sensors

MDPI

Article

Temporal and Spatial Nearest Neighbor Values Based Missing Data Imputation in Wireless Sensor Networks

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

[1] College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China; dyl@njupt.edu.cn (Y.D.); hc@njupt.edu.cn (C.H.); guoj@njupt.edu.cn (J.G.)
[2] Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks, Nanjing University of Posts and Telecommunications, Nanjing 210003, China
* Correspondence: sunlj@njupt.edu.cn

Abstract: Data missing is a common problem in wireless sensor networks. Currently, to ensure the performance of data processing, making imputation for the missing data is the most common method before getting into sensor data analysis. In this paper, the temporal and spatial nearest neighbor values-based missing data imputation (TSNN), a new imputation based on the temporal and spatial nearest neighbor values has been presented. First, four nearest neighbor values have been defined from the perspective of space and time dimensions as well as the geometrical and data distances, which are the bases of the algorithm that help to exploit the correlations among sensor data on the nodes with the regression tool. Next, the algorithm has been elaborated as well as two parameters, the best number of neighbors and spatial–temporal coefficient. Finally, the algorithm has been tested on an indoor and an outdoor wireless sensor network, and the result shows that TSNN is able to improve the accuracy of imputation and increase the number of cases that can be imputed effectively.

Keywords: wireless sensor networks; missing data; imputation; temporal and spatial nearest neighbor values; regression

Citation: Deng, Y.; Han, C.; Guo, J.; Sun, L. Temporal and Spatial Nearest Neighbor Values Based Missing Data Imputation in Wireless Sensor Networks. *Sensors* **2021**, *21*, 1782. https://doi.org/10.3390/s21051782

Received: 7 February 2021
Accepted: 28 February 2021
Published: 4 March 2021

1. Introduction

The dataset will become incomplete if some data are lost in the acquisition stage, which may be caused by many different reasons [1]. Generally, most of them are caused by the human errors especially when the data come from manual questionnaires or by the non-human errors when the data are obtained from a system automatically, for example, the faults of the sensors or communication units in wireless sensor networks. The latter will be focused on in this paper. However, most of the theories of data processing and related algorithms are founded on complete datasets so far. Missing data in the dataset which make the dataset incomplete will have an impact on the subsequent data processing stage, for instance, degrading the quality of classification, even making the processing task fail. For example, in human activity recognition where the Support Vector Machine (SVM) is applied as the classification tool. Two per cent of missing data will cause the recognition rate to decrease to 80%. When the missing rate increases to 5%, the recognition rate will drop down to 65%, and if the missing rate is over 10%, the recognition rate will be below 50%, which may make it unavailable in the recognition task [2]. In a very few of situations, we can get the complete dataset by simply discarding parts of it which contain the missing data, but in most of cases, the missing data can be imputed by the estimation values, and a complete dataset will be obtained before we start the data processing in the next step; therefore, it is important to improve the performance of missing data imputation.

Currently, wireless sensor networks (WSNs) contribute considerable data of the physical environment and events through sensors on the deployed nodes which are connected with a communication network. Generally, in WSNs, data are acquired by nodes and sent to the processing center, the unstable and even dramatic change of environment where the

WSNs deployed may increase the chance of sensor faults, and the capacity fade of their power system that is commonly composed of batteries will cause the malfunction of the nodes. This work principle of WSN increases the possibility to lose data during the data acquisition. That means the dataset obtained from WSN will be an incomplete dataset in a high likelihood. Among all methods to deal with the incomplete WSN dataset, one of them is to work on it directly by special algorithms without imputation. Some researchers design effective processing methods on their datasets. Bryan Conroy [3] proposed the Dynamic–Abstain–Boost. It is a dynamic classifier obtained from a two-stage machine learning algorithm, in which the low-dimensional classifiers are combined in the ensemble learning stage. The classifier works directly on the incomplete dataset and is tested on the intensive care unit (ICU) clinical dataset which consists of huge number of physiological sampled measurement data, where some of them are missing because of insufficient monitoring. Vladimir Gorodetskytichu [4] developed a lattice based direct mining method on the binary dataset with missing values. The approach is applied to extract the classification rules, based on upper and low bounds of the rules' sets. After testing on a classification criterion, it will select a subset of rules, which will be used for classification. The mining algorithm has been applied on raw data streams in the computer network. These previous work shows the possibility to process incomplete datasets without imputation. However, they are limited to designated data processing on special application scenarios whereas most of data processing tasks in WSNs still require the complete datasets, which fuels the demand for research on the imputation algorithms in WSNs.

The imputation methods are closely connected with the causes of the data missing and the characteristics of the incomplete datasets. There are lots of different reasons that contribute to the data missing in WSNs, such as the failures of the sensors on nodes, the abnormalities of the wireless communication or the malfunction of the power supply, for instance, the capacity fade of the battery may lead to the abnormalities of the measurements from sensors [5]. A complete dataset in WSNs should have measurement values on all nodes at all time points in an observed time window; if not, we can detect there is missing data in the test case, and the dataset is incomplete. Missingness mechanisms [1] are an important feature of the incomplete dataset; missing completely at random (MCAR) occurs when the probability of a case having a missing value for an observed attribute is independent of either the known values or the missing data, which is a typical situation when the data loss occurs in WSNs and accounts for large proportions of the cases with missing values. Therefore, most of the current researchers use MCAR for the theoretical study and experiments [6]; likewise, in this paper we carry out the study on the imputation algorithms for cases in the MCAR mechanism.

At present, machine learning techniques are the main tools for researchers to design imputation techniques of missing values in WSNs, which are applied to find features of the dataset, exploit the information in the non-missing data and figure out estimators for the missing values. Different research studies have been carried out on this area. In [7], Roman Tkachenko proposed the GRNN-SGTM. General regression neural networks have been used in the ensemble algorithm. Two successively connected general regression neural networks (GRNN) and a successive geometric transformation model (SGTM) neural-like structure are assembled. In addition, for improving the accuracy of estimation of missing values, a weighted summation is applied in the algorithm. Nan Jiang applied the association rule mining techniques in the closed item sets based association rule mining (CARM) [8], which is applied to extract the most recent association rules between sensors and work out the missing values of sensors in a traffic network.

It is worth noting that the data analysis proves that there are strong correlations among data in the dimensions of time and space in WSNs. Regression, as one of the classical machine learning algorithms, can exploit the correlation effectively and make the processing procedure simpler as well, which is conducive to reduce the expense in the process. In [9], Liqiang Pan introduced an adaptive missing data estimation algorithm (CIAM). It is a self-adaptive algorithm in which a multiple regression model is applied to

make the imputation based on the spatial correlation among sensor nodes and improve the accuracy of the missing value estimation. Another imputation method, a new estimation model based on a spatial temporal correlation analysis (STCAM), is presented by Xiaojun Ren [10], for obtaining better estimates of the missing value, four sub-algorithms based on regression are combined to exploit the spatial and temporal correlation in the datasets of an indoor sensor network and a traffic network.

In this paper, we focus on the research route to make imputation based on the spatial and temporal correlation and introduce a new imputation algorithm for missing values in WSNs, the temporal and spatial nearest neighbor values based missing data imputation (referred to as TSNN). Similar to the previous research work, regression tool is applied in our algorithm but in a new way to figure out the estimates based on the temporal and spatial nearest neighbor values which are defined on the bases of geometrical distance and the data distance. Through this approach, the correlation hidden in the data can be exploited more effectively, and the power of regression can be boosted to improve both the percentage of cases in which a missing value can be estimated and the accuracy of the estimation for missing values.

The rest of this paper is organized as follows. First, in Section 2, we make a review of typical imputation methods in WSNs and present the main contributions of our work. Then, in Section 3, the main reasons of data loss and the scenarios of data missing in WSNs have been described. After that, we give definitions of four kinds of nearest neighbor values from the perspective of time and space dimensions and analyze the correlations between these values and the raw value. Based on them, we elaborate our new imputation algorithm TSNN. Next, in Section 4, we describe the experiments and the results for our method. Finally, we make a discussion and conclude our research work in Sections 5 and 6.

2. Related Work

It is an effective way to make imputation for the missing values in WSNs from perspective of time and space dimensions. Many research studies have been carried out on this field. We have summarized representative methods proposed by previous researchers as follows:

- Linear interpolation model (LIN) Algorithm [11].

The temporal correlation among the measurements of a sensor on a node s_i is utilized in this algorithm. It applies the linear interpolation to figure out the estimated value $t(s_i, \hat{u}_{miss})$ based on two time points u' and u'' that are nearest to the time point u_{miss} at which the measurement is missed inside the time window W.

$$t(s_i, \hat{u}_{miss}) = t(s_i, u'') + \frac{(t(s_i, u') - t(s_i, u''))(u_{miss} - u'')}{u' - u''} \quad (1)$$

where $u', u'' \in W$.

- K-nearest temporal neighbors (TKNN) Algorithm.

TKNN is a special version of the nearest neighbors method (NNs) [12], in which the neighbor values come from the nearest time points. When there is more than one sensor on a same node s_i, the algorithm applies the measurements of the sensor without missing values to find K temporal nearest neighbors $I(i, j)$ inside the time window W, meanwhile, the weight coefficient W_{td} will contribute to the final calculation of the estimated value $t(s_i, \hat{u}_{miss})$.

$$t(s_i, \hat{u}_{miss}) = \sum_{I(i,j)} W_{td}(u_j, u_q) t(s_i, u_q) \quad (2)$$

where $u_q \in I(i, j)$.

- K-nearest spatial neighbors (SKNN) Algorithm,

Similar to TKNN, the algorithm is another version of NNs, in which the neighbor values come from the nearest spatial neighbors. It is based on the premise that there is

more than one sensor on the same node s_i. Instead of temporal nearest neighbors, K spatial nearest neighbors $G(i,j)$ inside the time window W will be found, and their measurements are used to figure out the estimated value $t(s_i, \hat{u}_{miss})$, combined with the weight coefficient W_{sd}. In [13], the algorithm is applied to impute the missing value of gene in an experiment where K non-missing values of the deoxyribonucleic acid (DNA) in other experiments are selected as spatial nearest neighbors.

$$t(s_i, \hat{u}_{miss}) = \sum_{G(i,j)} W_{sd}(s_i, g(i,j)) t(g(i,j), u_j) \tag{3}$$

where $g(i,j) \in G(i,j)$.

- Applying K-nearest neighbor estimation (AKE) Algorithm [14].

Assumed that there exist spatial nearest neighbors $G(i,j)$ of the sensor on a node s_i which loses data inside the time window W, the correlation can be find among the measurements of the node s_i and its spatial nearest neighbors on the time points without missing values; the algorithm uses them to calculate the estimated value $t(s_i, \hat{u}_{miss})$ by linear regression, where the weight coefficient is applied to figure out the contributions of the different spatial nearest neighbors.

$$t(s_i, \hat{u}_{miss}) = \sum_{G(i,j)} W_k t(s_i, \hat{u_{miss}})\ (k) \tag{4}$$

where

$$t(s_i, \hat{u_{miss}})\ (k) = \alpha_{s_i}(k) + \beta_{s_i}(k) t(g(i,j), u_{miss}) \tag{5}$$

$$W_k = \frac{R_{si}^2(k)}{\sum_{G(i,j)} {R_{s_i}}^2(g(i,j))} \tag{6}$$

$$R_{si}^2(k) = \frac{\sum_{U_{sample}} \left(t\left(s_i, \hat{u_{sample}}\right)(k) - \overline{t\left(s_i, u_{sample}\right)(k)}\right)^2}{\sum_{U_{sample}} \left(t\left(s_i, u_{sample}\right)(k) - \overline{t\left(s_i, u_{sample}\right)(k)}\right)^2} \tag{7}$$

where $g(i,j) \in G(i,j)$, $\alpha_{s_i}(k)$ and $\beta_{s_i}(k)$ are regression coefficients. W_k is the weight that is applied to calculate the estimated value, in which, $R_{si}^2(k)$ is the percentage of variation explained by the regression model.

- Data estimation using statistical model (DESM) Algorithm [15].

