: deterministic (objects that are for sure within the query range) and nondeterministic (objects that are possibly in the query range). Cells and devices that are fully contained in the query range contribute to the deterministic result set from their CDHT and DHT, contribute to the nondeterministic result set from CNHT. Cells and devices that partially intersect with the query range can only contribute to the nondeterministic result set. However, there is one more complication here: If an object's state is nondeterministic, and cells of its IDset are all fully contained in the query range, then the object is also in the deterministic result set. Figure 3.5 shows an example of two range queries. For query 2, if an object was last seen by device 16, it is in nondeterministic state and its cellID is (12, 13); query 2 happen to fully cover cell 12 and 13, so this object should be in the deterministic result set.

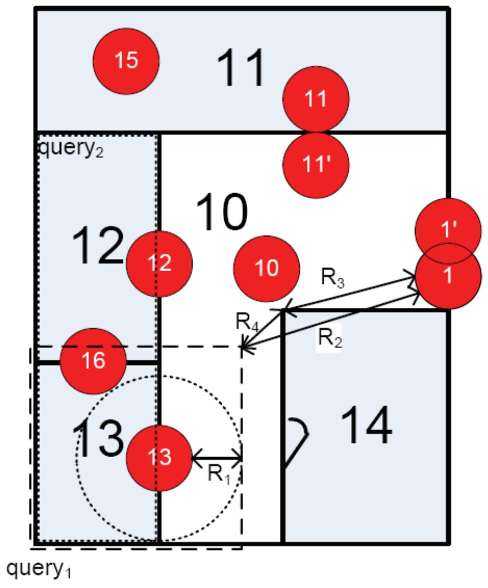


Figure 3.5: Range query example. [29]

3.3.3 Symbolic Model based Indoor *k*NN Query

With uniform distribution model at its core, symbolic model based indoor *k*NN query is not an easy task since there are many different combinations with different probabilities. Yang etc [30] proposed *k*NN query with user specified probability threshold, which can be used to filter out objects whose probability of being inside the result set is lower than the threshold. By pruning the non-candidate objects, the number of possible result sets is considerably reduced, and a large percent of the calculations can be thus saved in the final stage of probability evaluation of possible result sets.

First of all, a novel distance metric is proposed in Yang's work–minimum indoor walking distance. This new metric, as its name suggests, represents the minimum distance needed to walk from any two points in an indoor space. A precomputed door-to-door distance graph is used in their algorithm. The source/destination point first searches for the nearest door, and the shortest distance between the two doors is then calculated according to the door-to-door graph. Thus the shortest distance between source and destination points can be calculated accordingly.

A major contribution of Yang's work is that they proposed two effective pruning methods of non-candidate objects: distance based pruning and probability threshold based pruning. Distance based pruning will filter out objects whose uncertain regions are too far away from the query point that at least k objects are for sure closer to the query point. For example, figure 3.6 shows one kNN query with four objects currently in the indoor enviroment. Even the shortest distance from q to object 4's uncertain region is larger than the furthest distance of q to the uncertain regions of objects 1, 2, and 3. Therefore, object 4 can be safely pruned if this is a 3NN query. We borrow this idea in our system design to optimize particle filters based kNN queries in section 4.3, which has more formal mathematical definitions of distances. Probability threshold based pruning is one step further than the former pruning method, which calculates the probability of an object being in the result set, and filter out those whose probability is lower than threshold.

After pruning, probability evaluation of all different combinations of candidate objects is performed, and low probability combinations are eliminated. Suppose there are m objects left after initial pruning, then there are C_m^k different possible result sets needing to be evaluated. Yang etc. approximated continuous integration of pdf functions to be the sum of many discrete intervals. As their previous work [29], the main drawback of this approach is that all the probability evaluations are based on the simple but inaccurate uniform distribution assumptions. Also, the computation intensity of approximating the integration can get high when m is large.

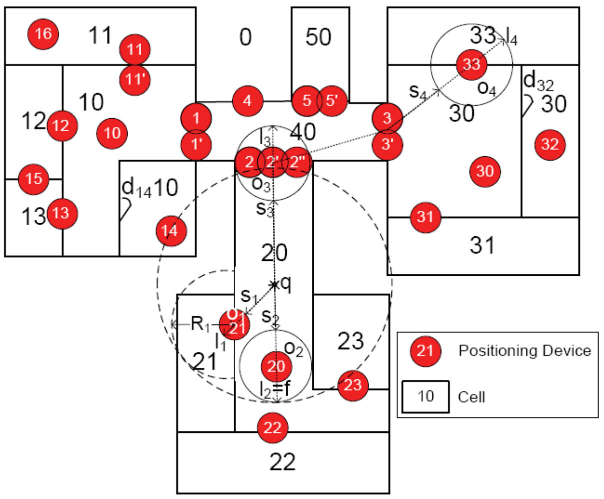


Figure 3.6: Distance based pruning. [30]

Chapter 4

Design

In this section, we will introduce the design of an RFID-based indoor range and kNN query evaluation system, which incorporates five modules: event-driven raw data collector, query aware optimization module, particle filter-based preprocessing module, cache management module, and query evaluation module. In addition, we introduce the underlying framework of two models: *indoor walking graph model* and *anchor point indexing model*. We will elaborate the function of each module and model in the following sections.

Figure 4.1 shows the overall structure of our system design. Raw readings are first fed into and processed by the event-driven raw data collector module, which then provides aggregated readings for each object at every second to the query aware optimization module, particle filter-based preprocessing module, and cache management module. The query aware optimization module filters out non-candidate objects according to registered queries and objects' most recent readings, and outputs a candidate set C to the particle filter-based preprocessing module. The particle Filter-based preprocessing module cleanses the noisy raw data for each object in C, stores the resulting probabilistic data in a hash table, and passes the hash table to the query evaluation module. At the same time, particle filter-based preprocessing module and the cache management module communicates data when necessary. The cache management module also requests data from the event-driven raw data collector module in order to age out old entries. At last, the query evaluation module answers registered range and kNN queries based on the hash table that contains filtered data.

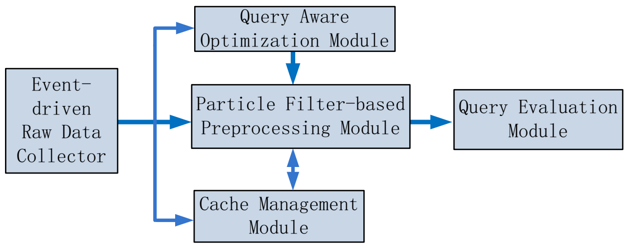

Query Aware Optimization Module
Event-driven Raw Data Collector
Particle Filter-based Preprocessing Module
Query Evaluation Module
Cache Management Module

Figure 4.1: Overall system structure.

