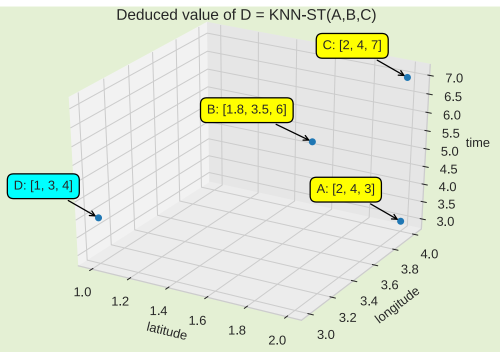
KNN-ST: Exploiting Spatio-Temporal Correlation for Missing Data Inference in Environmental Crowd Sensing

Ningrinla Marchang, *Senior Member, IEEE*, and Rakesh Tripathi, *Member, IEEE*



***Abstract*—Sparse mobile crowdsensing is a new crowdsensing paradigm which leverages the spatial and temporal correlation between data sensed at different locations over time to reduce the overall sensing cost by significantly reducing the number of sensing tasks. Consequently, only sparsely selected spatio-temporal cells would be reporting the sensed data, whereas data for the rest of the cells would have to be inferred from the sensed data. This process, which is largely known as missing data inference is the focus of this study. We examine the KNN (K-Nearest Neighbor) approach, which is known to be relatively faster and simpler. However, it is generally accepted to perform poorly when the sensed data is sparse. In the context of environmental crowd sensing, we examine whether it is a viable missing data inference approach if we incorporate the spatio-temporal correlation of data in the algorithm, instead of just exploiting either the spatial or the temporal correlation independently. Thus, we examine three variants of KNN: KNN-ST (KNN-Spatio-Temporal), KNN-S (KNN-Spatial), and KNN-T (KNN-Temporal) on sparse data. Besides, we find that voxelization is a natural way of exploiting the spatio-temporal properties of sensed data and thereby the spatio-temporal correlation between them. Interestingly, we find that KNN-ST indeed shows good performance (normalized absolute error of about 0.1) even when the loss probability is as high as 0.9. Additionally, we implement an existing method on the same experimental datasets and present corresponding comparative simulation results.**

***Index Terms*—Crowd sensing, KNN, missing data inference, voxelization.**

I. Introduction

MOBILE crowdsensing (MCS) is when 'individuals with sensing and computing devices collectively share data and extract information to measure and map phenomenon of common interest' [1]. Sparse mobile crowdsensing is a new crowdsensing paradigm which leverages the spatial and temporal correlation between data sensed at different locations over time to significantly reduce the number of sensing tasks [2]. The sensing area is divided into subareas or cells. Data sensed from these subareas tend to exhibit spatial and temporal correlation especially in crowdsensing of environmental conditions such as temperature. Hence, only some selected cells need to be sensed and data for the remaining cells can be inferred from the sensed data of the selected cells.

Clearly, one major function in sparse MCS is *Missing Data Inference*. How well missing data (data of cells which are not selected for sensing) can be inferred from sensed data will determine the amount of reduction in the number of sensing tasks and consequently the reduction in the overall sensing cost. Moreover, the accuracy of inference will determine the quality of data thus inferred and thereby the usability of the data. Therefore, it is paramount that missing data inference methods for sparse MCS should perform well. Additionally, it is equally important that the methods should not be too time-consuming so as to be applicable online, i.e., when the MCS application is running.

Some representative missing data inference approaches are K-Nearest Neighbors (KNN) [3], Delaunay Triangulation (DT) [4], Multi-channel Singular Spectrum Analysis (MSSA) [5] and Compressive Sensing (CS) [6]–[8]. Out of

Manuscript received August 24, 2020; accepted September 13, 2020. Date of publication September 21, 2020; date of current version January 6, 2021. This work was supported in part by Science and Engineering Research Board, Department of Science and Technology (SERB-DST), Government of India, under Grant EEQ/2017/000083. The associate editor coordinating the review of this article and approving it for publication was Prof. Chao Tan. *(Corresponding author: Ningrinla Marchang.)*

Ningrinla Marchang is with the Department of Computer Science and Engineering, North Eastern Regional Institute of Science and Technology, Itanagar 791109, India (e-mail: ningrinla@gmail.com; nm@nerist.ac.in).