While there are both temporal nearest neighbors $I(i,j)$ and spatial nearest neighbors $G(i,j)$ of the sensor on a node s_i, which loses data inside the time window W, both the temporal and spatial correlations can be used in the algorithm. Therefore, the estimated value $t(s_i, \hat{u}_{miss})$ will be figured out based on the measurements of temporal nearest neighbors and spatial nearest neighbors. A correlation coefficient is obtained from the node s_i, and its spatial neighbors inside the time window, which will be used as the weight coefficient to evaluate the contribution of them.

$$t(s_i, \hat{u}_{miss}) = (1-\alpha) t\left(s_i, u'\right) + \alpha t\left(s_i, u_j\right) \left(1 + \frac{t(g(i, u_{miss}), u_{miss}) - t\left(g(i, u_{miss}), u_j\right)}{t\left(g(i, u_{miss}), u_j\right)}\right) \tag{8}$$

where $u' \in W$, $g(i,j) \in G(i,j)$ and weight parameter α can be computed as:

$$\alpha = \frac{Cov(T(s_i), T(g(i)))}{\sigma T(s_i) \sigma T(g(i))} \tag{9}$$

Methods discussed in the above research work represent the typical ways to utilize the information hidden in the WSNs data from the perspective of time and space dimensions to

make estimates for the missing values; they are effective, but there are still some deficiencies and further work is required to correct them.

- The temporal and spatial correlations are considered separately or taken into account simultaneously in the research work. LIN and TKNN only make use of data in time dimension while SKNN and AKE utilize data in space dimension. DESM combines both of the time and space information to make imputation and balances them depending on a simple correlation coefficient. However, the way to fully exploit the time and space information is expected in order to improve the performance of the imputation;
- Most of research works focus on improving the accuracy of the imputation, but few of them consider the way to improve the percentage of cases in which a mission value can be estimated;
- Little research has been conducted to study the scenarios in which more than one sensor on the node has lost data during their work.

In this paper, we do research by addressing the problems above and propose a new imputation algorithm TSNN. The main contributions of this work are described as follows:

- Temporal and spatial nearest neighbor values are defined from two perspectives: geometrical distance and data distance, which make it available to exploit the temporal and spatial correlation hidden in the data more fully. In this situation, as the tool applied in the algorithm, linear regression shows better performance to make estimates for missing values, which contributes to better accuracy of imputation;
- Four temporal and spatial nearest neighbor values are combined into the algorithm, which makes it flexible to adapt to different situations in which parts of the temporal or spatial information may be unavailable. Benefiting from it, the percentage of cases in which a mission value is improved;
- Three scenarios in which there is more than one sensor on the WSNs node have been considered in the algorithm, which are consistent with actual conditions when the data get lost on the sensors;
- For evaluating the algorithm more reasonably, different from other research work, the raw datasets are utilized directly in most of our experiments without the preprocessing steps such as the mean imputation. Meanwhile, in an indoor and an outdoor WSNs datasets, the nodes have been chosen randomly in the experiments and the performance is evaluated on both sides: the percentage of cases in which a mission value can be estimated and the accuracy of the estimation for missing values.

3. Materials and Methods

3.1. Missing Data in WSNs

In WSNs, the data source are the nodes on which the sensors provide measurements. In this paper, we only discuss the centralized data processing WSNs, namely, all data coming from the nodes will be sent by a wireless communication network to the processing center of the WSN, where the data can be stored and processed. Usually, the processing center is a PC or server that has large computing power and enough memory for dealing with the data sent from nodes effectively. For collecting different physical data in the environment, generally, there is more than one sensor on the node. In most cases, each sensor is responsible for acquiring a type of physical quantity, and if it fails, the corresponding measurement value will be missing. One way to improve reliability is to equip the node with two or more same sensors for obtaining the same quantity, but it is beyond the scope of this paper. A typical example which has nodes equipped with multiple sensors is the WSN deployed in the Intel Berkeley Research lab [16], each node has a weather board equipped with four Mica2Dot sensors, collecting humidity, temperature, light and voltage values at the location of the nodes every 31 s. The measurements are all physical quantities of the environment except for voltage value. In this paper, only temperature and humidity data are used. In addition, the actual reliabilities of the sensors may be different, and they are affected by the devices or the circuits. Therefore, for simplicity in the later discussion,

the odds that data missing due to the fault of sensors are assumed to be able to preset in different scenarios. In actual WSNs, there are three main reasons the data are lost on the node. Suppose there are two sensors on a node for collecting temperature and humidity.

- The communication unit on the node fails at some time points, and the data cannot be sent to the data processing center, so both measurements of the sensors, the temperature and the humidity data will be missing at these time points;
- The communication unit on the node works properly, but due to the failure of the data processing center or the fault in the data transmission, the data coming from the node cannot be received by the data processing center; likewise, the temperature and the humidity data will be missing at this time point;
- The communication unit on the node works properly, but because of the fault of the sensors or the node itself, for instance, the capacity fade of battery, both measurements or one of them, becomes abnormal, the spike, for example [17]. This data has been sent to the data processing center but will be removed because it has been judged to be abnormal; therefore, the temperature and the humidity data or one of them will be missing at this time point.

According to the cases above, in a time window, the scenarios of data missing in a node can be classified into three types: both temperature and humidity data are missing at all time points; the temperature (humidity) data are missing, but the humidity (temperature) data are available at all time points; the temperature (humidity) data are missing, but the humidity (temperature) data are available at some of the time points in certain probability. These scenarios have been considered into our new imputation algorithm. Moreover, generally, there exist correlations among the measurements of sensors on the same node, for example, the temperature and the humidity data. For exploiting the correlative relationships, we define the nearest neighbor values from the perspective of the time and space dimensions, which can take better advantage of the correlations between sensors on the same node and on the neighboring nodes as well. For the convenience of discussion, in this paper, it is assumed that there are only two sensors on a node in the WSN, for collecting temperature and humidity data separately, and the three types of scenarios will be used in the algorithm design and experiments.

3.2. Temporal and Spatial Nearest Neighbor Values for a Node in WSNs

3.2.1. Definition and Computation of Temporal and Spatial Nearest Neighbor Values

As the basis of our TSNN imputation algorithm, we define four types of nearest neighbor values from the perspective of time and space dimensions; they are spatial nearest neighbor value in geometrical distance (SGNN), spatial nearest neighbor value in data distance (SDNN), temporal nearest neighbor value in time distance (TTNN) and temporal nearest neighbor value in data distance (TDNN).

In a wireless sensors network, nodes often work continuously, so we are usually concerned with the physical measurement values in a time period. Given a physical space P, let $S = \{s_1, s_2, \ldots, s_{n-1}, s_n\}$ be a set of n sensor nodes, $s_i \in S$, and a time window W and let $W = \{u_1, u_2, \ldots, u_{m-1}, u_m\}$ be a set of m time points, $u_j \in W$. For each node s_i, $G(i, j)$ represents its spatial nearest neighbors at time point u_j, and the number of the spatial nearest neighbors is k_s; $I(i, j)$ represents its temporal nearest neighbors at time point u_j, and the number of the temporal nearest neighbors is k_t. The node s_i has two sensors which are used to get temperature value and humidity value, respectively, at time point u_j on its location in the physical space; the temperature value is denoted by $t(s_i, u_j)$, and the humidity value is denoted by $h(s_i, u_j)$.

Definition 1. *Spatial Nearest Neighbor Value in Geometrical Distance (SGNN): Let $t_{sgnn}(s_i, u_j)$ and $h_{sgnn}(s_i, u_j)$ be the nearest spatial neighbor value of node s_i for the temperature value and the humidity value at time point u_j in geometrical distance, respectively.*

The geometrical distance is calculated by the positions of the node s_i and its spatial neighbors in space. The geometrical distance between two nodes is denoted by $D_{sg}(\cdot,\cdot)$, and the nearest spatial neighbor value for temperature value in geometrical distance $t_{sgnn}(s_i,u_j)$ can be given as:

$$t_{sgnn}(s_i,u_j) = \begin{cases} t(f_t(i,j),u_j) & G(i,j) \neq \varnothing \\ NA & G(i,j) = \varnothing \end{cases} \tag{10}$$

Similarly, the nearest spatial neighbor value for humidity value in geometrical distance is given as:

$$h_{sgnn}(s_i,u_j) = \begin{cases} h(f_h(i,j),u_j) & G(i,j) \neq \varnothing \\ NA & G(i,j) = \varnothing \end{cases} \tag{11}$$

where NA means the value is not available, and $f(i,j)$ describes the spatial nearest neighbor of node s_i at time point u_j in geometrical distance, which is computed as

$$f_t(i,j) = \left\{ g_x(i,j) \middle| \begin{array}{c} \forall g_y(i,j),\ t(g_y(i,j),u_j) \neq NA;\ \exists g_x(i,j)\ that\ makes \\ D_{sg}(g_x(i,j),s_i) \leq D_{sg}(g_y(i,j),s_i),\ \ t(g_x(i,j),u_j) \neq NA \end{array} \right\} \tag{12}$$

$$f_h(i,j) = \left\{ g_x(i,j) \middle| \begin{array}{c} \forall g_y(i,j), h(g_y(i,j),u_j) \neq NA;\ \exists g_x(i,j)\ that\ makes \\ D_{sg}(g_x(i,j),s_i) \leq D_{sg}(g_y(i,j),s_i),\ \ h(g_x(i,j),u_j) \neq NA \end{array} \right\} \tag{13}$$

where $g_x(i,j) \in G(i,j)$, $g_y(i,j) \in G(i,j)$ and $x \neq y$.

Definition 2. *Spatial Nearest Neighbor Value in Data Distance (SDNN): Let $t_{sdnn}(s_i,u_j)$ and $h_{sdnn}(s_i,u_j)$ be the nearest spatial neighbor values of node s_i for the temperature value and the humidity value at time point u_j in data distance, respectively.*

Different from the geometrical distance, the data distance between two nodes is calculated by the data of the nodes and is only available when there are more than one types of physical values in the data, and it indicates the relationship between different types of values acquired by nodes. The data distance between two nodes is denoted by $D_{sd}(\cdot,\cdot)$. For example, there are two nodes and each of them has obtained three types of values, A, B and C. Values a1, b1 and c1 are from node1, and values a2, b2 and c2 are from node2. Then, to value A, the data distance between node 1 and node 2 can be calculated as $D_{sd}(node1, node2) = \sqrt{(b1-b2)^2+(c1-c2)^2}$. In this paper, we only consider that there are two types of values obtained on nodes, the temperature and the humidity. To temperature value, the data distance between them can be calculated by humidity values and vice versa. Therefore, to the temperature value, the data distance between nodes s_p and s_q at time u_j can be computed as $D_{sd}(s_p,s_q) = \sqrt{(h(s_p,u_j)-h(s_q,u_j))^2} = |h(s_p,u_j)-h(s_q,u_j)|$; likewise, to the humidity value, the data distance between nodes s_p and s_q at time point u_j can be computed as $D_{sd}(s_p,s_q) = \sqrt{(t(s_p,u_j)-t(s_q,u_j))^2} = |t(s_p,u_j)-t(s_q,u_j)|$.

Then the spatial nearest neighbor value of node s_i for the temperature value at time point u_j in data distance can be computed as

$$t_{sdnn}(s_i,u_j) = \begin{cases} \sum\limits_{G(i,j)} W_{sd}(s_i,g(i,j))t(g(i,j),u_j) & h(s_i,u_j) \neq NA\ and\ \forall g(i,j) \in G(i,j), \exists h(g(i,j),u_j) \neq NA \\ NA & h(s_i,u_j) = NA \\ NA & \forall g(i,j) \in G(i,j), \exists h(g(i,j),u_j) = NA \end{cases} \tag{14}$$

Similarly, the spatial nearest neighbor value of node s_i for the humidity value at time point u_j in data distance can be computed as

$$h_{sdnn}(s_i,u_j) = \begin{cases} \sum\limits_{G(i,j)} W_{sd}(s_i,g(i,j))h(g(i,j),u_j) & t(s_i,u_j) \neq NA \text{ and } \forall g(i,j) \in G(i,j), \exists t(g(i,j),u_j) \neq NA \\ NA & t(s_i,u_j) = NA \\ NA & \forall g(i,j) \in G(i,j), \exists t(g(i,j),u_j) = NA \end{cases} \quad (15)$$

where $g(i,j) \in G(i,j)$, and the weight W_{sd} can be computed as

$$W_{sd}(s_i,g(i,j)) = \frac{K(\, D_{sd}\,(s_i,g(i,j)))}{\sum_{G(i,j)} K(\, D_{sd}\,(s_i,g(i,j)))} \quad (16)$$

where $K(\cdot)$ is a kernel function, for example Gaussian.