4.1 Event-Driven Raw Data Collector

In this section, we describe the event-driven raw data collector which is the front end of the entire system. The data collector module is responsible for storing RFID raw readings in an efficient way for the following query processing tasks. Considering the characteristics of particle filtering, readings of one detecting device alone cannot effectively infer an object's moving direction and speed, while readings of two or more detecting devices can. We define events in this context as the object either entering (ENTER event) or leaving (LEAVE event) the reading range of an RFID reader. To minimize the storage space for every object, the data collector module only stores readings during the most recent ENTER, LEAVE, ENTER events, and removes earlier readings. In other words, our system only stores readings of up to the two most recent consecutive detecting devices for every object. For example, if an object is previously identified by d_i and d_j, readings from d_i and d_j are stored in the data collector. When the object is entering the detection range of a new device d_k, the data collector will record readings from d_k while removing older readings from d_i.

The data collector module is also responsible for aggregating the raw readings to more concise entries with a time unit of one second. The reasons are twofold: RFID readers usually have a high reading rate of tens of samples per second. However, particle filters do not need such a high observation frequency. An update frequency of once per second would provide a good enough resolution. Therefore, aggregation of the raw readings can further save storage without compromising accuracy. Another advantage of aggregating is to significantly mitigate the effects of missing readings. With tens of samples per second, as long as an object is detected at least once during a second, an entry marking that event is inserted into the aggregated results. It is very unlikely that all the readings of an object during one second are totally missed by a reader. Thus aggregation can greatly reduce the detecting errors of false negatives.

It is worth noting that since this research focuses on snapshot queries launched at the present time, the data collector module can be designed as above to save storage space. For systems which are required to answer historical queries, the data collector module needs to be modified accordingly to keep a longer reading history.

4.2 Indoor Walking Graph Model and Anchor Point Indexing Model

This section introduces the underlying assumptions and backbone models of our system, which forms the basis for understanding subsequent sections. We propose two novel models in our system, indoor walking graph model and anchor point indexing model, for tracking object locations in indoor environments.

Indoor Walking Graph Model: we assume our system setting is a typical office building where the width of hallways can be fully covered by the detection range of sensing devices (which is usually true since the detection range of RFID readers can be as long as 3 meters), and RFID readers are deployed only along the hallways. In this case the hallways can simply be modelled as lines, since from RFID reading results alone, the locations along the width of hallways cannot be inferred. Furthermore, since no RFID readers are deployed inside rooms, the resolution of location inferences cannot be higher than a single room.

Based on the above assumptions, we propose an *indoor walking graph model*. The indoor walking graph $G\langle N, E\rangle$ is abstracted from the regular walking patterns of people in an indoor environment, and can represent any accessible path in the environment. The graph G comprises a set N of nodes together with a set E of edges. By restricting object movements and particle movements to be only on the edges E of G, we can greatly simplify the object movement model while at the same time still preserving the inference accuracy of particle filters. Also, the distance metric used in this paper, e.g., in kNN query evaluations, can simply be the shortest spatial network distance on G, which can then be calculated by many well-known spatial network shortest path algorithms [16, 19] as shown in Figure 4.2.

Anchor Point Indexing Model: the indoor walking graph edges E are by nature continuous. To simplify the representation of an object's location distribution on E, we propose an effective spatial indexing method: anchor point-based indexing. We define anchor points as a set AP of predefined points on E with a uniform distance (such as 1 meter) to each other. An example of anchor points is shown in Figure 4.2. In essence, the model of anchor points is a scheme of trying to discretize objects' locations. After particle filtering is finished for an object o_i, every particle of o_i is assigned to its nearest anchor point, so that the inferred object location can only be on discrete locations instead of anywhere on E. For an anchor point ap_j with a nonzero number n of particles, $p_i(o_i.location = ap_j) = n/N_s$, where p_i is the probability distribution function that o_i is at ap_j and N_s is the total number of particles for o_i.

A hash table `APtoObjHT` is maintained in our system with the key to be the coordinates of an anchor point ap_j and returned value the list of each object and its probability

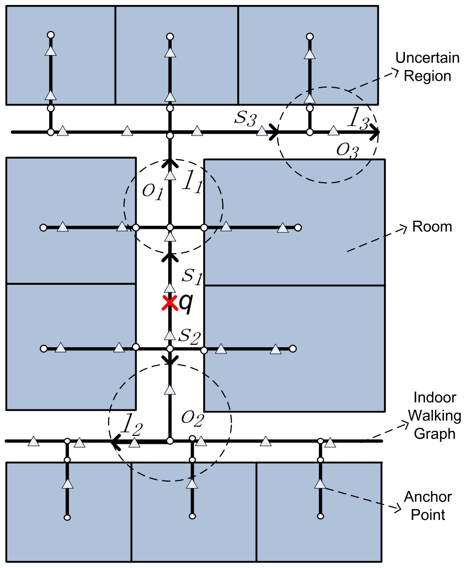


Figure 4.2: Example of filtering out kNN query non-candidate objects. Note that $s_i(l_i)$ is the minimum (maximum) shortest network distance from q to the uncertain region of o_i.

at the anchor point ($\langle o_i, p_i(ap_j) \rangle$). For instance, an entry of `APtoObjHT` would look like: $(8.5, 6.2), \{\langle o_1, 0.14\rangle, \langle o_3, 0.03\rangle,$
$\langle o_7, 0.37\rangle\}$, which means at the anchor point with coordinate (8.5, 6.2), there are three possible objects (o_1, o_3, and o_7), with probabilities of 0.14, 0.03, and 0.37, respectively. With the help of the above anchor point indexing model, the query evaluation module can simply refer to the hash table `APtoObjHT` to determine objects' location distributions.

4.3 Query Aware Optimization Module

To answer every range query or kNN query, a naive approach is to calculate the probability distribution of every object's location currently in the indoor setting. However, if query ranges cover only a small fraction of the whole area, then there will be a considerable percentage of objects who are guaranteed to be not in the result set of any query. We call those objects that have no chance to be in any result set "non-candidate objects". The computational cost of running particle filters for non-candidate objects should be saved. In this section we present two efficient methods to filter out non-candidate objects for range query and kNN query, respectively.

