Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 6937912, 12 pages
https://doi.org/10.1155/2020/6937912

Research Article

A Temporal and Spatial Data Redundancy Processing Algorithm for RFID Surveillance Data

Siye Wang[1,2,3] **Ziwen Cao**[2,3] **Yanfang Zhang**[2,3] **Weiqing Huang**[1,2,3] **and Jianguo Jiang**[1,2,3]

[1]*School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China*
[2]*Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China*
[3]*School of Cyber Security, University of Chinese Academy of Sciences, China*

Correspondence should be addressed to Siye Wang; wangsiye@iie.ac.cn

Received 9 September 2019; Accepted 7 February 2020; Published 24 February 2020

Academic Editor: Mario Kolberg

The Radio Frequency Identification (RFID) data acquisition rate used for monitoring is so high that the RFID data stream contains a large amount of redundant data, which increases the system overhead. To balance the accuracy and real-time performance of monitoring, it is necessary to filter out redundant RFID data. We propose an algorithm called Time-Distance Bloom Filter (TDBF) that takes into account the read time and read distance of RFID tags, which greatly reduces data redundancy. In addition, we have proposed a measurement of the filter performance evaluation indicators. In experiments, we found that the performance score of the TDBF algorithm was 5.2, while the Time Bloom Filter (TBF) score was only 0.03, which indicates that the TDBF algorithm can achieve a lower false negative rate, lower false positive rate, and higher data compression rate. Furthermore, in a dynamic scenario, the TDBF algorithm can filter out valid data according to the actual scenario requirements.

1. Introduction

Radio Frequency Identification (RFID) is a wireless communication technology that has the advantages of low cost, low power consumption, and easy deployment. As one of the key technologies pertaining to the Internet of Things, RFID technology is used in many fields in our daily lives, especially in the field of access monitoring. RFID devices are deployed in places where the need exists to detect personnel flow, such as the entrance of a building or an office. An RFID tag is adhered to objects such as documents and paper files and is capable of storing the identifying information of these objects [1]. One important issue is whether the tags can be detected at the building exit, as the items the tags are attached to are often critical corporate assets or confidential government documents. For example, many documents created during meetings are highly confidential. Once these documents are taken out of the meeting room, either intentionally or unintentionally, the probability of information leakage significantly increases, a leak that can cause losses to an enterprise or even a country.

Except for tag data detection accuracy, the RFID detection system requires real-time performance to decide instantaneously whether the tag carries the risk of escaping. Because loss of items, as well as information, can occur in an instant, the ability to detect the risk of loss in advance and handle it in real time is crucial. An RFID reader collects RFID data streams on the tags, keeps them within its detection range, and sends them to the server, and, as a result, a massive amount of data is generated, although most of it is redundant and useless. These redundant data not only increase network latency but also take up valuable system storage space. If not filtered, this redundant data can even lead to server overload and failure to detect important tag information. Therefore, it is vital to research redundancy filtering technology for improving the efficiency of the RFID systems and striking a balance between the needs for real-time performance and accuracy when filtering the data.

Although there are many approaches to filtering duplicate RFID data, existing approaches cannot satisfy the demands of RFID monitoring. Three problems need to be solved. First of all, most of these algorithms do not have a clear definition of what is redundancy in a specific scenario. For example, when inventory is counted, each item only needs to be counted once. If it is counted multiple times, it will cause an inventory error. If these same items are counted multiple times, it could be considered as redundancy. For instance, a reader in a supermarket is primarily concerned with the items on the target shelf, but the reader's wide detection range allows additional data to be read. We can also count these useless data as redundant data. Second, many of these approaches focus on redundant filtering in a single dimension, such as temporal or spatial, rather than combining several dimensions to achieve better filtration efficiency. Finally, the typical RFID system is more concerned with the collection of static data, which is more stable and easier to filter, than dynamic data, while, in reality, many tags are not static.

In this paper, we propose an innovative algorithm for the removal of redundant RFID data. The contributions of this paper are as follows: (a) propose an algorithm called Time-Distance Bloom Filter (TDBF) to eliminate redundant RFID data in multiple dimensions, (b) propose a novel filter efficiency measurement formula for RFID data, (c) perform experimental analysis of the proposed algorithm and its performance in static and dynamic scenarios, and (d) strike a balance between the need for real-time performance and accuracy when filtering the data in a monitoring scenario. The experiment's results show that our proposed approach demonstrates superior performance compared to the other baseline approaches.

The rest of the paper is organized as follows. In Section 2, we discuss background information, including the basic structure of the RFID system and the characteristics of RFID data streams. The related work is described in detail in Section 3. In Section 4, an RFID redundant data filter algorithm is proposed that takes into account time and space information simultaneously. In addition, we propose a new formula to measure the quality of filtration. Section 5 describes the experiments, whose results prove that the proposed approach significantly outperforms other existing approaches. Our conclusions and planned future research are presented in Section 6.

2. Background

2.1. System Model. A typical RFID system consists of three parts: an RFID tag, which is attached to the object to be identified and can be read from up to several feet away and does not need to be within the direct line of sight of the reader; a reader, which is responsible for generating interactive information with the tag, such as reading data from the tag or modifying tag information (communications between the reader and tags adhere to the EPC Class 1 Gen 2 RFID standard [2]); and software, such as the middleware that collects and processes the readings from readers and transforms raw data into meaningful information for the application [3].

RFID tags are generally categorized into four RFID frequency bands [4]: low frequency (LF): 125–134 kHz, high frequency (HF): 13.56 MHz, ultrahigh frequency (UHF): 433–956 MHz, and microwave frequency (MF): 2.45 GHz. In this paper, we focus on the UHF tags, as they are widely used in large quantities in many applications, such as supply chain management.

RFID readers obtain data by querying tags and forward the resulting information through the middleware to the back end applications or database servers. The middleware filters, aggregates, and transforms raw data (while also allowing heterogeneous devices to collaborate) and coordinates reader activities. The applications receive these filtered data and then decide to respond to these events and orchestrate corresponding actions such as sending theft alerts, raising alarms regarding harmful operations, or replacing fragile components before they fail [4].

2.2. RFID Data Characteristics. It is vital to understand the characteristics of RFID data, as our work is based on RFID data streams. Unlike traditional data streams, RFID data streams have the following characteristics [5, 6].

2.2.1. Simplicity. Most RFID devices in the UHF frequency band follow international standards [2] and have an embedded system to handle some complicated applications such as access control, networking, and formatting data. The RFID devices are capable of providing the ID of tags, capturing the time and antenna number of the captured tags, and providing the received signal strength (RSS), and signal phase of tags, as well as other tag-related information. All captured information is transmitted to back end with a formatted structure. For example, the RFID data are often formatted as <tag_id, location, time>, where tag_id means the unique code of the tag, location means the IP or number of the RFID reader that captured the tag, and time means the time of the capture. RFID data is better organized and formatted than other radio signals and can be received and stored in a predesigned database or data structures.