Especially, if $h(s_i,u_j)$ is missing, the $t_{sdnn}(s_i,u_j)$ will be unavailable. Similarly, $h_{sdnn}(s_i,u_j)$ will be unavailable when $t(s_i,u_j)$ is missing; the unavailable spatial nearest neighbor value is denoted as NA.

Definition 3. *Temporal Nearest Neighbor Value in Time Distance (TTNN): Let $t_{ttnn}(s_i,u_j)$ and $h_{ttnn}(s_i,u_j)$ be the nearest spatial neighbor values of node s_i for the temperature value and the humidity value at time point u_j in time distance, respectively.*

The time distance is calculated by the time point u_j and its near time points in the time window W. The time distance between two time points is denoted by $D_{tg}(\cdot,\cdot)$. It is worth noting that there are important differences between TTNN and SGNN. For the node s_i, SGNN comes directly from the value of one of its actual nearest spatial neighbors at the same time point. However, TTNN comes from the values of its two temporal nearest neighbors, i.e., the values of its two nearest time points on the same node; moreover, different from SGNN, these values are not applied directly; instead, they are used to calculate the final temporal nearest neighbor value by the formula presented below.

Supposed that u' and u'' are the two nearest time points of u_j for node s_i in window W.

Given u_j, we have

$$u' = \{u_x | \forall u_y \in W,\ t(s_i,u_y) \neq NA;\ \exists u_x,\ t(s_i,u_x) \neq NA \text{ that makes } D_{tg}(u_x,u_j) \leq D_{tg}(u_y,u_j)\ \} \quad (17)$$

where $u_x \in W$ and $x \neq y$.

$$u'' = \{u_x | \forall u_y \in W,\ t(s_i,u_y) \neq NA;\ \exists u_x,\ t(s_i,u_x) \neq NA \text{ that makes } D_{tg}(u',u_j) \leq D_{tg}(u_x,u_j) \leq D_{tg}(u_y,u_j)\ \} \quad (18)$$

where $u_y \in W$ and $x \neq y$, in which, the time distance between two time points is denoted by $D_{tg}(\cdot,\cdot)$.

Then the temporal nearest neighbor value of node s_i for the temperature value in time distance can be computed as

$$t_{ttnn}(s_i,u_j) = \begin{cases} \frac{t(s_i,u')(u''-u_j)+t(s_i,u'')(u_j-u')}{u''-u'} & \exists\, u' \in W, u'' \in W, u' \neq u'' \\ t(s_i,\ u') & \exists\, u' \in W \\ NA & \exists\, u' \in W \end{cases} \quad (19)$$

Similarly, the temporal nearest neighbor value of node s_i for the humidity value in time distance can be computed as

$$h_{ttnn}(s_i,u_j) = \begin{cases} \frac{h(s_i,u')(u''-u_j)+h(s_i,u'')(u_j-u')}{u''-u'} & \exists u' \in W, u'' \in W, u' \neq u'' \\ h(s_i,\ u') & \exists\, u' \in W \\ NA & \exists\, u' \in\ W \end{cases} \quad (20)$$

Definition 4. *Temporal Nearest Neighbor Value in Data Distance (TDNN): Let $t_{tdnn}(s_i, u_j)$ and $h_{tdnn}(s_i, u_j)$ be the nearest spatial neighbor values of node s_i for the temperature value and the humidity value at time point u_j in data distance, respectively.*

Similar to the SDNN, the data distance is applied in TDNN to exploit the relationship between different types of values acquired by nodes, but the difference is that the values used to calculate TDNN are obtained from the same node at different time points. Compared with TTNN, the final value calculated in TDNN is based on the values of its temporal nearest neighbors, i.e., the values at all these time points, rather than the values at two nearest time points used in TTNN. The data distance between two time points in the same window is denoted by $D_{td}(\cdot,\cdot)$, to the temperature value; the data distance of nodes s_i between two time points u_p and u_q can be computed as $D_{td}(u_p, u_q) = \sqrt{(h(s_i, u_p) - h(s_i, u_q))^2} = |h(s_i, u_p) - h(s_i, u_q)|$; likewise, to the humidity value, the data distance between u_p and u_q can be computed as $D_{td}(u_p, u_q) = \sqrt{(t(s_i, u_p) - t(s_i, u_q))^2} = |t(s_i, u_p) - t(s_i, u_q)|$.

Then the temporal nearest neighbor value of node s_i for the temperature value at time point u_j in data distance can be computed as

$$t_{tdnn}(s_i, u_j) = \begin{cases} \sum_{I(i,j)} W_{td}(u_j, u_q) t(s_i, u_q) & h(s_i, u_j) \neq NA \text{ and } \forall u_q \in I(i, j),\ \exists h(s_i, u_q) \neq NA \\ NA & h(s_i, u_j) = NA \\ NA & \forall u_q \in I(i, j),\ h(s_i, u_q) = NA \end{cases} \tag{21}$$

Likewise, the temporal nearest neighbor value of node s_i for the humidity value at time point u_j in data distance can be computed as

$$h_{tdnn}(s_i, u_j) = \begin{cases} \sum_{I(i,j)} W_{td}(u_j, u_q) h(s_i, u_q) & t(s_i, u_j) \neq NA \text{ and } \forall u_q \in I(i, j),\ \exists t(s_i, u_q) \neq NA \\ NA & t(s_i, u_j) = NA \\ NA & \forall u_q \in I(i,j), t(s_i, u_q) = NA \end{cases} \tag{22}$$

where $u_q \in I(i, j)$, and the weight W_{td} can be computed as

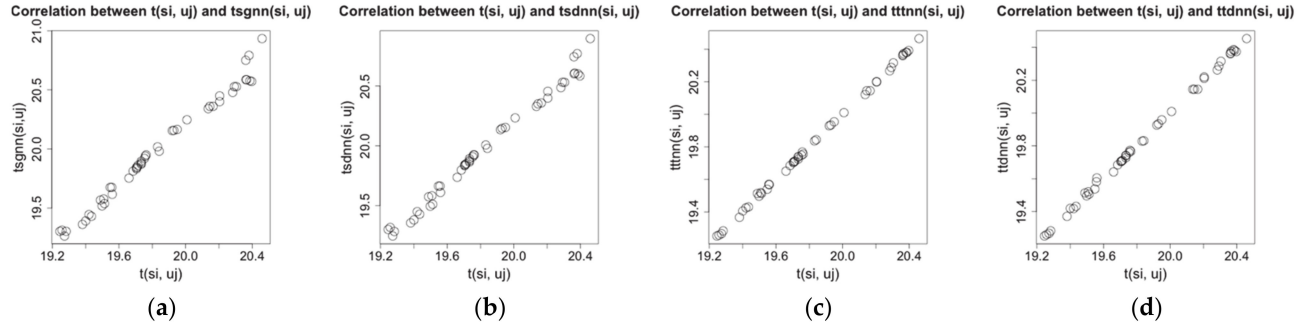
$$W_{td}(u_j, u_q) = \frac{K(D_{td}(u_j, u_q))}{\sum_{I(i,j)} K(D_{td}(u_j, u_q))} \tag{23}$$

where $K(\cdot)$ is a kernel function, for example, Gaussian.

Similar as the spatial nearest neighbor values in data distance, if $h(s_i, u_j)$ or $t(s_i, u_j)$ is missing, $t_{tdnn}(s_i, u_j)$ or $h_{tdnn}(s_i, u_j)$ will be unavailable, denoted by NA.

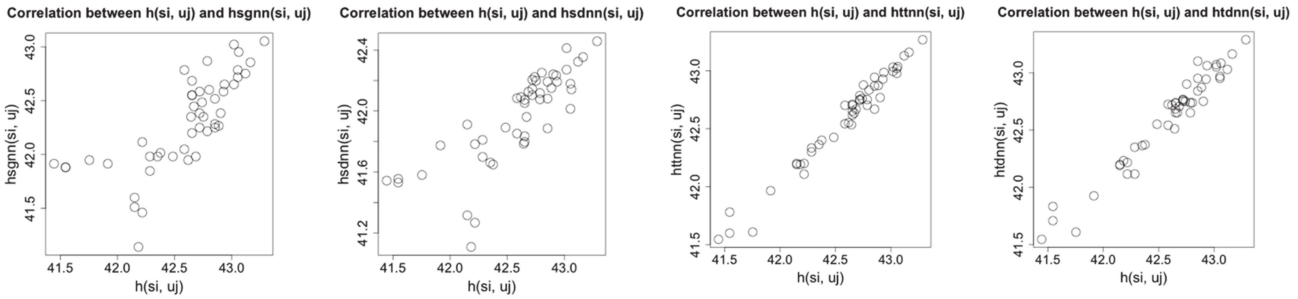
3.2.2. Correlations between a Node's Raw Value and Its Spatial (Temporal) Nearest Neighbor Values

The node s_i gets the temperature value $t(s_i, u_j)$ and the humidity value $h(s_i, u_j)$ at the time point u_j on its location in the physical space. Here $t(s_i, u_j)$ and $h(s_i, u_j)$ are raw values. By data analysis, we find that the raw values of a node have strong relationship with their spatial (temporal) nearest neighbor values. For example, on the experimental dataset of Intel Lab, after choosing a time window W and a node s_i randomly, we calculate spatial (temporal) nearest neighbor values for the node and compare them with the raw values $t(s_i, u_j)$ and $h(s_i, u_j)$; the results are shown as Figures 1 and 2.



(**a**) (**b**) (**c**) (**d**)

Figure 1. Correlation between raw temperature values and spatial (temporal) nearest neighbor values on Intel Lab dataset. (**a**) $t\left(s_i, u_j\right)$ and $t_{sgnn}\left(s_i, u_j\right)$. (**b**) $t\left(s_i, u_j\right)$ and $t_{sdnn}\left(s_i, u_j\right)$. (**c**) $t\left(s_i, u_j\right)$ and $t_{ttnn}\left(s_i, u_j\right)$. (**d**) $t\left(s_i, u_j\right)$ and $t_{tdnn}\left(s_i, u_j\right)$.



(**a**) (**b**) (**c**) (**d**)

Figure 2. Correlation between raw humidity values and spatial (temporal) nearest neighbor values on Intel Lab dataset. (**a**) $h\left(s_i, u_j\right)$ and $h_{sgnn}\left(s_i, u_j\right)$. (**b**) $h\left(s_i, u_j\right)$ and $h_{sdnn}\left(s_i, u_j\right)$. (**c**) $h\left(s_i, u_j\right)$ and $h_{ttnn}\left(s_i, u_j\right)$. (**d**) $h\left(s_i, u_j\right)$ and $h_{tdnn}\left(s_i, u_j\right)$.

The Figures 1 and 2 show that there are strong linear correlations between the raw values of a sensor node and their spatial (temporal) nearest neighbor values. Among them, the correlation between the raw values and temporal nearest neighbor values is higher than that between the raw values and spatial nearest neighbor values, but when we increase the sampling interval for the test data, the data density of the test dataset is decreased, and the correlation between the raw values and temporal nearest neighbor values becomes lower. In addition, the correlations on the humidity data are lower than those on the temperature data because the humidity has been affected by more complex environmental factors than the temperature. On another experimental dataset, the GreenOrbs [18], we can get the similar results on the correlations between raw values and spatial (temporal) nearest neighbor values. That means these correlations and their patterns of change, together with linear regression tool, can be applied to make estimation for missing values for a node. It is a new way to design the imputation algorithm.

