Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 4909476, 20 pages
https://doi.org/10.1155/2022/4909476

Research Article

Online Missing Data Imputation Using Virtual Temporal Neighbor in Wireless Sensor Networks

Yulong Deng,[1,2] **Chong Han**,[1,2] **Jian Guo**,[1,2] **Linguo Li**,[3] **and Lijuan Sun**[1,2]

[1]*College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China*
[2]*Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks, Nanjing University of Posts and Telecommunications, Nanjing 210003, China*
[3]*College of Information Engineering, Fuyang Normal University, Fuyang 236041, China*

Correspondence should be addressed to Lijuan Sun; sunlj@njupt.edu.cn

Received 18 December 2021; Accepted 11 January 2022; Published 8 February 2022

Academic Editor: Xingsi Xue

Copyright © 2022 Yulong Deng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A wireless sensor network (WSN) is one of the most typical applications of the Internet of Things (IoT). Missing values exist in the sensor data streams unavoidably because of the way WSNs work and the environments they are deployed in. In most cases, imputing missing values is the universally adopted approach before making further data processing. There are different ways to implement it, among which the exploitation of correlation information hidden in the sensor data interests many researchers, and lots of results have emerged. Researching in the same way, in this paper, we propose VTN imputation, an online missing data imputation algorithm based on virtual temporal neighbors. Firstly, the virtual temporal neighbor (VTN) in the sensor data stream is defined, and the calculation method is given. Next, the VTN imputation algorithm, which applies VTN to make estimates for missing values by regression is presented. Finally, we make experiments to evaluate the performance of imputing accuracy and computation time for our algorithm on three different real sensor datasets. The experiment results show that the VTN imputation algorithm benefited from the fuller exploitation of the correlation in sensor data and obtained better accuracy of imputation and acceptable processing time in the real applications of WSNs.

1. Introduction

With the development of the Internet of things (IoT) [1], nowadays more devices and sensors are deployed in the physical environments. Extracting hidden values from the data of IoT becomes challengeable and valuable tasks which make processing ranged from simple tasks such as queries, predictions and classifications to complex processing such as ontology matching [2, 3] and knowledge discovery [4]. As one of the most typical IoT applications, wireless sensor networks (WSNs) [5] are equipped with more nodes with different sensors and are deployed in wider areas with complicated situations. It enables us to obtain a huge amount of physical data in the environment, and these data become the basis for WSN applications. Most of the applications demand complete datasets, i.e., there do not exist missing values in the data obtained from the WSNs because the missing values degrade the performance of the processing algorithms and even make them inapplicable. For example, in an application that is applied to recognize human activities based on the measurement values obtained from the sensors, such as accelerometer and gyroscopes, where the random forest classifier and support vector machine (SVM) are used for classification, the research work shows that 5% missing rate of values in the dataset makes the performance of HASC recognition decrease to 83% and 84%, respectively, 20% missing rate makes them drop down to 45% and 46%, which is unacceptable for the application [6]. We expect to get complete datasets from WSNs, however, it is a fact that missing values exist commonly and widely in real situations. The way WSNs work and the environment they are deployed in make the data to get lost unavoidably, for example. Factors including the signal strength fading and interferences from the environment bring about 9% to 17%

packet loss over a wireless communication channel approximately [7], which causes some of the measurement values missing in the sensor stream. Previous research showed that 45% of the datasets in the UCI machine learning repository contained missing values [8]. For example, in one of the sensor datasets of this repository, the air quality dataset [9], which is used in our experiments, had approximately 14% of its measurement values missing.

Therefore, it is very important to deal with the missing values in a reasonable way, which can benefit the applications running on the incomplete data of WSNs [10]. Before we go further, there are several points we need to consider.

Firstly, the presentation form of the sensor data decides the way to process it. In the common architecture of WSNs, the way to collect data from the sensor nodes is centralized by the sink node, i.e., the measurement values acquired by the sensors on the nodes are directly forwarded to the sink node using hops or are transferred by the cluster heads till they get to the sink node [11]. Hence, on the sink node, which is usually a base station or a data center, if equipped with more powerful processing resources, the data sent from all nodes is collected and streamed in the form of a time series. In this paper, we focus on this presentation form of data, i.e., sensor data stream on the base station or the data center, and deal with the missing value in it using the online imputation algorithm. Different from the offline imputation that makes processing on the static longtime stored data, online imputation is triggered by the missing value once it occurs in the dynamic data stream.

Secondly, missingness mechanisms decide the design of the imputation algorithms. As a feature to describe the relationship between the missing variables and the underlying values of the variables in the dataset, it comprises of three types of mechanisms: missing completely at random (MCAR), missing at random (MAR), and not missing at random (NMAR) [12]. Among them, MCAR describes the case when the probability of a missing value occurrence for an observed attribute is independent of either the known values or the missing ones, i.e., in WSNs, whether the measurement value is missing does not depend on the values acquired by the sensors, including the missing ones. In most cases, it is a typical situation for the missing values in WSNs caused by communication errors or the faults of sensors. Therefore, most of the research works are based on MCAR so far [13]. In this paper, we carry out our work in the same way. MCAR is applied to generate sensor data streams with simulated missing values that are used to test imputation algorithms.

Thirdly, basic ideas to process missing values direct the way to tackle the incomplete data. There are three basic approaches to cope with missing values: deleting the missing values, continuing data processing with missing values, and imputing missing values to get complete data. The first method is seldom used because it makes the size of the dataset reduced, which brings more information loss. The second method is to make further operations directly on the incomplete dataset. A lattice-based direct mining method is proposed in [14]. It works out the subsets of rules to make classification on the binary dataset with the existence of missing values. Targeting at classification for incomplete data, a selective neural network ensemble (SNNE) classifying method applies the chosen feature subsets to train neural networks for making classification [15]. Although these methods exhibit good performance in the research work, they are all limited to specified applications, i.e., classifications, and require complex calculations that are not suitable for resource-constrained applications in WSNs. Therefore, the third method, i.e., to make an imputation for the missing values before further processing, is the most effective way to deal with the incomplete data in WSNs. Focusing on the imputing methods to the sensor data stream, lots of algorithms are proposed by researchers. Closed item sets-based association rule mining (CARM) [16] extracts the most recent patterns between the sensor nodes and applies them to impute the missing number of vehicles in a traffic sensor data stream. Besides the association rule mining, more machine learning methods are applied in the imputation in WSNs. Multidirectional recurrent neural network (M-RNN) based on deep learning is applied to make an estimation for the missing values in medical data streams [17]. A deep imputation network (Deep IN) utilizes deep learning to find a continuous missing pattern that contributes to imputation for sensor data streams in a smart space [18]. These methods exploit machine learning to improve the accuracy of imputation but increase the computational complexity as well. Compared with them, statistical technique-based methods are simpler for calculation and more explainable when they are applied in imputation. Mean imputation [19] is the simplest method, however, it presents the worst performance in most cases because it ignores the relationship between the adjacent values, whereas another simple method, i.e., the linear interpolation model (LIN) [20], applies the temporal relationship to interpolate the missing values and gets improved in accuracy. Lots of data analysis results prove that there exist correlations between measurement values in sensor data streams. Data estimation using a statistical model (DESM) [21] utilizes both spatial and temporal correlations between the sensor nodes and applies previous values to make estimates for the missing values. It is weighed by the Pearson correlation coefficient. In [22], the tensor-based description of the multiple attribute sensor data is used to exploit the correlation between the attributes to make imputation with higher accuracy. Moreover, regression tools are widely used by many researchers to design imputation algorithms. In the spatial correlation-based adaptive missing data estimation algorithm (AMR) [23], spatial correlations between the nodes in WSNs are exploited and adjustable regression models are applied to calculate the estimations for the missing values. For improving the accuracy of imputation, in a new estimation model based on a spatial-temporal correlation analysis (STCAM) [24], four subalgorithms based on regression are combined to deal with the missing values in the sensor datasets. Temporal and spatial nearest neighbor value-based missing data imputation (TSNN) [25] is proposed to make imputation in WSNs by the combination of four spatial and temporal nearest neighbor values, which can exploit the correlation information more effectively.

In this paper, we focus on the way that combines the correlation information with regression tools because it can obtain high imputation accuracy with relatively simple calculations. The aforementioned research works utilize correlations in different ways, however, there still exists room for improvement. Besides, when we deal with missing values in the sensor data stream with limited information, in other words, when the measurement values from spatial neighbor nodes are unavailable or partly available, some of these previous methods present reduced performance, which requires further research work to design a more suitable imputation algorithm. In addition, the computational time is seldom discussed in the previous research work, however, it is very important, especially for online imputation. It is analyzed in this paper based on experiment results.

The rest of this paper is organized as follows: firstly, in Section 2, we make a review of representative imputation algorithms in WSNs and summarize the main contributions of our work. Then, in Section 3, we give the definition of VTN, and based on it, we propose our VTN imputation algorithm. Next, the experiments for evaluating the VTN imputation algorithm and the results are elaborated in Section 4. Finally, in Section 5 and 6, we make a discussion and conclude our research work.

2. Related Work

In this section, we present the typical algorithms in previous research works. Firstly, for discussing the related work conveniently, we propose the description of the sensor data stream in WSNs. Then, the representative algorithms can be redescribed briefly by math expressions. Finally, based on the discussion of the deficiencies of previous works, we present our contributions in this paper.

The imputation for missing values in the sensor data stream is one of the attractive research areas in WSNs, and many researchers have proposed a variety of imputing algorithms in recent decades. They make the imputation for missing values from the perspective of time, space, or both dimensions. Before introducing our VTN algorithms, several representative algorithms are summarized as follows, and they are also used as comparative algorithms in this paper.

The sensor data stream in WSNs is the common object that all imputation algorithms deal with for redescribing the implementation of these typical algorithms, and to present our algorithm later easily, we give the subsequent description.

In general, the nodes in a wireless sensor network are working continuously. Let N be a set of m sensor nodes, $N = \{n_1, n_2, \ldots, n_{m-1}, n_m\}$, $n_i \in N$, and let T be a time point series T, $T = \{t_1, t_2, \ldots, t_{m-1}, t_m, \ldots\}$, $t_j \in T$. For each node n_i, $G(i, j)$ represents its spatial nearest neighbors at the time point t_j, and the number of the spatial nearest neighbors is k_s. $U(i, j)$ represents its temporal nearest neighbors at the time point t_j, and the number of the temporal nearest neighbors is k_t. Let S be a set of p sensors equipped on each of the node, $S = \{s_1, s_2, \ldots, s_{p-1}, s_p\}$, $s_k \in S$, and let V be a set of measurement values acquired by p sensors, respectively, on a node, $V = \{v_1, v_2, \ldots, v_{p-1}, v_p\}$, $v_k \in V$, and $V(n_i, t_j)$ is the vector of the sensor values on the node n_i at time point t_j.

The measurement data obtained by the nodes can be transferred to the base station or data center, where it is collected in the form of sensor data stream, denoted by $S\ DS$, which can be described as follows:

$$SDS = \begin{bmatrix} V(n_1, t_i) & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & V(n_m, t_i) & \cdots \end{bmatrix}. \quad (1)$$

The measurement value on the sensor s_k of node n_i at the time point t_j can be denoted by $v(n_i, t_j, s_k)$, and if there is a missing value on the sensor s_k of node n_i at the time point t_{miss}, the imputation algorithm calculates an estimated value for it, denoted by $v(n_i, \widehat{t_{\text{miss}}}, s_k)$.