2.2.2. Massive Amounts of Data. RFID devices for article surveillance generally capture data in a high frequency for the sake of loss protection. However, frequent capturing will lead to redundant data to some degree. The cost of communications between tags and readers is quite low. And readers continuously read tag information and lead to a large amount of RFID data.

2.2.3. Redundant Data. Although a large amount of data is collected by the reader, most of these data are replicate or redundant. There are three reasons for the redundancy of RFID data: the crossreading and writing areas exist between multiple adjacent readers, so that the same tag is read repeatedly; the same tag stays in the range of a fixed reader for a long time and is read many times, resulting in a large amount of duplicate or invalid data; and multiple tags are attached to the same item to improve detection accuracy [7].

2.2.4. Real Time. When the reader acquires the RFID tag information, the data stream flows into the reader

continuously. The interaction between the tag and the reader occurs at a very short interval, so the data is generated in real time and needs real-time processing.

2.2.5. Relevance. RFID data generally does not exist independently but is interrelated. RFID data has a space-time feature, meaning that the timestamp of the same tag reflects its timing relationship and the recorded position, and state reflects the spatial change process of the tag. The space-time association also reflects the related events of the object as identified by the tag.

Considering the characteristics of RFID data streams, some challenges need to be solved when processing RFID data streams in the field of monitoring. On the one hand, due to the real-time requirements of application software, the traditional method of saving all data to the database is no longer feasible. Therefore, our work focuses on how to filter redundant RFID data in real time to meet the needs of the surveillance sector, as these redundant data often lack practical value and reduce a system's operational efficiency. On the other hand, when filtering redundant data, all the above characteristics should be considered so that we can perform comprehensive filtering, including the RFID signal or the space-time relevance hidden in the data. This paper will focus on how to perform efficient real-time filtering on redundant data.

2.2.6. RFID Signal Characteristics. RSS is an important eigenvalue of RFID signals, which can be obtained by the RFID reader when reading the RFID tags. Figure 1 depicts the test we performed with a continuously increasing distance of 0–6 meters by recording one data per 0.2 meter. Through testing, we found that the RSS value shows a decreasing trend with the increase of the distance, but it is not strictly attenuated. This means that the relationship between the RSS value and the distance is not one-to-one mapping, and there is a certain degree of jitter. When the transmission distance is close, the RSS value decays faster. The farther the transmission distance, the slower the RSS value attenuated.

Due to the severe jitter in received signal strength (RSS), several methods have been proposed to avoid this instability when using RSS to represent distance. Xu et al. in [8] processed the RSS by using a Gaussian filter to filter the abnormal value. Shangguan et al. in [9] focused on the reading rate of the RSS in a short distance. In our research, we found that it was not effective to preprocess RSS values at the expense of real-time performance, although these algorithms above perform well in filtering RSS. We found that the RSS value can indicate a region while tolerating fluctuations in RSS values within the region.

3. Related Work

Many filtering approaches have been proposed to eliminate redundant data in the RFID data stream efficiently [10]. Table 1 summarizes these approaches.

Traditional data redundancy processing technology stores all the data in a data warehouse or database and then returns the query results according to the database query

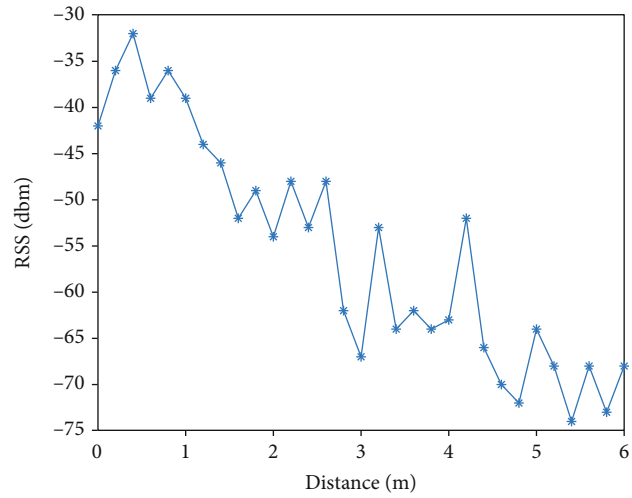

Figure 1: Received signal strength (RSS) value changes according to distance.

statement [11]. RFID Cube is a data warehouse model based on the assumption that RFID tags tend to move and stay together, and therefore, a large amount of redundant data can be filtered, based on the data's locations. However, as the data scale expands, query efficiency will decrease, and the real-time requirements cannot be met. In that circumstance, a window-based filtering method should be used, which does not need to save all the data; it only needs to maintain a much smaller scale than its size [12]. As the new data stream arrives, the data in the window is continuously updated, the expiration data overflows, and the new data is inserted synchronously. In this manner, the RFID stream data can be continuously processed in real time. However, the effectiveness of the sliding window method depends on the selection of attribute fields and the size of the window. Too large of a window size may result in higher time complexity, while a small window size may result in incomplete filtering [13].

In addition, there are many redundant processing algorithms designed for the characteristics of RFID data streams. Jeffery et al. and Le each proposed a framework for pipelined data cleansing, called ESP and PDC, respectively [14, 15]. Both cleaning frames consist of certain cleaning steps with their own cleaning task, such as correcting missed readings or detecting outliers for the sensor streams. These two methods are proposed for cleaning the sensor data stream when the complexity and nature of RFID objects that dynamically move make them unsuitable for RFID data streams.

Paper [16] uses the finite state machine model, which focuses on the moving tags for the cleaning of redundant data. The method divides the real-world area into different states in the state machine model, so that the valid data is defined as the data in which the state is migrated and so that a large amount of redundant data generated by static tags can be filtered. This method is suitable for the monitoring and access control scenarios. However, the method is not suitable for complex environments, because it limits the number of state machines and because different state models need to be developed for different scenarios.

Table 1: Weaknesses of existing approaches for filtering redundant RFID data.

Approach	Weakness
RFID Cube [11]	Needs to wait for all readings to be completed before duplicate readings can be filtered
Sliding window [12]	Subject to a selection of parameters and needs to scan along the sliding windows for every new reading
Extensible receptor Stream Processing (ESP) and pipelined data cleaning (PDC) [14, 15]	Complex filtering framework, delayed process, and not suitable for dynamic tag filtering
Finite state machine [16]	Limited by scenario and lack of generalization ability
Dynamic Bayesian networks (DBN) [17]	Cleaning effect of the dynamic tag is not obvious
Bloom filter [19, 20]	Does not allow deletion and filtering in a single dimension

In the area of machine learning [17, 18], many RFID dataset cleaning methods based on dynamic Bayesian networks (DBNs) have been proposed. Paper [17] introduced a cleaning framework to generate a globally optimal cleaning scheme to minimize cost and also introduced a decision tree-based cleaning method. However, the disadvantage of the DBN cleaning method is that it obtains the relationship between the predicted value and the observed value from the historical data and cannot be dynamically updated.