Rakesh Tripathi is with the Department of Information Technology, National Institute of Technology Raipur, Raipur 492010, India (e-mail: rtripathi.it@nitrr.ac.in).

Digital Object Identifier 10.1109/JSEN.2020.3024976

these approaches, KNN is known to have the least time complexity [9]. Therefore, it is an attractive candidate for missing data inference in MCS. However, the downside is that it is known to exhibit relatively poor performance when the missing values are more [9], [10]. This is where the motivation of this study lies. We find that although KNN has been used in comparative study of inference methods, either the spatial correlation or the temporal correlation of data has been used [10], [11], [17]. The question of interest is 'If we would incorporate both of these correlations together in KNN, would the inference accuracy improve?'. Hence, we formulate three variants of KNN: a) KNN-S (K-Nearest Neighbors-spatial), b) KNN-T (K-Nearest Neighbors-Temporal), and c) KNN-ST (K-Nearest Neighbors-Spatio-Temporal). Besides, we find that voxelization is a natural way of exploiting the spatio-temporal properties of sensed data and thereby the spatio-temporal correlation between them. Voxelization can be defined simply as 'dividing a volume into a collection of smaller fixed-size volumes'. Thus, by voxelization, we can easily incorporate the spatio-temporal correlation in KNN as will be seen in later sections. The summary of the contribution of this study is:

1. We present three variants of KNN, viz., KNN-S, KNN-T and KNN-ST for missing data inference in environmental crowdsensing.
2. We exploit the spatio-temporal correlation of sensed data using voxelization.
3. We show how the extent of spatial and temporal correlation of sensed data can be exploited by judiciously setting the weighting parameter in KNN-ST.
4. We compare the proposed approach with an existing method [17] with the help of simulation results.

The rest of the paper is organized as follows. Section II reviews related existing work, We explain the development of the proposed schemes in section III. Section IV gives a performance comparison study, which is followed by the conclusions in section V.

II. Related Work

Missing data and inference (or interpolation) is a well-studied subject. Quite a large number of methods have been proposed for solving the problem of spatio-temporal interpolation [18]–[23]. Neural network is used in [18] and [21]. K-Nearest-Neighbor (KNN) [3] is a simple local interpolation method, which uses the values of the K nearest neighbors to deduce the missing value. KNN is used for short-term traffic flow prediction in [24] and [25]. However, we are interested in missing data inference w.r.t. sparse MCS. Delaunay Triangulation (DT) methods build virtual triangles for data interpolation by considering the gathered data as vertices. Multi-channel Singular Spectrum Analysis (MSSA) [5], which is a data-adaptive and non-parametric method is mainly used for geographic and meteorological data. Compressing Sensing (CS) [6]–[8] is a well-used method. It has been used for missing data reconstruction in wireless sensor networks [9], urban traffic sensing with probe vehicles [17] and Internet traffic [10]. It is also used for online task allocation with the help of Compressing Crowdsensing [11]. Matrix completion is also used for missing data inference. It is a method of filling missing values in the presented partially filled data matrix. Several approaches have been discussed in literature for matrix completion such as convex optimization [12], low rank matrix completion [13], heuristic greedy algorithm [14], and Neural Network [15], [16]. Out of all these methods, KNN is known to run relatively faster. Moreover, the algorithm is very simple. However, it tends to perform poorly when the data is sparse. In scenarios when running time is an issue, it definitely is an attractive method. This study explores voxelization for exploiting the spatio-temporal properties of data and incorporating them in KNN, resulting in three variants KNN-S, KNN-T and KNN-ST. Voxelization is used in [26] for detecting cluster outliers in aerial LIDAR (Light Detection and Ranging) point clouds.