For example, suppose there are only two sensors s_t and s_h on each node for acquiring temperature and humidity values, respectively. In this case, $S = \{s_t, s_h\}$, and hence, the measurement values are $v(n_i, t_j, s_t)$ and $v(n_i, t_j, s_h)$, and the sensor data stream can be reduced as follows:

$$SDS = \begin{bmatrix} (v(n_1, t_1, s_t), v(n_1, t_1, s_h)) & \cdots & (v(n_1, t_m, s_t),\ v(n_1, t_m, s_h)) & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ (v(n_m, t_1, s_t), v(n_m, t_1, s_h)) & \cdots & (v(n_m, t_m, s_t),\ v(n_m, t_m, s_h)) & \cdots \end{bmatrix}. \quad (2)$$

We assume that there is a missing value on the sensor s_k of the node n_i at the time point t_{miss}, and it can be imputed by following algorithms using $v(n_i, \widehat{t_{\text{miss}}}, s_k)$.

2.1. Linear Interpolation Model (LIN) Algorithm. The linear interpolation is utilized to calculate the estimated value $v(n_i, \widehat{t_{\text{miss}}}, s_k)$. It applies two previous values that have been

read in from the data stream at the time points t_a and t_b, which are the nearest to the time point t_{miss}.

$$v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right) = v\left(n_i, t_b, s_k\right) + \frac{\left(v\left(n_i, t_a, s_k\right) - v\left(n_i, t_b, s_k\right)\right)\left(t_{\text{miss}} - t_b\right)}{t_a - t_b}, \tag{3}$$

where $t_b < t_a < t_{\text{miss}}$.

2.2. Mean of Temporal Neighbors (MEAN) Algorithm. MEAN is a simple imputation algorithm, in which the estimated value for imputation is the average of the p values of temporal neighbors at the nearest time points before the occurrence of the missing value.

$$v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right) = \frac{1}{|U|} \sum_{U} v\left(n_i, t_{\text{non-miss}}, s_k\right), \tag{4}$$

where $t_{\text{non-miss}} \in U$, $t_{\text{non-miss}} < t_{\text{miss}}$, $|U| = p$.

2.3. Temporal and Spatial Nearest Neighbor Value-Based Missing Data Imputation (TSNN) Algorithm. Assume that there exist spatial nearest neighbors $G(i, j)$ and temporal nearest neighbors $U(i, j)$ for the missing value of sensor s_k on node n_i at the time point t_{miss}. It has four different nearest neighbor values: $N_{sgnn}(s_i, t_j)$, $N_{s\,dn\,n}(s_i, t_j)$, which come from spatial nearest neighbors in the geometrical distance and in data distance, $N_{ttnn}(s_i, t_j)$ and $N_{t\,dn\,n}(s_i, t_j)$, which come from the temporal nearest neighbors in time distance and in data distance. Then, the correlations among the measurement value of the node n_i and its four nearest neighbors can be applied to calculate the estimated value for imputation by regression.

$$v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right) = \lambda v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right)(S) + (1 - \lambda) v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right)(T), \tag{5}$$

where,

$$\begin{cases} v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right)(S) = \dfrac{R^2_{s_k}(1)}{R^2_{s_k}(1) + R^2_{s_k}(2)} v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right)(1) + \dfrac{R^2_{s_k}(2)}{R^2_{s_k}(1) + R^2_{s_k}(2)} v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right)(2), \\ v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right)(T) = \dfrac{R^2_{s_k}(3)}{R^2_{s_k}(3) + R^2_{s_k}(4)} v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right)(3) + \dfrac{R^2_{s_k}(4)}{R^2_{s_k}(3) + R^2_{s_k}(4)} v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right)(4), \end{cases} \tag{6}$$

$$\begin{cases} v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right)(1) = \alpha_{s_k}(1) + \beta_{s_k}(1) N_{sgnn}\left(n_i, t_{\text{miss}}\right), \\ v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right)(2) = \alpha_{s_k}(2) + \beta_{s_k}(2) N_{sdnn}\left(n_i, t_{\text{miss}}\right), \\ v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right)(3) = \alpha_{s_k}(3) + \beta_{s_k}(3) N_{ttnn}\left(n_i, t_{\text{miss}}\right), \\ v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right)(4) = \alpha_{s_k}(4) + \beta_{s_k}(4) N_{tdnn}\left(n_i, t_{\text{miss}}\right), \end{cases} \tag{7}$$

where $\alpha_{s_k}(1)$ to $\alpha_{s_k}(4)$, $\beta_{s_k}(1)$ to $\beta_{s_k}(4)$ are regression coefficients. $R^2_{s_k}(1)$ to $R^2_{s_k}(4)$ are the measures of the fit of the regression models. λ is the spatial-temporal coefficient, which can be calculated based on the contribution ratio of the spatial-temporal neighbors.

Particularly, when the spatial neighbors and the values of the other sensors on the same node are not available, TSNN algorithm degrades to,

$$\begin{aligned} v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right) = v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right)(3) = \alpha_{s_k}(3) \\ + \beta_{s_k}(3) N_{ttnn}\left(n_i, t_{\text{miss}}\right). \end{aligned} \tag{8}$$

2.4. Data Estimation Using Statistical Model (DESM) Algorithm. The estimated value $v(n_i, \widehat{t_{\text{miss}}}, s_k)$ can be calculated based on the temporal nearest neighbors and spatial nearest neighbors of the missing value of sensor s_t on node n_i at time point t_{miss}. A correlation coefficient is obtained from the node n_i, and its spatial neighbors inside the sensor streams can be used as the weight coefficient.

$$\begin{aligned} v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right) = (1 - \alpha) v\left(n_i, t_a, s_k\right) + \alpha v\left(n_i, t_a, s_k\right) \\ \cdot \left(1 + \frac{v\left(g\left(i, t_{\text{miss}}\right), t_{\text{miss}}, s_k\right) - v\left(g\left(i, t_{\text{miss}}\right), t_a, s_k\right)}{v\left(g\left(i, t_{\text{miss}}\right), t_a, s_k\right)}\right), \end{aligned} \tag{9}$$

where $t_a < t_{\text{miss}}$, $g(i,j) \in G(i,j)$, $g(i) = g(i,j)_{j=1}^{j=a}$, and the weight parameter α can be computed as follows:

$$\alpha = \frac{\text{Cov}\left(RV\left(n_i, s_k\right), RV\left(g\left(i\right), s_k\right)\right)}{\sigma RV\left(n_i, s_k\right) RV\left(g\left(i\right), s_k\right)}, \tag{10}$$

$$\begin{cases} RV\left(n_i, s_k\right) = \left\{v\left(n_i, t_1, s_k\right), v\left(n_i, t_2, s_k\right), \ldots, v\left(n_i, t_{a-1}, s_k\right), v\left(n_i, t_a, s_k\right)\right\}, \\ RV\left(g\left(i\right), s_k\right) = \left\{v\left(g\left(i\right), t_1, s_k\right), v\left(g\left(i\right), t_2, s_k\right), \ldots, v\left(g\left(i\right), t_{a-1}, s_k\right), v\left(g\left(i\right), t_a, s_k\right)\right\}, \end{cases} \tag{11}$$

Particularly, when the spatial neighbors are not available, the estimated value alone can be calculated based on temporal neighbors as follows:

$$v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right) = v\left(n_i, t_a, s_k\right). \tag{12}$$

The above algorithms can be applied to make an imputation for the missing values in sensor data streams in WSNs, and their effectiveness is verified in experiments or real situations. However, there are still some deficiencies that should be corrected in further research work.

The temporal neighbors, spatial neighbors, or both are utilized to calculate the estimated value in some ways for imputation. As we present above, the sensor data stream (SDS) is a data stream combined with spatial and temporal data. If they are all available to make an imputation, the performance of the estimation algorithm can be improved because of more information extracted from more nonmissing values of the spatial and temporal neighbors. However, in WSNs, affected by the following problems occurring in the network, the spatial data may be unavailable or be difficult to be applied by the imputation algorithm. One situation is that there are also existing missing values in the spatial neighbors, which makes it impossible to obtain data from them at some time points of SDS. Another common situation is that the neighbors can be changing in the self-organized network, which makes it hard to obtain continuous measurement data from a fixed neighbor of a node at all time points of SDS. Moreover, when the node is out of sync with its spatial neighbors when transferring data to the base station or data center, it causes the rows in the matrix of SDS to be asynchronous. In other words, it is unable to obtain the data from neighbors at these time points in the data stream.

LIN and MEAN algorithms are not affected by the above problem because they only utilize temporal neighbors of the missing value. However, the performances of DESM and TSNN algorithms are reduced without the support of the data from spatial neighbors.

LIN and TSNN algorithms apply temporal neighbors to calculate the estimated value by linear interpolation. It makes them more accurate than MEAN, which only gets the estimated value by the arithmetic average of temporal neighbor values. When they two are applied in offline data, which are already stored on the base station or data center, the nonmissing values at all time points in the observed data window can be applied to calculate the estimated value, however, when they are applied in the online stream, the nonmissing values at the time points after the missing point are not available to them, which causes the calculation to be less precise.

TSNN has better performance than LIN, DESM, and MEAN when they are working on offline data because it utilizes the correlation between the nearest neighbor values and raw values, and the regression is applied to make the final estimated value. In the regression step, the training data are selected from the nonmissing values alone based on the distance between the two values. If we can find a new way to select more suitable training data by more reasonable rules for regression, it can get better accuracy for imputation.

In this paper, addressing the above problems, we propose a new imputation algorithm VTN that works for online sensor data stream in WSNs. The main contributions of our work are described as follows:

VTN works based on the temporal neighbors in SDS and only requires the measurement data from one senor on the node. Hence, it is not affected by the availability of spatial neighbors, and the algorithm can be deployed in WSNs with multiple sensor nodes or with a single sensor node.

Similar to LIN and TSNN, the technology of linear interpolation based on temporal neighbors is also applied in the VTN algorithm. However, different from them, in VTN, the nonmissing value at the next time point of the current missing point is required to read in to obtain the extra data, and therefore, based on the past information provided by the previous nonmissing values and the future information provided by the nonmissing value at the next time point, the VTN algorithm can calculate the virtual temporal neighbor with relatively higher accuracy, and it builds up the foundation to make a precise estimation for the missing value.

Compared with TSNN, VTN makes an improvement in the rule of selecting nonmissing temporal neighbors. The temporal neighbors that are eligible for training data not only have the value nearer to the temporal neighbor at the missing time point but also have a

similar change rate with it. This new way to choose training data helps the VTN algorithm get higher accuracy because the change rate extracted extra information hidden in the data, and it is beneficial to the estimation of missing values.

Besides the accuracy of imputation, the computational time is another evaluation standard for imputation algorithms, especially for the online imputation of the sensor data stream. We make a detailed analysis for it based on the experiment results and give the final evaluation of the VTN algorithm in the real application in WSNs.

3. Materials and Methods

In this section, we present our VTN algorithm in detail. Firstly, we define the virtual temporal neighbor (VTN) and give the algorithm to calculate it. Then, the correlation between the VTNs and their raw values is studied as the basis of our algorithm designing. Next, we describe the VTN imputation algorithm based on VTNs. Finally, the method to set the parameter θ for the algorithm is suggested.