Due to the dynamic generation of RFID data, it is impractical to store all the data. How to clean these data dynamically with limited time and space is a challenge. Approaches that explore the Bloom filter for filtering redundant readings in RFID have recently emerged. Compared with the other above approaches, the Bloom filter is more sensitive to real-time applications. Generally, when a new reading in the format of <tag_id, location, time> comes at the reader, it will be inserted into the Bloom filter and can be filtered without delay. Bloom filters perform well even in the face of huge RFID data streams. However, existing methods of removing RFID redundancy based on Bloom filters are usually filtered for the classic triplet with the attribute such as time or tag id. Table 2 shows the algorithm of Bloom filter in RFID data stream redundancy filtering methods.

Based on the characteristics of RFID data streams, the TBF modifies the bit array into an integer array. The timestamp of the latest data is saved in the array, which ensures that the tuple in the filter will not be full, so that the error rate will increase sharply by updating the time. The Time Interval Bloom Filter works in a similar manner, while it stores the timestamp of the start and end by using a two-dimensional array.

The Compared Bloom Filter (CBF) considers the problem of data filtering in multireader environment. In the environment of multiple readers existing, the tag data may be collected and stored by multiple readers for many times. The Compared Bloom Filter determines which reader is the reader of the tag data. The principle is that in a time span, when a tag is read by multiple readers, the reader with the least number of reads is redundant.

The proposed Time and Space Bloom Filter algorithm solves the problem of how to judge the mobility of the tag. This filter extends the one-dimensional array in the standard Bloom filter to a two-dimensional array that includes location and time information.

The Approximate Probability Synthesis Bloom Filter extends the number of dimensions to four. This approach handles the situations of location movement and staying at the overlapping areas among multiple readers with accuracy.

Although the aforementioned method considers the characteristics of time dimension and space dimension of multiple readers, there is no mention of filtering redundant data within a certain range. The work presented in this paper takes into account the filtering of redundant data within a certain range and can be applied into the actual scenario.

4. Time-Distance Bloom Filter Algorithm

4.1. Bloom Filters. A Bloom filter is a space-saving probabilistic data structure proposed by Burton Howard Bloom in 1970 to test whether an element is in a collection or not [20]. A Bloom filter consists of a bit array of size m and k independent hash functions $h_1, h_2, \cdots, h_k$. The array of m bit is initialized to 0, and the value range of these hash functions is $\{1, 2, \cdots, m\}$. Suppose that the set $S = \{x_1, x_2, \cdots, x_n\}$ has n elements, each element x_i will be mapped to different positions in the array via each hash function in order and set the corresponding value to 1, denoted as $\forall x_i \in S, BF_{h_j(x_i)} = 1, 1 \le i \le n, 1 \le j \le k$. To test whether an element x_i is a member of the existing set S, it will be run through the same hash functions. The element is said to be a member of the set if all the bits in which the element was mapped to are 1. If there are one or more mapping values of 0, then the element is considered to be new.

The specific operation of the Bloom filter is shown in Figure 2. All bits in a Bloom filter are initially set to 0 and will be replaced by 1 when it is hashed by the element. "TagID1" is a newly arrived element that is mapped to the first, fourth, and sixth bits of the array via the Bloom filter shown in Figure 2(a). When checking whether "TagID2" is duplicated, the first hash function puts the corresponding mapping position at 0 to 1, which means that "TagID2" has not yet been inserted into the bit array.

Compared to other data structures, a Bloom filter has significant advantages in terms of space and time because the storage space and query time in the filter are constants [24, 25]. In addition, the hash functions have no relationship with each other and can be implemented in parallel by hardware. Furthermore, because the Bloom filter does not require the

Table 2: Algorithms based on the Bloom filter.

Algorithm	Characteristics
Time Bloom Filter (TBF)/Time Interval Bloom Filter (TIBF) [21]	Reading timestamp of stored data to determine whether it is redundant data (TIBF stores the timestamp of the start and end)
Compared Bloom Filter (CBF) [7]	Focus on data filtering in the physical space dimension
Time and Space Bloom Filter (TSBF) [22]	Judges the mobility of the tag with consideration of the time-space characteristics
Approximate Probability Synthesis Bloom Filter (PSBF) [23]	Not only filters redundant probability events but also handles positional motion, even in overlapping areas

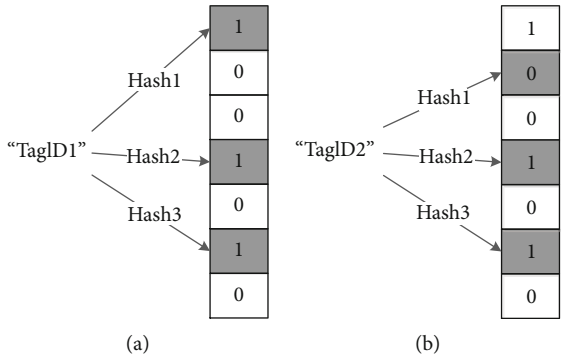

Figure 2: Basic principle of a Bloom filter. (a) Newly arrived data mapped in different positions in the array; (b) duplicate detection via Bloom filter.

storage element itself, it has advantages in certain situations where the privacy requirements are very strict.

4.2. Definitions of Redundant RFID Data. Generally speaking, RFID data redundancy refers to that in a time period, a certain tag is read many times, and these data are sorted by time. Except the first reading data is valid, other data are redundant. But in the scenario of warehouse monitoring, data redundancy has a richer meaning. The RFID-based electrical article surveillance (EAS) system of the warehouse is deployed at the entrance of the warehouse to monitor the items in and out of the warehouse. However, due to the RFID technology itself, the RFID system deployed at the entrance is very likely to read not only the tags of items entering the entrance but also the tags of items close to the entrance in the warehouse. These items near the entrance bring a lot of invalid redundant data.

As mentioned previously, RFID data are always described as a tuple of <tag_id, location, time> which stands for the ID of the captured tag, the ID of the capturing reader, and capturing time, respectively. Furthermore, the Bloom filter has some variants for different purposes. Here, we introduce a new collected data RSS and provide some definitions of RFID data stream redundancy in this algorithm:

Definition 1. Use S to denote a series of RFID data streams, $S = \{e_1, e_2, \cdots, e_n\}$, where e_i denotes an RFID four-tuple <tid, loc, time, RSS>, where tid stands for the unique identifier of the tag, loc is the ID of the reader that captured the data, time is the timestamp of the data acquisition, and RSS is the acquired signal strength value, which can represent the distance of the data.

Definition 2 (temporal redundancy). Set w as the time window in data stream S. If there are two tag data $x \in S, y \in S$ in w, if $x.\text{tid} = y.\text{tid}$, $x.\text{loc} = y.\text{loc}$, $y.\text{time} - x.\text{time} \le \tau\ (\tau > 0)$, where τ is a set time threshold, at this time data y is considered to be temporal redundant data. The tagged object is detected multiple times by a reader within a time window. Therefore, only the earliest arriving data in the time window can be considered as redundant while others are regarded as nonredundant.