3.3. TSNN Imputation Algorithm

3.3.1. The Methods to Calculate the Imputation Value

Here we consider the temperature value $t\left(s_i, u_j\right)$ measured by node s_i at time point u_j; it has four different nearest neighbor values, $t_{sgnn}\left(s_i, u_j\right)$, $t_{sdnn}\left(s_i, u_j\right)$, $t_{ttnn}\left(s_i, u_j\right)$ and $t_{tdnn}\left(s_i, u_j\right)$; the first two values come from spatial nearest neighbors in geometrical distance and in data distance, respectively, and the last two values come from the temporal nearest neighbors in time distance and in data distance, respectively.

Then the $t(s_i, u_j)$ can be described in four different equations as

$$\begin{cases} t(s_i, u_j) &= \alpha_{s_i}(1) + \beta_{s_i}(1) t_{sgnn}(s_i, u_j) + noise(1) \\ t(s_i, u_j) &= \alpha_{s_i}(2) + \beta_{s_i}(2) t_{sdnn}(s_i, u_j) + noise(2) \\ t(s_i, u_j) &= \alpha_{s_i}(3) + \beta_{s_i}(3) t_{ttnn}(s_i, u_j) + noise(3) \\ t(s_i, u_j) &= \alpha_{s_i}(4) + \beta_{s_i}(4) t_{tdnn}(s_i, u_j) + noise(4) \end{cases} \tag{24}$$

As in the discussion in Section 3.2.2, we can use linear regression to make estimation for missing values of a node. Supposed a temperature value measured by node s_i at time point u_{miss} is missing, $u_{miss} \in U_{miss}$, $U_{miss} \subset W$. We can find four estimated values for it, denoted by $t(s_i, \hat{u}_{miss})$ (1), $t(s_i, \hat{u}_{miss})$ (2), $t(s_i, \hat{u}_{miss})$ (3) and $t(s_i, \hat{u}_{miss})$ (4). They can be computed as

$$\begin{cases} t(s_i, \hat{u}_{miss})\ (1) &= \alpha_{s_i}(1) + \beta_{s_i}(1) t_{sgnn}(s_i, u_{miss}) \\ t(s_i, \hat{u}_{miss})\ (2) &= \alpha_{s_i}(2) + \beta_{s_i}(2) t_{sdnn}(s_i, u_{miss}) \\ t(s_i, \hat{u}_{miss})\ (3) &= \alpha_{s_i}(3) + \beta_{s_i}(3) t_{ttnn}(s_i, u_{miss}) \\ t(s_i, \hat{u}_{miss})\ (4) &= \alpha_{s_i}(4) + \beta_{s_i}(4) t_{tdnn}(s_i, u_{miss}) \end{cases} \tag{25}$$

Applying the spatial nearest neighbor value of node s_i at time point u_{miss} in geometrical distance, we have the estimated value, denoted by $t(s_i, \hat{u}_{miss})$ (1); the model is

$$t(s_i, \hat{u}_{miss})\ (1) = \alpha_{s_i}(1) + \beta_{s_i}(1) t_{sgnn}(s_i, u_{miss}) \tag{26}$$

$\alpha_{s_i}(1)$ and $\beta_{s_i}(1)$ can be computed by minimized residual sum of squares as

$$\alpha_{s_i}(1), \beta_{s_i}(1) = \underset{\alpha_{s_i}'(1), \beta_{s_i}'(1)}{\operatorname{argmin}} \sum_{U_{sample}} \left(t\left(s_i, u_{sample}\right)(1) - \alpha_{s_i}'(1) - \beta_{s_i}'(1) t_{sgnn}\left(s_i, u_{sample}\right) \right)^2 \tag{27}$$

where u_{sample} is the sample time points from non-missing data in the window W, $U_{sample} \subset W$, $U_{sample} \cap U_{miss} = \phi$, $u_{sample} \in U_{sample}$.

Meanwhile, the linear relationship between $t\left(s_i, u_{sample}\right)(1)$ and $t\left(s_i, \hat{u_{sample}}\right)(1)$ can be measured by R^2 as

$$R_{si}^2(1) = \frac{\sum_{U_{sample}} \left(t\left(s_i, \hat{u_{sample}}\right)(1) - \overline{t\left(s_i, u_{sample}\right)(1)} \right)^2}{\sum_{U_{sample}} \left(t\left(s_i, u_{sample}\right)(1) - \overline{t\left(s_i, u_{sample}\right)(1)} \right)^2} \tag{28}$$

Similarly, applying the spatial nearest neighbor value of node s_i at time point u_{miss} in data distance, we have another estimated value, denoted by $t(s_i, \hat{u}_{miss})$ (2), the model is

$$t(s_i, \hat{u}_{miss})(2) = \alpha_{s_i}(2) + \beta_{s_i}(2) t_{sdnn}(s_i, u_{miss}) \tag{29}$$

where

$$\alpha_{s_i}(2), \beta_{s_i}(2) = \underset{\alpha_{s_i}'(2), \beta_{s_i}'(2)}{\operatorname{argmin}} \sum_{U_{sample}} \left(t\left(s_i, u_{sample}\right)(1) - \alpha_{s_i}'(2) - \beta_{s_i}'(2) t_{sdnn}\left(s_i, u_{sample}\right) \right)^2 \tag{30}$$

where $U_{sample} \subset W$, $U_{sample} \cap U_{miss} = \phi$, $u_{sample} \in U_{sample}$.

The linear relationship between $t_{s_i, u_{sample}}(2)$ and $t_{s_i, \hat{u_{sample}}}(2)$ can be measured by R^2 as

$$R_{si}^2(2) = \frac{\sum_{U_{sample}} \left(t\left(s_i, \hat{u_{sample}}\right)(2) - \overline{t\left(s_i, u_{sample}\right)(2)} \right)^2}{\sum_{U_{sample}} \left(t\left(s_i, u_{sample}\right)(2) - \overline{t\left(s_i, u_{sample}\right)(2)} \right)^2} \tag{31}$$

The estimated value using spatial nearest neighbor values of node s_i at time point u_{miss}, denoted by $t(s_i, \hat{u}_{miss})\ (S)$, can be computed as

$$t(s_i, \hat{u}_{miss})(S) = \begin{cases} \frac{R^2_{si}(1)}{R^2_{si}(1)+R^2_{si}(2)} t(s_i, \hat{u}_{miss})(1) + \frac{R^2_{si}(2)}{R^2_{si}(1)+R^2_{si}(2)} t(s_i, \hat{u}_{miss})(2) & t_{sdnn}(s_i, u_{miss}) \neq NA, t_{sdnn}(s_i, u_{miss}) \neq NA \\ t(s_i, \hat{u}_{miss})(S) = t(s_i, \hat{u}_{miss})(1) & t_{sdnn}(s_i, u_{miss}) = NA \\ NA & t_{sdnn}(s_i, u_{miss}) = NA \end{cases} \quad (32)$$

Following the similar steps, the estimated value using temporal nearest neighbor values of node s_i at time point u_{miss}, denoted by $t(s_i, \hat{u}_{miss})\ (T)$, can be computed as

$$t(s_i, \hat{u}_{miss})(T) = \begin{cases} \frac{R^2_{si}(3)}{R^2_{si}(3)+R^2_{si}(4)} t(s_i, \hat{u}_{miss})(3) + \frac{R^2_{si}(4)}{R^2_{si}(3)+R^2_{si}(4)} t(s_i, \hat{u}_{miss})(4) & t_{sdnn}(s_i, u_{miss}) \neq NA, t_{sdnn}(s_i, u_{miss}) \neq NA \\ t(s_i, \hat{u}_{miss})(S) = t(s_i, \hat{u}_{miss})(3) & t_{sdnn}(s_i, u_{miss}) = NA \\ NA & t_{sdnn}(s_i, u_{miss}) = NA \end{cases} \quad (33)$$

The final estimated temperature value for node s_i at time point u_{miss} is denoted by $t(s_i, \hat{u}_{miss})$, which can be computed based on the two values: $t(s_i, \hat{u}_{miss})\ (S)$ and $t(s_i, \hat{u}_{miss})\ (T)$. The contributions of the two values are different to calculate the final estimation for the node s_i at time point u_{miss}, which are determined by the physical environment factors in the time window W of the space P, where the node s_i owns its spatial and temporal nearest neighbors. In fact, the experiment results demonstrate that the contribution ratio basically keeps unchanged in the observed time window. Therefore, a spatial–temporal coefficient, denoted by λ, can be used to describe the contribution ratio of the $t(s_i, \hat{u}_{miss})\ (S)$ and $t(s_i, \hat{u}_{miss})\ (T)$, which will be discussed in the next section.

The final estimated temperature value for node s_i at time point u_{miss}, denoted by $t(s_i, \hat{u}_{miss})$ can be computed as

$$t(s_i, \hat{u}_{miss}) = \begin{cases} \lambda t(s_i, \hat{u}_{miss})(S) + (1-\lambda) t(s_i, \hat{u}_{miss})(T) & t(s_i, \hat{u}_{miss})(S) \neq NA, t(s_i, \hat{u}_{miss})(T) \neq NA \\ t(s_i, \hat{u}_{miss})(S) & t(s_i, \hat{u}_{miss})(S) \neq NA, t(s_i, \hat{u}_{miss})(T) = NA \\ t(s_i, \hat{u}_{miss})(T) & t(s_i, \hat{u}_{miss})(S) \neq NA, t(s_i, \hat{u}_{miss})(T) \neq NA \\ NA & t(s_i, \hat{u}_{miss})(S) = NA \end{cases} \quad (34)$$

Then, $t(s_i, \hat{u}_{miss})$ will be used as the estimated value for $t(s_i, u_{miss})$.

The implementation of TSNN is shown in Algorithm 1.

3.3.2. The Spatial–Temporal Coefficient λ

In the observed time window W, the value measured by node s_i at time point u_{miss} is missing, $u_{miss} \in U_{miss}$, $U_{miss} \subset W$. The experiment results demonstrate that the contribution ratio of the $t(s_i, \hat{u}_{miss})\ (S)$ and $t(s_i, \hat{u}_{miss})\ (T)$ basically keeps unchanged in the time window. Therefore, the spatial–temporal coefficient λ in time window W can be obtained based on the non-missing test data in the window. We put non-missing test data into different new assistant window V_i, all assistant windows form a windows group, denoted by VG, $VG = \{V_1, V_2, \ldots, V_{r-1}, V_r\}$, $V_i \in VG$, $V_i = \{v_1, v_2, \ldots, v_{m-1}, v_m\}$, $V_i \subset W$, $V_i \cap U_{miss} = \varnothing$. Then we can generate a dataset with hypothetical random missing values based on time window V_i. For each raw value $t(s_i, v_{miss})$ marked as a missing value in the dataset, we will calculate its estimated values as the $t(s_i, \hat{v}_{miss})\ (S)$ and $t(s_i, \hat{v}_{miss})\ (T)$ using Equations (32) and (33). In the next step, for all missing values in time window V_i, using the equation 40, we can calculate the spatial and temporal RMSE values for the given window V_i, which is denoted by $RMSE(S, V_i)$ and $RMSE(T, V_i)$, respectively.

Algorithm 1: TSNN algorithm.

Input: sensor node s_i, $s_i \in S$; u_{miss}, $u_{miss} \in W$; temperature dataset ; $T(S,W)$, humidity dataset $H(S,W)$,

$S = \{s_1, s_2, \ldots, s_{n-1}, s_n\}$, $W = \{u_1, u_2, \ldots, u_{m-1}, u_m\}$ initial number of spatial nearest neighbors k_s;

initial number of temporal nearest neighbors k_t; the spatial—temporal coefficient λ.

Output: $t(s_i, \hat{u}_{miss})$.