Range Query: to decrease the computational cost, we employ a simple approach based on the Euclidian distance instead of the minimum indoor walking distance [30] to filter out non-candidate objects. An example of the optimization process is shown in Figure 4.3. For every object o_i, its most recent detecting device d and last reading time stamp t_{last} are first retrieved from the data collector module. We assume the maximum walking speed of people to be u_{max}. Within the time period from t_{last} to the present time $t_{current}$, the maximum walking distance of a person is $l_{max} = u_{max} * (t_{current} - t_{last})$. We define o_i's uncertain region $UR(o_i)$ to be a circle centered at d with radius $r = l_{max} + d.range$. If $UR(o_i)$ does not overlap with any query range then o_i is not a candidate and should be filtered out. On the contrary, if $UR(o_i)$ overlaps with one or more query ranges then we add o_i to the result candidate

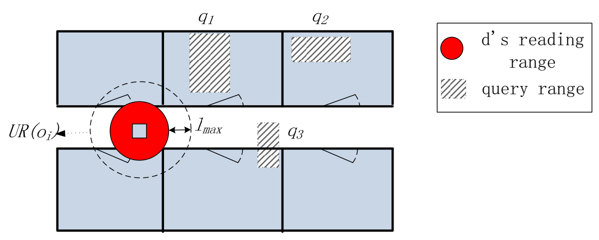


Figure 4.3: Example of filtering out range query non-candidate objects.

set C. In Figure 4.3, the only object in the figure should be filtered out since its uncertain region does not intersect with any range query currently evaluated in the system.

***k*NN Query:** by employing the idea of distance-based pruning in [30], we perform a similar distance pruning for kNN queries to identify candidate objects. We use $s_i(l_i)$ to denote the minimum (maximum) shortest network distance (with respect to the indoor walking graph) from a given query point q to the uncertain region of o_i:

$$s_i = \min_{p \in UR(o_i)} d_{shortestpath}(q,p), l_i = \max_{p \in UR(o_i)} d_{shortestpath}(q,p). \tag{4.1}$$

Let f be the k-th minimum of all objects' l_i values. If s_i of object o_i is greater than f, object o_i can be safely pruned since there exist at least k objects whose entire uncertain regions are definitely closer to q than o_i's shortest possible distance to q. Figure 4.2 is an example pruning process for a 2NN query: There are 3 objects in total in the system. We can see $l_1 < l_2 < l_3$ and consequently $f = l_2$ in this case; s_3 is greater than f, so o_3 has no chance to be in the result set of the 2NN query. We run the distance pruning for every kNN query and add possible candidate objects to C.

Finally, a candidate set C is produced by this module, containing objects that might be in the result set of one or more range queries or kNN queries. C is then fed into the particle filter preprocessing module which will be explained in the next section.

4.4 Particle Filter-Based Preprocessing Module

We design a particle filter-based algorithm (Algorithm 2) with the prior knowledge of the indoor walking graph $G\langle N, E\rangle$, anchor points set AP, and the deployment information of sensing devices D. This algorithm receives the output candidates set C from the query aware optimization module as input, infers the probability distribution of candidate objects' locations, and smooths out the result by assigning particles' locations to the nearest anchor point.

For every candidate in C, the Particle Filter algorithm first retrieves its most recent readings detected by up to two RFID readers (this number can be adjusted by users for supporting other query types) from the data collector module. If the object has just entered the system and is only detected by one sensing device, the algorithm still runs, although one device's readings alone can hardly determine the object's moving direction.

If the object is undetected by any device for a long time, it is highly likely that the object stays in a room. In this case if the particle filter continues running for a while without any observation, particles will become dispersed over a large area and the filtering result will become unusable. Line 6 restricts the particle filter from running more than 60 seconds beyond the last active reading.

The particle filter method consists of 3 steps: initialization, particle updating, and particle resampling. In the first step, a set of particles are generated and uniformly distributed on the graph edges within the detection range of d_i, and each particle picks its own moving direction and speed as in line 5. In our system, particles' speeds are drawn from a Gaussian distribution with $\mu = 1$ m/s and $\sigma = 0.1$. In the updating step, lines 8 to 16 are particles' location updates; at every time interval (1 second), particles move along the graph edges according to their speed and direction. Particles pick a random direction at intersections; if particles are inside rooms, they continue to stay inside with probability 0.9 and move out with probability 0.1. After location updating, lines 21 to 27 update particles' weights according to their consistency with reading results. In other words, particles within the

Algorithm 2 Particle Filter(C)

```
1.  for each object $o_i$ of $C$ do
2.      retrieve $o_i$'s aggregated readings from the data collector module
3.      $t_0$, $t_d$ = the starting/ending time of the aggregated readings
4.      $d_i$, $d_j$ = the second most/most recent detecting devices for $o_i$ //$d_j$ may not exist
5.      generate particles for $o_i$ within $d_i.activationRange$
6.      $t_{min} = \min(t_d + 60, t_{current})$
7.      for every second $t_j$ from $t_0$ to $t_{min}$ do
8.          for every particle $p_m$ of $o_i$ do
9.              Let $p_m$ move along graph edges $E$ with $p_m.speed$ and $p_m.direction$
10.             if $p_m$ meets intersection then
11.                 $p_m$ randomly choose a direction
12.             end if
13.             if $p_m$ resides in a room node of G then
14.                 $p_m$ moves out of room with probability 0.1
15.             end if
16.         end for
17.         retrieve the aggregated reading entry $reading$ of $t_j$
18.         if $reading.Device$=$null$ then
19.             continue
20.         else
21.             for every particle $p_m$ of $o_i$ do
22.                 if $p_m \in reading.Device.activationRange$ then
23.                     assign a high weight to $p_m$
24.                 else
25.                     assign a low weight to $p_m$
26.                 end if
27.             end for
28.             normalize the weights of all particles of $o_i$
29.             Resampling() // Algorithm 1
30.         end if
31.     end for
32.     assign particles of $o_i$ to their nearest anchor points
33.     for each anchor point $ap$ with a nonzero number of particles $n$ do
34.         calculate probability $p_i(o_i.location = ap) = n/N_s$
35.         update Hash Table APtoObjHT
36.     end for
37. end for
```

detecting device's range are assigned a high weight, while others are assigned a low weight. In the resampling step, particles' weights are first normalized as in line 28. We then employ the Resampling algorithm to replicate highly weighted particles and remove lowly weighted particles as in line 29.

Lines 33 to 36 discretize the filtered probabilistic data and build the hash table `APtoObjHT` as described in Section 4.2.

4.5 Cache Management Module

The cache management module is optional for the system functionality, but will improve the query evaluation performance if queries are frequent and geographically adjacent to previous queries. We design the cache management module to store the particle states of objects from the Particle Filter algorithm. Consequently, insertion to the cache happens every time when Algorithm 2 is done for an object o_i. In case near future queries need to determine the location distribution for the same object o_i again, we do not need to run the Particle Filter algorithm from the start; instead, previous computation is reused by retrieving the particles of o_i from the cache and resuming the Particle Filter algorithm from the cache-stored time stamp.