Definition 3 (distance redundancy). If $x \in S, y \in S$ exists and the tag data x and y are, respectively, satisfied $x.\text{tid} = y.\text{tid}$, $x.\text{loc} = y.\text{loc}$, $y.\text{time} > x.\text{time}$, and $y.\text{RSS} < \varepsilon$, where ε is the distance threshold of RSS. If the RSS value is smaller than the threshold ε, it indicates that the tag is not in the detection range, so the data y can be considered as meaningless redundant data. Tagged objects scattered in every corner of space, no matter if the tag data is in the time window or not, only if it is detected out of the detection range ε will it be considered as redundant data.

Definition 4. Although we give a simple definition of RFID redundancy, some special cases need further consideration. If $x \in S, y \in S$ exists and the tag data x and y are, respectively, satisfied $x.\text{tid} = y.\text{tid}$, $x.\text{loc} = y.\text{loc}$, $y.\text{time} > x.\text{time}$, $y.\text{RSS} > \varepsilon$, and $y.\text{RSS} > x.\text{RSS}$, we believe that y is not redundant data. In the actual scenario, the value of RSS keeps changing with the movement of the tag. When the tag enters and leaves the detection area, the distance between the tag and the reader is from far, near, and far. In theory, in this process, the RSS value of the tag will change from small to big and big to small. The tag stored in the warehouse for a long time will not change like this. Therefore, this trend can be used to determine whether the label left the warehouse.

4.3. Basic Principle of the Time-Distance Bloom Filter Algorithm. The Bloom filter can be extended to RFID data streams by taking advantage of its high performance and acceptable error rate. Based on time and space considerations, we designed an algorithm called Time-Distance Bloom Filter (TDBF). Based on the TBF, the TDBF algorithm increases the unique distance judgment in RFID data filtering and achieves the function of filtering RFID redundant data at a certain time and space.

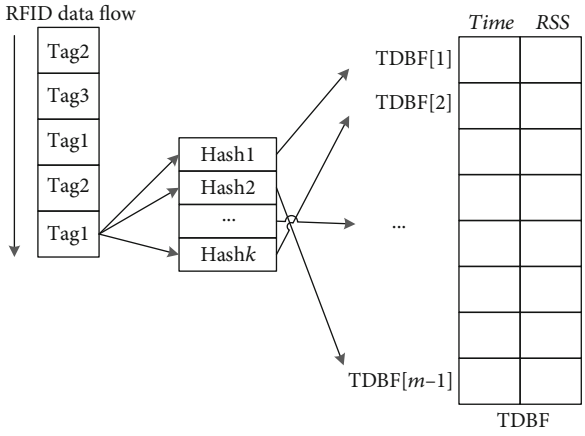

Figure 3: Time-Distance Bloom Filter (TDBF) data structure diagram.

The structure of TDBF is shown in Figure 3. The TDBF array size is initialized to m, and each cell represents a two-dimensional integer array. The first column stores the timestamp information of the read tag, and the second column stores the RSS value of the read tag. The data of the unit can be represented as $M_i[\text{Time}][\text{RSS}]$, and the RSS value is represented as a negative number, so that when the TDBF is initialized, the value of the first column is initialized to 0, and the value of the second column is initialized to the distance threshold ε.

When new RFID data arrives, k independent hash functions are used for filtering the tid value of RFID tags. The timestamp value stored in the mapping and the RSS information unit are used to determine whether the data is redundant data. Once we determine which data are redundant data, we can delete those data and obtain the data we want. The principle of redundancy judgment is as follows:

(1) For the newly arrived RFID data x, k independent hash functions are used for mapping its unique ID into the corresponding array unit

(2) If there is $i \in \{1, 2, \cdots, k\}$, $M_i[\text{Time}] = 0$, $x.\text{RSS} > \varepsilon$, then we can think of this data x as the new arrival data within the detection range. At this time, the time information and signal strength information in k different array units need to be updated, which is $M_i[\text{Time}] = x.\text{Time}$, $M_i[\text{RSS}] = x.\text{RSS}$

(3) If there is $i \in \{1, 2, \cdots, k\}$, $x.\text{Time} - M_i[\text{Time}] > \tau$, $x.\text{RSS} > \varepsilon$, then the newly arrived data is outside the specified time interval and within the distance range, so data x is not redundant data. At this time, the time information and signal strength information in k different array units need to be updated, which is $M_i[\text{Time}] = x.\text{Time}$, $M_i[\text{RSS}] = x.\text{RSS}$

(4) A special case that needs to be considered is that if there is $i \in \{1, 2, \cdots k\}$, $x.RSS > M_i[\text{RSS}] \geq \varepsilon$, $x.\text{Time} > M_i[\text{Time}]$, we can ascertain that the newly arrived data x is within the distance threshold ε and the value of RSS is larger compared to the previous detected value. Therefore, we are given a reason to think that the data has a tendency to approach the reader. In this case, whether the data is within the prescribed time interval or not, we consider data x as paramount and that it needs to be saved for further processing. At this time, the time information and signal strength information in k different array units do not need to be updated, which is $M_i[\text{Time}] = x.\text{Time}$, $M_i[\text{RSS}] = x.\text{RSS}$

(5) Otherwise, the data is considered redundant or invalid data and is directly deleted

Combined with the definitions of redundancy introduced earlier and the basic principles of TDBF, the pseudocode representation of TDBF is shown in Algorithm 1.

4.4. Performance Evaluation Standard. Approaches to evaluate the performance of TDBF are difficult because in the real world it is sometimes hard to say what is useful data. Here, we consider three main factors as evaluation indicators: the data compression ratio (CR), the false positive (FP) rate, and the false negative (FN) rate. The data compression rate is the number of original tags collected per minute, divided by the number of filtered tags, which can reflect the filtering efficiency. The higher the compression ratio, the more the data filtered. However, an excessive data compression ratio leads to loss of valid information. A false positive in TDBF refers to the misidentification of RFID tags that are not redundant data into redundant data, and a false negative in TDBF means that redundant RFID tag data is mistaken for nonredundant data.

For the redundant filtering algorithm, we want to get the highest possible data compression rate with the smallest possible error rate. Therefore, we use the following formula to express filtering performance [26]:

$$Score = \log \frac{CR}{N} * \frac{M - FN - FP * W}{M}. \quad (1)$$

where N denotes the number of RFID tags in the environment; FP and FN indicate the number of detected RFID tags; W denotes the weight of the FP, because we believe that FP should be minimized to avoid the loss of valid information; M is the maximum ideal number of read tags; and log function here is to prevent the data compression ratio from being too high, resulting in loss of effective information. According to the formula, higher scores indicate better algorithm performance. This formula can produce a specific evaluation of the filtering algorithm, which is convenient for evaluating the filtering performance.

