RFID data cleansing with Bayesian Filters

by

Zhitao Gong

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
November 7, 2016

Keywords: data cleansing, range query, kNN query, particle filter, Kalman filter

Approved by

Wei-Shinn Ku, Associate Professor of Computer Science and Software Engineering
James Cross, Professor of Computer Science and Software Engineering
Cheryl Seals, 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 Bayesian filtering techniques 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 Bayesian filtering-based location inference methods 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

I would like to thank Ms. Jiao Yu for her insightful discussion and contribution. We both work on this work. Ms. Yu worked on the particle filter and snapshot queries, while I focus on Kalman filter and continuous queries.

In addition, I would also like to thank Dr. Ku for his invaluable guidance during my research. I learned how to think like a researcher and how to narrow down areas of focus during our weekly discussion. I also greatly appreciate his patience for my really slow research progress.

Table of Contents

List of Figures

List of Tables

Part I

Introduction

Today most people spend a significant portion of their time in indoor spaces such as subway systems, office buildings, shopping malls, convention centers, and etc. In addition, indoor spaces are increasingly larger and more complex. For instance, the New York City Subway has 469 stations and 233 miles (375 km) of routes [1]. In 2014, the subway system delivered over 1.75 billion rides, averaging approximately 5.6 million rides on weekdays [2]. Therefore, users will have more and more demand for launching spatial queries for finding friends or Points Of Interest (POI) in indoor places. Moreover, users are usually moving around when issuing queries. Thus we need to properly support indoor spatial queries *continuously*, e.g., reporting nearby friends in a mall when a user is shopping. However, existing spatial query evaluation techniques for outdoor environments (based on either Euclidean distance or network distance) [3, 4, 5, 6, 7] cannot be applied in indoor spaces because these techniques assume that user locations can be acquired from GPS signals, which are not available in indoor spaces. Furthermore, indoor spaces are usually modeled differently from outdoor spaces. In indoor environment, user movements are constrained by topology 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 [8], 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 where objects carry RFID tags. When a tag enters the detection range of a reader, the reader recognizes the presence of the tag and generates a record in the backend database. However, the raw data collected by RFID readers is inherently unreliable [9], with false negatives as a result of RF interference, limited detection range, tag orientation, and other environmental phenomena [10]. In addition, readers cannot cover all areas of interest because of their high cost or privacy concerns. Therefore, with RFID raw data, we cannot provide reliable support for commonly used spatial query types, e.g., range and kNN in indoor environments. Several other types of wireless communication technologies such as WiFi and Bluetooth have been employed for indoor positioning [11, 12]. However, each aforementioned technology has considerable positioning uncertainty. Furthermore, WiFi and Bluetooth are mainly utilized for locating individual users rather than supporting a centralized indoor location tracking system. It is too expensive to attach WiFi or Bluetooth transmitters ($5 per device) to monitored objects. Therefore, we focus on RFID in this research.

In this paper, 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 enters 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. We mainly focus on four types of spatial queries, range query, kNN query, continuous range query, and continuous kNN query.

Bayesian filtering techniques [13, 14] 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 Bayesian filtering-based location inference methods, 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 spatial queries can be evaluated efficiently by our algorithms with high accuracy. The contributions of this study are as follows:

- We design the Bayesian filtering-based location inference methods 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, kNN, continuous range, and continuous 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 [15] through extensive simulations using real-world parameters.

The rest of this paper is organized as follows. In Section II, we survey previous work for indoor object monitoring and spatial queries. Background knowledge of particle filters and the Kalman filter is provided in Section III. In Section IV, we introduce our Bayesian filter-based indoor spatial query evaluation system. The experimental validation of our design is presented in Section V. Section VI concludes this paper with a discussion of future work.

Part II

Related Work

Chapter 1

Indoor Spatial Queries

Outdoor spatial queries, e.g., range and kNN queries, have been extensively studied both for Euclidean space [3, 4] and road networks [5, 6, 7]. However, due to the inherent difference in spatial characteristics, indoor spatial queries need different models and cannot directly borrow mature techniques from their outdoor counterparts. Therefore, indoor spatial queries are drawing more and more research attention from industry and academia. To answer continuous range queries in indoor environments, [16] 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 in [17]. Query results are returned in two forms: certain results and uncertain results. To reduce the workload of maintaining and updating the query results, [15] further proposed the concept of *critical devices*. Only from the ENTER and LEAVE observations of the critical devices can a query's results be affected. However, the probability model utilized in [15] 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 simplified 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, [15] also addressed the problem of kNN queries over moving objects in indoor spaces. Unlike [18] which defines nearest neighbors by the minimal number of doors to go through, they proposed a novel distance metric, i.e., minimum indoor walking distance, as the underlying metric for indoor kNN queries. Moreover, [15] 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 [15] to speed up PTkNN query processing. Similarly, [17] employs the same simplified probabilistic model, thus suffering from deficiencies in probability evaluation.

Chapter 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 [19]. An RFID-based system consists of a large number of low-cost tags that are attached to objects, and readers which can identify tags through RF communications without a direct line-of-sight. RFID technologies enable exceptional visibility to support numerous track and trace applications in different fields [20]. However, the raw data collected by RFID readers is inherently noisy and inconsistent [21, 9]. Therefore, middle-ware systems are required to correct readings and provide cleansed data [22]. 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.

To overcome the above limitations, RFID data cleansing is a necessary step to produce consistent data to be utilized by high-level applications. [23] proposed a probabilistic distance-aware graph model to handle false negative in RFID readings. The main limitation is that their generative model relies on the long tracking history to detect and possibly correct RFID readings. [24] 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, their approach of adapting and applying particle filters cannot be directly applied to our settings. Another limitation is that they did not explore further utilization of the output event streams for high-level applications. [25] 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 it is not suitable for our problem of general indoor settings. [26, 27, 28] 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. [29] also proposed using particle filters for indoor tracing with RFID. however, they assumed a grid layout of RFID readers instead of only along the hallways. Thus their algorithms cannot be applied to our problem.

Part III

Preliminary

In this section, we briefly introduce the mathematical background of Bayesian filters, including the Kalman filter and particle filters, and location inference based on the two filters. Notations used in this paper are summarized in Table 2.1.

Table 2.1: Symbolic Notations

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
E	The edge set of G
N	The node (i.e., intersection) set of G
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
d_i	The ith RFID reader

Chapter 3

The Kalman Filter

Kalman filter is an optimal recursive data processing algorithm, which combines a system's dynamics model, known control inputs, and observed measurements to form an optimal estimate of system states. Note here the control inputs and observed measurements are not deterministic, but rather with a certain degree of uncertainty. The Kalman filter works by making a prediction of the future system state, obtaining measurements for that future state, and adjusting its estimate by moderating the difference between the two. The result of the Kalman filter is a new probability distribution of system state which has reduced its uncertainty to be less than either the original predicted values or measurements alone.