We also need to design a proper life time for entries in the cache. Intuitively, we know that moving patterns from a distant past provide little help to current location inferences. The same is true for particle filtering. Suppose for an object o_i, the cache stores its particles due to a previous query at time t_{prev}. In addition, assume the situation in the period from t_{prev} to $t_{current}$, o_i is detected by two or more readers. According to Algorithm 2, the old particles in the cache are useless since we only focus on readings of the most recent two readers after t_{prev}. Furthermore, in order to make the filtering process among objects consistent (*i.e.*, the particle filtering for each object is based on the readings of the most recent two devices), we decide to discard processed particles of o_i from the cache every time o_i is detected by a new device. Otherwise the particle filtering for some objects will be based on readings of more than two devices.

If the cache management module is implemented, the Particle Filter algorithm needs to be slightly modified by looking up the cache first before running from t_0. If there is a cache hit, then particle filters should run from the cache-stored time stamp. After the particle filtering step in the Particle Filter algorithm, the object ID, particle states, and current time stamp are inserted into the cache.

4.6 Query Evaluation

In this section we are going to discuss how to evaluate range and kNN queries efficiently with the filtered probabilistic data in the hash table `APtoObjHT`. For kNN queries, without loss of generality, the query point is approximated to the nearest edge of the indoor walking graph for simplicity.

4.6.1 Indoor Range Query

To evaluate indoor range queries, the first thought would be to determine the anchor points within the range, then answer the query by returning objects and their associated probabilities indexed by those anchor points. However, with further consideration, we can see that since anchor points are restricted to be only on graph edges, they are actually the 1D projection of 2D spaces; the loss of one dimension should be compensated in the query evaluation process. Figure 4.4 shows an example of how the compensation is done with respect to two different types of indoor entities: hallways and rooms.

In Figure 4.4, query q is a rectangle which intersects with both the hallway and room R_1, but does not directly contain any anchor point. We denote the left part of q which overlaps with the hallway as q_h, and the right part which overlaps with R_1 as q_r. We

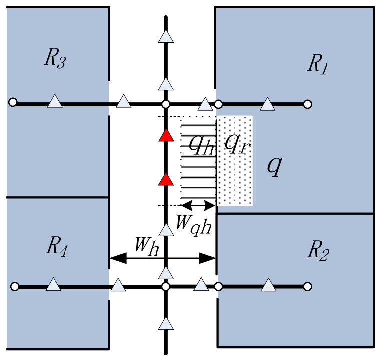


Figure 4.4: Example of indoor range query.

Algorithm 3 Indoor Range Query(q)

```
1.  resultSet=∅
2.  cells=getIntersect($q$)
3.  for every cell in cells do
4.      if cell.type=HALLWAY then
5.          anchorpoints=cell.getCoveredAP($q$)
6.          ratio=cell.getWidthRatio($q$)
7.      else if cell.type=ROOM then
8.          anchorpoints=cell.getInsideAP()
9.          ratio=cell.getAreaRatio($q$)
10.     end if
11.     result=∅
12.     for each ap in anchorpoints do
13.         result=result+APtoObjHT.get($ap$)
14.     end for
15.     result=result*ratio
16.     resultSet=resultSet+result
17. end for
18. return resultSet
```

first look at how to evaluate the hallway part of q. The anchor points which fall within q's vertical range are marked red in Figure 4.4, and should be considered for answering q_h. Since in our assumptions no differentiation along the width of hallways can be inferred about an object's true location, objects in hallways can be anywhere along the width of hallways with equal probability. With this assumption, the ratio of w_{q_h} (the width of q_h) and w_h (the width of hallway) will indicate the probability of objects in hallways within the vertical range of q being in q_h. For example, if an object o_i is in the hallway and in the vertical range of q with probability p_1, which can be calculated by summing up the probabilities indexed by the red anchor points, then the probability of this object being in q_h is $p_i(o_i.location \in q_h) = p_1 * w_{q_h}/w_h$.

Then we look at the room part of q. The anchor points within room R_1 should represent the whole 2D area of R_1, and again we assume objects inside rooms are uniformly distributed. Similar to the hallway situation, the ratio of q_r's area to R_1's area is the probability of an object in R_1 happening to be in q_r. For example, if o_i's probability of being in R_1 is p_2, then its probability of being in q_r is $p_i(o_i.location \in q_r) = p_2 * Area_{q_r}/Area_{R_1}$, where p_2 can be calculated by summing up the indexed probabilities of o_i on all the anchor points inside R_1 and $Area_i$ stands for the size of a given region i.

Algorithm 3 summarizes the above procedures. In line 15, we define the multiply operation for resultSet to adjust the probabilities for all objects in it by the multiplying constant. In line 16, we define the addition operation for resultSet to be: if an object probability pair $\langle o_i, p\rangle$ is to be added, we check whether o_i already exists in resultSet. If so, we just add p to the probability of o_i in resultSet; otherwise, we insert $\langle o_i, p\rangle$ to resultSet. For instance, suppose resultSet originally contains $\{(o_1, 0.2), (o_2, 0.15)\}$, and result stores $\{(o_2, 0.1), (o_3, 0.05)\}$. resultSet is updated to be $\{(o_1, 0.2), (o_2, 0.25), (o_3, 0.05)\}$ after the addition in line 16.

4.6.2 Indoor kNN Query

For indoor kNN queries, we present an efficient evaluation method with statistical accuracy. Unlike previous work [30, 4], which involves heavy computation and returns multiple result sets for users to choose, our method is user friendly and returns a relatively small number of candidate objects. Our method works as follows: starting from the query point q, anchor points are searched in ascending order of their distance to q; the search expands from q one achor point forward per iteration, until the sum of the probability of all objects indexed by the searched anchor points is no less than k. The result set has the form of $\langle(o_1, p_1), (o_2, p_2), ...(o_m, p_m)\rangle$ where $\sum_{i=1}^{m} p_i \geq k$. The number of returned objects will be at least k. From the sense of statistics, the probability p_i associated with object o_i in the result set is the probability of o_i being in the kNN result set of q. The algorithm of the indoor kNN query evaluation method in our work is shown in Algorithm 4.

In Algorithm 4, lines 1 and 2 are initial setups. Line 3 adds two entries to a vector V, whose elements store the edge segments expanding out from query point q. In the following for loop, line 5 finds the next unvisited anchor point further away from q. If all anchor points are already searched on an edge segment e, lines 6 to 11 remove e and add all adjacent unvisited edges of e.node to V. Line 12 updates the result set by adding ⟨object

Algorithm 4 Indoor kNN Query(q, k)

```
1.  resultSet=$\emptyset$
2.  $\overline{n_i n_j}$=find_segment($q$)
3.  vector V=$\langle(n_i, q), (n_j, q)\rangle$ // elements in V have the form (node, prevNode)
4.  for every entry $e$ in V do
5.     anchorpoint=find_nextAnchorPoint($e$) // return the next unsearched anchor point from
       e.prevNode to e.node
6.     if anchorpoint=$\emptyset$ then
7.        remove $e$ from $V$
8.        for each unvisited adjacent node $n_x$ of e.node do
9.           add ($n_x$, e.node) to V
10.       end for
11.       continue
12.    end if
13.    resultSet=resultSet+APtoObjHT.get(anchorpoint)
14.    $prob_{total}$=resultSet.getTotalProb() //calculate the probability sum of all objects in resultSet
15.    if $prob_{total}$ >= $k$ then
16.       break
17.    end if
18. end for
19. return  resultSet
```

ID, probability⟩ pairs indexed by the current anchor point to it. In lines 13 to 16, the total probability of all objects in the result set is checked, and if it equals or exceeds k, the algorithm ends and returns the result set. Note that the stopping criteria of our kNN algorithm do not require emptying the frontier edges in V.