4.5. Algorithm Analysis. The uncertainty of an RFID data stream makes it difficult to avoid false positives and false negatives.

(1) The false positive rate of TDBF consists of two parts:

(a) Assume that there is an RFID data stream $S = \{s_1, s_2, s_3, \cdots, s_n\}$, $s_1 = <\text{TagId}_1, \text{Time}_1, \text{RSS}>$,$s_2 = <\text{TagId}_2, \text{Time}_2, \text{RSS}>$,$s_3 = <\text{TagId}_3, \text{Time}_2,$

Input: RFID data x: x.tid, x.Time, x.RSS.
Output: Whether x is redundant data.
1: BEGIN
2: //New data arrives
3: FOR (i = 1; i ≤ k; i ++)
4: p[i] = $Hash_i$(x.tid) //Hash mapping the tag ID
5: FOR (i = 1; i ≤k; i ++)
6: IF $TDBF_{p[i]}$[Time] = 0 and x.RSS > w //If the time value in the mapped array is the initial value of 0 and the RSS value is greater than a threshold within the specified range
7: Update TDBF (x.Time, x.RSS) // update TDBF
8: Send x to the event // send data to the event module
9: break
10: ELSE IF x.Time – $TDBF_{p[i]}$[Time] > τ and x.RSS > w // If the data x in the range and the previously stored data have a time interval of τ
11: Update TDBF(x.Time, x.RSS)
12: Send x to the event
13: break
14: ELSE IF x.Time > $TDBF_{p[i]}$[Time] and x.RSS > $TDBF_{p[i]}$[RSS] // If the newly arrived data has a higher RSS value
15: Update TDBF (x.Time, x.RSS)
16: Send x to the event
17: break
18: ELSE IF
19: END FOR
20: Drop x // this data is redundant data, directly drop
21: END

Algorithm 1: TDBF.

RSS > . Among these, the entire value of RSS is greater than the distance threshold, and the difference value between $Time_2$ and $Time_1$ is greater than the time threshold. For example, two hash functions are used in the case. $TagId_1$ is mapped to the TDBF array of position r and s by each hash function, and the probability of each bit being mapped is $1/m$. Similarly, $TagId_2$ is mapped to the position s and t of the array while $TagId_3$ is mapped to the position r and t. At this time, although $TagId_3$ is different on the tag ID, it also should be judged as redundant data because the time and distance information of the mapping location satisfies the condition for redundancy. At this time, the repetition probability of a certain bit is

$$1 - \left(1 - \frac{1}{m}\right)^{kn}. \tag{2}$$

Since there are k hash functions, there are k bits mapped, and the probability of a false positive is

$$f_1 = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k}. \tag{3}$$

However, the size of the TDBF array can be set to be much larger than the number of tags. In this case, the probability of a false positive can be ignored.

(b) The RSS value itself has a certain volatility. Ideally, the closer the RSS value is to the reader, the larger it is. However, due to the complexity of the real-world environment, there may be a situation where the tag is close to the reader, and the RSS value decreases. The probability of false positives and false negatives at this time is based on the environment

(2) The false negative rate of TDBF also consists of two parts:

(a) Assume that there is an RFID data stream $S = \{s_1, s_2, s_3, \cdots, s_n\}$, $s_1 = <TagId_1, 2, -40>$,$s_2 = <TagId_2, 3, -50>$,$s_3 = <TagId_1, 4, -45>$. Note that the second term in the triplet should be expressed as an integer format of datetime, because in Algorithm 1, it is necessary to calculate the time interval. Here, we use small integers for short. We define the RSS distance threshold as -60, the time threshold is 3, $TagId_1$ is mapped to the position of r and s in the array of size m, and $TagId_2$ is mapped to the position of s and t in the array. According to the TDBF algorithm, the data s_3 has the RSS value -45, which is greater than the RSS value -50 stored in the position of s in the array, so it is determined to be nonredundant data. However, in the actual

scenario, although this tag is within the distance range, the tag is not moving to the reader in the time threshold. Therefore, it should be considered as redundant data. The possibility of this situation is very low, because the TDBF array can be very large, and it is hard for the data to be mapped to the same cell

(b) Due to the complexity of the actual environment and the unreliability of the RSS value, when the tag is far away from the reader, its RSS value may become smaller, so the redundant data is mistaken as nonredundant data. In the actual scenario, the miss rate of the tag should be minimized. If so, it is less likely to lose key information in the data collection

In addition, the set address in the traditional Bloom filter cannot be deleted because the incoming data may share the same address of the Bloom filter. As a result, there is a case where the Bloom filter is full; that is, as with the increase of time, the addresses of Bloom filters are all set to 1, resulting in a significant increase in the rate of false positives. Some improved algorithms, such as CBF [27], are proposed to use the counting method to differentiate the situation where different data share the same address. However, our algorithm uses the time parameter instead. The time is gradually increased, and we in turn update the time parameter in the TDBF. In this way, we effectively solved the challenging situation of a full Bloom filter.

TDBF achieves a balance of time, space efficiency, and accuracy. Spatially, TDBF uses a two-dimensional integer array with a size of m, so the space complexity is $\theta(m)$, which is twice the size of the Time Bloom Filter (TBF). In terms of time, when new data arrives, a comparison can be done only once at least to determine whether redundancy exists and the processing of n data only needs to be compared n times. The best time complexity is $\Omega(n)$, while the worst case is that it needs to be compared k times for each time that new data arrives, where k is the number of hash functions, and n data needs to be compared $k * n$ times. The worst time complexity at this time is $O(n)$, and therefore, the average time complexity is $\theta(n)$.

5. Algorithm Design and Implementation

Our experiment uses the Impinj Revolution series of passive readers and several supporting tags. In order to verify the effectiveness of the algorithm, the experimental environment is an open room without interference, thereby reducing the environmental impact on RSS. The reader and the computer are connected by a network cable. The other components of the experimental environment are shown in Table 3.

5.1. Experiment in Static Scenario. The static experimental scenario deployment is shown in Figure 4. To ensure that there are no obstacles, we chose an empty room to simulate the ideal experimental environment. In the experiment, we used the basic Bloom filter, the Time Bloom Filter (TBF), and the Time-Distance Bloom Filter (TDBF) to filter and analyze the same data stream. The RSS distance threshold shown in Figure 4 is roughly represented by a curve. Since the RSS value has a certain volatility and the RSS value generated by the tag at the same position is slightly different, the RSS distance threshold should be expressed as a distance range. We selected a distance threshold of approximately 3 meters from the reader and set the time threshold to 2 seconds. Then, we attempted to verify the filtering efficiency of the algorithm in a static environment.

Table 3: Components of experimental environment.