3.1. Virtual Temporal Neighbor (VTN)

3.1.1. Definition and Calculation Method of Virtual Temporal Neighbor. VTN is the basis of VTN algorithm. Based on our previous description of sensor data stream in Section 2, we give the definition of VTN as follows:

Definition. Virtual temporal neighbor (VTN): in the sensor data stream of a WSN, the VTN of the sensor s_k of the node n_i at the time point t_j exists and is denoted by $vtn(n_i, t_j, s_k)$ if and only if there exist two measurement values acquired by the same sensor of the same node at the time point t_a and t_b, and they satisfy the following conditions:

$$\begin{cases} t_a = \left\{ t_x \,\middle|\, \begin{array}{c} \forall t_y < t_j, t_y \in T,\ v(n_i, t_y, s_k) \neq NA; \\ \exists t_x, t_x < t_j, t_x \in T, v(n_i, t_x, s_k) \neq \text{NA that makes } |t_j - t_x| \leq |t_j - t_y| \end{array} \right\}, \\ t_b = \left\{ t_x \,\middle|\, \begin{array}{c} \forall t_y > t_j, t_y \in T,\ v(n_i, t_y, s_k) \neq NA; \\ \exists t_x, t_x > t_j, t_x \in T, v(n_i, t_x, s_k) \neq \text{NA that makes } |t_x - t_j| \leq |t_y - t_j| \end{array} \right\}, \end{cases} \tag{13}$$

where NA denotes the missing value at the time point.

If the VTN at the time point t_j exists, it can be calculated by linear interpolation as follows:

$$vtn(n_i, t_j, s_k) = \frac{v(n_i, t_a, s_k)(t_b - t_j) + v(n_i, t_b, s_k)(t_j - t_a)}{t_b - t_a}, \tag{14}$$

and the change rate of VTN denoted by $\Delta vtn(n_i, t_j, s_k)$ can be computed as follows:

$$\Delta vtn(n_i, t_j, s_k) = \frac{v(n_i, t_b, s_k) - v(n_i, t_a, s_k)}{t_b - t_a}. \tag{15}$$

For calculating VTN, we need to implement the data structures on the base station or data center in the WSN, shown as Figure 1. The input data matrix (IDM) is used to store the data read from the sensor data stream SDS by the stream reader program, and it is also the data source for the imputation program, for a specified sensor of a node, the measurement value at the time point t_j is reduced as $v(j)$. The values and the time points are stored in different rows of the matrix and are referred to by the *s_index* for the stream reader. In addition, three indies, namely the *u_index*, *c_index,* and *v_index,* are applied to access the matrix in the VTN algorithm. The VTN matrix (VTNM) is another matrix for storing the calculated VTN and its change rate and is accessed by the *vtn_index*. Waiting indies vector (WIV) stores the *c_index* referring to the time point at which the VTN cannot be calculated temporarily.

The calculation of VTN outputs the VTN and its change rate of the value at the previous time point when it gets a new value from the stream. Moreover, it scans the waiting indies vector to calculate the VTN for those previous values that are waiting for getting their VTN calculated. The flow chart of VTN calculation is shown as Figure 2.

The pseudocode for calculating VTN is shown in Algorithm 1.

3.1.2. Correlations between Raw Values and Temporal Neighbor Values. As discussion in Section 2, similar to TSNN, in the VTN algorithm, we also utilize the correlation between the temporal neighbors and their raw values. However, we make further improvements by changing the method to calculate the temporal neighbors. To verify the effectiveness of our methods, we make exploratory experiments on the real WSNs dataset.

Firstly, we get a test dataset by reading the streamed data in a random time frame on the Intel lab dataset [26]. Then, we calculate the temporal neighbors of the raw values according to the methods of TSNN and VTN, respectively. Next, the correlation between the temporal neighbors and their raw values is plotted along with the Pearson correlation coefficient between them. The results are shown in Figure 3.

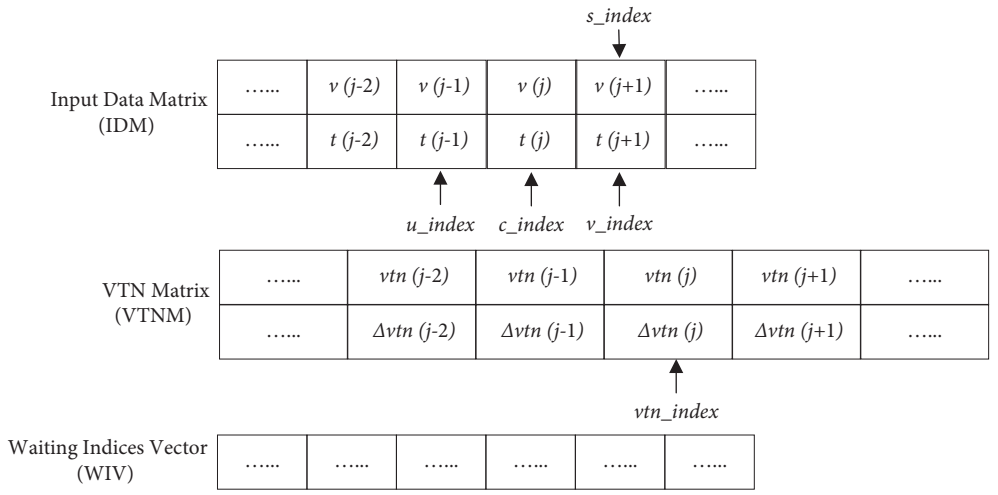

Figure 1: Data structures used in the calculation of VTN.

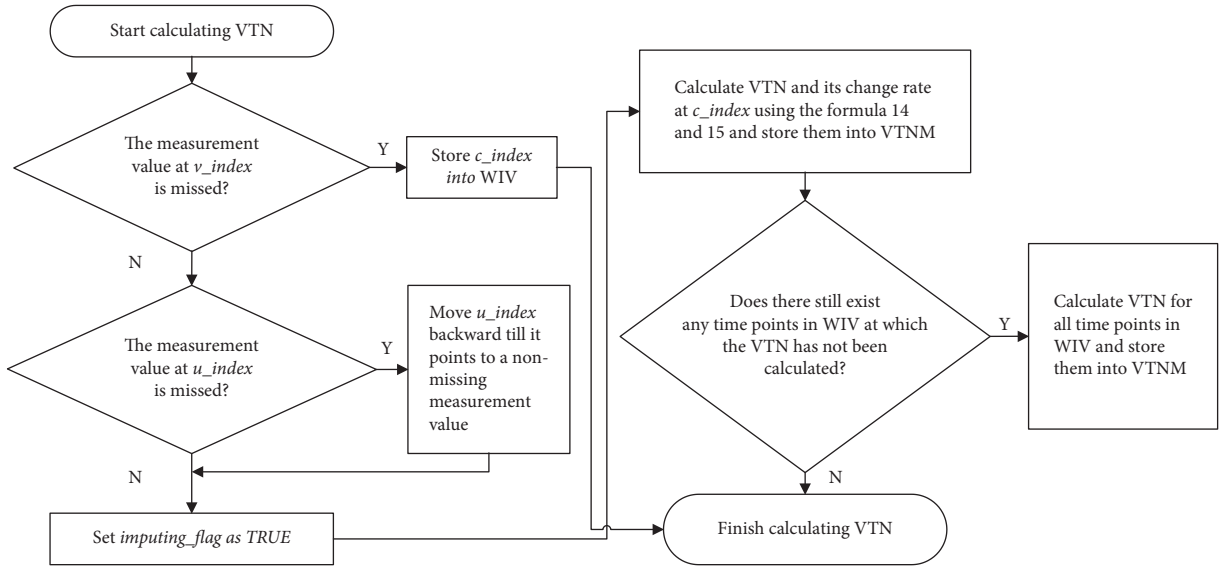

Figure 2: Flow chart of VTN calculation.

Figure 3 shows that the correlation in temperature data is stronger than that in humidity data. It conforms to the real situation because the humidity has more fluctuation than temperature because of more affecting physical factors in the environment. Compared with TSNN, we can find that the data points are less scattered along the diagonal and that the correlation is much stronger in VTN. Upon repeating the same experiments on different time frames or on other real datasets, we get similar results. It indicates that the method to calculate the temporal neighbors in VTN is better than the one in TSNN, which helps compute a more accurate estimation value by regression in the next step.

3.2. VTN Imputation Algorithm

3.2.1. The Calculation of the Estimated Value Based on VTN. Once we have VTNs for all nonmissing values in the sensor data stream, the estimated value for the missing time point can be calculated by regression. The estimation equation is as follows:

$$v\left(n_i, \widehat{t_{\text{miss}}}, s_k\right) = \alpha_{s_k} + \beta_{s_k} vtn\left(n_i, t_{\text{miss}}, s_k\right), \tag{16}$$

α_{s_k} and β_{s_k} can be calculated by minimized residual sum of squares as follows:

Input: input data matrix (IDM), VTN matrix ($VTNM$), u_index, v_index, c_index, vtn_index, waiting indices vector (WIV), imputing_flag.
Output: VTN matrix ($VTNM$), imputing_flag. 1.
if $IDM[1, v\_index] = NA$ then
(2) Add c_index into WIV
(3) else if $IDM[1, v\_index] \neq NA$ and $IDM[1, u\_index] \neq NA$ then
(4) $imputing\_flag \leftarrow TRUE\, VTNM[1, vtn\_index] \leftarrow (IDM[1, v\_index] - IDM[1, u\_index]) * (IDM[2, c\_\text{index}] - IDM[2, u\_index]) // (IDM[2, v\_index] - IDM[2, u\_index]) + IDM[1, u\_index]$
(5) $VTNM[2, vtn\_index] \leftarrow (IDM[1, v\_index] - IDM[1, u\_index]) / (IDM[2, v\_index] - IDM[2, u\_index])$
(6) if $WIV \neq NULL$ then
(7) Reverse WIV
(8) for each element i in WIV do
(9) $c\_index\_temp \leftarrow i$
(10) $u\_index\_temp \leftarrow u\_index$
(11) $v\_index\_temp \leftarrow v\_index$
(12) if $c\_index\_temp = u\_in\ \ dex\_temp$ then
(13) $u\_index\_temp \leftarrow u\_index\_temp - 1$
(14) while $VTNM[u\_index\_temp, 1] = NA$ do
(15) $u\_index\_temp \leftarrow u\_index\_temp - 1$
(16) end while
(17) end if
(18) $VTNM[1, c\_index\_temp] \leftarrow (IDM[1, v\_index\_temp] - IDM[1, u\_index\_temp]) * (IDM[2, c\_index\_temp] - IDM[2, u\_index\_temp]) / (IDM[2, v\_index\_temp] - IDM[2, u\_index\_temp]) + IDM[1, u\_index\_temp]$
(19) $VTNM[2, c\_index\_temp] \leftarrow (IDM[1, v\_index\_temp] - IDM[1, u\_index\_temp]) / (IDM[2, v\_index\_temp] - IDM[2, u\_index\_temp])$
(20) end for
(21) end if
(22) end if
(23) return VTN matrix ($VTNM$), imputing_flag.