1. Get k_{sb} and k_{tb}
2. $k_s \leftarrow k_{sb}$, $k_t \leftarrow k_{tb}$
3. $T'(S,W'), H'(S,W') \leftarrow$ *all elements in* $T(S,W), H(S,W)$ which do not contain missing values for node s_i
4. $T''(S,W''), H''(S,W'') \leftarrow$ *time sampling from* $T'(S,W'), H'(S,W')$
5. $m'' \leftarrow |W''|$
6. $A \leftarrow \varnothing$, $B \leftarrow \varnothing$, $C \leftarrow \varnothing$, $D \leftarrow \varnothing$, $E \leftarrow \varnothing$
7. **for** j from 1 to m'' do
8. On On $T''(S,W'')$ *and* $H''(S,W'')$, get $t_{sgnn}\left(s_i, u_j\right)$, $t_{sdnn}\left(s_i, u_j\right)$, $t_{ttnn}\left(s_i, u_j\right)$ and $t_{tdnn}\left(s_i, u_j\right)$
9. $A \leftarrow A + \left\{t\left(s_i, u_j\right)\right\}$
10. **if** $t_{sgnn}\left(s_i, u_j\right) \neq NA$ **then**
11. $B \leftarrow B + \left\{t_{sgnn}\left(s_i, u_j\right)\right\}$
12. **else if** $t_{sdnn}\left(s_i, u_j\right) \neq NA$ **then**
13. $C \leftarrow C + \left\{t_{sdnn}\left(s_i, u_j\right)\right\}$
14. **else if** $t_{ttnn}\left(s_i, u_j\right) \neq NA$ **then**
15. $D \leftarrow D + \left\{t_{ttnn}\left(s_i, u_j\right)\right\}$
16. **else if** $t_{tdnn}\left(s_i, u_j\right) \neq NA$ **then**
17. $E \leftarrow E + \left\{t_{tdnn}\left(s_i, u_j\right)\right\}$
18. **end if**
19. **end for**
20. On $T(S,W)$ *and* $H(S,W)$, get $t_{sgnn}(s_i, u_{miss})$, $t_{sdnn}(s_i, u_{miss})$, $t_{ttnn}(s_i, u_{miss})$ and $t_{tdnn}(s_i, u_{miss})$
21. **if** $t_{sgnn}(s_i, u_{miss}) = NA$ and $t_{ttnn}(s_i, u_{miss}) = NA$ **then**
22. $t(s_i, \hat{u}_{miss}) \leftarrow NA$
23. **return** $t(s_i, \hat{u}_{miss})$
24. **else if** $t_{sgnn}(s_i, u_{miss}) \neq NA$ **then**
25. Construct the estimation equation $E(1)(s_i, u_{miss})$
26. Using A and B, regress the coefficients $\alpha_{s_i}(1), \beta_{s_i}(1)$ of $E(1)(s_i, u_{miss})$ and compute $R^2_{si}(1)$
27. **else if** $t_{sdnn}(s_i, u_{miss}) \neq NA$ **then**
28. Construct the estimation equation $E(2)(s_i, u_{miss})$
29. Using A and C, regress the coefficients $\alpha_{s_i}(2), \beta_{s_i}(2)$ of $E(2)(s_i, u_{miss})$ and compute $R^2_{si}(2)$
30. **else if** $t_{ttnn}(s_i, u_{miss}) \neq NA$ **then**
31. Construct the estimation equation $E(3)(s_i, u_{miss})$
32. Using A *and* D, regress the coefficients $\alpha_{s_i}(3), \beta_{s_i}(3)$ of $E(3)(s_i, u_{miss})$ and compute $R^2_{si}(3)$
33. **else if** $t_{tdnn}(s_i, u_{miss}) \neq NA$ **then**
34. Construct the estimation equation $E(4)(s_i, u_{miss})$
35. Using A and E, regress the coefficients $\alpha_{s_i}(4), \beta_{s_i}(4)$ of $E(4)(s_i, u_{miss})$ and compute $R^2_{si}(4)$
36. **end if**
37. Compute the $t(s_i, \hat{u}_{miss})$ (1), $t(s_i, \hat{u}_{miss})(2)$ *to* get $t(s_i, \hat{u}_{miss})$ (S)
38. Compute the $t(s_i, \hat{u}_{miss})$ (3), $t(s_i, \hat{u}_{miss})(4)$ to get $t(s_i, \hat{u}_{miss})$ (T)
39. Compute the $t(s_i, \hat{u}_{miss})$ using $t(s_i, \hat{u}_{miss})$ (S), $t(s_i, \hat{u}_{miss})$ (T) and λ
40. **return** $t(s_i, \hat{u}_{miss})$

Then the spatial–temporal coefficient λ can be computed as

$$\lambda = \frac{\sum_{VG} N(S, V_i)}{\sum_{VG}(N(S, V_i) + N(T, V_i))} \tag{35}$$

where

$$N(S, V_i) = \begin{cases} 1 & if\ RMSE(S, V_i) \leq RMSE(T, V_i) \\ 0 & if\ RMSE(S, V_i) > RMSE(T, V_i) \end{cases} \tag{36}$$

$$N(T, V_i) = \begin{cases} 1 & if\ RMSE(T, V_i) \leq RMSE(S, V_i) \\ 0 & if\ RMSE(T, V_i) > RMSE(S, V_i) \end{cases} \tag{37}$$

where $V_i \in VG,\ VG = \{V_1, V_2, \ldots, V_{r-1}, V_r\}$.

3.3.3. The Best k_s for Spatial Nearest Neighbors and k_t for Temporal Nearest Neighbors

As in discussion in Section 3.3.1, before computing the final estimated value the $t(s_i, \hat{u}_{miss})$, we must get the spatial nearest neighbor value in data distance $t_{sdnn}(s_i, u_{miss})$ and temporal nearest neighbor value in data distance $t_{tdnn}(s_i, u_{miss})$. The two values can be worked out based on the values of their neighbors at the time point, respectively, that is, k_s spatial nearest neighbors in $G(i, miss)$ and k_t temporal nearest neighbors in $I(i, miss)$. The experimental results show that different k_s and k_t will have different effects to the deviation between $t_{sdnn}(s_i, u_j)$, $t_{tdnn}(s_i, u_j)$ and the real value $t(s_i, u_j)$; meanwhile, there exist the best k_s and k_t which make the least deviations. Basically, in an observed time window with missing values at some time points, the relationships between the real value $t(s_i, u_j)$ and its spatial nearest neighbor value $t_{sdnn}(s_i, u_j)$ or its temporal nearest neighbor value $t_{tdnn}(s_i, u_j)$ keep stable; therefore, the best k_s and k_t obtained by the samples from non-missing values will still make the least deviation when they are applied to calculate neighbor values $t_{sdnn}(s_i, u_{miss})$ and $t_{tdnn}(s_i, u_{miss})$ for the missing value.

In the observed time window W, let $T\left(s_i, U_{sample}\right)$ be the vector composed of $t\left(s_i, u_{sample}\right)$, $T_{sdnn}\left(s_i, k_s', U_{sample}\right)$ be the vector composed of $t_{sdnn}\left(s_i, k_s', u_{sample}\right)$ which is computed based on k_s' and $T_{tdnn}\left(s_i, k_t', U_{sample}\right)$ be the vector composed of $t_{tdnn}\left(s_i, k_t', u_{sample}\right)$, which is computed based on k_t'. Given the initial k_s and k_t, the best k_s and k_t can be computed as

$$k_{sb} = \underset{k_s'}{\operatorname{argmax}} Pearson\left(T_{sdnn}\left(s_i, k_s', U_{sample}\right), T\left(s_i, U_{sample}\right)\right) \tag{38}$$

$$k_{tb} = \underset{k_t'}{\operatorname{argmax}} Pearson\left(T_{tdnn}\left(s_i, k_t', U_{sample}\right), T\left(s_i, U_{sample}\right)\right) \tag{39}$$

where $Pearson(\cdot)$ is the function used to calculate the Pearson relationship coefficient between two vectors.

4. Experimental Results

4.1. Evaluation Platform Used in Experiments

To make the evaluation for our algorithm, all experiments are running on a laptop with Intel Core i7 2.9 GHz CPU and 16 GB RAM. The codes are implemented by R language.

4.2. Evaluation Methods Used in Experiments

Two measuring sticks have been applied for evaluating the experimental results. The first one is the root mean square error (RMSE), which will be used to measure the accuracy of the estimation for missing values. The other one is the percentage of cases in which a mission value can be estimated (PCE), which will be used to measure the applicability of the imputation algorithm in a real application.

The RMSE is computed as

$$RMSE = \sqrt{\frac{\sum(real\ value\ -\ estimated\ value)^2}{the\ numer\ of\ estimations}} \tag{40}$$

The PCE is computed as:

$$PCE(\%) = \frac{the\ number\ of\ cases\ that\ a\ missing\ value\ can\ be\ estimated}{the\ total\ number\ of\ attempts\ to\ estimate\ a\ missing\ value} * 100\% \tag{41}$$

4.3. Experimental Datasets

We test our TSNN algorithm and other algorithms based on two real world datasets: the Intel Lab dataset and the GreenOrbs dataset. They represent two kinds of typical real wireless network: the indoor network with unchanged common spatial neighbors and the outdoor network with changing common spatial neighbors. The nodes deployed in the two different WSNs have similar structures that each node is equipped with more than one sensor for obtaining different physical quantities at their locations. In addition, all the nodes are working continuously in the real environment. The datasets provide an ideal platform for experiments where our algorithm can obtain and exploit the information hidden in the data. Moreover, the two WSNs have different topological structures for the node to build its neighbors, which make the node in the different datasets have fixed neighbors and unfixed neighbors, respectively, and this change can be adapted by our algorithm and does not degrade its performance.

4.3.1. Intel Lab Dataset

Intel Lab dataset contains indoor data collected from 54 sensor nodes deployed in the Intel Berkeley Research lab. Each sensor node will get the data including temperature, humidity, light and voltage values once every 31 s. To the data we collected using the in-network query processing system, a monotonically increasing sequence number is applied to make sure the data processing center can get the data from different nodes at the same time point. We only use the temperature and humidity data in our experiments. In the raw dataset, data at some time points on some nodes have not been recorded. When we evaluate applicability of imputation algorithms using PCE and the accuracy of imputation algorithms using RMSE on the real-world dataset, all data in the raw dataset are used directly; in other words, the raw dataset is used as the experimental dataset.

In this paper, we consider the situation that the test node has missing values at time points randomly selected in the time window. Therefore, in the following experiments, some data of the test node have been marked as dummy missing values (not available, denoted by NA) following the Missing Completely at Random (MCAR) [19]. In real situations, the data coming from the temperature sensor and the humidity sensor on the same node will have three possible scenarios: the temperature (humidity) data are missing but the humidity (temperature) data are available at all time points; the temperature (humidity) data are missing but the humidity (temperature) data are available at some of the points in certain probability, and the temperature and humidity data are both missing at all time points. Accordingly, in the experiments, the three test scenarios have been defined as follows: For the test noted at the time point, humidity (temperature) value is not NA when temperature (humidity) value is marked as NA; humidity (temperature) value is NA in certain probability (set as 50% in the experiments) when temperature (humidity) value is marked as NA; humidity value is NA when temperature value is marked as NA. Next, we will apply imputation algorithms to estimate missing temperature (humidity) values and compare them with real values.

In the Intel Lab dataset, the common spatial neighbors will keep unchanged at time points in a time window. As one of the parameters used in algorithm, the initial number of neighbors in the group will be decided by distance range between the test node and its neighbor node as well as whether there is a wall or door between nodes when creating a

neighbor group for the test node. For example, the distance between node 8 and node 54 is 2.83 m, and the distance between node 8 and node 9 is 3.61 m, when the distance range is set as 5 m, the node 9 and node 54 should both be the neighbor nodes of node 8, but the wall between the node 54 and node 8 will make node 54 excluded from the neighbor group of node 8, as Figure 3 shows.