To help readers better understand how the Kalman filter works for location estimation, we use a simple example of one dimensional movement and location estimation. Suppose an object is moving along a horizontal line, and we are interested in estimating the object's location x with the Kalman filter. We assume the object's speed can be expressed by $d_x/d_t = u + w$, where u is a constant and w is a Gaussian random variable with a mean of zero and variance of σ_w^2. We also assume the object's initial location at t_0 follows a Gaussian distribution with mean $\hat{x}_0$ and variance σ_0^2. At a later time t_{1^-}, just before an observation is made, we get a prediction of the object's location x_{1^-} which follows a Gaussian distribution:

$$\hat{x}_{1^-} = \hat{x}_0 + u * (t_1 - t_0) \tag{3.1}$$

$$\sigma_{1^-}^2 = \sigma_0^2 + \sigma_w^2 * (t_1 - t_0) \tag{3.2}$$

As indicated by Equation (3.2), the uncertainty in the predicted location x_1 increases with the time span $t_1 - t_0$, since no measurements are made during the time span and the uncertainty in speed accumulates with time.

After the observation at t_1 is made, suppose its value turns out to be z_1 with variance $\sigma_{z_1}^2$. The Kalman filter combines the predicted value with the measured value to yield an optimal estimation with mean and variance:

$$\hat{x}_1 = \hat{x}_{1^-} + K_1 * (z_1 - \hat{x}_{1^-}) \tag{3.3}$$

$$\sigma_1^2 = \sigma_{1^-}^2 - K_1 * \sigma_{1^-}^2 \tag{3.4}$$

where $K_1 = \sigma_{1^-}^2 / (\sigma_{1^-}^2 + \sigma_{z_1}^2)$. Please refer to [14] for the derivation details.

As we can see from Equation (3.3), the optimal estimate $\hat{x}_1$ is the optimal predicted value before the measurement plus a correction term. The variance σ_1^2 is smaller than either $\sigma_{1^-}^2$ or $\sigma_{z_1}^2$. The optimal gain K_1 gives more weights to the better value (with lower variance), so that if the prediction is more accurate than the measurement then $\hat{x}_{1^-}$ is weighted more, otherwise z_1 is weighed more.

Chapter 4

The Particle Filter

Particle filter is a method that can be applied to nonlinear recursive Bayesian filtering problems [13]. The system under investigation is often modeled as a state vector x_k, which contains all relevant information about the system at time k. The observation z_k at time k is nonlinear to the true system state x_k, also the system evolves from x_k to x_{k+1} in a nonlinear fashion.

The objective of the particle filter method is to construct a discrete approximation to the probability density function (pdf) $p(x_k|z_{1:k})$ by a set of weighted random samples. We denote the weight of the i^{th} particle at time k by w_k^i. According to the equations of particle filter [13], the new weight w_k^i is proportional to the old weight w_{k-1}^i augmented by the observation likelihood $p(z_k|x_k^i)$. Thus, the particles that are more likely to result in an observation consistent with the true observation z_k will gain higher weights than the others.

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

$$p(x_k|z_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(x_k - x_k^i) \tag{4.1}$$

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 would have weights nearly zero. 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. Our work adopts sampling importance resampling filters, which performs the resampling step at every time index.

In our application, particles update their locations according to the object motion model employed in our work. Briefly, the object motion model assumes objects move forward with constant speeds, and can either enter rooms or continue to move along hallways. Weights of particles are updated according to the device sensing model [30] used in this research. An example of applying particle filters to the problem of RFID-based indoor location inferences can be found in [31].

Part IV

System Design

In this section, we will introduce the design of an RFID-based indoor range and kNN query evaluation system, which incorporates four modules: event-driven raw data collector, query aware optimization module, Bayesian filtering-based preprocessing 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 subsections.

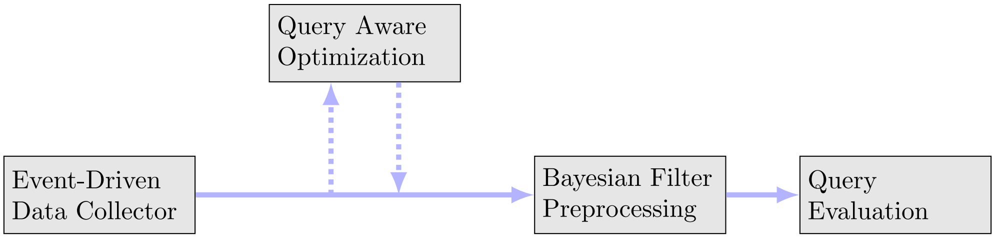


Figure 4.1: Overall system structure

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 Bayesian filtering-based preprocessing module. Before running the preprocessing module, the reading data may be optionally sent to the query aware optimization module which filters out non-candidate objects according to registered queries and objects' most recent readings, and outputs a candidate set C to the Bayesian filtering-based preprocessing module. The preprocessing module cleanses 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 last, the query evaluation module answers registered queries based on the hash table that contains filtered data.

Chapter 5

Event-Driven Raw Data Collector

In this subsection, 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 Bayesian 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 previous readings have negligible effects on the current prediction.

The data collector module is also responsible for aggregating the raw readings to more concise entries with a time unit of one second. RFID readers usually have a high reading rate of tens of samples per second. However, Bayesian filtering does 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.

Chapter 6

Indoor Walking Graph Model and Anchor Point Indexing Model

This subsection 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.

6.1 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 modeled 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 (i.e., intersections) together with a set E of edges (i.e., hallways). By restricting object 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 Bayesian filtering. 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 [5, 6], as shown in Figure 6.1.

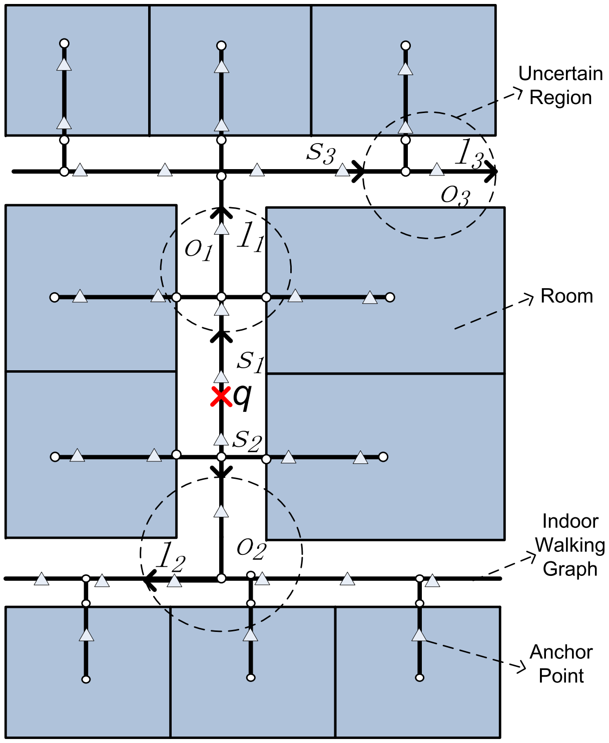


Figure 6.1: Example of filtering out kNN query non-candidate objects.