III. Proposed Schemes

This section presents the development of the proposed schemes. First, we illustrate how voxelization is done to exploit the spatio-temporal properties of data. Then, we explain why both spatial and temporal correlation of sensed environmental data have to be considered and how they are incorporated in KNN.

A. Voxelization Using Normalization and Scaling

As mentioned earlier, we find that voxelization is a natural way of exploiting the spatio-temporal properties of sensed data and thereby the spatio-temporal correlation between them. Voxelization is simply the process of dividing a volume object into fixed-size smaller volumes.

Consider a collection (dataset) of sensed data with spatio-temporal properties. For instance, a data-point could be temperature sensed at a particular location and a particular instant of time. So, each data-point has at least three properties, viz., latitude, longitude and time associated with it. We can say that the dataset as a whole occupies a volume in the 3D-space, in which the 3 dimensions are these three properties. This volume is divided into fixed-size smaller volumes (or voxels), resulting in a 3D structure consisting of fixed-volume cuboids. The value of a voxel is the sensed value that falls in that particular voxel. If there is no sensed value in that voxel, the value of the voxel is said to be NULL.

Voxelization helps in computation of the *distance* between two sensed data-points easily. In this work, by distance between two data-points, we mean a measure of the correlation between two data-points. For instance, in the context of environmental sensing, consider temperature sensed at the same time instant at three locations (lo_1, lo_2, lo_3). If lo_1 is nearer to lo_3 than to lo_2, then the temperature value at lo_1 is expected to be closer to that at lo_3 than at lo_2. Similary, considering temperature sensed at a location at three time instants (t_1, t_2, t_3), if t_1 is closer to t_3 than to t_2, then the temperature value at t_1 is expected to be closer to that at t_3 than at t_2. Thus, we have two distance measures, viz., temporal distance and spatial distance. However, we cannot use the absolute values of latitude, longitude and time. We need to scale them so that they are of similar order of magnitude. Otherwise, one will overshadow the others during calculation. The scaling process is explained in detail below.

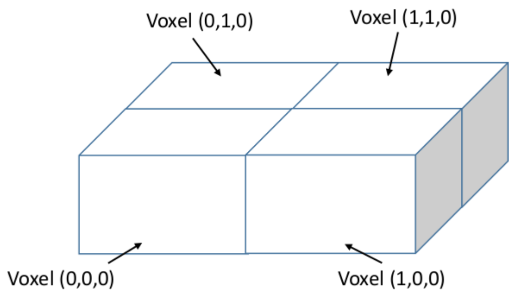


Fig. 1. Four neighboring voxels.

The target area is the area for which sensing is done. We also consider the temporal property of the sensed data. Hence, target volume would be more appropriate a term. First, we decide on the size of granularity of the voxels (i.e., how much area and time duration a voxel represents). The precision (the number of decimal places taken) for the latitude, longitude and time values will determine the granularity. Next, we multiply with the required powers of 10 to convert the values into whole numbers. For time, prior to this, the values are converted to seconds and then to time slots based on a slot-size. The slot-size is a parameter that can be set (in our experiments, we have taken 300s). We then normalize the range of these values. For instance, a longitude value, l is normalized to $l - l_{min}$. Hence, the range of longitude values becomes $[0, l_{max} - l_{min}]$ where l_{max} and $_{min}$ represent the maximum and minimum of all the longitude values in the target volume respectively. Similarly, it is done so for latitude and time. After normalization, the two smaller ranges out of the three are scaled to the the largest range. This is done so that one will not overshadow the other during distance calculation which is described in a later section. Henceforth, whenever longitude, latitude or time values are mentioned, they are values which are already normalized and scaled and not absolute values. It may be noted that it is not necessary to use the absolute values as distance is only used relatively in the proposed approach. Henceforth, by distance, we mean relative distance between two points. Fig. 1 illustrates four neighboring voxels and how they are named.