Algorithm 1: Calculation of VTN.

$$\alpha_{s_k}(1), \beta_{s_k}(1) = \underset{\alpha'_{s_k}(1), \beta'_{s_k}(1)}{argmin} \sum_{T_{\text{sample}}} \left(v\left(n_i, t_{\text{sample}}, s_k\right) - \alpha_{s_k}'(1) - \beta_{s_k}' vtn\left(n_i, t_{\text{sample}}, s_k\right) \right)^2, \tag{17}$$

where $t_{\text{sample}} \in T_{\text{sample}}$. T_{sample} is the selected timepoint set of nonmissing data in the input data matrix IDM.

To have the closest values and change rates in the same direction to the VTN of the missing value, the VTNs in the T_{sample} can be obtained as follows:

$$T_{\text{sample}} = \left\{ T' \left| \forall T'' \subset T \right|, \left| T'' \right| = \theta,\ t' \in T''; \exists \Delta vtn\left(n_i, t', s_k\right) * \Delta vtn\left(n_i, t_{\text{miss}}, s_k\right) \geq 0 \text{ and } T' = \underset{T''}{\text{argmin}} \sum_{T''} \left| \Delta vtn\left(n_i, t', s_k\right) - \Delta vtn\left(n_i, t_{\text{miss}}, s_k\right) \right| \right\}. \tag{18}$$

θ is the preset number of VTNs of nonmissing values that are used as training data of regression in the VTN imputation algorithm. As the result, $v(n_i, \widehat{t_{\text{miss}}}, s_k)$ can be used to impute the missing value in the input data matrix IDM. The flow chart of VTN imputation is shown in Figure 4.

The implementation of VTN imputation is shown in Algorithm 2.

3.2.2. Discussion about Parameter θ in VTN Imputation Algorithm. In the VTN imputation algorithm, the number of VTNs of nonmissing values is preset by θ. Firstly, all VTNs in the past time points of the input data matrix, including the VTN of the missing value are sorted by values. Next, centered on the VTN of the missing value, twice the number of θ VTNs are selected to be sorted again by their change rates if they have the same changing direction as that of the missing value. Finally, θ VTNs are picked out and are applied to the regression in the next step. To find the relationship between the two performance indicators, the imputing accuracy and the computational time and the parameter θ, we make experiments as follows:

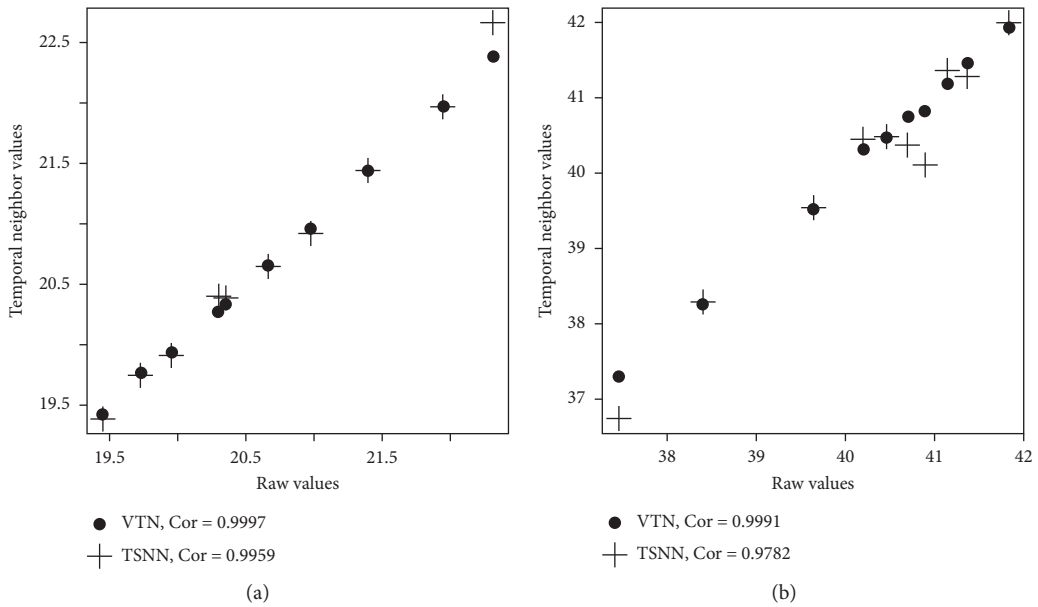

Figure 3: Correlation between raw values and their temporal neighbor values on Intel Lab dataset. (a) Temperature. (b) Humidity.

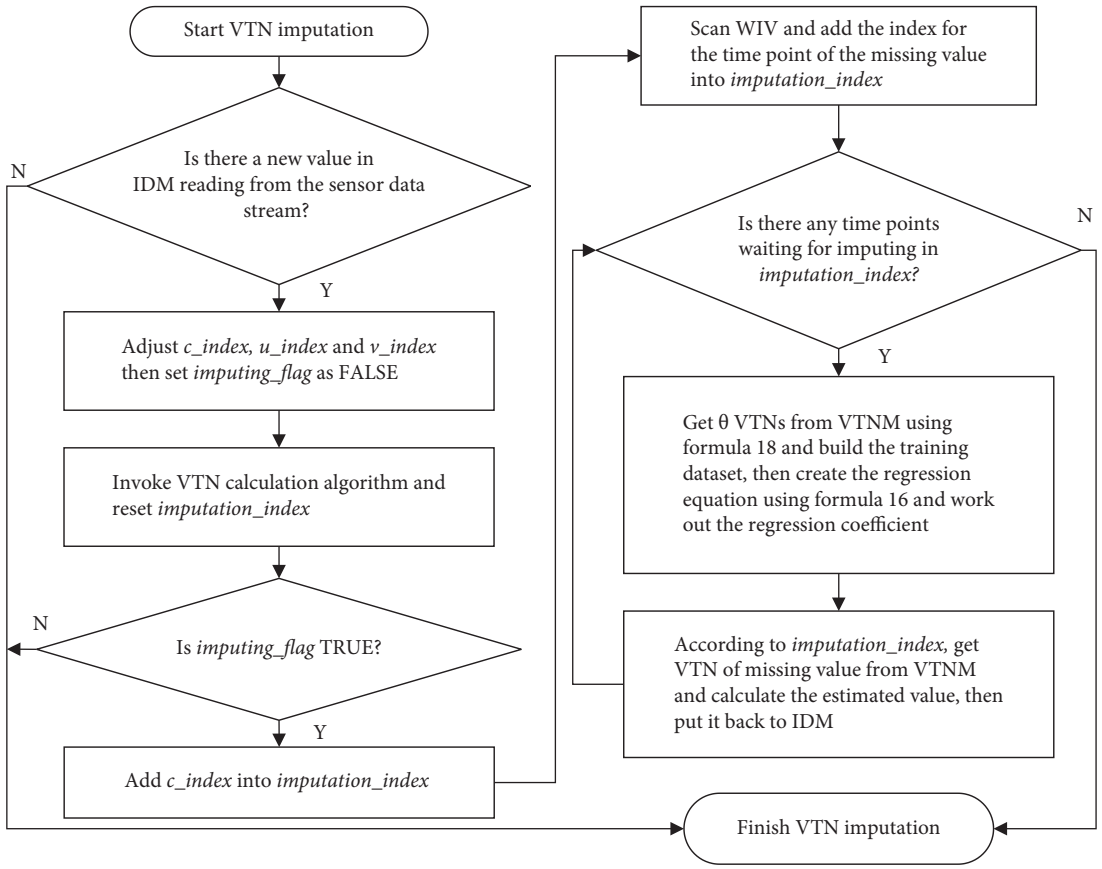

Figure 4: Flow chart of VTN imputation.

Firstly, the parameter θ is set as 5, and we randomly select one of the subsets from the Intel lab dataset. Apply the MCAR method to temperature values to get the subsets with missing values at each of the missing percentages ranging from 5% to 50%. Then, we stream the dataset by a rate, such as one data per minute, and apply the VTN imputation

Input: input data matrix ($I\ DM$), node n_i, sensor s_j, θ,
Output: input data matrix ($I\ DM$)
if there exists a new data in IDM then
$c\_index \leftarrow c\_index + 1$
U $\_index \leftarrow c\_index - 1$
V $\_index \leftarrow c\_index + 1$
vtn $\_index \leftarrow vtn\_index + 1$
$imputing\_flag \leftarrow FALSE$ Get the new data from IDM by v_index
Call Calculation of VTN
$imputation\_index \leftarrow NULL$
if $imputing\_flag = TRUE$ then
Add c_index into imputation_index
end if
if $WIV \neq NULL$ then
for each element i in WIV do
if $I\ DM[1, i] = NA$ then
Add $I\ DM[2, i]$ into imputation_index
end if
end for
end if
if $imputation\_index \neq NULL$ then
Sort imputation_index by increasing index
for each element j in imputation_index do
for each k in 1: j do
if $VTNM[2, k] * VTNM[2, j] \geq 0$ then
Add $VTNM[, k]$ into $PAST\_VTNM$
end if
end for
sort $PAST\_VTNM$ by increasing order of $PAST\_VTNM[1,]$
$new\_imputation\_index \leftarrow new\ index\ for\ imputing\ value\ in\ PAST\_VTNM$
$lower\_bound \leftarrow new\_imputation\_indx - \theta$
high $er\_bound \leftarrow new\_imputation\_indx + \theta$
if $lower\_bound < 1$ then
$lower\_bound \leftarrow 1$
else if high $er\_bound > |PAST\_VTNM| - 1$
high $er\_bound \leftarrow |PAST\_VTNM| - 1$
end if
$PAST\_VTNM\_CAN\ DI\ DATE \leftarrow PAST\_VTNM$[, $lower\_bound$: $higher\_bound$]
$new\_imputation\_index \leftarrow new\ index\ for\ imputing\ value\ in\ PAST\_VTNM\_CAN\ DI\ DATE$
for each element m in PAST_VTNM_CANDADATE do
CHANGE_RATE_DIST
$\leftarrow |PAST\_VTNM\_CAN\ DI\ DATE[2, new\_imputation\_index] - PAST\_VTNM\_CAN\ DI\ DATE[2, m]|$
end for
sort $PAST\_VTNM\_CAN\ DI\ DATE$ by increasing order of CHANGE_RATE_DIST
Remove imputing value from $PAST\_VTNM\_CAN\ DI\ DATE$
$PAST\_VTNM\_CAN\ DI\ DATE \leftarrow PAST\_VTNM\_CAN\ DI\ DATE[, 1: \theta]$
$I\ DM\ \_CAN\ DI\ DATE \leftarrow raw\ values\ of\ PAST\_VTNM\_CAN\ DI\ DATE\ in\ I\ DM$
Construct the estimation equation using PAST_VTNM_CANDADATE and IDM_CANDIDATE to
regress the coefficients $\alpha_{s_k}, \beta_{s_k}$
Compute $v(n_i, \widehat{t_{\text{miss}}}, s_k)$ using α_{s_k}, β_{s_k} and $VTNM[1, j]$
$I\ DM[1, j] \leftarrow v(n_i, t_{\text{miss}}, s_k)$
end for
end if
end if
return IDM

Algorithm 2: VTN imputation algorithm.

algorithm to process the stream. After repeating the experiment 10 times for different subsets, we get the results of the average RMSE and the computational time for the imputation. Increasing the θ by step size 5 till 35, we repeat the experiments, and the results are shown in Figure 5.