6.2 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 6.1. In essence, the model of anchor points is a scheme of trying to discretize objects' locations. After Bayesian filtering is finished for an object o_i, its location probability distribution is aggregated to discrete anchor points. Specifically, for the Kalman filter, an integration of an object's bell-shaped location distribution between two adjacent anchor points is calculated. For particle filters, suppose ap_j is an anchor point 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 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.

Chapter 7

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 not be 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 Bayesian filters for non-candidate objects should be saved. In this subsection we present two efficient methods to filter out non-candidate objects for range query and kNN query, respectively.

7.1 Range Query

To decrease the computational cost, we employ a simple approach based on the Euclidian distance instead of the minimum indoor walking distance [15] to filter out non-candidate objects. An example of the optimization process is shown in Figure 7.1. 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 set C. In Figure 7.1, 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.

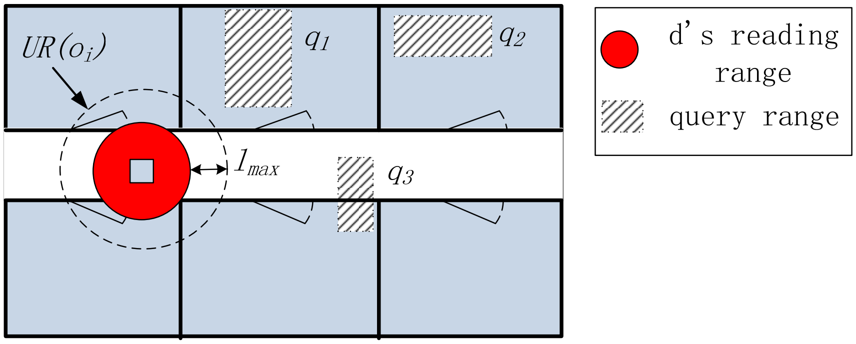


Figure 7.1: Example of filtering out range query non-candidate objects.

7.2 kNN Query

By employing the idea of distance-based pruning in [17], 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:

$$\begin{aligned} s_i &= \min_{p \in UR(o_i)} d_{shortestpath}(q, p) \\ l_i &= \max_{p \in UR(o_i)} d_{shortestpath}(q, p) \end{aligned} \tag{7.1}$$

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 6.1 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 Bayesian filtering-based preprocessing module which will be explained in the next subsection.

Chapter 8

Bayesian Filtering-based Preprocessing Module

The preprocessing module estimates an object's location distribution according to its two most recent readings, calculates the discrete probability on anchor points, and stores the results to the hash table `APtoObjHT`. We introduce two preprocessing approaches based on two famous algorithms in the Bayesian Filtering family: the *Kalman filter* and the *Particle filter*.

8.1 Kalman Filter-Based Preprocessing Module

In this section, we extend the basic 1-D example of the Kalman filter in Section 3 to be suitable for more complex 2D indoor settings. Due to the irregularity of indoor layout, the main challenge here is that an object's moving path may diverge to multiple paths. For example, in Figure 8.1, assume that an object was detected first by reader d_1 at t_1 then by reader d_2 at t_2, it could have entered R_2 or R_6 before proceeding to d_2. When we conduct a prediction with the Kalman filter, we need to consider all possible paths, each of which will give a separate prediction. Algorithm 1 formulates our approach of applying the Kalman filter to estimate objects' locations, which is elaborated in the rest of this subsection with the example in Figure 8.1.

The Kalman filter algorithm starts by first retrieving most recent readings for each candidate from the data collector module. Line 5 of Algorithm 1 restricts the Kalman filter from running more than 60 seconds beyond the last active reading, otherwise its location estimation will become dispersed over too large a area and the filtering result will become unusable.

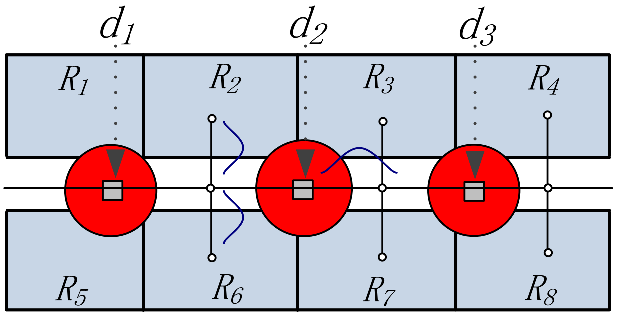


Figure 8.1: Example of Kalman filter-based prediction.

We assume objects' speed v is a Gaussian variable with mean $\mu = 1$ m/s and variance $\sigma = 0.1$, and the time of an object staying inside a room t_{room} also follows Gaussian distribution. From line 6 to 11, we assume that objects rarely enter the same room more than once. Suppose there are m rooms from d_1 to d_2, then there are $m+1$ different predictions $\hat{x}_{2^-} = \hat{x}_1 + v * (t_2 - t_1 - i * \mu_{t_{room}})$ where $i = 0, \ldots, m$ represents the number of rooms the object entered during t_1 to t_2. Note that we simplify $\hat{x}_{2^-}$ by replacing t_{room} with its mean value $\mu_{t_{room}}$.

When the observation at t_2 is made, we combine the observation with only reasonable predictions to get a final estimation. By "reasonable", we mean predictions with a good portion of pdf overlapping with d_2's reading range. For example, in Figure 8.1, the two predictions for the two paths entering R_2 and R_6 respectively are hardly overlapping with d_2's reading range, so we can safely prune them and only consider the rightmost prediction. After pruning, the average of remaining predictions is used to calculate the object's location estimation at t_2 according to Equations (3.3) and (3.4).

From the latest detected time t_2 to current, the object can take every possible path from d_2 going forward. Line 15 uses recursion to enumerate all the possibilities and line 16 calculates the probability distribution of $\hat{x}_{min^-}$ by counting the number of cases of the object in a particular room or at a particular location along the hallway divided by the total number of cases. At last, from line 18 to 21, we calculate the integration of the object's location probability distribution function from the current anchor point to its adjacent point, and

Algorithm 1 Kalman Filter(C)

```
1: for each object $o_i$ of $C$ do
2:     retrieve $o_i$'s aggregated readings from the data collector module
3:     $t_1$, $t_2$ = the starting/ending time of the aggregated readings
4:     $d_1$, $d_2$ = the second most/most recent detecting devices for $o_i$
5:     $t_{min} = \min(t_2 + 60, t_{current})$
6:     $m$ = number of rooms from $d_1$ to $d_2$
7:     for $i = 0, \ldots, m$ do
8:         $\hat{x}_{2^-} = \hat{x}_1 + v * (t_2 - t_1 - i * \mu_{t_{room}})$
9:         $\sigma^2_{2^-} = \sigma^2_1 + \sigma^2_v * (t_2 - t_1)$
10:        prune if this distribution's overlap with $d_2$'s range is below threshold
11:    end for
12:    average all the predictions
13:    calculate $\hat{x}_2$ and $\sigma^2_2$ by employing Equations 3.3 and 3.4
14:    recursively enumerate all possible paths from $\hat{x}_2$ going forward until $t_{min}$
15:    estimate $o_i$'s location $\hat{x}_{min^-}$ by counting
16:    $\sigma^2_{min^-} = \sigma^2_2 + \sigma^2_v * (t_{min} - t_2)$
17:    for each anchor point $ap_j$ with a nontrivial probability under estimated location distribution do
18:        calculate probability $p_i(o_i.location = ap_j)$
19:        update Hash Table APtoObjHT
20:    end for
21: end for
```

store the discrete probability of the object's location being on a certain anchor point to `APtoObjHT`.