An example kNN query is shown in Figure 4.5, which is a snapshot of the running status of Algorithm 4. In Figure 4.5, red arrows indicate the searching directions expanding from q, and red anchor points indicate the points that have already been searched. Note that the

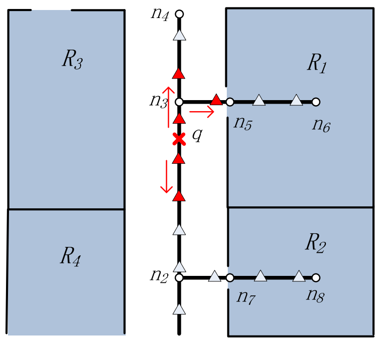


Figure 4.5: Example of indoor kNN query.

edge segment from q to n_3 is already removed from V and new edges $\overline{n_3 n_4}$, $\overline{n_3 n_5}$ are currently in V as well as $\overline{n_3 n_2}$. The search process is to be continued until the total probability of a result set is no less than k.

Chapter 5

Experiment

In this section, we evaluate the performance of the proposed RFID and particle filter-based indoor spatial query evaluation system using the data generated by real-world parameters, and compare the results with the symbolic model-based solution [30]. We implemented the proposed algorithms and related experimental components in C++. All the experiments were conducted on an Ubuntu Linux server equipped with an Intel Xeon 2.4GHz processor and 16GB memory. The settings of our experiment validation include 30 rooms and 4 hallways on a single floor, in which all rooms are connected to one or more hallways by doors. A total of 19 RFID readers are deployed on hallways with uniform distance to each other.

The whole simulator consists of seven components, including true trace generator, raw reading generator, particle filter module, symbolic model module, ground truth query evaluation, top-k success module, and KL divergence module. Figure 5.1 shows the relationship of different components in the simulation system. The true trace generator module is responsible for generating the ground truth traces of moving objects and records the true location of each object every second. We let each object randomly select a room as its destination, and walk along the shortest path on the indoor walking graph from its current location to the destination node. We simulate the objects' speeds using a Gaussian distribution with

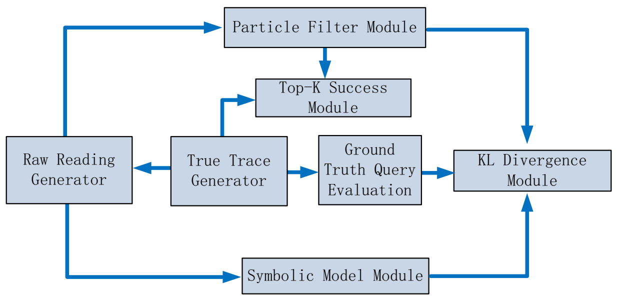


Figure 5.1: The simulator structure.

$\mu = 1$ m/s and $\sigma = 0.1$. At the same time, the raw reading generator module checks whether each object is detected by a reader according to the deployment of readers and the current location of the object. Whenever a reading occurs, the raw reading generator will feed the reading, including detection time, tag ID, and reader ID, to the two probabilistic query evaluation modules (particle filter module and symbolic model module). We also implemented the ground truth query evaluation module in order to form a basis to evaluate the accuracy of the results returned by the two probabilistic query evaluation modules.

5.1 Simulator Implementation

The query results are evaluated by the following metrics: (1) We calculated the top-1 and top-2 success rate of particle filters inferred objects' locations with respect to their true locations. The top-k success rate is a percentage of the number of objects whose true locations match the top k predicted locations of the reconstructed distribution over the total number of objects. Note that this metric only applies to the particle filter-based method. (2) For range queries, we employed the metric of Kullback-Leibler (KL) divergence to measure the accuracy of query results based on the two different probabilistic models. KL divergence is a metric commonly used to evaluate the difference between two probability distributions [11]. KL divergence is defined in Equation 5.1 with two probability distributions P and Q of a discrete random variable. In the following experiments, smaller KL divergence indicates better accuracy of the results with regard to the ground truth query results. (3) For kNN queries, KL divergence is no longer a suitable metric since the result sets returned from the symbolic model module do not contain object-specific probability information. Instead, we simply count the hit rates of the results returned by the two probabilistic methods over the ground truth result set. We only consider the maximum probability result set generated by the symbolic model module when calculating hit rate.

$$D_{KL}(P||Q) = \sum_i P(i) \ln \frac{P(i)}{Q(i)} \tag{5.1}$$

Parameters	Default Values
Number of particles	64
Query window size	2%
Number of moving objects	200
k	3
Activation range	2 meters

Table 5.1: Default values of parameters.

In all the following experimental result figures, we utilize PF and SM to represent the curves of the particle filter-based method and the symbolic model-based method, respectively. The default parameters of all the experiments are listed in Table 5.1.

5.2 Effects of Query Window Size

We first evaluate the effects of query window size on the accuracy of range queries. The window size is measured by percentage with respect to the total area of the simulated indoor space. 100 query windows are randomly generated as rectangles at each time stamp, and the results are averaged over 50 different time stamps. As shown in Figure 5.2, the KL divergence of both methods does not seem to be affected by the query window size, but the KL divergence of the particle filter-based method is significantly below that of the symbolic model-based method.

5.3 Effects of k

In this experiment we evaluate the accuracy of kNN query results with respect to the value of k. We choose 30 random indoor locations as kNN query points and issue queries on

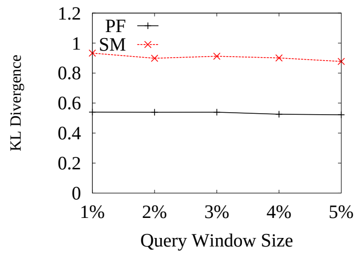


Figure 5.2: Effects of query window size.

these query points at 50 different time stamps. As k goes from 2 to 9, we can see in Figure 5.3 that the average hit rate of the symbolic model-based method grows slowly. As k increases, the number of objects returned by the method increases as well, resulting in a higher chance of hits. On the contrary, the average hit rate of the particle filter-based method is relatively stable with respect to the value of k, and the particle filter-based method always outperforms the symbolic model-based method in terms of the average hit rate.