From Figure 5, we can find that larger θ can bring lower RMSE, i.e., the higher accuracy of imputation, however, it increases the time costs as well. Moreover, the performance of accuracy is improved slowly when the θ is great than 10. We repeat the same experiments on the different sensor data for humidity on the Intel lab dataset and repeat all experiments on other datasets: the GreenORB dataset [27] and the Air Quality dataset. All of them give similar results.

Therefore, in the VTN algorithm, it is reasonable that the parameter θ is preset as the number that is a bit larger than 10 so that we can get relatively higher accuracy and shorter computational time for imputation.

4. Experiment Results

In this section, we make evaluation of the performance of the proposed VTN algorithm. Firstly, we introduce evaluation methods and datasets applied in experiments. Then, the experiment steps are given, and results for imputing accuracy and computational time on three different real datasets are described in detail.

4.1. Evaluation Methods. Generally, the accuracy of imputation can be evaluated with the root mean square error (RMSE), which can be computed by,

$$\text{RMSE} = \sqrt{\frac{\sum(\text{real value} - \text{estimated value})^2}{\text{the number of estimations}}}. \quad (19)$$

The smaller RMSE indicates the higher accuracy of the imputation algorithm. In addition, the computational time should be taken into consideration for online imputation, and the microbenchmark tool [28] is applied to evaluate them in experiments. We write codes in *R* language, and all experiments are running on a computer with Intel Core i7 2.9 Ghz CPU and 16 GB RAM.

4.2. Datasets and Their Preprocessing. Three real world sensor datasets are applied in experiments: the Intel Lab dataset, the GreenOrbs dataset, and the Air Quality dataset. They represent different application scenarios, which ensure the sufficient tests of our imputation algorithm.

The Intel lab dataset contains data from an indoor wireless sensor network deployed in the Intel Berkeley Research lab, which is composed of 54 nodes. Each node is equipped with multiple sensors and is working continuously to obtain different physical quantities on its location every 31 s, including temperature, humidity, light, and voltage values. The GreenOrbs dataset provides us with the outdoor data collected from 120 wireless sensor nodes scattered in a forest. Each node obtains the data, including temperature, humidity, and light values every 80 to 85 s. The Air Quality dataset contains the data of a single multisensory node network, which gets the air quality data hourly, including temperature, humidity, the concentration of CO, NO2 concentration, Non Metanic HydroCarbons concentration, and benzene concentration.

Before applying these three datasets to the experiments, we preprocess them as follows: firstly, the missing data contained in these raw datasets are deleted to get complete datasets. Then, the data values, except the temperature and humidity, are removed because only temperature and humidity values are applied in our experiments. Secondly, for every dataset, we randomly divide it into subsets in which the measurement values are continuous, i.e., the raw dataset and the number of values in the subsets are the same for the experiments. It is worth noting that for the Intel Lab dataset and the GreenOrbs dataset, the measurement values can be obtained from more than one note in the WSNs. For testing our algorithm reasonably, we get subsets from 20% of all nodes chosen randomly. Thirdly, in this paper, we only consider the situation that the value obtained by the sensor of the node is missing at the time points randomly. Therefore, some values of the subsets have been marked as dummy missing values (not available, denoted by NA) using the Missing Completely at Random (MCAR). Then, we get two versions for each subset, one is with missing values and is applied to test algorithms, and the other is without the missing values. After the missing values in the subset are imputed in experiments, RMSE can be calculated based on the raw values in the latter one. In experiments, the missing percentage can be changed to observe the different results. The upper bound of the missing percentage is set as 50% in our experiments because it is risky that over 50% of data is missing for the observed values and imputation should not be used in this situation [29]. In addition, in the experiments, the measurement values can be extracted in different intervals to make the data density of the dataset changeable. The dataset with the shorter sampling interval gets the higher data density. Then, we can observe the results and evaluate the algorithms. Finally, to simulate the actual working of the sensor network, the base station or data center collects the data from the sensor nodes in the form of a data stream, and the imputation algorithm deployed on it checks the data and makes imputation for the missing values. In each experiment, the imputation algorithm reads the measurement value one by one at the original rate from the subsets and makes an imputation when there is a missing value. After finishing the imputation for all missing values in the subset, the imputation result can be used to evaluate the performance of algorithms.

4.3. Evaluation of RMSE

4.3.1. Experiments and Results on Intel Lab Dataset. Firstly, the sampling interval is set to 1 min, and the missing percentages are from 5% to 50%. Next, we randomly select one of the subsets of the Intel lab dataset. Apply the MCAR method to the temperature values to get the subsets with missing values at each of the missing percentages. Then, we apply different algorithms to make the imputation for the

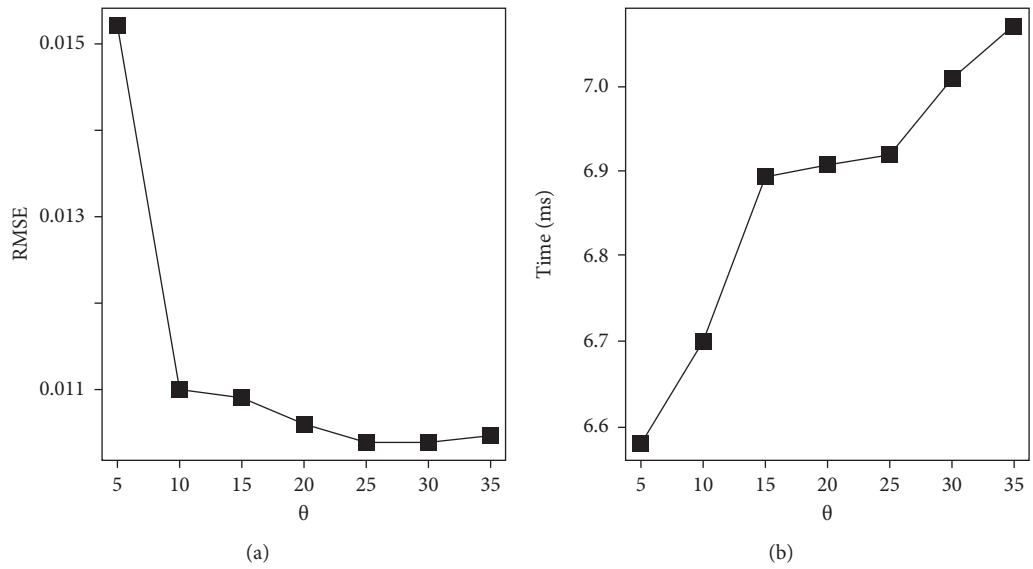

Figure 5: The relationship between parameter θ and performance of VTN algorithm. (a) RMSE and ?. (b) Computational time and ?.

missing values in the stream from the subset, each one as a test case. The experiment is repeated 10 times for different subsets. Finally, the average RMSE for each case at the same missing percentage is calculated based on the test results of all cases. Similarly, we apply the MCAR method to humidity values and repeat the same experiments.

The parameter θ is set as 15 in the VTN algorithm. To evaluate all imputation algorithms under the same condition, the spatial data from the neighbor nodes and the temporal data from other sensors on the same node are not applied in the TSNN and DESM algorithms. Therefore, in the TSNN algorithm, the spatial-temporal coefficient λ is set to 0, and the best k_t for the temporal nearest neighbors will be calculated and applied by the algorithm. Similarly, in the DESM algorithms, the weight parameter α is set to 0.

The experiment results are shown in Figure 6.

Figure 6 shows that for all imputation algorithms, the RMSE rises as the percentage of the missing values increases. The reason is that the reliable information from the non-missing values decreases as the number of missing values increases, and the imputed values are applied in the calculation of the estimated value of the next missing value when the algorithms make the online imputation. Therefore, for imputation algorithms that depend on more than one previous value, such as LIN, MEAN, and TSNN, the performance of accuracy is dropping down and is lower than DSEM, which only requires one previous value. Among the algorithms based on the calculated neighbor value, i.e., LIN, TSNN, and VTN, LIN has a lower performance than the latter two algorithms because it does not utilize the correlation between the measurement value and its neighbors. It is worth noting that although TSNN and VTN are the algorithms applying regression to estimate the missing values, the TSNN algorithm has poorer performance because the temporal neighbor is calculated merely based on the previous values. Besides, it cannot get benefits from the spatial neighbor and the data from other sensors on the same node because they are unavailable in the experiment. However, the VTN algorithm has the smallest RMSE in the entire range of percentages of missing values and shows its best performance. It benefits from the new algorithm design: the method to calculate the virtual temporal neighbors exploits the information from the value next to the current value, and the information of previous values is fully exploited than other algorithms. In addition, compared with temperature, the RMSE for humidity data is bigger. It is caused by the more dramatic variety of the humidity data than the temperature data, which is because of more physical factors affecting humidity than the temperature in real environments.

To evaluate the performance of algorithms in different data densities, we set the sampling intervals from 1 min to 30 mins. Before making the MCAR operation, we extract measurement values based on them to make the subsets with data density from high to low. Then, for different imputation algorithms, we follow the same steps as the previous examples and calculate the average RMSE of cases within the entire missing percentage range. The experiment results are shown in Figure 7.

Figure 7 displays that, for all imputation algorithms, with the longer intervals, the data density is getting lower, and the RMSE is fluctuating toward the bigger direction. Among all algorithms, MEAN has the poorest performance because precise information obtained from the previous values is becoming much lesser when the data density is decreased, and it has the greatest impact on the MEAN algorithm. However, similar to the previous analysis, the VTN algorithm is slightly affected and maintains its best performance among all algorithms.

4.3.2. Experiments and Results on GreenOrbs Dataset. Similar to the experiments on the Intel lab dataset, we make the same experiments on the GreenOrbs dataset. Because of

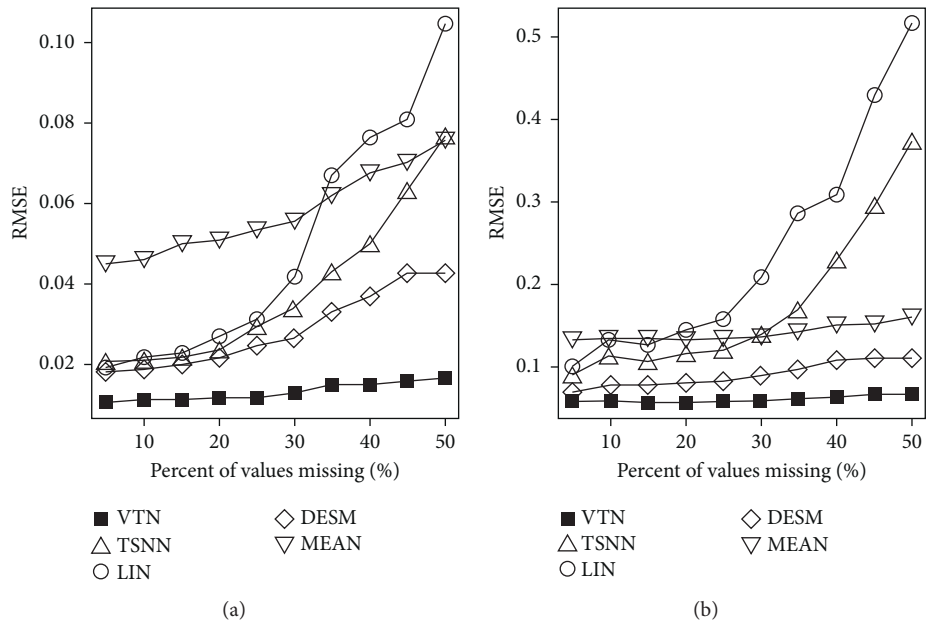

Figure 6: RMSE of imputation algorithms on Intel lab dataset. (a) Temperature data. (b) Humidity data.