8.2 Particle Filter-Based Preprocessing Module

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_2, 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 mean $\mu = 1$ m/s and $\sigma = 0.1$. In the location updating step in line 9, particles move along graph edges according to their speed and direction, and will 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, in line 16 particles' weights are updated according to their consistency with reading results. In other words, particles

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.      $t_1$, $t_2$ = the starting/ending time of the aggregated readings
4.      $d_1$, $d_2$ = the second most/most recent detecting devices for $o_i$
5.      initialize particles with random speed and direction within $d_2.activationRange$
6.      $t_{min} = \min(t_2 + 60, t_{current})$
7.      for every second $t_j$ from $t_1$ to $t_{min}$ do
8.          for every particle $p_m$ of $o_i$ do
9.              $p_m$ updates its location
10.         end for
11.         retrieve the aggregated reading entry $reading$ of $t_j$
12.         if $reading.Device$=$null$ then
13.             continue
14.         else
15.             for every particle $p_m$ of $o_i$ do
16.                 update $p_m$'s weight
17.             end for
18.             normalize the weights of all particles of $o_i$
19.             Resampling()
20.         end if
21.     end for
22.     assign particles of $o_i$ to their nearest anchor points
23.     for each anchor point $ap_j$ with a nonzero number of particles $n$ do
24.         calculate probability $p_i(o_i.location = ap_j) = n/N_s$
25.         update Hash Table APtoObjHT
26.     end for
27. end for
```

within the 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 18. We then employ the Resampling Algorithm [31] to replicate highly weighted particles and remove lowly weighted particles as in line 19. Lines 23 to 26 discretize the filtered probabilistic data and build the hash table `APtoObjHT` as described in Section 6.

Chapter 9

Query Evaluation

In this subsection 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.

9.1 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 9.1 shows an example of how the compensation is done with respect to two different types of indoor entities: hallways and rooms.

In Figure 9.1, 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 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 9.1, 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 the hallway) will indicate the probability of objects in hallways within the vertical range of q being in q_h. For

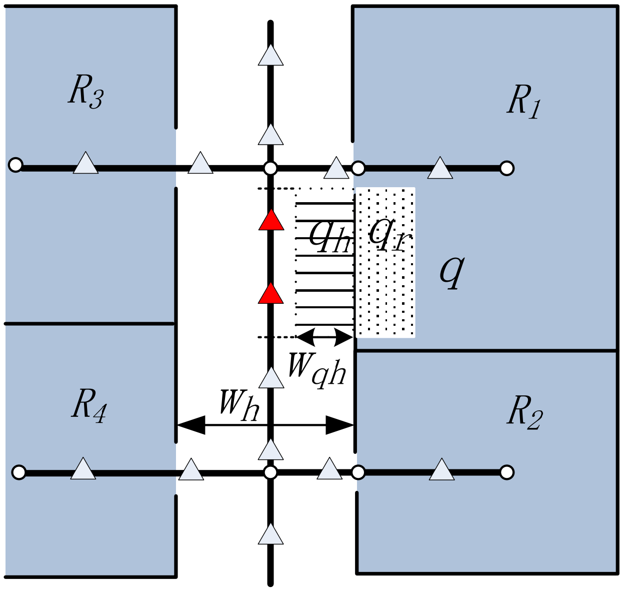


Figure 9.1: Example of indoor range query.

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

Algorithm 3 Indoor Range Query(q)

```
1.  resultSet=$\emptyset$
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=$\emptyset$
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
```

$\{(o_2, 0.1), (o_3, 0.05)\}$. `resultSet` is updated to be $\{(o_1, 0.2), (o_2, 0.25), (o_3, 0.05)\}$ after the addition in line 16.

9.2 Indoor kNN Query

For indoor kNN queries, we present an efficient evaluation method with statistical accuracy. Unlike previous work [15, 32], 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 $\langle(o_1, p_1), (o_2, p_2), \ldots, (o_m, p_m)\rangle$ 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.

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()
15.   if $prob_{total}$ >= $k$ then
16.     break
17.   end if
18. end for
19. return resultSet
```

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 12 remove e and add all adjacent unvisited edges of e.node to V. Line 13 updates the result set by adding ⟨object ID, probability⟩ pairs indexed by the current anchor point to it. In lines 14 to 17, 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 9.2, which is a snapshot of the running status of Algorithm 4. In Figure 9.2, red arrows indicate the searching directions expanding from q, and red anchor points indicate the points that have already been searched. Note that the 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

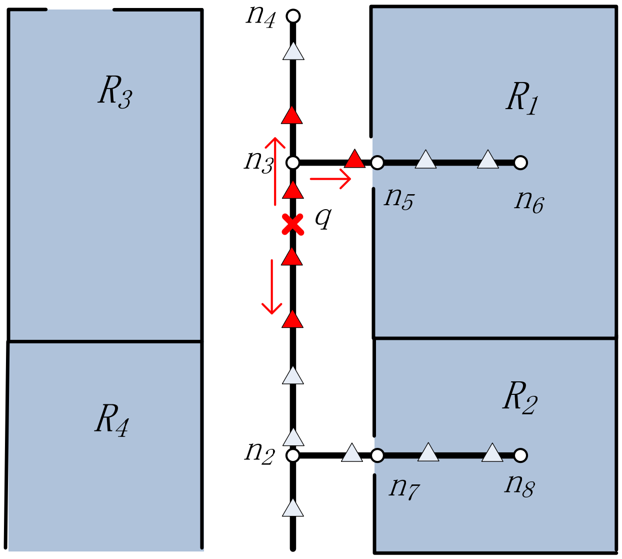


Figure 9.2: Example of indoor kNN query.

in V as well as $\overline{n_2 q}$. The search process is to be continued until the total probability of the result set is no less than k.

9.3 Continuous Indoor Range Query

In this subsection, we aim to solve the problem of continuous indoor range query on filtered probabilistic data. To efficiently monitor the result set, we use a similar concept *critical device* as in [17], which can save considerable computations rather than constantly repeating the snapshot algorithm. We define *critical devices* for a query to be only the set of devices whose readings will affect the query results. Our continuous monitoring algorithm is distinct from Yang's work [17] in two aspects: first, we leverage the Indoor Walking Graph to simplify the identification process of critical devices; second, the probability updating process is Bayesian filter-based, which is more accurate and very different from Yang's 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:

Case 1 the whole query range is contained within one room or adjacent rooms, then we project from the doors of end rooms to E along hallways. For example, q_1 in Figure 9.3 is fully contained in room R_1, so it is projected to a point (the red point) on E through the door of R_1.