5.4 Effects of Number of Particles

From the mathematical analysis of particle filters in Section 3.1, it is known that if the number of particles is too small, the accuracy of particle filters will degenerate due to insufficient samples. On the opposite, keeping a large number of particles is not a good choice either since the computation cost may become overwhelming, as the accuracy improvement is no longer obvious when the number of particles is beyond a certain threshold. In this subsection, we conduct extensive experiments to exploit the effects of the number of particles on query evaluation accuracy in order to determine an appropriate size of particle set for the application of indoor spatial queries.

As shown in Figure 5.4, we can see that when the number of particles is very small, the particle filter-based method has a larger KL divergence for range queries and a smaller average hit rate for kNN queries than the symbolic model-based method. As the number of particles grows beyond 8, the performance of the particle filter-based method begins to exceed the symbolic model-based method. Another observation is that when the number of particles is beyond 64, the top-k success rates of our solution are relatively stable. In

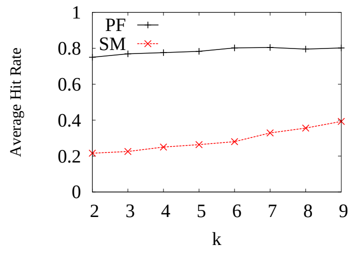
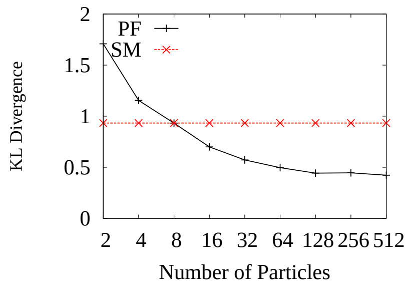
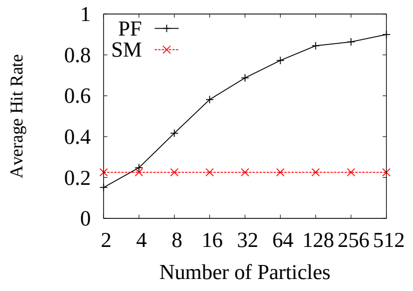


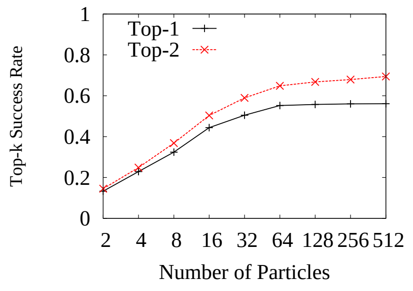
Figure 5.3: Effects of k.



(a) KL divergence.

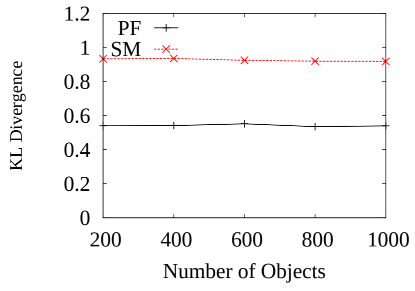


(b) kNN success ratio.

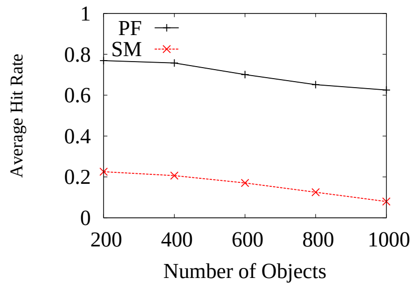


(c) Top-k success rate.

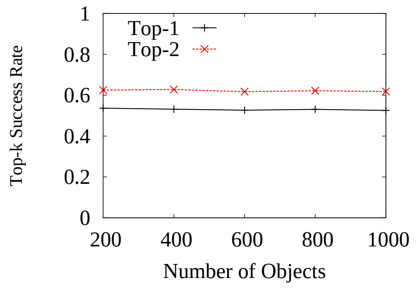
Figure 5.4: The impact of the number of particles.



(a) KL divergence.

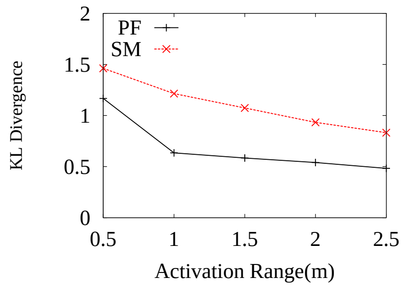


(b) kNN success ratio.

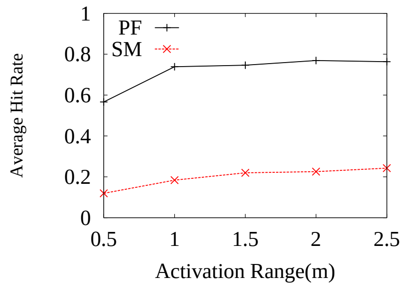


(c) Top-k success rate.

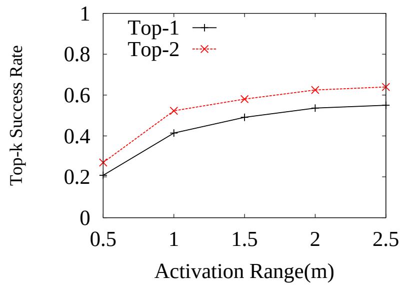
Figure 5.5: The impact of the number of moving objects.



(a) KL divergence.



(b) kNN success ratio.



(c) Top-k success rate.

Figure 5.6: The impact of activation range.

addition, both the KL divergence and the average hit rate change slowly when the number of particles grows beyond 64. We conclude that in our application, the appropriate size of particle set is around 60, which guarantees a good accuracy while not costing too much in computation.

5.5 Effects of Number of Moving Objects

In this subsection, we evaluate the scalability of our proposed algorithm by varying the number of moving objects from 200 to 1000. All the result data are collected by averaging an extensive number of queries over different query locations and time stamps. Figure 5.5 shows a comparison of the query results from the two probabilistic methods, and also the top-k success rates of particle filters' inferred locations. The KL divergence of the two methods and top-k success rates of the particle filter-based method are relatively stable, but the average hit rate of kNN queries decreases for both methods. The decrease of kNN hit rate is due to more objects being distributed in the same indoor space. A finer resolution algorithm is required to accurately answer kNN queries. In all, our solution demonstrates good scalability in terms of accuracy when the number of objects increases.

5.6 Effects of Activation Range

Finally, we evaluated the effects of reader's activation range by varying the range from 50 cm to 250 cm. The results are reported in Figure 5.6. As the activation range increases, the performance of the two methods gets better because uncertain regions not covered by any reader essentially get reduced. In addition, even when the activation range is small (e.g., 100 cm), the particle filter-based method is still able to achieve relatively high accuracy. Therefore, the particle filter-based method is more suitable than the symbolic model-based method when the physical constrains limit readers' activation ranges.