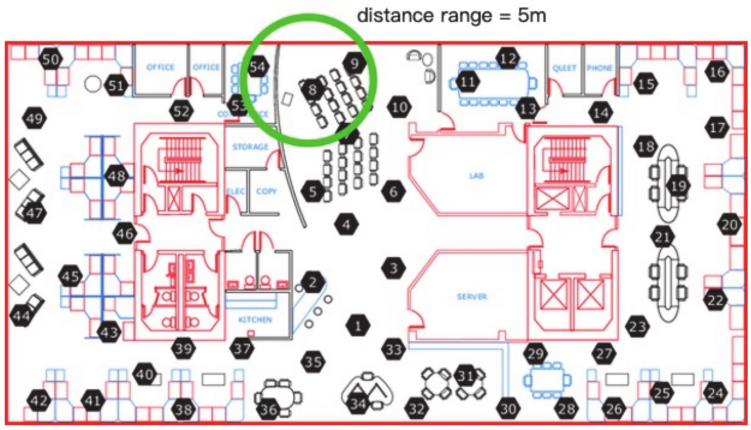


Figure 3. The example of a neighbor group affected by walls in Intel Lab dataset.

4.3.2. GreenOrbs Dataset

GreenOrbs dataset contains outdoor data collected from 120 wireless sensor nodes scattered in the forest, shown in Figure 4. Each sensor node will get the data including temperature, humidity and light values once every 80 to 85 s. In experiments, we only use the temperature and humidity data. Similar as the Intel Lab dataset, the raw GreenOrbs dataset is not integrated; data at some time points from some nodes have not been recorded. Similarly, the raw dataset will be used as the experimental dataset directly when we evaluate applicability of imputation algorithms using PCE and the accuracy of imputation algorithms using RMSE.

Figure 4. Wireless sensor nodes in GreenOrbs dataset.

Similar as experiment on Intel Lab dataset, some data of the test nodes have been marked as dummy missing values following MCAR in the three test scenarios before testing different imputation algorithms.

In the GreenOrbs dataset, different from the Intel Lab dataset, the neighbor group of a node will change in a time window because the neighbors are obtained according to the sensed RSSI values [20]. For example, the neighbors group of node 3 is {51, 7,2, 59, 87, 70, 50, 75, 4, 37, 95}, {72, 29, 70, 50, 120, 75, 113, 4, 37, 95} and {29, 70, 50, 75, 113, 4, 6, 84} at three continuous time points in the time window. Therefore the common spatial neighbors at these time points are {70, 50, 75, 4}.

4.4. Evaluation of PCE

On the experimental dataset of Intel Lab, we choose a time window randomly, in which some of the temperature measuring data are supposed to be missing. Then, we randomly choose 20% of all nodes as test nodes; next, a missing percentage is applied to mark some of measuring data as lost status using MCAR method for one of the nodes. Because over 50% of data missing is risky for observed values, and imputation should not be used in this situation [21], the upper bound of missing percentage is set as 50% in our experiments for both PCE and RMSE evaluations. Then, on the three test scenarios, we apply different imputation algorithms to make the estimation for the missing values, each one as a case. The experiment has been repeated for each note, and the PCE can be calculated. Finally, for different imputation algorithms, we will work out the average PCE for all test nodes on each of missing percentages from 5% to 50%. The experimental results are shown in Figure 5.

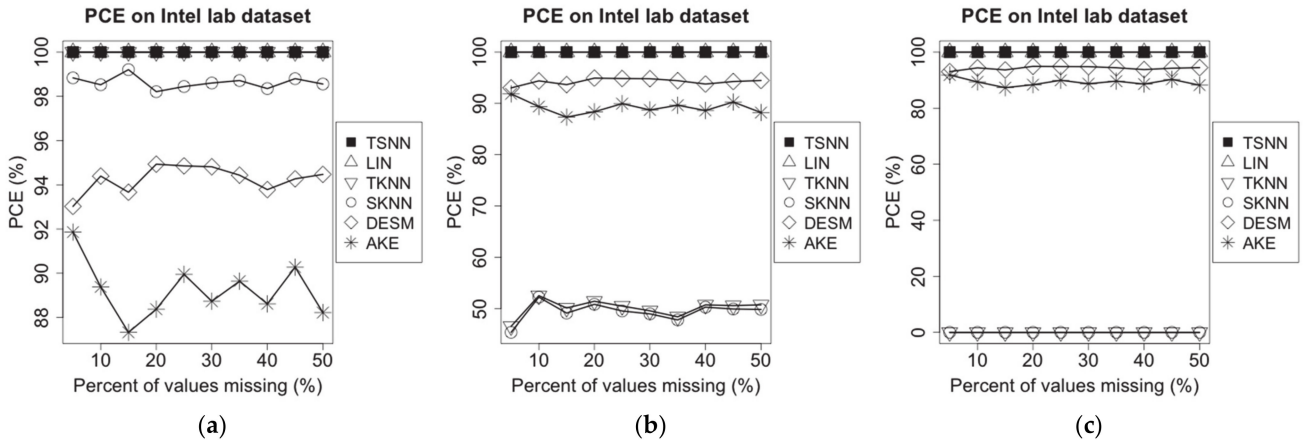


Figure 5. PCE of different algorithms on Intel lab dataset for the three scenarios: (**a**) temperature NA and humidity not NA, (**b**) temperature NA and humidity 50% NA and (**c**) temperature NA and humidity NA.

On GreenOrbs dataset, the same experiments have been made, and the results are shown in Figure 6.

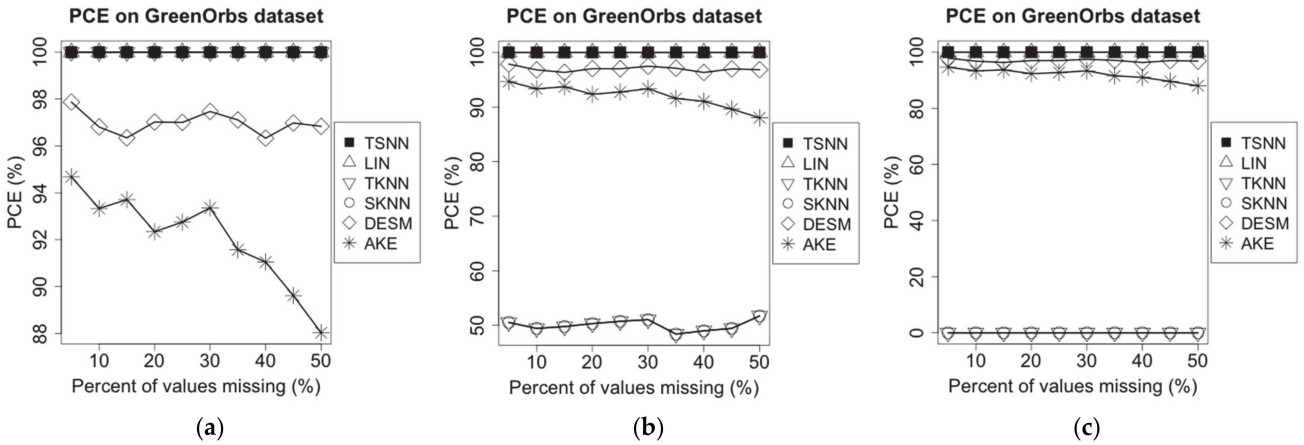


Figure 6. PCE of different algorithms on GreenOrbs dataset for the three scenarios: (**a**) temperature NA and humidity not NA, (**b**) temperature NA and humidity 50% NA and (**c**) temperature NA and humidity NA.

Figure 5 shows that TSNN and LIN are able to make imputation for all cases on all the three scenarios because LIN only uses the temporal neighbor values, and TSNN only uses the temporal nearest neighbor value in time distance when other three values are

unavailable. In other words, as long as there are at least two neighbor values in the time window, the preconditions of TSNN and LIN can be satisfied, and these two algorithms can work correctly. Actually, in our experiments, even if the percentage of values missing reaches the upper bound of 50%, the node still owns more than two temporal neighbor values in the time window; therefore, TSNN and LIN can make imputation in all cases and are not affected by the percentage of values missing. However, TKNN, SKNN, AKE and DESM cannot make imputation for some of the scenarios or some of the cases. TKNN is able to make imputation for all cases in scenario (a), but its PCE drops down to about 50% in scenario (b) and becomes unavailable in scenario (c) because the temporal nearest neighbor value in data distance on which it relied decreases in scenario (b) until unavailable in scenario (c). SKNN, AKE and DESM all need non-missing values coming from spatial neighbors of the test node in the test time window. However, on Intel lab dataset, for the whole test window, almost all nodes have some time points where the raw values are not recorded. In other words, although spatial neighbors existed throughout the entire time window, there are not raw values recorded on some of the time points. SKNN, DESM and AKE require at least one neighbor with non-missing values; besides, for the test note, DESM and AKE require at least one common spatial neighbor with non-missing values on at least two and three time points in the window; in scenarios (a), (b) and (c), both DESM and AKE have the same performance. In scenario (a), SKNN has better performance than DESM and AKE, but similar to TKNN; its PCE drops down over 50% in scenarios (b) and becomes unavailable in scenario (c) because the spatial nearest neighbor value in data distance on which it relied decreases in scenario (b) until unavailable in scenario (c).

Because the time points without recorded values are spread mainly evenly in the raw dataset, all algorithms have slightly changed as the percent of missing values increases.

On the experimental dataset of GreenOrbs, the situation is different from Intel lab dataset because the common spatial neighbors for a test node may be unavailable on a test window due to the neighbors being obtained according to the sensed RSSI values. It may cause one or more common spatial neighbors not to be available at time points selected randomly on the test window, which will make DESM or AKE not applicable in this case, and the degree of PCE degradation for AKE is the most prominent as the percent of missing values increases because of the increasing difficulty to get common spatial neighbors among less time points without missing values. However, in all three scenarios, SKNN always can get at least one spatial neighbor, so it has the same performance as TKNN, which always can get at least one temporal neighbor. Figure 6 shows TSNN and LIN are still able to make imputation for all cases on all three scenarios.

Because the applicability of the imputation algorithms is only related to temporal and spatial neighbors, to humidity measurement data on Intel Lab dataset and GreenOrbs dataset, we can obtain the similar experimental results.

The experimental results on two test datasets show that TSNN has the best PCE performance.

4.5. Evaluation of RMSE

On the experimental dataset of Intel Lab dataset, the sampling interval is set to 1min, and we choose a time window randomly, on which some of the temperature measurement data are supposed to be missing. After that, we randomly choose 20% of all nodes as test nodes. Next, a missing percentage is applied to make the MCAR operation for one of the nodes; then, on the three test scenarios, we apply different imputation algorithms to make the estimation for the missing values, each one as a case. The experiment will be repeated for each test note. In experiments, we will compare the performance of TSNN with other imputation algorithms based on the common cases in which a missing value can be estimated by all algorithms, although they have different PCE. Finally, based on the test results of all cases, for different imputation algorithms, we work out the average RMSE for all test nodes on each of missing percentages from 5% to 50%. The experimental results have been shown in Figure 7.

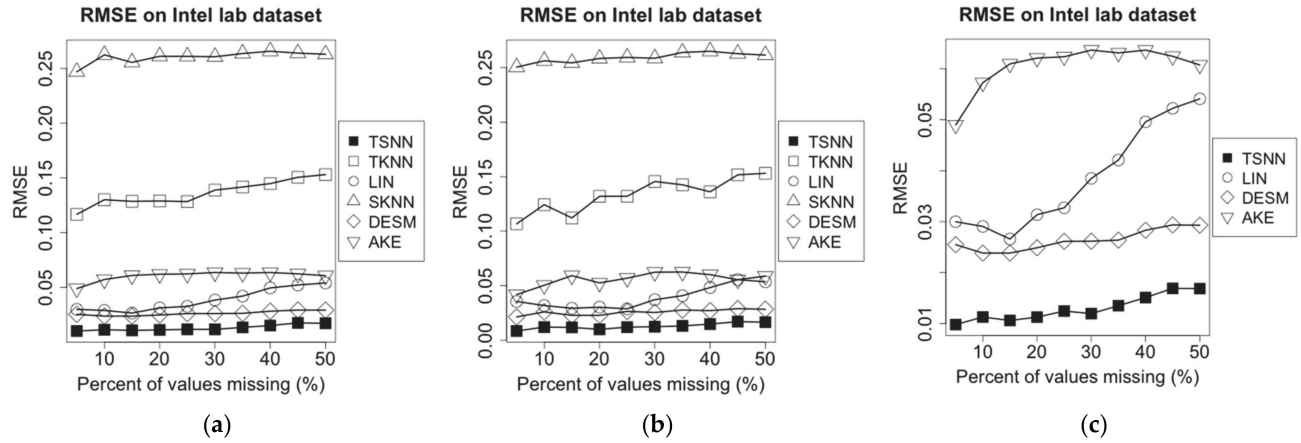


Figure 7. RMSE of imputation algorithms on temperature data of Intel lab dataset for the three scenarios: (**a**) temperature NA and humidity not NA, (**b**) temperature NA and humidity 50% NA and (**c**) temperature NA and humidity NA.