Case 2 the query range overlaps with both rooms and hallways, then the endpoints of mapped edge segment(s) should take whichever makes the covered segment longer among projected points of query range ends and end rooms' doors. q_2 in Figure 9.3 is an example of this case. It is mapped to an edge segment, $\overline{ab}$, along the hallway as marked in red. Point a, room R_1 door's projected point, is chosen instead of c, the query range end projected point. Similarly, point b is chosen instead of d.

For the searching step, an expansion starting from the mapped endpoint(s) is performed along E until the activation range of an RFID reader or deadend is reached.

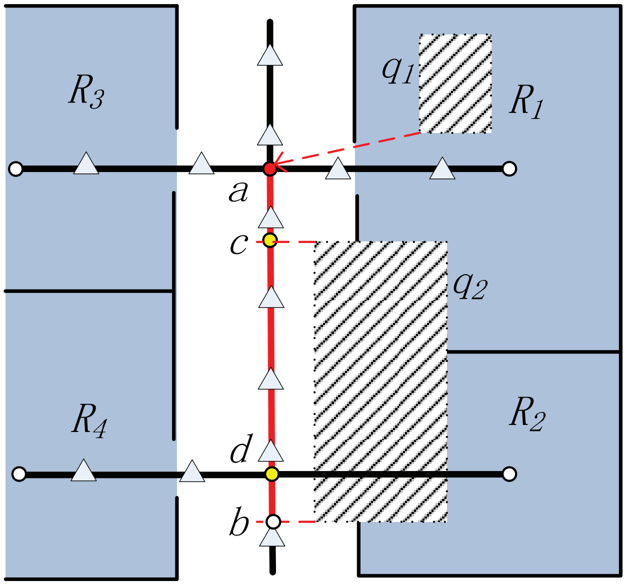


Figure 9.3: Mapping process to identify critical devices.

For the initial evaluation of a query, we change the optimization algorithm in Section 7 of the snapshot query to fully take advantage of critical devices. For an object to be in the query range, it must be most recently detected by a critical device or any device that is bounded by the critical devices. Other than the difference in identifying the candidate object set, other parts of the initial evaluation algorithm are the same as its snapshot counterpart. After initial evaluation, we continuously monitor the candidate set by performing Bayesian filters for them at every time step.

During the lifetime of a query, the candidate set may change due to candidates moving out or non-candidates moving into the critical device bounded region. If a candidate object is detected by a critical device, or the object's probability of still residing in the bounded region falls to 0, then we assume that it is moving out and should be removed from the candidate set. On the other hand, if a non-candidate object enters the detection range of a critical device, we assume it is moving into the bounded region and should be added to the candidate set.

The proposed continuous indoor range query is formalized in Algorithm 5. Lines 1 to 6 initialize the critical devices and candidate set for query q. In line 4 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 update the candidate set according to the readings of critical devices, and also objects' probabilities of presence within the bounded region. Line 21 executes Algorithms 1 or 2 to update candidate objects' location distribution probabilities. Line 22 calculates the result set using Algorithm 3. Note that for Algorithm 3 there is no need to recompute anchor point set since it remains unchanged until the query is unregistered from the system.

9.4 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 the candidate set at every time step, we buffer a certain number of extra candidates, and only recompute the candidate set according to the optimization approach in Section 7 when the total number of candidates is less than k.

Algorithm 5 Continuous Range Query(q)

```
$D_{cd} = getCriticalDevices(q)$
$C = \emptyset$
for every $reader$ in or bounded by $D_{cd}$ do
    $C = C \bigcup DtoObj(reader)$
end for
Bayesian Filter($C$)
$R_{init}$=Indoor Range Query($q$)
for every time step from $t_{reg}$ to $t_{unreg}$ do
    for every $o_i$ detected by any reader in $D_{cd}$ do
        if $o_i \in C$ then
            $C$.remove($o_i$)
        else
            $C$.add($o_i$)
        end if
    end for
    for every $o_i \in C$ do
        if $p(o_i.location \in boundedregionofD_{cd}) = 0$ then
            $C$.remove($o_i$)
        end if
    end for
    Bayesian Filter($C$)
    $R$=Indoor Range Query($q$)
end for
```

Recall from Section 7, by examining the minimum (s_i)/maximum (l_i) shortest network distance from the query point q to an object's uncertain region, the snapshot optimization approach excludes objects with $s_i > f$. Note that the candidate set identified by this method contains at least k objects (usually more than k). 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 tradeoff between the size of candidate set and the recomputation frequency. We accomplish this by calculating the $(k + y)$-th minimum l_i among all objects, and use this value as a threshold to cut off non-candidate objects.

During continuous monitoring, we need to make sure that the candidate set gets updated accordingly when objects move away or towards q. We still use critical devices to monitor candidates, but now the critical devices may change each time the candidate set is recomputed. The identification process of critical devices goes like the following: after calculating the

Algorithm 6 Continuous kNN Query(q, k, y)

```
1.  $C = getCandidateObjects(k + y)$
2.  $D_{cd} = getCriticalDevices(C)$
3.  Bayesian Filter($C$)
4.  $R_{init}$=Indoor $k$NN Query($q$, $k$)
5.  for every time step from $t_{reg}$ to $t_{unreg}$ do
6.    for every $o_i$ detected by any reader in $D_{cd}$ do
7.      if $o_i \in C$ then
8.        $C$.remove($o_i$)
9.      else
10.       $C$.add($o_i$)
11.     end if
12.   end for
13.   if $C.count < k$ then
14.     $C = getCandidateObjects(k + y)$
15.     $D_{cd} = getCriticalDevices(C)$
16.   end if
17.   Bayesian Filter($C$)
18.   $R$=Indoor $k$NN Query($q$, $k$)
19. end for
```

candidate set, a search is performed from q along E to cover all the uncertain regions of candidate objects, until reaching readers (critical devices) or deadend. As we can see, critical devices form a bounded region where at least $k + y$ candidate objects are for sure inside it.

The proposed continuous indoor kNN query is formalized in Algorithm 6. Note that in lines 13 to 16, when the total number of candidates falls below k, we need to recompute a new candidate set of at least $k + y$ objects, and identify new critical devices accordingly.

Part V

Experiment

In this section, we evaluate the performance of the proposed Bayesian filtering-based indoor spatial query evaluation system using the data generated by real-world parameters and compare the results with the symbolic model-based solution [15]. The proposed algorithms are implemented in `C++`. All the experiments were conducted on an Ubuntu Linux server equipped with an Intel Xeon 2.4GHz processor and 16GB memory. In our experiments, the floor plan, which is of the second floor of the Haley Center on Auburn University campus, includes 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.