Chapter 6

Continuous Spatial Queries

This chapter presents tentative solutions to evaluate continuous indoor range query and continuous indoor kNN query, but their accuracy and performance has not been validated by simulation yet. Our ongoing research will conduct further simulation to compare the performance of our proposed methods to current state-of-art methods [29, 30].

6.1 Continuous Indoor Range Query

In this section, we aim to solve the problem of continuous indoor range query on filtered probabilistic data. We give users the option of setting a probability threshold to truncate low probability candidate objects.

To efficiently monitor the result set, we use similar concept *"critical device"* as in [29], which can save considerable computations than constantly repeating the snapshot algorithm. We define *critical devices* for a query to be the set of devices whose readings will affect the query results. Our continuous monitoring algorithm is distinct from Yang's work [29] in two aspects: 1. We leverage the Indoor Walking Graph to simplify the identification process of critical devices; 2. The probability updating process is particle filters based, which are more accurate and very different from their approach in nature.

To identify *critical devices* for a range query, we propose an approach consisting of two steps, mapping and searching. For the mapping step, we categorize two different cases and treat them differently: *case 1*, the whole query range is within one room or adjacent rooms, and we map the query range to the door(s) of the containing room(s). The reason is that doors are the only entrances/exits of rooms. *Case 2*, the query range also overlaps with hallways, and it is mapped to an edge segment of Indoor Walking Graph edges E lying along the hallway. For the searching step, an expansion starting from the mapped points or edge segments is performed along E until reaching the activation range of an RFID reader or deadend. Notice that in the second case of mapping step, we do not specially deal with the room part of queries, since the subsequent expansion from the mapped edge segment will eventually go through the doors of affected rooms.

Figure 6.1 shows an example of the mapping process for two queries q_1 and q_2. Since q_1 is fully contained in room R_1, it is mapped to a point at the door of R_1. q_2 intersects with not only rooms but also hallway, therefore it is mapped to an edge segment along hallway as marked red in figure 6.1;

For the initial evaluation of a query, we change the optimization algorithm in section 4.3 of the snapshot query to fully take advantage of critical devices. For an object to be in the query range, it must be mostly recently detected by a critical device or any device that is bounded by the critical devices. Therefore, we no longer need to calculate the maximum bounding circle of each object to determine whether it is a candidate as in section 4.3. Other than the difference in identifying the candidate object set, other parts of the initial evaluation algorithm are the same as its snapshot counterparts. After initial evaluation, we continuously monitor the candidate set by performing particle filters for them at every time step. Note here for continuous query evaluation, particle filters algorithm should run with cache in order to avoid repetitive calculations and thus gain higher speed.

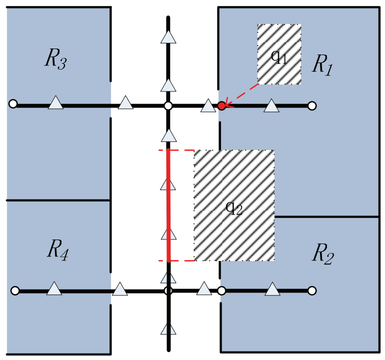


Figure 6.1: Mapping process to identify critical devices.

Algorithm 5 Continuous Range Query(q)

```
1.  $D_{criticaldevices} = getCriticalDevices(q)$
2.  $C = \emptyset$
3.  for every $reader$ in or bounded by $D_{criticaldevices}$ do
4.      $C = C \bigcup DtoObj(reader)$
5.  end for
6.  Particle Filter with Cache($C$)
7.  $R_{init}$=Indoor Range Query($q$)
8.  for every second from $t_{reg}$ to $t_{unreg}$ do
9.      for every $o_i$ detected by any reader in $D_{criticaldevices}$ do
10.         if $o_i \in C$ then
11.             $C$.remove($o_i$)
12.         else
13.             $C$.add($o_i$)
14.         end if
15.     end for
16.     for every $o_i \in C$ do
17.         if $o_i$.particles are all outside the bounded region of $D_{criticaldevices}$ then
18.             $C$.remove($o_i$)
19.         end if
20.     end for
21.     Particle Filter with Cache($C$)
22.     $R$=Indoor Range Query($q$)
23. end for
```

At the same time, the candidate set may change due to candidates moving out or non candidate objects moving into the critical device bounded region. Therefore, how to efficiently update the candidate set becomes a critical problem. Again, we rely on critical devices to gain information about possible outgoing and incoming objects. During continuous monitoring, if a candidate object enters the activation range of critical devices, or none of its particles still reside in the bounded region, then we assume it is moving out and should be removed from the candidate set. On the other hand, if a non candidate object enters the range of critical devices, we assume it is moving into the bounded region, and should be added to the candidate set.

Algorithm 5 summarizes our proposed continuous indoor range query algorithm. Lines 1 to 6 initializes the critical devices and candidate set for query q. In line 5 we use a new hash table $DtoObj$, which maps a device to objects whose most recent readings are from this device. Lines 9 to 20 are updating the candidate set according to the readings of critical devices, and also particles' presence within the bounded region. Line 21 run Algorithm 2 with cache to update candidate objects' location distribution probability. Line 22 calculates the result set using Algorithm 3. Note there is no need to recompute the anchor point set affecting query q in line 22, since it is already calculated in line 7 and remain unchanged until the query unregisters from the system.

6.2 Continuous Indoor kNN Query

Similar to continuous indoor range query, how to update the candidate set of continuous indoor kNN query is crucial. To reduce the overhead of computing candidate set at every time step, we buffer certain number of extra candidates than necessary, and only recompute the candidate set according the optimization approach in section 4.3 when the total number of candidates is less than k.

Recalling from section 4.3, by examining the minimum/maximum shortest network distance from the query point q to an object's maximum speed circle, the snapshot optimization approach excludes objects whose minimum shortest network distance are larger than the kth largest minimum shortest network distance. Note here the candidate set identified by this method contains $k + m$ candidates, where $m >= 0$. Based on this snapshot optimization approach, we extend it to include at least $k + y$ candidates where y is a user configurable parameter. Obviously, y represents a compromise between the size of candidate set and the recomputation frequency. We accomplish this by calculating the $k + y$th largest minimum shortest network distance among all objects, and use this value as a threshold to cut off non-candidate objects. In this way, the candidate set would contain $k + y + m$ objects, ensuring that we have at least $k + y$ candidates.

During continuous monitoring, we need to make sure the candidate set gets updated accordingly when a candidate object moves away or non candidate object moves towards q. We use a similar concept of "critical device" as in continuous indoor range query, although the critical devices here may change each time the candidate set is recomputed. The identification process of critical devices goes like following: after calculating the candidate set, a search is performed from q along E to cover all the maximum speed circle of candidate objects, until reaching readers (critical devices) or dead ends. As we can see, critical devices form a bounded region where at least $k + y$ candidate objects are for sure inside it.