B. Capturing Spatial-Temporal Correlation in Distance Metric

In KNN, distance between two data-points (sensed values) form the basis of the algorithm. In this subsection, we explain the development of the distance metric. First, we show that considering only the spatial distance does not capture the spatio-temporal correlation between two data-points. Associated with each data-point is a tuple: (latitude, longitude, time) which represents a voxel in the target volume. It may be noted here again that these three values are already normalized and scaled as explained in the last subsection (refer section III-A). The value of the data-point represents the sensed value in that particular voxel.

Please refer to Fig. 2. The figure is used only to give an idea about how far apart the depicted data-points are in the 3-dimensional space. The 3-D space depicted is only a fraction of the target volume. Data-point p_1 is at latitude 2, longitude

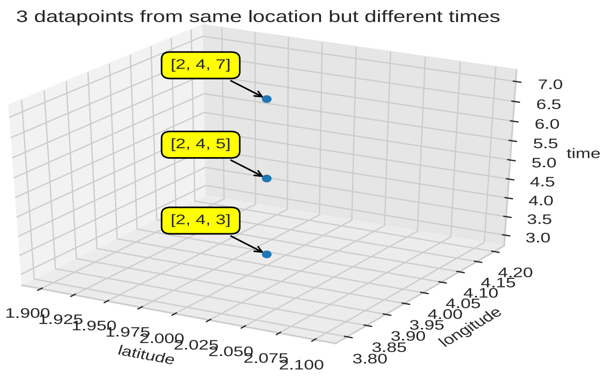


Fig. 2. Distance between data-points decided by time.

4 and time 7. Similarly, data-points p_2 and p_3 are at (2,4,5) and (2,4,3) respectively. Let $d_S(p_i, p_j)$ denote the spatial distance between two data-points, p_i and p_j. Here, we consider the Euclidean distance using the spatial 'coordinates' of p_i and p_j. It is mentioned again here that only relative distance is required. Hence, Euclidean distance is a good enough measure as the coordinate values are already normalized. Calculation of actual distance between any two locations on the earth would require use of methods such as the Haversine function. Then, $d_S(p_1, p_2) = \sqrt{(2-2)^2 + (4-4)^2} = 0$. Similarly, $d_S(p_1, p_3) = d_S(p_2, p_3) = 0$. However, since we started with the assumption that the sensed data exhibit spatio-temporal correlation, we would expect data-point p_1 to be nearer to p_2 than to p_3. In other words, we would expect the sensed value associated with p_1 to be closer to that of p_2 than to that of p_3. This illustrates that d_S does not capture the spatio-temporal correlation between the data-points effectively. Similarly, let $d_T(p_i, p_j)$ denote the temporal distance between two data-points, p_i and p_j. Here, we take the absolute value of the difference between the temporal 'coordinates' of p_i and p_j. Thus, $d_T(p_1, p_2) = |7-5| = 2$; $d_T(p_1, p_3) = |7-3| = 4$ and $d_T(p_2, p_3) = |5-3| = 2$. In other words, as seen in the figure, spatio-temporal distance between p_1 and p_2 (or p_2 and p_3) is indeed smaller than that between data-points p_1 and p_3. Thus, we conclude that d_T is able to capture the spatio-temporal correlation.