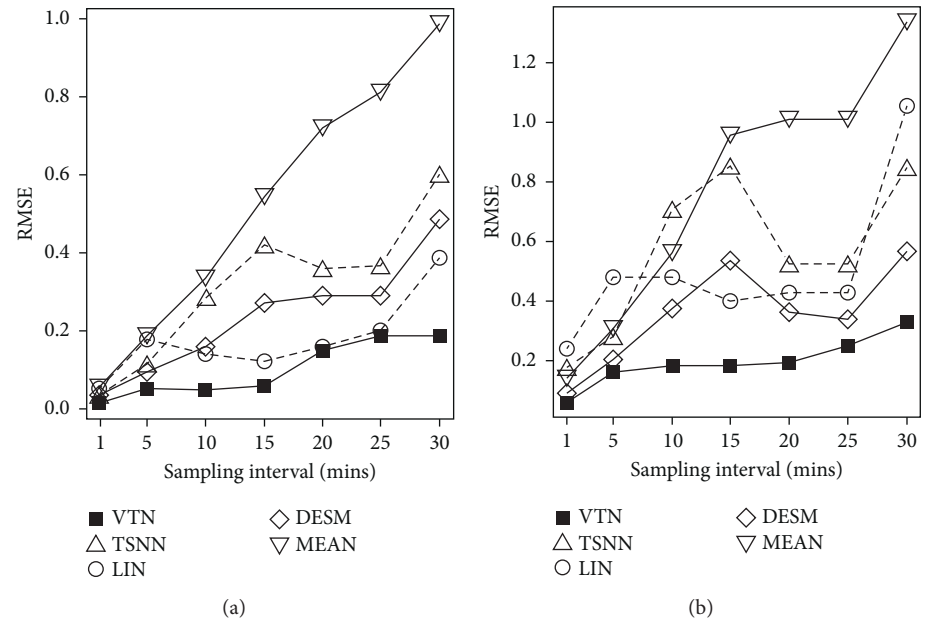

Figure 7: RMSE of imputation algorithms in different data density on Intel lab dataset. (a) Temperature data. (b) Humidity data.

the different data rate in the raw dataset, in experiments on the GreenOrbs dataset, the sampling interval is set to 3 mins, and the missing percentages are still from 5% to 50%.

The experiment results are shown in Figure 8.

Figure 8 demonstrates that the RMSE of all imputation algorithms becomes relatively bigger in the GreenOrbs dataset than in the Intel lab dataset. It is because of the more intensive change of the temperature and humidity data in the outdoor forest than indoors, and it is also the reason that MEAN gets relatively lower performance. However, VTN still obtains the highest performance because the neighbors chosen based on the change rate contribute to the accurate estimated values, which is affected less severely in this situation.

Next, for evaluating the performance of algorithms in the different data density, we set the sampling intervals from 3 mins to 21 mins and make the same experiments. The results are shown in Figure 9.

Similarly, from Figure 9, we find that MEAN is still the worst one among all algorithms. The performance of VTN is getting degraded as the sampling interval is getting longer. However, it still maintains the best accuracy in RMSE.

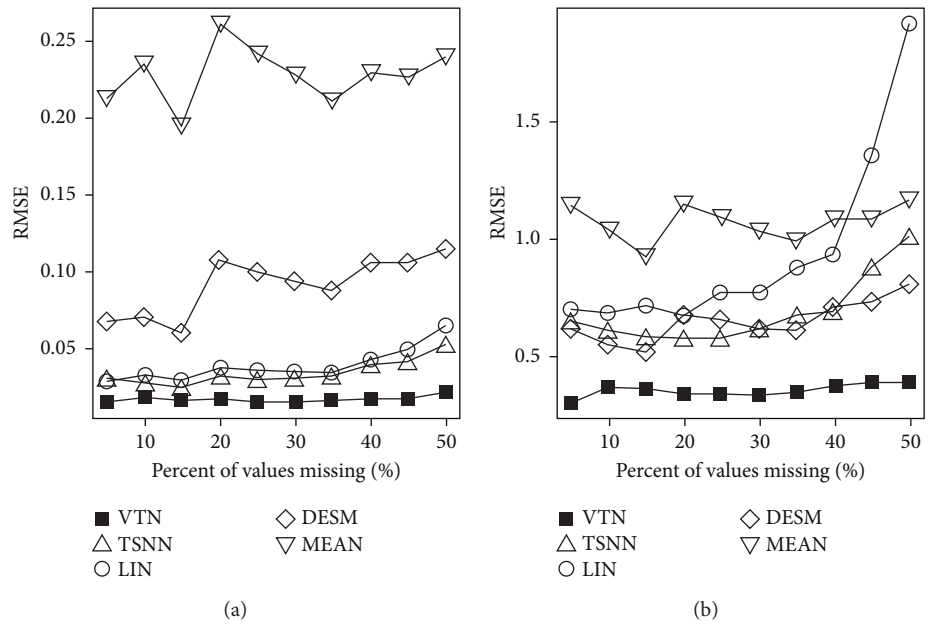

Figure 8: RMSE of imputation algorithms on GreenOrbs dataset. (a) Temperature data. (b) Humidity data.

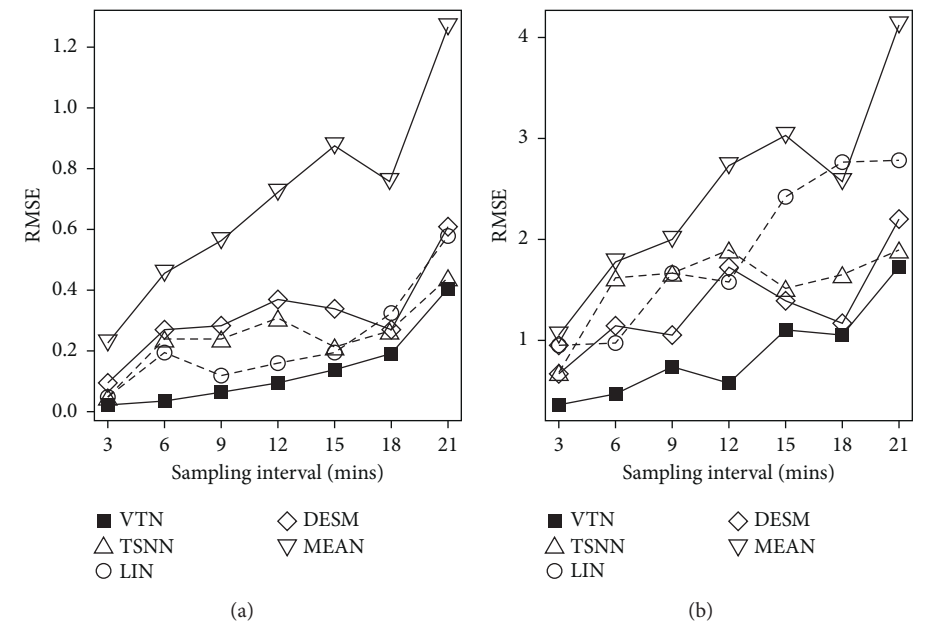

Figure 9: RMSE of imputation algorithms in different data density on GreenOrbs dataset. (a) Temperature data. (b) Humidity data.

4.3.3. Experiments and Results Air Quality Dataset. In experiments on the Air Quality dataset, the sampling interval is set to 1 hour according to the raw dataset. Same as before, the missing percentages are from 5% to 50%. We make the same experiments on the Air Quality dataset and get results as shown in Figure 10.

As shown in Figure 10, for all algorithms, the RMSE is getting bigger with the increasing percentage of missing values. Moreover, for all algorithms, the average RMSE is the biggest in the Air Quality dataset, because in the Intel lab dataset and the GreenOrbs dataset, the intervals in which the sensor obtains the measurement are less than two minutes. However, in the Air Quality dataset, the less interval of the measurement values is one hour, which causes the difference between the two adjacent measurement values to be bigger than the one in the other two datasets.

Then, we set the sampling intervals from 1 hour to 7 hours and make the same experiments to evaluate the performance of algorithms in different data densities. The results are shown in Figure 11.

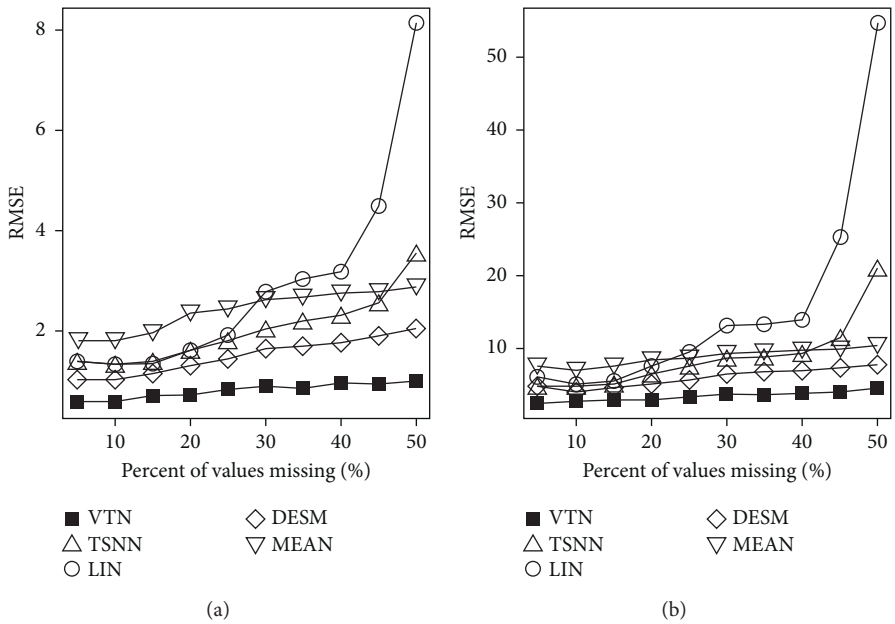

Figure 10: RMSE of imputation algorithms on air quality dataset. (a) Temperature data. (b) Humidity data.

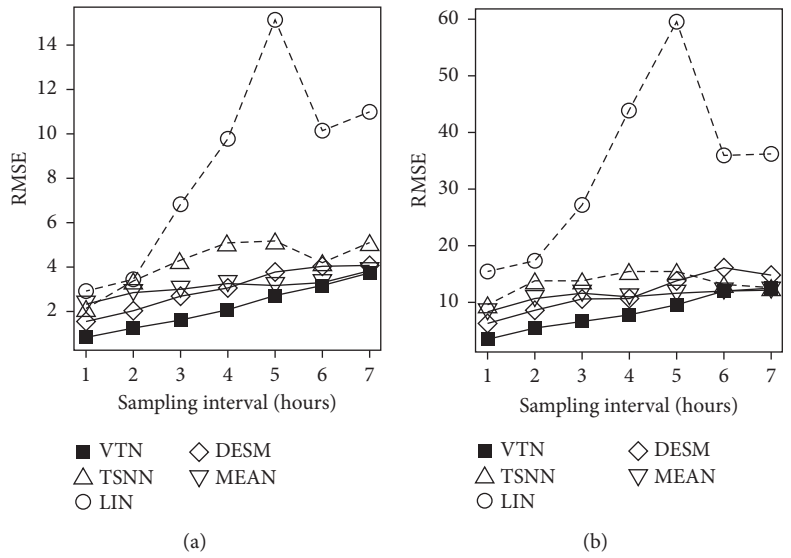

Figure 11: RMSE of imputation algorithms in different data densities in the Air Quality dataset. (a) Temperature data. (b) Humidity data.