Chapter 10

Simulator Implementation

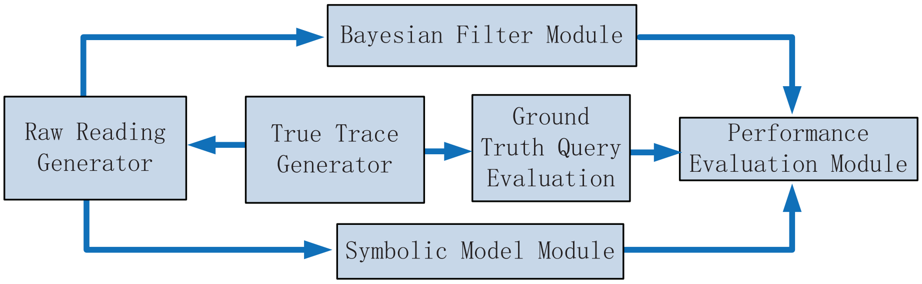


Figure 10.1: The simulator structure.

The whole simulator consists of six components, including true trace generator, raw reading generator, Bayesian filter module, symbolic model module, ground truth query evaluation, and performance evaluation module. Figure 10.1 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 recording the true location of each object every second. Each object randomly selects its destination, and walks 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 $\mu = 1$ m/s and $\sigma = 0.1$. 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 query evaluation modules (Bayesian filter module and symbolic model module). The ground truth query evaluation module forms a basis to evaluate the accuracy of the results returned by the two aforementioned query evaluation modules.

The query results are evaluated by the following metrics:

1. For range queries, we employed Kullback-Leibler (KL) divergence [33] to measure the accuracy of query results from the two modules based on their similarity with the true result. KL divergence is a metric commonly used to evaluate the difference between two probability distributions. The discrete form of KL divergence of Q from P given in Equation (10.1) measures the information loss when Q is used to approximate P. As a result, in the following experiments, smaller KL divergence indicates better accuracy of the results with regard to the ground truth.

$$D_{KL}(P||Q) = \sum_i P(i) \ln \frac{P(i)}{Q(i)} \tag{10.1}$$

2. 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 modules over the ground truth result set. We only consider the maximum probability result set generated by the symbolic model module when calculating hit rate.

In all the following experimental result figures, we use PF, KF, and SM to represent the curves of the particle filter-based method, Kalman filter-based method, and symbolic model-based method, respectively. The default parameters of all the experiments are listed in Table 10.1.

Table 10.1: Default values of parameters.

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

Chapter 11

Effects of Parameters

11.1 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 simulation space. 100 query windows are randomly generated as rectangles at each time stamp, and the results are averaged over 100 different time stamps. As shown in Figure 11.1, their accuracy is not significantly affected by the query window size. However, the KL divergence of the particle filter-based method is lower than both of the Kalman filter-based and symbolic model-based methods.

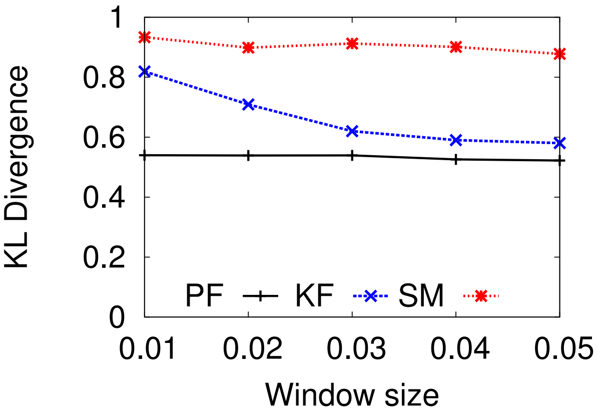


Figure 11.1: Effects of query window size.

11.2 Effects of k

In this experiment we evaluate the accuracy of kNN query results with respect to the value of k. We choose 100 random indoor locations as kNN query points and issue queries on these query points at 100 different time stamps. As k goes from 2 to 9, we can see in Figure 11.2 that the average hit rates of Kalman filter-based and symbolic model-based methods grow slowly. As k increases, the number of objects returned by the methods increase 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 other two methods in terms of the average hit rate.

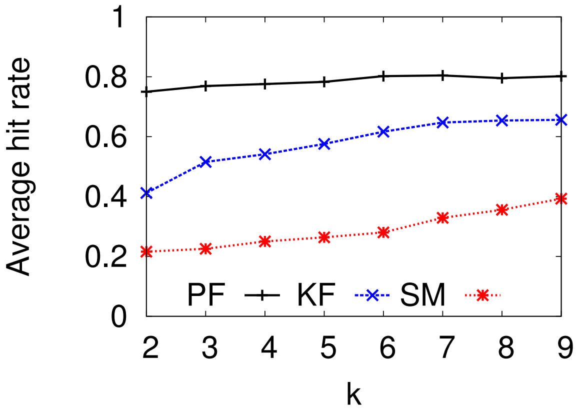
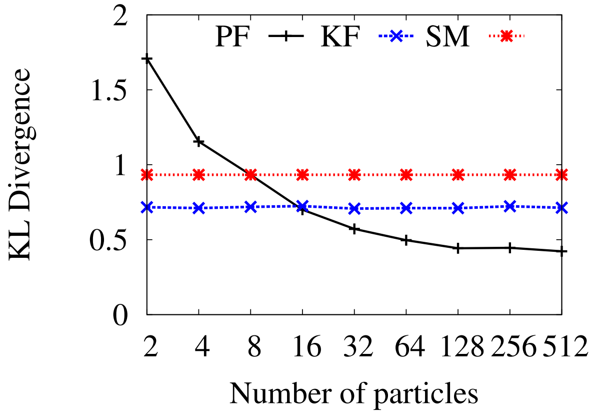


Figure 11.2: Effects of k

11.3 Effects of Number of Particles

From the mathematical analysis of particle filters in Section 4, we knew 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 result accuracy in order to determine an appropriate size of the particle set for the application of indoor spatial queries.

As shown in Figure 11.3, 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



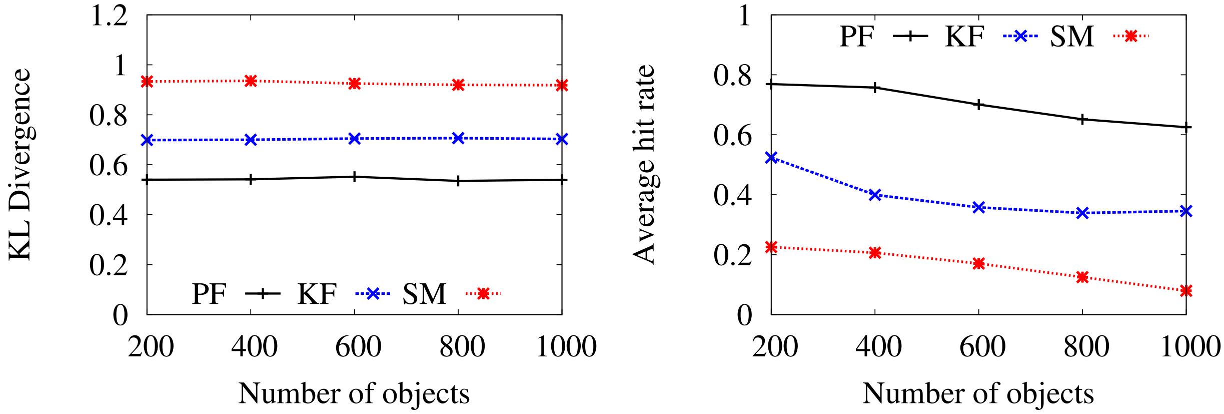
(a) KL divergence

(b) kNN success ratio

Figure 11.3: The impact of the number of particles.

average hit rate for kNN queries than the other two methods. As the number of particles grows beyond 16, the performance of the particle filter-based method exceeds the other two. However, the performance gain with more than 64 particles slows down as we already have around 90% accuracy. Therefore, we conclude that in our application, the appropriate size of the particle set is around 60, which guarantees a good accuracy while not costing too much in computation.