Hardware	CPU	Intel Core i5-3320M CPU
	Hard disk	256 GB
	RAM	4 GB
	Network card	Eth0: Intel@82579LM gigabit network connection
Software	Operating system	Windows 7 Ultimate (64 bits)
	Software version	Visual Studio 2017

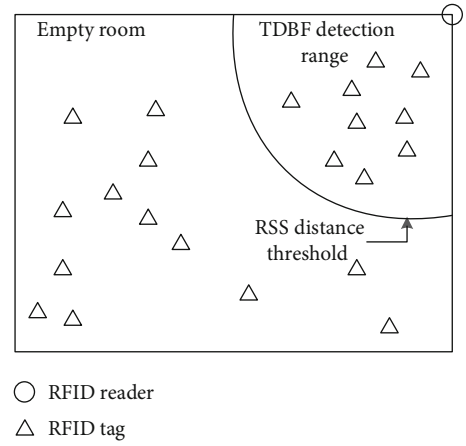

Figure 4: Static scene deployment.

5.1.1. Experiment 1. First, we put 10 tags in the TDBF detection range and counted the total amount of data filtered by the BF, TBF, and TDBF algorithms in 1 minute. To reduce the impact of RSS fluctuations, we placed the tags approximately 1 meter from the threshold to minimize the uncertainty of the RSS. Next, we put all 10 tags out of the distance threshold and performed the same operation. The experimental results are shown in the form of the histogram in Figure 5.

As can be seen in Figure 5, in the same time, the number of tags outside the threshold is slightly lower than the number of tags in the distance threshold, but the total amount of data is basically the same. This shows that our experimental environment is less affected by external interference and can effectively collect most of the data information, but the distance of the tags will affect the data reading rate to some extent. After the Bloom filter, TBF, and TDBF algorithms, the tags are well filtered.

When the tags are all placed within the distance threshold, we observe that the RSS value of the original data is very stable. In theory, the number of valid tags at this time is 300,

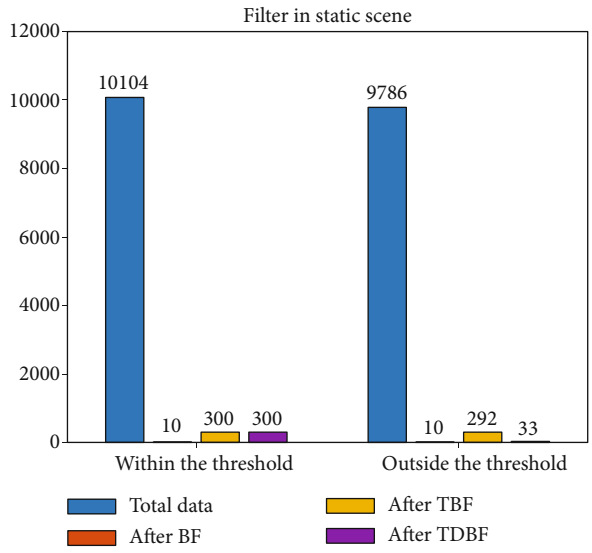

Figure 5: Comparison of different filtering algorithms in static scenes.

which is calculated by multiplying the number of tags by the number of readings in one minute. At this time, both the TDBF algorithm and the TBF algorithm reach the theoretical value, and the same data compression rate is maintained. This shows that when the TDBF algorithm is within the range of the threshold, there is basically no leakage of the tags. Because the Bloom filter algorithm does not take into account the time and space information of the tags, it only performs coarse-grained filtering on the tag ID. Although it has a high data compression rate, it also loses valuable information. Table 4 reflects the evaluation scores of different filtering algorithms, where W is 10 and M is 300. The Bloom filter has the lowest score and poor filtering performance. TBF and TDBF have the same score within the distance threshold, indicating that the filtering performance is similar.

When all the tags are outside the distance threshold, the number of theoretical valid tags is 0 (because all the tag data outside the threshold can be regarded as redundant data), and because the TBF algorithm only considers the time characteristics of the RFID data stream, it generated 292 false negatives. The reason why TBF did not reach the theoretical value of 300 is that the reading distance is too far, and some data cannot be obtained within 2 seconds. After TDBF filtering, the number of false negatives was reduced to 33. From Table 5, we can see that TDBF has the highest score and the best performance because it has a low false negative rate and high compression ratio, while the Bloom filter has a high compression ratio and high false negative rate. TBF only works well within the threshold and has a high false negative rate and low compression ratio compared with TDBF.

The experiment proves that the TDBF algorithm and the TBF algorithm maintain the same zero false positive rate when the tags are within the distance threshold. However, when the tags are outside the distance threshold, the false negative rate of the TDBF algorithm is much smaller than the false negative rate of the TBF algorithm. Therefore, the TDBF algorithm can achieve a 0 false positive rate, a low false negative rate, and a high data compression ratio in a static ideal environment.

5.2. Experiment in Dynamic Scenario. The dynamic experimental scenario deployment is shown in Figure 6. In the experiment, the operator carries a number of tags, from the position outside the threshold shown in the figure, slowly getting closer to the reader at the gate. We use the basic Bloom filter (BF), the TBF, and the TDBF to filter and analyze the same data stream. We selected the distance threshold at approximately 2 meters from the reader and set the time threshold to 1 second. We then attempted to verify the filtering efficiency of the algorithm in a dynamic environment.

5.2.1. Experiment 2. We first placed all five tags outside the distance threshold, and then, the experimenter carried these tags along the motion trajectory shown in Figure 6, slowly approaching the gate. The tag data filtered by TBF and TDBF is calculated separately to see if TDBF can filter out valid data.

As shown in Figure 7, the tag carrier is slowly moving closer to the RFID reader along the trajectory. Slow movement can make the tags approximately static, which can simulate the efficiency of redundant data filtering in an ideal environment. At this time, the TBF does not consider the spatial characteristics, and the filtered data volume has a linear growth trend. The deviation of the data from the expected value is often due to the missing or misreading caused by the tags being unstable when the experimenter moves. When the tag carrier is far away from the threshold, TDBF will produce very little false negative data. As the tag carrier approaches the distance threshold, the RSS value gradually increases and has some fluctuation, resulting in a low false negative rate. About 16 seconds after the tag carrier enters the distance threshold, the RSS fluctuation causes the TDBF to have a certain degree of false positive rate. But after 20 seconds, once the tag carrier has completely entered the distance threshold range, the TDBF maintains a tag filtering rate similar to that of the TBF.

The experiment shows that there is a range of RSS fluctuations in the vicinity of the distance threshold. In this range, the TDBF algorithm will produce a certain degree of false negative rate and false positive rate due to the instability of the RSS value. TDBF can produce a good filtering effect, and within the distance threshold, it can produce a filtering effect similar to that of the TBF algorithm. The spatial granularity of the RFID redundant data is further filtered by using the distance information.