For TSNN algorithm, the spatial–temporal coefficient λ, the best k_s for spatial nearest neighbors and k_t for temporal nearest neighbors will be calculated and applied to the algorithm. For TKNN algorithm, which requires k_t for temporal nearest neighbors, and SKNN, DESM and AKE algorithms, which require k_s for spatial nearest neighbors, they will use their own methods to select the suitable k_t or k_s to make the algorithm get the best performance in the experiment case.

Supposing that the humidity measurement data are missing in the time window, we repeated the same experiments, and the results are shown in Figure 8.

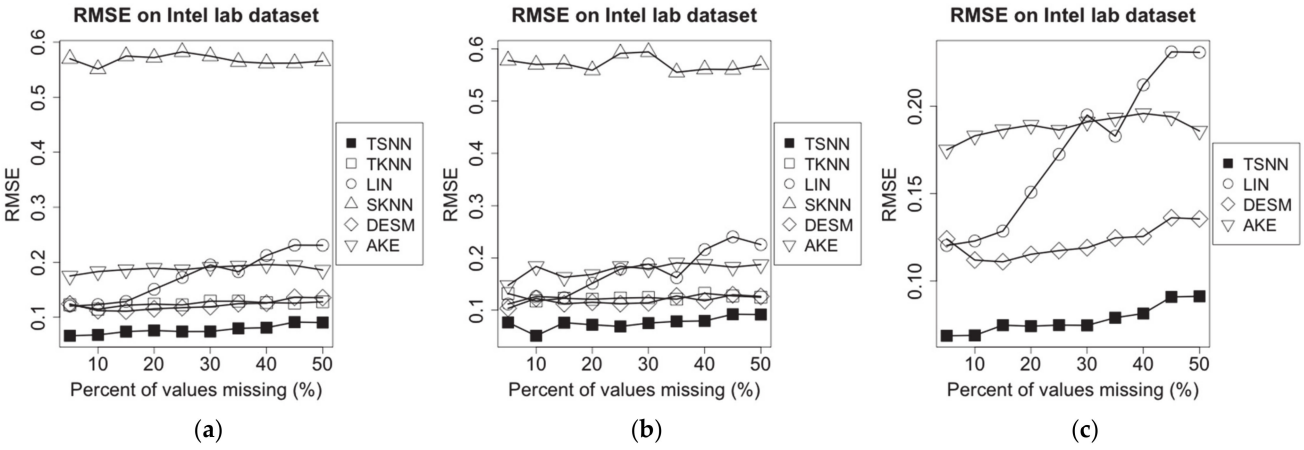


Figure 8. RMSE of imputation algorithms on humidity data of Intel lab dataset for the three scenarios: (**a**) humidity NA and temperature not NA, (**b**) humidity NA and temperature 50% NA and (**c**) humidity NA and temperature NA.

To evaluate the performance of algorithms on different data density, we extract data from the experimental dataset based on different sampling intervals from 1 min to 30 min, to make the new test datasets and repeat the same experiments. To ensure that there is still enough data when we make sampling on the long interval, before making the new test datasets, the data that have not been recorded at some time points on some nodes in the raw dataset will be filled with the average of the non-missing value nearby. We test all algorithms on the most typical scenario: The temperature (humidity) data are missing, but the humidity (temperature) data are available at some of the time points in certain probability which is set as 50% in the experiments. For different imputation algorithms, we work out the average RMSE for all test nodes on all missing percentages from 5% to 50%. The experimental results have been shown in Figure 9.

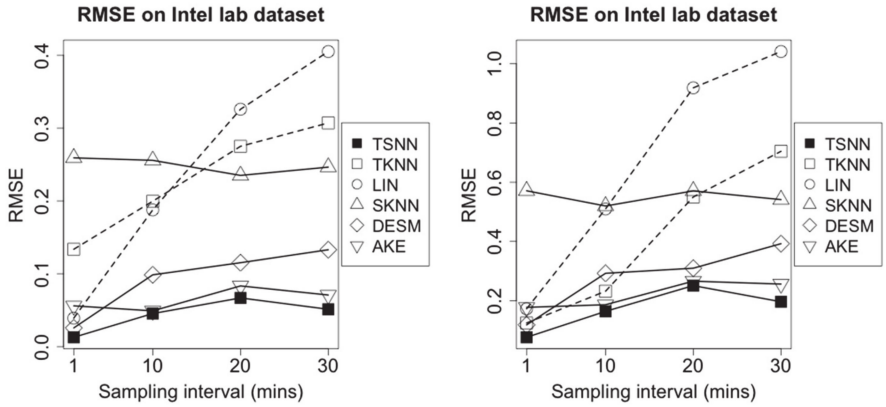


Figure 9. RMSE of imputation algorithms on different data density for Intel lab dataset: (**a**) temperature data and (**b**) humidity data.

Figure 7 shows experimental results for missing temperature measurement data. We find that TKNN and SKNN become unavailable in scenario (c). The RMSE for all imputation algorithms fluctuates as the percent of missing values increases; LIN, TKNN and SKNN do not use spatial or temporal correlation information; LIN has relatively lower RMSE because it benefits from continuity of values at the low sampling interval. AKE and DESM both are based on spatial neighbors' correlation information, so they have lower RMSE. Compared with them, TSNN can use time distance and temporal neighbors' correlation information as well, which make it get the lowest RMSE. Among all algorithms, TSNN has the lowest RMSE.

Figure 8 shows experimental results for missing humidity measuring data. Similarly, it shows that the RMSE for all imputation algorithms fluctuates as the percent of missing values increases, LIN has a notable growing trend because the temporal neighbors will become lesser with the increasing percent of missing values, and the humidity data have more dramatic fluctuation than temperature data. In comparison, using data distance rather than time distance, TKNN gets lower RMSE in this situation. TSNN still has the lowest RMSE because it can use both data distance and the temporal neighbors' correlation information. In addition, RMSE for missing humidity data is higher than that for missing temperature data because the humidity measuring data have more dramatic variety than temperature measuring data, which is due to more affected factors in the physical environment.

Figure 9 demonstrates the RMSE for all imputation algorithms for missing temperature measurement data (a) and humidity measurement data (b) in the different data density. With the increasing sampling interval, among algorithms without correlation information, SKNN has less fluctuate because it has no connection with temporal neighbors, which will be affected by sampling interval significantly, while the RMSE of LIN and TKNN go up sharply because temporal neighbors will become lesser. In contrast, the correlation-based algorithms, AKE, DESM and TSNN, have better performance. AKE has lower RMSE than DESM because of more weight-based spatial neighbors' contributions, but TSNN will obtain more benefits from values of spatial neighbors in geometrical and data distance when the sampling interval is increasing, which makes it have the lowest RMSE among all algorithms.

On GreenOrbs dataset, for the missed temperature measurement data, we set the sampling interval as 1.5 min and missing percentages range as from 5% to 50%; then, we make the same experiments on the three test scenarios, and the results are shown in Figure 10. Similarly, we get experimental results for the missing humidity measurement data shown in Figure 11.

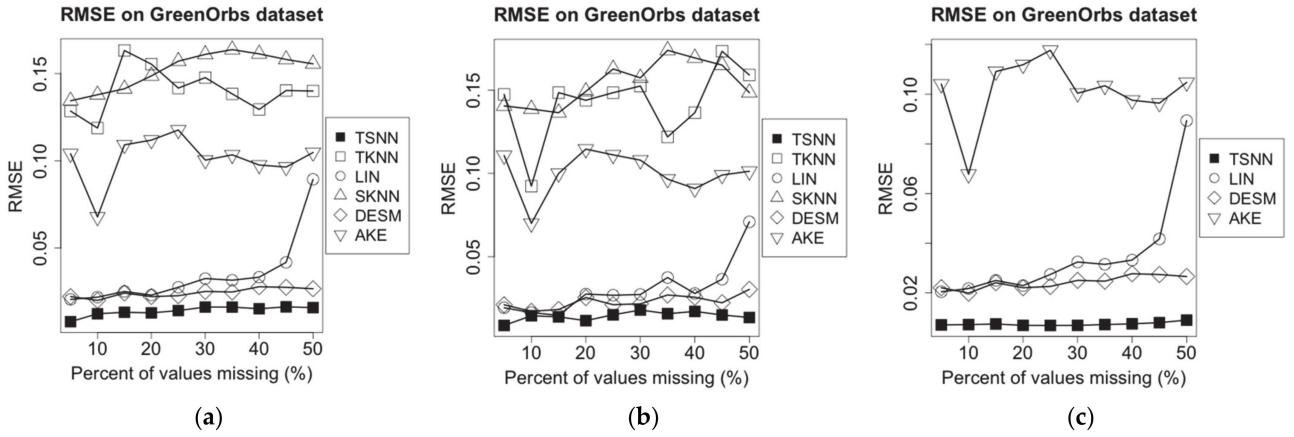


Figure 10. RMSE of imputation algorithms on temperature data of GreenOrbs dataset for the three scenarios: (**a**) temperature NA and humidity not NA, (**b**) temperature NA and humidity 50% NA and (**c**) temperature NA and humidity NA.

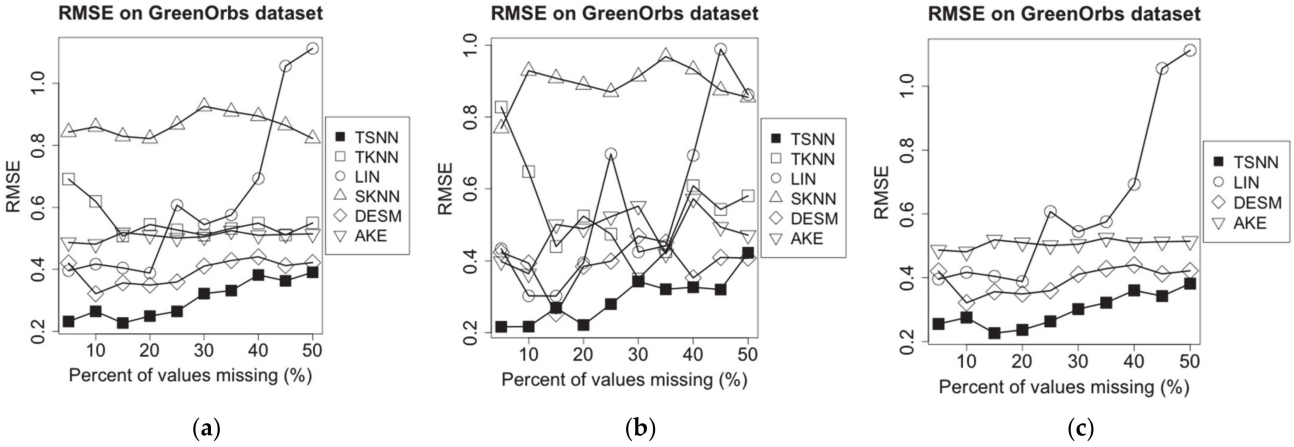


Figure 11. RMSE of imputation algorithms on humidity data of GreenOrbs dataset for the three scenarios: (**a**) temperature NA and humidity not NA, (**b**) temperature NA and humidity 50% NA and (**c**) temperature NA and humidity NA.

Next, we set sampling intervals from 1.5 min to 27 min, to make the new test datasets and repeat the same experiments for evaluating the performance of algorithms on different data density. The results are shown as Figure 12.