11.4 Effects of Number of Moving Objects



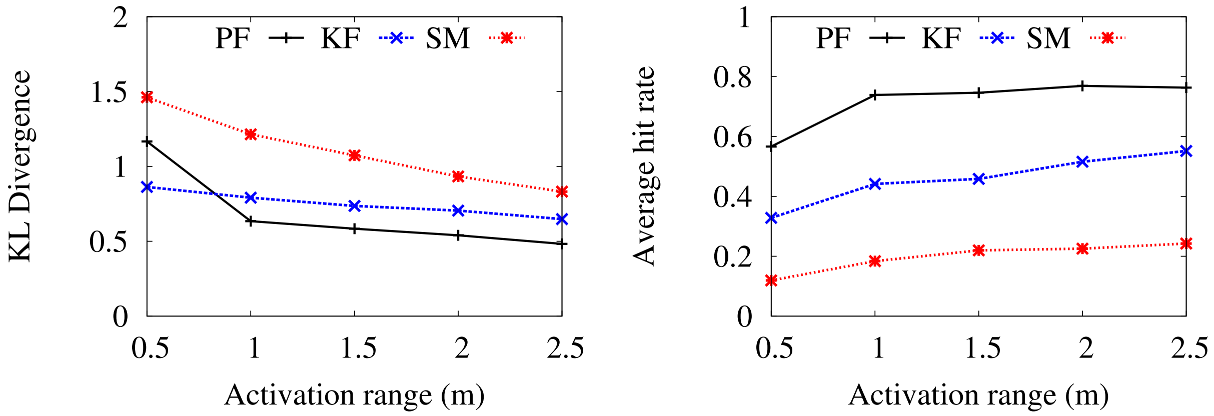
(a) KL divergence

(b) kNN success ratio

Figure 11.4: The impact of the number of moving objects.

In this subsection, we evaluate the scalability of our proposed algorithms 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 11.4 shows that the KL divergence of the three methods is relatively stable, while the average hit rate of kNN queries decreases for all the methods. The decrease of kNN hit rate is caused by increasing density of objects. 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.

11.5 Effects of Activation Range



(a) KL divergence

(b) kNN success ratio

Figure 11.5: The impact of activation range.

In this subsection, 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 11.5. As the activation range increases, the performance of all the three 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 other two methods when the physical constraints limit readers' activation ranges.

11.6 Continuous Query Performance Evaluation

The previous subsections show the performance of snapshot queries, i.e., queries at a specific time stamp. This subsection demonstrates our algorithms' performance across a duration of time. The application scenarios are described as follows:

1. For continuous range query, a user registers a query window at time t_0, and unregisters at t_1. During the time interval (between t_0 and t_1), we keep updating the user of the objects in the query window whenever a change is detected.

2. For continuous kNN query, a user registers a query point q on the walking graph (a query point which is not on the walking graph can be projected to its closest edge of the graph) at t_0, and unregisters at t_1. During the time interval, every time there is a change in the k nearest neighbor query result set, we will update the user with the new query result.

   We develop two criteria to measure the performance

   Change Volume It is defined as the number of changes of objects in the query range between two consecutive time stamps, including departing and arriving objects. Suppose at t_0, the objects in the query range are $\{a, b, c\}$; at t_1, the result set changes to $\{a, b, d\}$, then the number of changes equals to 2, because one of the objects, c, is departing and another object, d, just arrived. The rationale behind this is that higher change volume could potentially impair query result accuracy.

   Query Duration It is the interval between t_0 and t_1, where t_0 denotes the time a user registers a continuous query, and t_1 denotes the time a user unregisters the query. The rationale for this criteria is that the proposed algorithms can be evaluated as stable and reliable if they can maintain a satisfactory accuracy for a long duration.

Figure 11.6 shows the performance of our proposed algorithms with different number of changes. It is clear from the figure that our algorithms' accuracy is not heavily influenced by

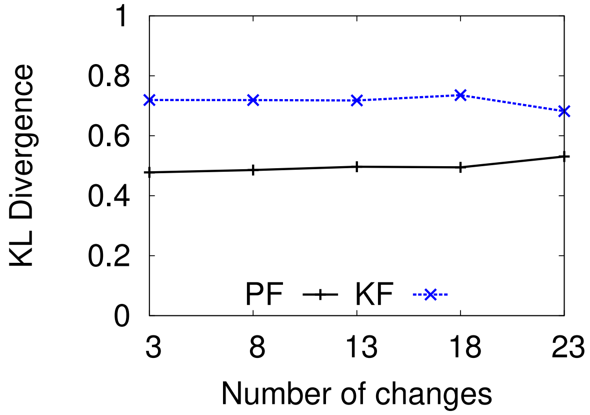
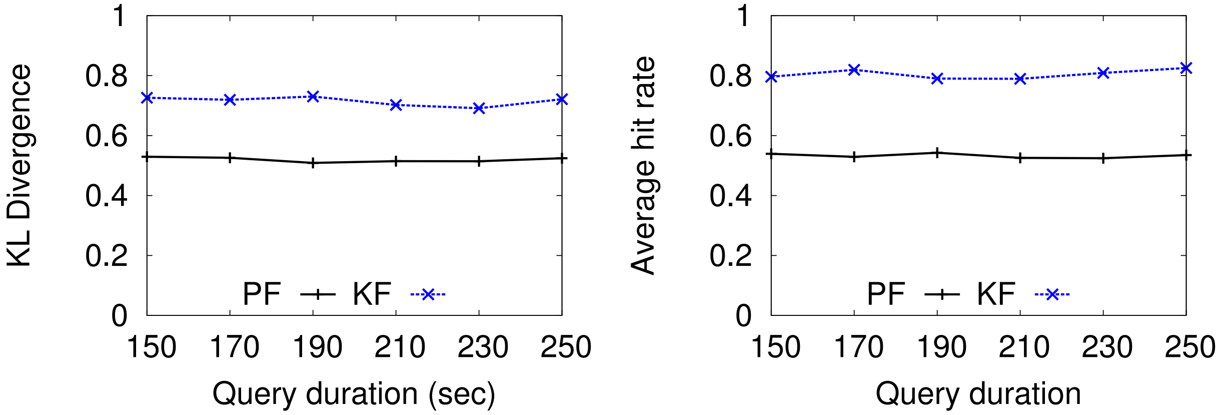


Figure 11.6: The impact of number of changes.

the change volume, although there are some fluctuations. Furthermore, Figure 11.7 shows the accuracy of our algorithms against the query duration. Once the system is stable, the accuracy of our algorithms is not affected by the duration of query time.