It is interesting that different from the previous experiments, we can find that LIN replaces MEAN as the worst algorithm, especially when the percentage of the missing values is getting higher or the sampling interval is getting longer as shown in Figures 10 and 11, because LIN gets the greatest impact among all algorithms because of the drastically variational neighbor values, which makes the linear interpolation produce the poorest result. However, VTN is still the best algorithm with the highest imputation accuracy.

In sum, evaluated by RMSE, the above experimental results on three different datasets show that VTN gets the best accuracy in imputation.

4.4. Evaluation of Computational Time. In this paper, the computational time is denoted by T_C, which is the elapsed time from the imputation algorithm is triggered by a missing data to the end of the imputation. The running time of an imputation algorithm for outputting an estimated value for a missing value read in is denoted by T_R. Generally, T_C is longer than T_R because the base station must spend some time for other required system processing tasks, for example, time for data I/O, and it is denoted by T_S. Hence, the computational time T_C is the sum of T_R and T_S. Moreover, there is extra time required by some imputation algorithms, which is used to obtain the data they need. For example, to

the VTN algorithm, it needs to get the next value in the sensor data stream to make imputation for the current missing value. Hence, it must wait for the next coming data, and the delay time is denoted by T_D. Therefore, the actual total processing time for completing an imputation for a missing value, denoted by T_P, can be computed as follows:

$$T_P = T_C + T_D = T_R + T_S + T_D. \quad (20)$$

In our experiments, we do not need to know about the actual running time T_R and system time T_S. Hence, the computational time T_C is enough for us to evaluate the performance of algorithms, and it can be measured by the microbenchmark tool on the *R* platform in our experiments.

Firstly, we randomly select one of the subsets of the Intel lab dataset and set the sampling interval as 1 min. The interval does not affect the computational time of imputation algorithms, however, it must ensure that the number of values inside the dataset is kept the same for all experiments so we can compare the test results of the different algorithms. Next, the missing percentages are set from 5% to 50%. We apply the MCAR method to temperature values to get the subsets with the missing values at each of the missing percentages. Then, we apply different algorithms to make the imputation for the missing values in the stream, each one as a test case. The experiment is repeated 10 times for different subsets, and finally, based on the test results of all cases, the average computational time for each case at the same missing percentage is measured. As the data type is unrelated to the computational time, we do not need to repeat the same experiments on humidity data.

Similarly, we set the sampling interval as 3 mins for the GreenOrbs dataset and 1 hour for the Air Quality dataset and ensure these subsets have the same number of values. Then, we repeat the same experiments, and the results are shown in Figure 12.

In Figure 12, we can find that each of the imputation algorithms shows similar performance on three different datasets, which indicates there is no relationship between the computational time of imputation algorithm and the dataset. The time consumed by all algorithms is increasing with the percentage of values missing because of more missing values required to be imputed, while VTN and TSNN are rising sharply among them.

The bar charts describe the average computational time for a missing value with different imputation algorithms. DESM has the shortest computational time, and MEAN and LIN have relatively shorter computational times. By contrast, VTN and TSNN still have prominently longer computational times, and VTN requires the longest time for imputing a missing value.

The reason that VTN and TSNN have poorer performances in computational time is that they apply regression to calculate the estimated value for missing data, which is a time-consuming operation. In addition, computing virtual neighbors for each value in the streaming data brings more cost of computational time for the VTN algorithm and makes it the top time-consuming algorithm. Actually, the experiment results only demonstrate the computational time T_C, and we also need to consider the delay time T_D to get the final imputation processing time T_P. Compared with VTN, the other four algorithms, namely TSNN, LIN, MEAN, and DESM, are not required to wait for getting the next value in the sensor data stream. T_D can be ignored for them, however, VTN does need it, which depends on the data stream rates on the sensor network, and it makes the final processing time T_P for the VTN algorithm longer.

5. Discussion

It is an essentiality of the imputation algorithms that they calculate the estimation for the missing value in some way based on the nonmissing values.

Firstly, the accuracy of imputation, which is evaluated by RMSE, is determined by the level of similarity between the estimated values and raw values if they are not missing. The only available information for estimating the operations comes from the nonmissing values. The best precise imputation algorithm should fully exploit the information hidden in the nonmissing values and build the best connection between the valuable information and the missing value. It includes two meanings: one is to effectively exploit the current information to extract the useful one for estimation. The other is to obtain extra information that helps improve the performance of estimation if possible. For the first one meaning, the MEAN algorithm utilizes the average number of previous nonmissing values, which has relatively less accuracy because of its inherent statistical deficiency, as the results shown in experiments. DESM applies the nonmissing value next to the missing value directly. Depending on the deviation level between the adjacent values, it represents different performance as the results shown in experiments. LIN applies more calculation to get neighbor values for making linear interpolation. Hence, its performance depends on the fluctuant of the temporal neighbors and becomes worse when the data density gets lower or the numbers of nonmissing values get lesser. TSNN makes further utilization on the relationship between the missing value and its temporal neighbors by regression. Compared with LIN, its performance is improved in most cases. LIN and TSNN extract more useful information from the previous nonmissing values, but because the temporal neighbors they relied on are calculated based on the nonmissing values on the time points before the time when there is a missing value, and it makes the calculated temporal neighbors less accurate, which leads to the poor preciseness of the regression in the next step. Eventually, LIN and TSNN cannot get ideal performance. They even become worse, as shown in the experiments results. However, by doing the way in the second meaning, in other words, to get extra information from the next measurement value improves the accurate of imputation in VTN algorithm. There are two crucial points in VTN, one is that the virtual temporal neighbor of a value is calculated based on the two values before and after the time point of the value, which boosts the accuracy of the virtual temporal neighbor. The other is that only the virtual temporal neighbors, which are the closest to the temporal neighbor of the missing value in values and change rates are used in the regression to calculate the

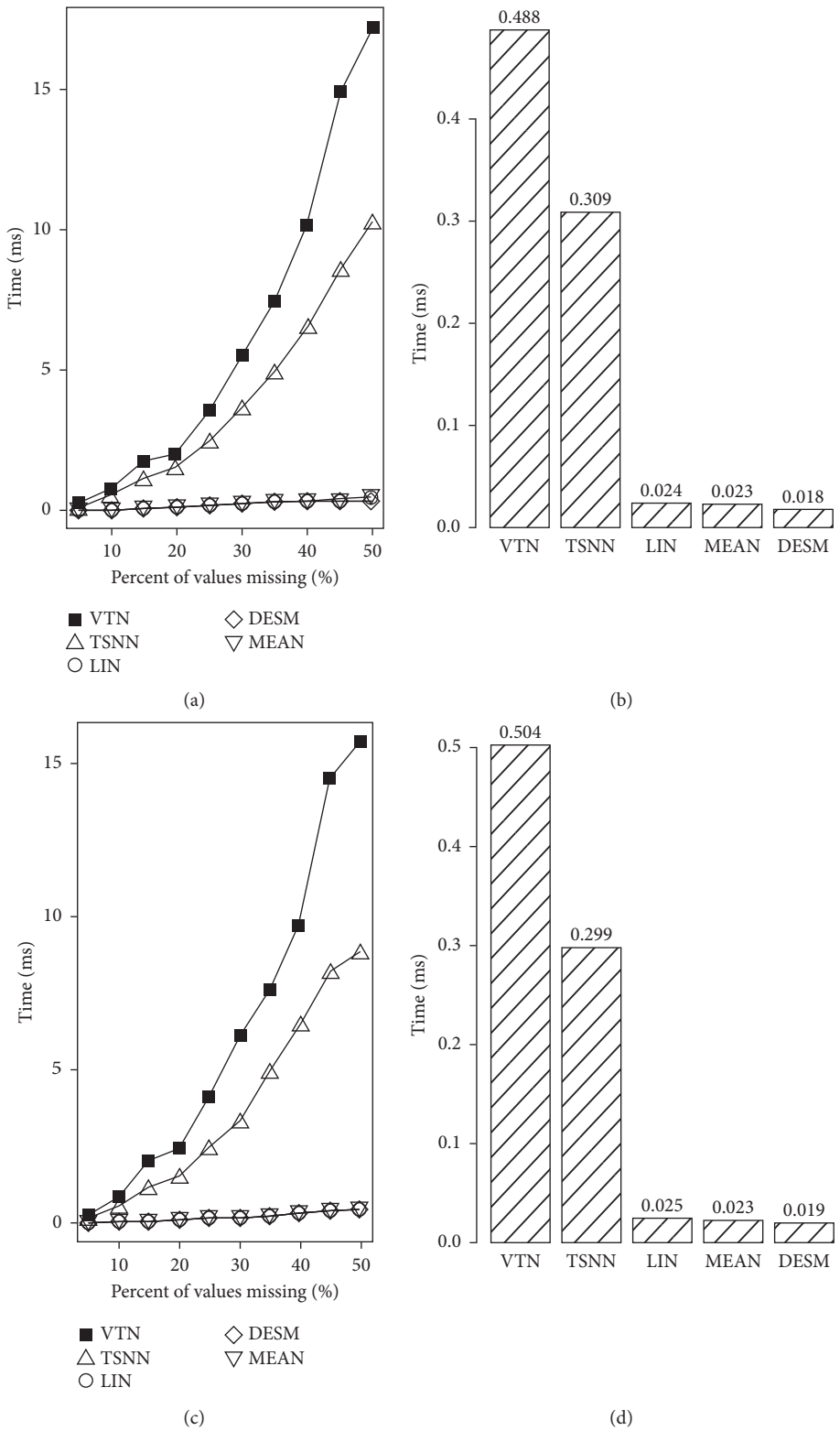

Figure 12: Continued.