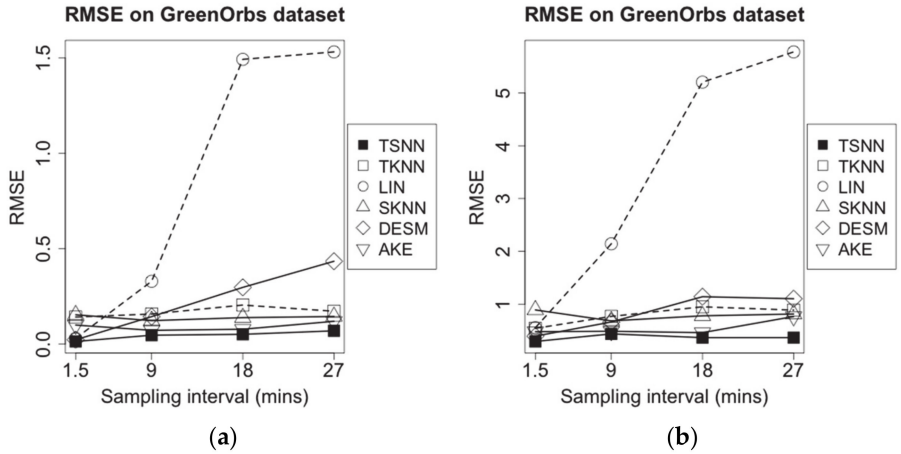


Figure 12. RMSE of imputation algorithms on different data density for GreenOrbs dataset: (**a**) temperature data and (**b**) humidity data.

From Figures 10 and 11, we find that in GreenOrbs dataset, both data distance-based algorithms, TKNN and SKNN, have the relative low performance in RMSE because the relationship between the temperature and the humidity data has been affected by more complex environment factors in the outdoor woods than the indoor room. Meanwhile, the temperature and the humidity data in the outdoor woods have more intensive change than that in the indoor room; that causes the time distance-based algorithm, LIN, get worse performance as the percent of missing values increases. In these situations, TSNN algorithm can optimally utilize four kinds of nearest neighbor values in geometrical or time distance and data distance and make effective estimation based on more information, so it still gets the lowest RMSE among all algorithms.

Moreover, as Figure 12 shows, among all algorithms, the dramatic changes of temperature and humidity measuring data make the performance of LIN, which is only based on time distances, reduce sharply with increasing sampling interval. SKNN and AKE, which are based on spatial nearest neighbor values, get the better performance. Benefitting from both the temporal and spatial nearest neighbor values, TSNN maintains the best performance in RMSE.

The above experimental results on two different test datasets show that TSNN has the best RMSE performance.

5. Discussion

The basic idea to make imputation for missing data on nodes of WSNs is to estimate the missing data using the information of non-missing data. Therefore, we can improve PCE with less information during imputation. It is a flexible way to make imputation by combining spatial and temporal nearest neighbor values because the algorithm is still applicable when one of them, spatial or temporal information, is unavailable for some reason. For example, when there are not spatial neighbors, the AKE is unusable, but the TSNN may be still available if it can find temporal neighbors.

On the contrary, to improve RMSE of imputation, we need to maximize the available information, in other words, more information of non-missing data will be helpful to improve the accuracy of estimation. Both of the spatial and temporal information are utilized in the TSNN. Actually, it is not a brand-new way to make imputation using both of them. For example, in DESM, spatial and temporal information are both used as well. However, the crucial point in TSNN is to make the calculated contribution ratio of spatial and temporal information to the estimation approach the actual contribution ratio in the real case. Different from the contribution ratio calculated on the basis of the correlation between the node with missing value and its neighbors in other algorithms, for example, DESM. We define the spatial–temporal coefficient λ, which makes it more accurate to evaluate the actual contribution of spatial and temporal information of non-missing data. It is obtained based on the subset of non-missing information that is chosen randomly in the observed time window W, while the regression tool and average root mean square error is applied in the calculation. The coefficient λ is more reasonable for evaluation the contribution ratio, but it has its own downside: It is costly to compute. Generally, the data processing center of WSNs is able to support this computation but how to optimize the method to get this coefficient and alleviate the cost of computation will be considered in our further research work.

In addition, the correlations among more than one sensor on the same node are utilized in TSNN. They have been described from the perspective of data distances in time and space dimensions. In our algorithm, k-nearest neighbors method is applied in the calculation in which the Gaussian kernel function is used to work out the weight of neighbors. The amounts of spatial and temporal nearest neighbors, k_s and k_t, are key parameters because more neighbors may bring noise whereas less neighbors will provide insufficient information. TSNN can find the best number of neighbors by calculating the correlations among measurements of the node with missing values and its neighbors on the subset without missing values. However, the range of the neighbors still requires being

set by users because it is strongly related to the spatial distribution of the nodes in WSNs, especially in the situation that the nodes are deployed in a building with complex structure. In addition, the change of the node's state is another problem in building the group of neighbors; because of power failure or other reasons, the nodes which are in the previous initial neighbors group may have problems and become unavailable. One way to solve the problem is to maintain the topological structure of all the neighbor groups and update them regularly, but the task will consume lots of computational resources. How to optimize the initial selection of neighbors and reduce the computation is another research objective for the next stage.

6. Conclusions

In this paper, we present TSNN, a new algorithm for imputation of missing values in WSNs. As the basis of the algorithm, the temporal and spatial nearest neighbor values have been defined. The missing values of the node have been estimated by utilizing the non-missing values of the nodes in the observed time window, in which the regression tool is applied, and the best number of nearest neighbors and the spatial–temporal coefficient are figured out to make the algorithm obtain its best performance. Two evaluation methods, PCE and RMSE, are applied to evaluate the performance of the algorithms, and we test our algorithm on two WSNs dataset, the indoor dataset Intel LAB and the outdoor dataset GreenOrbs, and compare it with other typical algorithms. The performance study shows that TSNN is able to exploit the spatial and temporal information and impute missing sensor data with higher imputation accuracy and reduce the number of cases that cannot be imputed as well.

Author Contributions: Idea and conceptualization, Y.D.; algorithm design, Y.D.; coding for experiments, Y.D., C.H. and J.G.; data validation, C.H.; writing—original draft preparation, Y.D.; writing—review and editing, Y.D. and L.S.; supervision, L.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partly supported by the National Natural Science Foundation of China under Grants No. 61873131 and 61702284, the Anhui Science and Technology Department Foundation under Grant 1908085MF207, and the Postdoctoral Found of Jiangsu Province under Grant No. 2018K009B.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

Acknowledgments: The authors would like to thank the reviewers for their comments, which helped to improve the paper. The authors also acknowledge Linguo Li, of Fuyang Normal University, China, who gave us kind help in the data validation.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Little, R.J.A.; Rubin, D.B. *Statistical Analysis with Missing Data*, 2nd ed.; John Wiley & Sons: Hoboken, NJ, USA, 2002.
2. Hossain, T.; Inoue, S. 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), Spokane, WA, USA, 30 May–2 June 2019; pp. 124–129. [CrossRef]
3. Conroy, B.; Eshelman, L.; Potes, C.; Xu-Wilson, M. A dynamic ensemble approach to robust classification in the presence of missing data. *Mach. Learn.* **2016**, *102*, 443–463. [CrossRef]
4. Gorodetsky, V.; Karsaev, O.; Samoilov, V. Direct Mining of Rules from Data with Missing Values. In *Foundations of Data Mining and Knowledge Discovery*; Studies in Computational Intelligence; Young Lin, T., Ohsuga, S., Liau, C.J., Hu, X., Tsumoto, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; Volume 6, pp. 233–264. [CrossRef]
5. Tolle, G.; Polastre, J.; Szewczyk, R.; Culler, D.; Turner, N.; Tu, K.; Burgess, S.; Dawson, T.; Buonadonna, P.; Gay, D. A macroscope in the redwoods. In Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys '05), Association for Computing Machinery, New York, NY, USA, 2–4 November 2005; pp. 51–63. [CrossRef]

6. Lin, W.C.; Tsai, C.F. Missing value imputation: A review and analysis of the literature (2006–2017). *Artif. Intell.* **2020**, *53*, 1487–1509. [CrossRef]
7. Tkachenko, R.; Izonin, I.; Kryvinska, N.; Dronyuk, I.; Zub, K. An Approach towards Increasing Prediction Accuracy for the Recovery of Missing IoT Data based on the GRNN-SGTM Ensemble. *Sensors* **2020**, *20*, 2625. [CrossRef] [PubMed]
8. Jiang, N. A Data Imputation Model in Sensor Databases. In *High Performance Computing and Communications: Third International Conference, HPCC 2007, Houston, USA, September 2007 Proceedings*; Perrott, R., Chapman, B.M., Subhlok, J., de Mello, R.F., Yang, L.T., Eds.; Lecture Notes in Computer Science; 2007; Volume 4782, pp. 26–28. [CrossRef]
9. Pan, L.; Gao, H.; Li, J.; Gao, H.; Guo, X. CIAM: An adaptive 2-in-1 missing data estimation algorithm in wireless sensor networks. In Proceedings of the 2013 19th IEEE International Conference on Networks (ICON), Singapore, 11–13 December 2013; pp. 1–6. [CrossRef]
10. Ren, X.; Sug, H.; Lee, H. A New Estimation Model for Wireless Sensor Networks Based on the Spatial-Temporal Correlation Analysis. *J. Inf. Commun. Converg. Eng.* **2015**, *13*, 105–112. [CrossRef]
11. Pan, L.; Gao, H.; Gao, H.; Liu, Y. A Spatial Correlation Based Adaptive Missing Data Estimation Algorithm in Wireless Sensor Networks. *Int. J. Wirel. Inf. Netw.* **2014**, *21*, 280–289. [CrossRef]
12. Tutz, G.; Ramzan, S. Improved methods for the imputation of missing data by nearest neighbor methods. *Comput. Stat. Data Anal.* **2015**, *90*, 84–99. [CrossRef]
13. Troyanskaya, O.; Cantor, M.; Sherlock, G.; Brown, P.; Hastie, T.; Tibshirani, R.; Botstein, D.; Russ, B. Altman, Missing value estimation methods for DNA microarrays. *Bioinformatics* **2001**, *17*, 520–525. [CrossRef] [PubMed]
14. Pan, L.; Li, J. K-Nearest Neighbor Based Missing Data Estimation Algorithm in Wireless Sensor Networks. *Wirel. Sens. Netw.* **2010**, *2*, 115–122. [CrossRef]
15. Li, Y.; Ai, C.; Deshmukh, W.P.; Wu, Y. Data Estimation in Sensor Networks Using Physical and Statistical Methodologies. In Proceedings of the 28th International Conference on Distributed Computing Systems, Beijing, China, 17–20 June 2008; pp. 538–545. [CrossRef]
16. Madden, S. Intel Lab Data. Available online: http://db.csail.mit.edu/labdata/labdata.html (accessed on 25 February 2021).
17. Ni, K.; Ramanathan, N.; Chehade, M.N.H.; Balzano, L.; Nair, S.; Zahedi, S.; Kohler, E.; Pottie, G.; Hansen, M.; Srivastava, M. Sensor network data fault types. *ACM Trans. Sen. Netw.* **2009**, *5*, 25. [CrossRef]
18. GreenOrbs. Available online: http://www.greenorbs.org/ (accessed on 25 February 2021).
19. Rubin, D.B. Inference and Missing Data. *Biometrika* **1976**, *63*, 581–592. [CrossRef]
20. Bo, C.; Ren, D.; Tang, S.; Li, X.Y.; Mao, X.; Huang, Q.; Mo, L.; Jiang, Z.; Sun, Y.; Liu, Y. Locating sensors in the forest: A case study in GreenOrbs. In Proceedings of the 2012 IEEE INFOCOM, Orlando, FL, USA, 25–30 March 2012; pp. 1026–1034. [CrossRef]
21. Garson, G.D. *Missing Values Analysis and Data Imputation*; Statistical Associates Publishers: Asheboro, NC, USA, 2015.