(a) Continuous range query

(b) Continuous kNN query

Figure 11.7: The impact of query duration.

Part VI

Conclusion

In this paper, we introduced a Bayesian filtering-based RFID data cleansing method in order to support accurate indoor spatial queries with noisy RFID data. In addition we proposed the indoor walking graph model and the anchor point indexing model to simplify the Bayesian filtering process. After the cleansing, indoor range query and kNN query can be evaluated efficiently and effectively via our algorithms. Our extensive experiment with data generated by real-world parameters demonstrates that our solution outperforms the symbolic model-based method by large margin in query result accuracy.

There are, however, a few limitations in our current solutions which will be addressed in our future work. For example, current solution are evaluated on synthesized data, We plan to conduct further analysis of our system with real data collected in the RFID lab. In addition, we intend to extend our framework to support more spatial query types such as spatial skyline, spatial joins, closest-pairs, etc.

Bibliography

[1] Wikipedia. New york city subway, 2016. Retrieved on March 15, 2016.

[2] Metropolitan Transportation Authority. Subway and bus ridership statistics 2014, 2014. Retrieved on March 15, 2016.

[3] Nick Roussopoulos, Stephen Kelley, and Frédéic Vincent. Nearest neighbor queries. In *SIGMOD Conference*, pages 71–79, 1995.

[4] Gísli R. Hjaltason and Hanan Samet. Distance browsing in spatial databases. 24(2):265–318, 1999.

[5] Dimitris Papadias, Jun Zhang, Nikos Mamoulis, and Yufei Tao. Query processing in spatial network databases. In *VLDB*, pages 802–813, 2003.

[6] Hanan Samet, Jagan Sankaranarayanan, and Houman Alborzi. Scalable network distance browsing in spatial databases. In *SIGMOD Conference*, pages 43–54, 2008.

[7] Ken C. K. Lee, Wang-Chien Lee, Baihua Zheng, and Yuan Tian. Road: A new spatial object search framework for road networks. 24(3):547–560, 2012.

[8] Brian L. Dos Santos and Lars S. Smith. RFID in the Supply Chain: Panacea or Pandora's Box? 51(10):127–131, 2008.

[9] Shawn R. Jeffery, Minos N. Garofalakis, and Michael J. Franklin. Adaptive Cleaning for RFID Data Streams. In *VLDB*, pages 163–174, 2006.

[10] Evan Welbourne, Leilani Battle, Garrett Cole, Kayla Gould, Kyle Rector, Samuel Raymer, Magdalena Balazinska, and Gaetano Borriello. Building the Internet of Things Using Rfid: The Rfid Ecosystem Experience. 13(3):48–55, 2009.

[11] Giuseppe Anastasi, Renata Bandelloni, Marco Conti, Franca Delmastro, Enrico Gregori, and Giovanni Mainetto. Experimenting an Indoor Bluetooth-Based Positioning Service. In *ICDCS Workshops*, pages 480–483, 2003.

[12] Scott Bell, Wook Rak Jung, and Vishwa Krishnakumar. Wifi-based enhanced positioning systems: accuracy through mapping, calibration, and classification. In *Indoor Spatial Awareness Workshop*, pages 3–9, 2010.

[13] M. Sanjeev Arulampalam, Simon Maskell, Neil J. Gordon, and Tim Clapp. A Tutorial on Particle Filters for Online Nonlinear/non-Gaussian Bayesian tracking. 50(2):174–188, 2002.

[14] Peter S. Maybeck. *Stochastic models, estimation, and control*, volume 141 of *Mathematics in Science and Engineering*. 1979.

[15] Bin Yang, Hua Lu, and Christian S. Jensen. Probabilistic threshold k nearest neighbor queries over moving objects in symbolic indoor space. In *EDBT*, pages 335–346, 2010.

[16] Christian S. Jensen, Hua Lu, and Bin Yang. Graph Model Based Indoor Tracking. In *MDM*, pages 122–131, 2009.

[17] Bin Yang, Hua Lu, and Christian S. Jensen. Scalable continuous range monitoring of moving objects in symbolic indoor space. In *CIKM*, pages 671–680, 2009.

[18] Dandan Li and Dik Lun Lee. A lattice-based semantic location model for indoor navigation. In *MDM*, pages 17–24, 2008.

[19] Roy Want. The magic of rfid. 2(7):40–48, 2004.

[20] Lei Yang, Jiannong Cao, Weiping Zhu, and ShaoJie Tang. A hybrid method for achieving high accuracy and efficiency in object tracking using passive rfid. In *PerCom*, pages 109–115, 2012.

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

[22] Shawn R. Jeffery, Michael J. Franklin, and Minos N. Garofalakis. An Adaptive RFID Middleware for Supporting Metaphysical Data Independence. 17(2):265–289, 2008.

[23] Asif Iqbal Baba, Hua Lu, Torben Bach Pedersen, and Xike Xie. Handling False Negatives in Indoor RFID Data. In *MDM*, pages 117–126, 2014.

[24] Thanh T. L. Tran, Charles Sutton, Richard Cocci, Yanming Nie, Yanlei Diao, and Prashant J. Shenoy. Probabilistic inference over rfid streams in mobile environments. In *ICDE*, pages 1096–1107, 2009.

[25] Wei-Shinn Ku, Haiquan Chen, Haixun Wang, and Min-Te Sun. A bayesian inference-based framework for rfid data cleansing. 25(10):2177–2191, 2013.

[26] Christopher Ré, Julie Letchner, Magdalena Balazinska, and Dan Suciu. Event queries on correlated probabilistic streams. In *SIGMOD Conference*, pages 715–728, 2008.

[27] Evan Welbourne, Nodira Khoussainova, Julie Letchner, Yang Li, Magdalena Balazinska, Gaetano Borriello, and Dan Suciu. Cascadia: a system for specifying, detecting, and managing rfid events. In *MobiSys*, pages 281–294, 2008.

[28] Julie Letchner, Christopher Ré, Magdalena Balazinska, and Matthai Philipose. Access methods for markovian streams. In *ICDE*, pages 246–257, 2009.

[29] Li Geng, Mónica F. Bugallo, Akshay Athalye, and Petar M. Djuric. Indoor Tracking With RFID Systems. 8(1):96–105, 2014.

[30] Haiquan Chen, Wei-Shinn Ku, Haixun Wang, and Min-Te Sun. Leveraging spatio-temporal redundancy for rfid data cleansing. In *SIGMOD Conference*, pages 51–62, 2010.

[31] Jiao Yu, Wei-Shinn Ku, Min-Te Sun, and Hua Lu. An rfid and particle filter-based indoor spatial query evaluation system. In *EDBT*, pages 263–274, 2013.

[32] Reynold Cheng, Lei Chen, Jinchuan Chen, and Xike Xie. Evaluating probability threshold k-nearest-neighbor queries over uncertain data. In *EDBT*, pages 672–683, 2009.

[33] Solomon Kullback and Richard A. Leibler. On information and sufficiency. 22:49–86, 1951.