On every new reading of critical devices, if the involved object is non candidate, we assume it is entering the bounded region and moving towards q. Accordingly, this object should be added to the candidate set. Otherwise, if it is a candidate object, we assume it is moving out of the bounded region and should be removed from the candidate set. As long as the total number of candidates is no less than k, there is no need to recompute the candidate set or critical devices. And the query result can be calculated according to Algorithm 4 for every time step. However, if there are more outgoing objects than incoming objects, the total number of candidates may fall below k. In this case, we restart the optimization algorithm in section 4.3 to get a new candidate set of at least $k + y$ objects.

Chapter 7

Conclusion

In this paper, we introduced an RFID and particle filter-based indoor spatial query evaluation system. In order to evaluate indoor spatial queries with unreliable data collected by RFID readers, we proposed the particle filter-based location inference method, the indoor walking graph model, and the anchor point indexing model for cleansing noisy RFID raw data. After the data cleansing process, indoor range and kNN queries can be evaluated efficiently and effectively by our algorithms. Our experiments with data generated by real-world parameters demonstrate that our solution outperforms the symbolic model-based method significantly in query result accuracy.

For future work, we plan to conduct further analysis of our system with more performance evaluation metrics. In addition, we intend to extend our framework to support more spatial query types such as continuous range, continuous kNN, closest-pairs, etc.

Bibliography

[1] M. S. Arulampalam, S. Maskell, N. J. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[2] L. Carlone, M. E. K. Ng, J. Du, B. Bona, and M. Indri. Rao-Blackwellized Particle Filters multi robot SLAM with unknown initial correspondences and limited communication. In *ICRA*, pages 243–249, 2010.

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

[4] R. Cheng, L. Chen, J. Chen, and X. Xie. Evaluating probability threshold k-nearest-neighbor queries over uncertain data. In *EDBT*, pages 672–683, 2009.

[5] N. Gordon, D. Salmond, and A. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F on Radar and Signal Processing*, 140(2):107–113, apr 1993.

[6] G. R. Hjaltason and H. Samet. Distance Browsing in Spatial Databases. *ACM Trans. Database Syst.*, 24(2):265–318, 1999.

[7] S. R. Jeffery, M. J. Franklin, and M. N. Garofalakis. An Adaptive RFID Middleware for Supporting Metaphysical Data Independence. *VLDB J.*, 17(2):265–289, 2008.

[8] S. R. Jeffery, M. N. Garofalakis, and M. J. Franklin. Adaptive Cleaning for RFID Data Streams. In *VLDB*, pages 163–174, 2006.

[9] C. S. Jensen, H. Lu, and B. Yang. Graph Model Based Indoor Tracking. In *Mobile Data Management*, pages 122–131, 2009.

[10] W.-S. Ku, H. Chen, H. Wang, and M.-T. Sun. A Bayesian Inference-Based Framework for RFID Data Cleansing. *IEEE Trans. Knowl. Data Eng.*, In press, 2012.

[11] S. Kullback and R. A. Leibler. On Information and Sufficiency. *Annals of Mathematical Statistics*, 22:49–86, 1951.

[12] K. C. K. Lee, W.-C. Lee, B. Zheng, and Y. Tian. ROAD: A New Spatial Object Search Framework for Road Networks. *IEEE Trans. Knowl. Data Eng.*, 24(3):547–560, 2012.

[13] J. Letchner, C. Ré, M. Balazinska, and M. Philipose. Access Methods for Markovian Streams. In *ICDE*, pages 246–257, 2009.

[14] D. Li and D. L. Lee. A Lattice-Based Semantic Location Model for Indoor Navigation. In *MDM*, pages 17–24, 2008.

[15] Metropolitan Transportation Authority. Subway and Bus Ridership Statistics 2011. `http://mta.info/nyct/facts/ridership/index.htm`. Retrieved on October 22, 2012.

[16] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query Processing in Spatial Network Databases. In *VLDB*, pages 802–813, 2003.

[17] C. Ré, J. Letchner, M. Balazinska, and D. Suciu. Event queries on correlated probabilistic streams. In *SIGMOD Conference*, pages 715–728, 2008.

[18] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest Neighbor Queries. In *SIGMOD Conference*, pages 71–79, 1995.

[19] H. Samet, J. Sankaranarayanan, and H. Alborzi. Scalable network distance browsing in spatial databases. In *SIGMOD Conference*, pages 43–54, 2008.

[20] B. L. D. Santos and L. S. Smith. RFID in the Supply Chain: Panacea or Pandora's Box? *Commun. ACM*, 51(10):127–131, 2008.

[21] L. Sullivan. RFID Implementation Challenges Persist, All This Time Later. *InformationWeek*, October 2005.

[22] T. T. L. Tran, C. Sutton, R. Cocci, Y. Nie, Y. Diao, and P. J. Shenoy. Probabilistic Inference over RFID Streams in Mobile Environments. In *ICDE*, pages 1096–1107, 2009.

[23] R. Want. The Magic of RFID. *ACM Queue*, 2(7):40–48, 2004.

[24] E. Welbourne, L. Battle, G. Cole, K. Gould, K. Rector, S. Raymer, M. Balazinska, and G. Borriello. Building the Internet of Things Using RFID: The RFID Ecosystem Experience. *IEEE Internet Computing*, 13(3):48–55, 2009.

[25] E. Welbourne, N. Khoussainova, J. Letchner, Y. Li, M. Balazinska, G. Borriello, and D. Suciu. Cascadia: a system for specifying, detecting, and managing rfid events. In *MobiSys*, pages 281–294, 2008.

[26] E. Welbourne, K. Koscher, E. Soroush, M. Balazinska, and G. Borriello. Longitudinal study of a building-scale rfid ecosystem. In *MobiSys*, pages 69–82, 2009.

[27] J. Welle, D. Schulz, T. Bachran, and A. B. Cremers. Optimization techniques for laser-based 3D particle filter SLAM. In *ICRA*, pages 3525–3530, 2010.

[28] Wikipedia. New York City Subway. `http://en.wikipedia.org/wiki/New_York_City_Subway`. Retrieved on October 22, 2012.

[29] B. Yang, H. Lu, and C. S. Jensen. Scalable continuous range monitoring of moving objects in symbolic indoor space. In *CIKM*, pages 671–680, 2009.

[30] B. Yang, H. Lu, and C. S. Jensen. Probabilistic threshold k nearest neighbor queries over moving objects in symbolic indoor space. In *EDBT*, pages 335–346, 2010.

[31] L. Yang, J. Cao, W. Zhu, and S. Tang. A hybrid method for achieving high accuracy and efficiency in object tracking using passive rfid. In *PerCom*, pages 109–115, 2012.