5.2.2. Experiment 3. Typically, the data that we are most concerned about is the data that is close to the reader. Such data tends to reflect the tendency of the tag carrier to carry the tags away from the area. In Experiment 2, the movement of the tag is in a straight line. Therefore, it is not possible to simulate the motion trajectory of a person in a real-world environment. Therefore, in Experiment 3, we express the trajectory of a character in a curve. The specific trajectory is shown in Figure 8.

Table 4: Filter score within the distance threshold.

	Compression rate (CR)	False positive (FP)	False negative (FN)	Score
Bloom filter (BF)	1010.4	9804	0	-1503.6
Time Bloom Filter (TBF)	33.7	0	0	1.2
Time-Distance Bloom Filter (TDBF)	33.7	0	0	1.2

Table 5: Filter score outside the distance threshold.

	Compression rate (CR)	False positive (FP)	False negative (FN)	Score
Bloom filter (BF)	978.6	0	9786	-144.9
Time Bloom Filter (TBF)	33.5	0	292	0.03
Time-Distance Bloom Filter (TDBF)	296.5	0	33	5.2

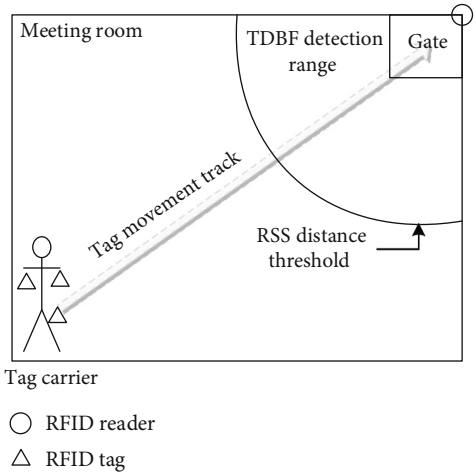

Figure 6: Dynamic scenario deployment.

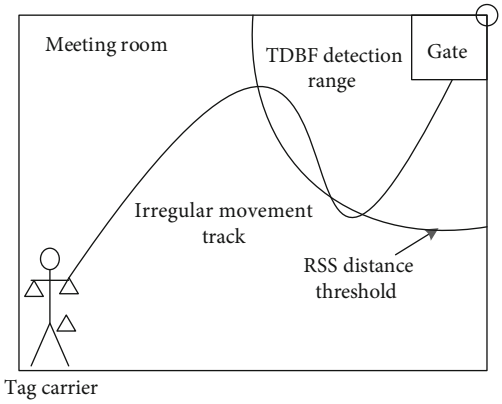

Figure 8: Irregular movement track.

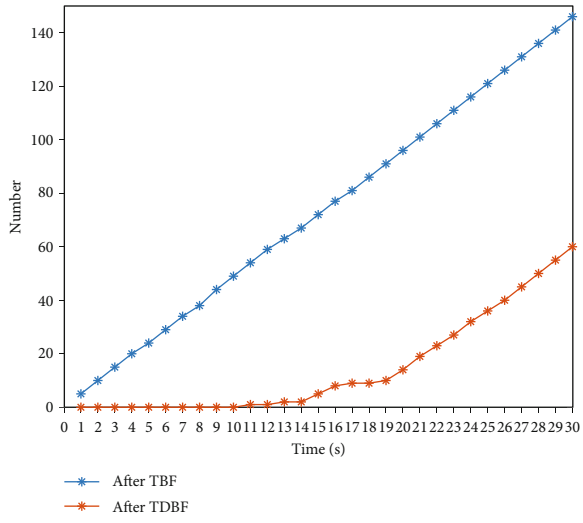

Figure 7: Filtering tags in dynamic scene with TBF and TDBF.

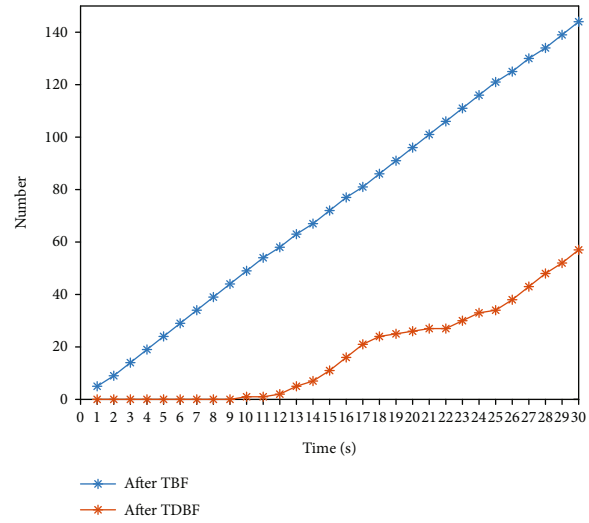

Figure 9: Filter irregularly moving tags with TBF and TDBF.

The experimental results are shown in Figure 9. It can be observed that the data filtered by the TBF maintains a linear growth trend because it is not bound by the spatial conditions. After TDBF filtering, the amount of data after filtering in the first 10 seconds is almost 0, indicating that the tag is still far from the distance threshold, and can therefore be

directly filtered. After 12–17 seconds, it can be observed that the filtered data is in a growing trend, indicating that the tag is near the distance threshold and is approaching the reader. After 18 seconds to 23 seconds, the filtered data is in a relatively flat state, and there is no new data filtering. At this time, the experimenter may be in a state away from the reader. After 24 seconds to 30 seconds, the data showed a linear growth trend again, indicating that the tag was suspected of being carried out of the meeting place. The upper node can generate alarm information according to the data filtered by the TDBF, thereby alerting the administrator of the possibility of a suspicious person who wants to carry the tags outside the monitoring area.

In summary, the TDBF algorithm can maintain a low false negative rate, a low false positive rate, and a high data compression rate. In a dynamic scenario, the TDBF algorithm can filter out valid data according to actual scenario requirements. Therefore, the algorithm can be practically applied to access monitoring.

6. Conclusions

An RFID data stream contains a large amount of redundant data, which often has no practical value and reduces the operating efficiency of the system. It is difficult for traditional redundant processing algorithms to filter the data in time and space accurately. In order to balance the real-time processing efficiency of data, this paper uses the Bloom filter algorithm to propose an algorithm called Time-Distance Bloom Filter (TDBF), which takes into account the reading time and reading distance of RFID tags. Both time granularity redundancy and spatial granularity redundancy are considered, which greatly reduces the redundancy of RFID data and improves the data compression rate. Experiments show that compared with the TBF algorithm, the TDBF algorithm greatly reduces the false negative rate of the data and ensures a lower false positive rate, which improves the performance. In addition, in dynamic scenarios, we find that the algorithm can filter out valid tag information, which can be applied to redundant filtering in the monitoring scenario.

However, the TDBF algorithm still has several limitations. First of all, the algorithm is only applicable to a single reader; it is not suited for large-scale application scenarios. The distributed reader architecture scheme is one of the feasible ideas for future work. Second, there is some uncertainty in the RSS value used in this paper, and it is greatly affected by the environment and tag quality. How to use the algorithm to further reduce the error impact caused by the fluctuation of RSS value or to use the new tuple model instead of the RSS value to estimate the distance will be the next step in our research [28].