Second, we show that considering only the temporal distance does not capture the spatio-temporal correlation between two data-points. Fig. 3 shows another scenario in which there are 3 data-points: p_1, p_2 and p_3 denoted by (2,2,3), (3,4,3) and (4,6,3) respectively at different locations but the same time. Once again, the figure is used only to give a picture of how the data-points are distant from each other in 3-dimensional space. Here, $d_S(p_1, p_2) = 2.24$; $d_S(p_1, p_3) = 4.47$ and $d_S(p_2, p_3) = 2.24$. But, $d_T(p_1, p_2) = d_T(p_1, p_3) = d_T(p_2, p_3) = 0$. In this scenario, d_T does not capture the spatio-temporal correlation between the data-points effectively whereas d_S does. Hence, we conclude that both the temporal and the spatial distances must be considered in the design of the distance function. So, we formulate a new distance metric, which is the weighted average of the two as follows:

$$d_{ST}(x, y) = \alpha . d_S(x, y) + (1 - \alpha) . d_T(x, y) \qquad (1)$$

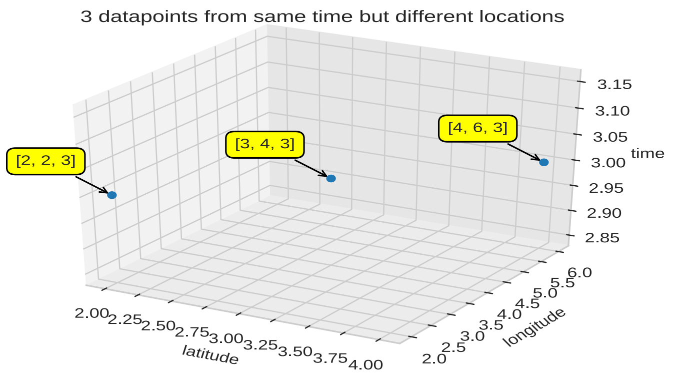


Fig. 3. Distance between data-points decided by location.

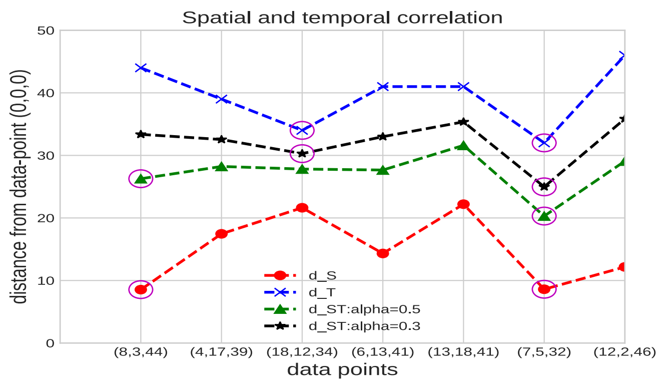


Fig. 4. Relative distance from data-point (0,0,0).

where α is the weight given to d_S, which is a measure of the degree of spatial correlation. Here, $\alpha = [0, 1]$. Though at first sight, d_S and d_T may seem to be different entities, due to normalization and scaling performed in section III-A, they can be combined using a weighting factor (α). Based on the application, α can be set. For instance, if the application is such that there is more spatial correlation than temporal correlation between the sensed data, then α can be set to a value a value greater than 0.5. Such an example application can be sensing of Oxygen level in the environment. Oxygen level at two nearby locations will be more similar than at two faraway locations (say, one location at the top of a mountain and one in the plain). However, time has not much relation with the Oxygen level in a place. Guidelines for setting the value of α is given in section IV-A.1. Note that d_S is a special case of d_{ST} (i.e., when $\alpha = 1$, d_{ST} becomes d_S). Similarly, d_T is nothing but d_{ST} when $\alpha = 0$.

The validity of Eqn. (1) is shown by the simulation results presented in section IV-A.1. Additionally, we present a graph to show the correctness of Eqn. (1). Please refer to Fig. 4. It shows how distance is calculated using Eqn. (1) from a reference data-point (here, (0,0,0)) to 7 data-points (see the x-axis labels). The two nearest data-points from (0,0,0) are shown using circles for each distance metric in the corresponding plots. Note that the 6th data-point, (7,5,32) is the nearest point from (0,0,0) both for distance metric d_S and d_T. This means that it is the nearest data-point from (0,0,0) both spatially and temporally as compared to all the other points. Hence, whatever may be the value of α in the calculation of the d_{ST} metric, it will still be the nearest point from (0,0,0), which illustrates that Eqn. (1) correctly measures the spatio-temporal correlation between two data-points.