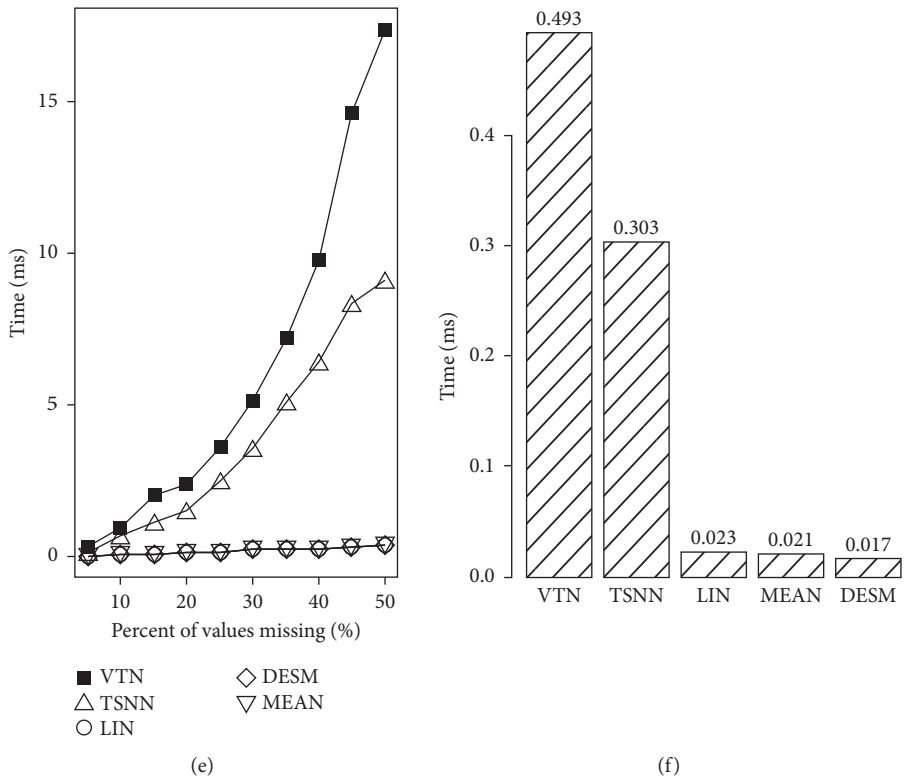

Figure 12: Computational time (a) (c) (e) and the average computational time for a missing value (b) (d) (f). (a) Intel lab dataset. (b) Intel lab dataset. (c) GreenOrbs dataset. (d) GreenOrbs dataset. (e) Air quality dataset. (f) Air quality dataset.

estimated value for imputing the missing value. The first point obtains extra information from the next measurement value read in, and the second point makes further exploitation of all information, which can explain the reason that VTN obtains the best performance in the accuracy of imputation in our experiments.

Secondly, the computational time is another important evaluation strategy for the imputation algorithm, especially for the online imputation of the sensor data stream. Because of more time expenses for more complex calculations than other algorithms, VTN has the longest computational time as shown in the experiment results. If considering about the delay time for waiting the next coming measurement values, the actual total processing time for VTN is quite longer than other algorithms. However, it does not mean VTN is not applicable in real circumstances. On the one side, in sensor networks, because of the limited computing power, storage, and power supply of sensor nodes, the imputation algorithm usually is deployed on the base station or data center, which has enough processing resources to support the running of the VTN imputation algorithm at a reasonable time cost. On the other hand, based on the results of experiments, the final total processing time for VTN is the sum of the computational time and the delay time. For example, on the Intel lab dataset, the original rate of the sensor data stream is 1/31 value per second, i.e., the interval between the two values is 31 s. Hence, the processing time for a missing value is approximately 31.0005 s. It indicates the delay time for obtaining the next value accounts for the most part of processing time. Generally, in the applications of the sensor network, the acceptable response time depends on the type of application. The query application expects the shortest response time, whereas the prediction can accept a little longer response time. For simplicity's sake, we use the query application as an example. As the imputation algorithm is working continuously to impute the possible missing value and output the stream without missing values, the query application can immediately get all complete data without the missing values unless the missing value happens to be in the end point of the data that has been querying currently, which adds the extra delay time of reading one value or more than one values (if the value read in is also a missing value) into the final response time. However, the likelihood of this situation is quite low when the percentage of value missing is not too big. In fact, in real applications, the percentage of value missing is usually small in most cases. Therefore, the extra time cost brought by the delay time in VTN has relatively minor impacts on the application. Compared with minor extra processing time, the improved accuracy of imputation makes the gains outweigh the loss for applications in WSNs, which makes VTN a valuable imputation algorithm.

Finally, as a new imputation algorithm, there is still plenty of room to improve the performance of VTN. The first point is that compared with other algorithms, its computational time is relatively long, and it can be shorter if we make further optimization in the implementation of the algorithm. The next point is that we are continuing to work is

the accuracy of imputation. By designing better methods to choose more suitable virtual temporal neighbors used in the regression step, we expect to improve the imputing preciseness of the VTN algorithm in the future. In addition, in this paper, the method to choose the parameter θ is based on experiments on limited datasets. More experiments on a larger range of datasets are required to be made for finding a better method to preset the parameter in the further study. The last point that is that we have not considered a lot about is the memory space used by the algorithm. In the current VTN, we apply extra space for storing the virtual temporal neighbors, and how to make it less without affecting the computational time should be studied in the next stage. In our research objective to make VTN an online imputation algorithm with faster running time, more imputing accuracy, and lesser memory space, we still have a lot of work to do in our further research work.

6. Conclusions

In this paper, we propose the VTN imputation algorithm to cope with the missing values in the sensor data stream in WSNs. Spatial information is not required for the algorithm, and only the previous temporal values and a next value reading from the stream are used to create the VTN. Next, the missing value can be estimated by regression based on the VTNs, which are closed to the VTN at the missing time point in values and change rates. The RMSE and the computational time are evaluated for the VTN imputation algorithm on three different sensor datasets. Compared with other representative algorithms, the VTN imputation algorithm presents higher imputing accuracy and acceptable time costs when it is applied to online imputation for the sensor data stream.

Data Availability

The system parameters used to support the findings of this study are included within the article. The experiment data used to support the study is available through the web links in the references.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

The authors would like to thank the reviewers for their comments, which helped to improve the paper. This work is partly supported by the National Natural Science Foundation of China under Grants nos. 61872194 and 61902237, the Anhui Science and Technology Department Foundation under Grant 1908085MF207, the Postdoctoral Found of Jiangsu Province under Grant no. 2018K009B, and the Postgraduate Research and Practice Innovation Program of Jiangsu Province under Grant no. KYLX15_0837.

References

[1] S. H. Shah and I. Yaqoob, "A survey: internet of things (IOT) technologies, applications and challenges," in *Proceedings of the 2016 IEEE Smart Energy Grid Engineering (SEGE)*, pp. 381–385, Oshawa, ON, Canada, August 2016.

[2] X. Xue, J. Lu, and J. Chen, "Using NSGA-III for optimising biomedical ontology alignment," *CAAI Transactions on Intelligence Technology*, vol. 4, no. 3, pp. 135–141, 2019.

[3] X. Xue, P.-W. Tsai, and Y. Zhuang, "Matching biomedical ontologies through adaptive multi-modal multi-objective evolutionary algorithm," *Biology*, vol. 10, no. 12, p. 1287, 2021.

[4] A. R. Ganguly, O. A. Omitaomu, and R. M. Walker, "Knowledge discovery from sensor data for security applications," in *Learning from Data Streams*Springer, Berlin, Heidelberg, 2007.

[5] I. F. Akyildiz, W. Weilian Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[6] T. Hossain and S. Inoue, "A comparative study on missing data handling using machine learning for human activity recognition," in *Proceedings of the 2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pp. 124–129, Spokane, WA, USA, June 2019.

[7] J. Zhao, R. Govindan, and D. Estrin, "Computing aggregates for monitoring wireless sensor networks," in *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, pp. 139–148, Anchorage, AK, USA, June 2003.

[8] C. T. Tran, M. Zhang, P. Andreae, and B. Xue, "Multiple imputation and genetic programming for classification with incomplete data," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 521–528, Spokane, WA, USA, 2017.

[9] S. De Vito, E. Massera, M. Piga, L. Martinotto, and G. Di Francia, "On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario," *Sensors and Actuators B: Chemical*, vol. 129, no. 2, pp. 750–757, 2008.

[10] E. Elnahrawy, "Research directions in sensor data streams: Solutions and challenges," Tech. Rep. DCIS-TR-527, Rutgers University, New Brunswick, Canada, 2003.

[11] S. Chavhan and N. A. Chavhan, "A review on data collection method with sink node in wireless sensor network," *International Journal of Distributed and Parallel systems*, vol. 4, no. 1, pp. 67–74, 2013.

[12] R. J. A. Little and D. B. Rubin, *Statistical Analysis with Missing Data*, John Wiley & Sons, Hoboken, NJ, USA, 3rd ed. edition, 2019.

[13] W.-C. Lin and C.-F. Tsai, "Missing value imputation: a review and analysis of the literature (2006-2017)," *Artificial Intelligence Review*, vol. 53, no. 2, pp. 1487–1509, 2020.

[14] V. Gorodetsky, O. Karsaev, and V. Samoilov, "Direct mining of rules from data with missing values," in *Foundations of Data Mining and knowledge Discovery, Studies in Computational Intelligence*Springer, Berlin, Germany, 2005.

[15] Y.-T. Yan, Y.-P. Zhang, Y.-W. Zhang, and X.-Q. Du, "A selective neural network ensemble classification for incomplete data," *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 5, pp. 1513–1524, 2017.

[16] N. Jiang, "A data imputation model in sensor databases," in *Proceedings of the High Performance Computing and Communications Third International Conference*, pp. 86–96, Houston, USA, 2007.

[17] J. Yoon, W. R. Zame, and V. Mihaela, "Estimating missing data in temporal data streams using multi-directional recurrent neural networks," *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 5, pp. 1477–1490, 2018.

[18] M. Lee, J. An, and Y. Lee, "Missing-value imputation of continuous missing based on deep imputation network using correlations among multiple IoT data streams in a smart space," *IEICE - Transactions on Info and Systems*, vol. 102, no. 2, pp. 289–298, 2019.

[19] P. J. García-Laencina, JL. Sancho-Gómez, and A. R. Figueiras-Vidal, "Pattern classification with missing data: a review," *Neural Computing & Applications*, vol. 19, no. 2, pp. 263–282, 2010.

[20] L. Pan and J. Li, "K-nearest neighbor based missing data estimation algorithm in wireless sensor networks," *Wireless Sensor Network*, vol. 2, no. 2, pp. 115–122, 2010.

[21] Y. Li, C. Ai, W. P. Deshmukh, and Y. Wu, "Data estimation in sensor networks using physical and statistical methodologies," in *Proceedings of the 28th International Conference on Distributed Computing Systems*, pp. 538–545, Beijing, China, 2008.

[22] Y. Shao, Z. Chen, F. Li, and C. Fu, "Reconstruction of big sensor data," in *Proceedings of the 2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pp. 1–6, New Brunswick, Canada, 2016.

[23] L. Pan, H. Gao, H. Gao, and Y. Liu, "A spatial correlation based adaptive missing data estimation algorithm in wireless sensor networks," *International Journal of Wireless Information Networks*, vol. 21, no. 4, pp. 280–289, 2014.

[24] X. Ren, H. Sug, and H. Lee, "A new estimation model for wireless sensor networks based on the spatial-temporal correlation analysis," *Journal of information and communication convergence engineering*, vol. 13, no. 2, pp. 105–112, 2015.

[25] Y. Deng, C. Han, J. Guo, and L. Sun, "Temporal and spatial nearest neighbor values based missing data imputation in wireless sensor networks," *Sensors*, vol. 21, no. 5, p. 1782, 2021.

[26] S. Madden, "Intel lab data," 2013, http://db.csail.mit.edu/labdata/labdata.html.

[27] GreenOrbs. Available online: http://www.greenorbs.org/.

[28] Microbenchmark. Available online: https://cran.r-project.org/package=microbenchmark.

[29] G. D. Garson, *Missing Values Analysis and Data Imputation*, Statistical Associates Publishers, Asheboro, NC, USA, 2015.