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Conflicts of Interest

The authors declare no conflict of interest.

Authors' Contributions

Siye Wang and Ziwen Cao conceived the presented idea, developed the theory, and performed the computations. Ziwen Cao and Yanfang Zhang conducted the experiment to verify the idea. They made the proposed differential position algorithm to the software program and collected the experiment data. Weiqing Huang and Jianguo Jiang helped supervise the project. Siye Wang and Ziwen Cao wrote the original draft in consultation with Weiqing Huang and Jianguo Jiang. All authors provided critical feedback and helped shape the research, analysis, and manuscript.

Acknowledgments

This research was funded by the National Natural Science Foundation of China grant No. 61501458 and Youth Innovation Promotion Association of Chinese Academy of Sciences grant No. 1104CX0204.

References

[1] R. Derakhshan, M. Orlowska, and X. Li, "RFID data management: challenges and opportunities," in *2007 IEEE International Conference on RFID*, pp. 175–182, Grapevine, TX, USA, 2007.

[2] *EPC Class 1 Gen 2 RFID standards*, https://www.gs1.org/sites/default/files/docs/epc/uhfc1g2_2_0_0_standard_20131101.pdf.

[3] M. A. El Khaddar, M. Boulmalf, H. Harroud, and M. Elkoutbi, "RFID middleware design and architecture," in *Designing and Deploying RFID Applications*, pp. 305–326, IntechOpen, 2011.

[4] K. Domdouzis, B. Kumar, and C. Anumba, "Radio-frequency identification (RFID) applications: a brief introduction," *Advanced Engineering Informatics*, vol. 21, no. 4, pp. 350–355, 2007.

[5] W. Tian, R. Xue, X. Dong, and H. Wang, "An approach to design and implement RFID middleware system over cloud computing," *International Journal of Distributed Sensor Networks*, vol. 9, no. 10, Article ID 980962, 2013.

[6] F. Vahdati, R. Javidan, and A. Farrahi, "A new method for data redundancy reduction in RFID middleware," in *2010 5th International Symposium on Telecommunications*, pp. 175–180, Tehran, Iran, 2010.

[7] H. Mahdin and J. Abawajy, "An approach for removing redundant data from RFID data streams," *Sensors*, vol. 11, no. 10, pp. 9863–9877, 2011.

[8] H. Xu, Y. Ding, P. Li, R. Wang, and Y. Li, "An RFID indoor positioning algorithm based on Bayesian probability and K-nearest neighbor," *Sensors*, vol. 17, no. 8, p. 1806, 2017.

[9] L. Shangguan, Z. Li, Z. Yang, M. Li, and Y. Liu, "OTrack: order tracking for luggage in mobile RFID systems," in *2013 Proceedings IEEE INFOCOM*, pp. 3066–3074, Turin, Italy, 2013.

[10] S. P. G. Jasil and V. Ulagamuthalvi, "A survey on duplicate data filtering methods in big data," *International Journal of Engineering & Technology*, vol. 7, no. 3.1, pp. 90–92, 2018.

[11] H. Gonzalez, J. Han, X. Li, and D. Klabjan, "Warehousing and analyzing massive RFID data sets," in *22nd International Conference on Data Engineering (ICDE'06)*, Atlanta, GA, USA, 2006.

[12] Y. Bai, F. Wang, and P. Liu, "Efficiently filtering RFID data streams," in *Proceedings Cleandb Workshop*, pp. 50–57, 2006.

[13] H. Xu, W. Shen, P. Li, D. Sgandurra, and R. Wang, "VSMURF: a novel sliding window cleaning algorithm for RFID networks," *Journal of Sensors*, vol. 2017, Article ID 3186482, 11 pages, 2017.

[14] S. R. Jeffery, G. Alonso, M. J. Franklin, W. Hong, and J. Widom, "A pipelined framework for online cleaning of sensor data streams," in *22nd International Conference on Data Engineering (ICDE'06)*, pp. 140–153, Atlanta, GA, USA, 2006.

[15] Z. Le, *Research and development of RFID middleware data processing [Ph.D. thesis]*, Shanghai Jiao Tong University, 2008.

[16] Y. Luo, J. Jiang, S. Wang et al., "Filtering and cleaning for RFID streaming data technology based on finite state machine," *Journal of Software*, vol. 25, no. 8, pp. 1713–1728, 2014.

[17] G. Hector, J. Han, and X. Shen, "Cost-conscious cleaning of massive RFID data sets," in *2007 IEEE 23rd International Conference on Data Engineering*, pp. 1268–1272, Istanbul, Turkey, 2007.

[18] I. F. Ilyas and X. Chu, *Machine learning and probabilistic data cleaning*, Data Cleaning, 2019.

[19] H. Mahdin, "A review on bloom filter based approaches for RFID data cleaning," in *Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013)*, vol. 285 of *Lecture Notes in Electrical Engineering*, pp. 79–86, Singapore, 2014Springer.

[20] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[21] C. H. Lee and C. W. Chung, "An approximate duplicate elimination in RFID data streams," *Data Knowledge Engineering*, vol. 70, no. 12, pp. 1070–1087, 2011.

[22] W. Rui, L. Guoqiong, and D. Guoqiang, "Filtering redundant RFID data based on sliding windows," in *2014 International Conference on Management of e-Commerce and e-Government*, pp. 187–191, Shanghai, China, 2014.

[23] L. Guoqiong, Z. Jun, H. Ni et al., "Approximately filtering redundant data for uncertain RFID data streams," in *2017 18th IEEE International Conference on Mobile Data Management (MDM)*, pp. 56–61, Daejeon, South Korea, 2017.

[24] L. Luo, D. Guo, R. T. B. Ma, O. Rottenstreich, and X. Luo, "Optimizing Bloom filter: challenges, solutions, and comparisons," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1912–1949, 2018.

[25] G. Einziger and R. Friedman, "TinySet—an access efficient self adjusting Bloom filter construction," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2295–2307, 2017.

[26] B. Lv, H. Xu, J. Wu et al., "LiDAR-enhanced connected infrastructures sensing and broadcasting high-resolution traffic information serving smart cities," *IEEE Access*, vol. 7, pp. 79895–79907, 2019.

[27] P. Pandey, M. A. Bender, R. Johnson, and R. Patro, "A general-purpose counting filter: making every bit count," in *Proceedings of the 2017 ACM International Conference on Management of Data - SIGMOD '17*, pp. 775–787, Chicago, Illinois, 2017.

[28] H. Hadj, M. R. Touhami, and S. Tedjini, "Cleansing RFID data based on RSSI estimation," in *2017 IEEE 17th International Conference on Ubiquitous Wireless Broadband (ICUWB)*, pp. 1–4, Salamanca, Spain, 2017.