Based on the d_{ST} distance metric, the KNN-ST algorithm is developed. There are three input parameters to the algorithm, viz., k, α and $type$. Here, α is the weight in equation (1). Thus, when $\alpha = 1$ and $\alpha = 0$ (refer equation (1)), KNN-ST becomes KNN-S and KNN-T respectively. The parameter $type$ denotes the type of KNN-ST, viz., naive or weighted. In naive KNN-ST, the deduced value is simply the average of the k nearest neighbors' values. However, in weighted KNN-ST, the weighted average of the neighbors' values is the deduced value. Here, the nearer a neighbor is, it is given greater weight. The weight of a neighbor is inversely proportional to its distance from the data-point to be deduced.

KNN-ST Algorithm

```
//$k$, $\alpha$ and $type$ are input parameters.
for each missing data-point x do
  //Find deduced value of x, $x_d$
  Get the values of the $k$ nearest points into array S[1..k]
  and the associated $d_{ST}$ distances into array D[1..k].
  $x_d \leftarrow 0$
  if type = $'naive'$ then
    $sum_s \leftarrow 0$
    for $i = 1$ to $k$ do
      $sum_s \leftarrow sum_S + S[i]$
    end for
    $x_d \leftarrow sum_s/k$
  else
    if type = $'weighted'$ then
      $sum_s \leftarrow 0; sum_d \leftarrow 0$
      for $i = 1$ to $k$ do
        $sum_d \leftarrow sum_d + 1/D[i]$
      end for
      for $i = 1$ to $k$ do
        $x_d \leftarrow x_d + ((1/D[i])/sum_D).S[i]$
      end for
    end if
  end if
end for
```

IV. Performance Evaluation

This section gives numerical simulation results of the proposed schemes and an existing technique, Compressive sensing as implemented in [17]. The programming language used is Python (version 3.7.4, anaconda) on ubuntu18 OS. The algorithms are applied on two datasets, *Temperature* [27] and *Air-pollution* [28]. *Temperature* [27] is a dataset of outdoor temperature recorded by sensors fixed on 289 taxis in Rome over 4 days. We have taken data of one day. The *Air-pollution* [28] dataset consists of measurement data of air pollution parameters such as $pm2.5$, $pm10$, co and so on generated by 25 stations in Seoul. We have taken data for one day.

The section is divided into three parts: i) results obtained using *Temperature* dataset, ii) results obtained using *Air-pollution* dataset, and iii) distribution of the *Temperature* and *air-pollution* data and the impact on results.

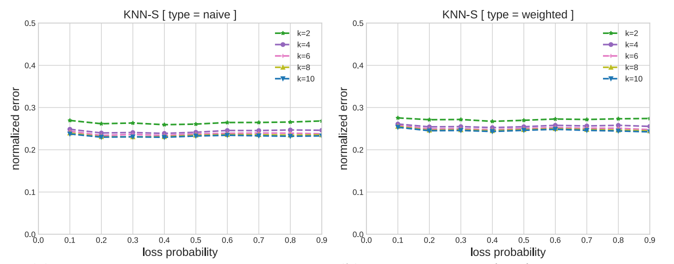

(a) *KNN-S: naive; Temperature* (b) *KNN-S: weighted; Temperature*

Fig. 5. Error vs. loss probability [KNN-S; Temperature]. (a) Naïve. (b) Weighted.

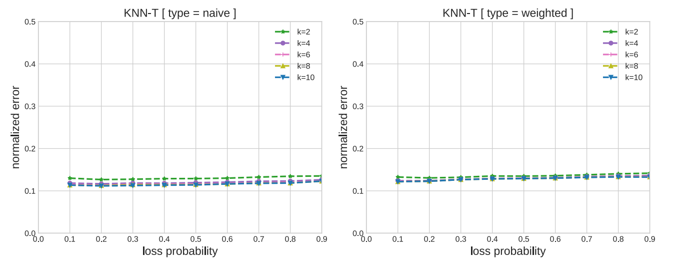

(a) *KNN-T: naive; Temperature* (b) *KNN-T: weighted; Temperature*

Fig. 6. Error vs. loss probability [KNN-T; Temperature]. (a) Naïve. (b) Weighted.

The loss model used is the random loss model, wherein each data-point is deleted (and hence later deduced) with the same loss probability. The performance metric used is the normalized error (NE), which is given by:

$$NE = \frac{\sum_{i,j,t:s(i,j,t)=0} |S(i,j,t) - \hat{S}(i,j,t)|}{\sum_{i,j,t:s(i,j,t)=0} |S(i,j,t)|} \tag{2}$$

where, $S(i, j, t)$ is the actual sensed value at voxel (i, j, t) and $\hat{S}(i, j, t)$ is the deduced value at voxel (i, j, t). When $s(i, j, t) = 0$, it means value at voxel (i, j, t) is missing (and has to be deduced).

A. Temperature Dataset

In this subsection, we show the results for the *Temperature* dataset [27]. Figs. 5a and 5b illustrate the normalized error vs. loss probability for varying values of k for naive and weighted KNN-S respectively. The error is between 0.2 and 0.3 for all values of k which shows that increasing the value of k does not affect the performance. Besides, there is no observable difference between naive and weighted KNN-S. The corresponding plots for KNN-T are found in Figs. 6a and 6b. KNN-T performs much better that KNN-S as the error has been reduced to within the range 0.1 to 0.15. Next, the corresponding plots for KNN-ST when $\alpha = 0.5$ are shown in Figs. 7a and 7b. We observe that KNN-ST performs marginally better that KNN-T and thus outperforms both KNN-S and KNN-T.

Fig. 8a gives better comparison between the schemes. It shows the error for varying values of k when loss probability is 0.8 (i.e., when 80% of data is missing). We can clearly observe that KNN-ST (with $\alpha = 0.5$) gives the best performance. Fig. 8b illustrates the error for varying loss probability when $k = 4$. In this scenario also, KNN-ST outperforms

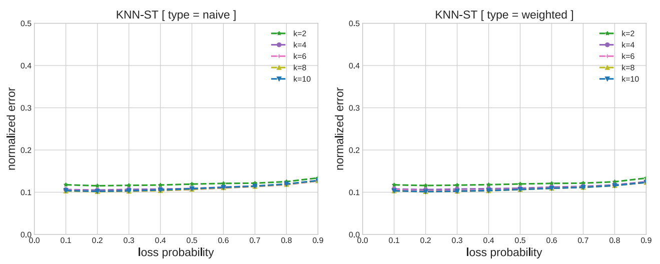

(a) *KNN-ST: naive; Temperature* (b) *KNN-ST: weighted; Temperature*

Fig. 7. Error vs. loss probability [KNN-ST; Temperature]. (a) Naïve. (b) Weighted.

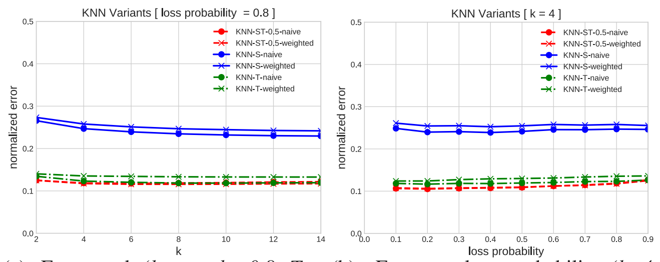

(a) Error vs. k (*loss prob.=0.8; Temperature*) (b) Error vs. loss probability (*k=4; Temperature*)

Fig. 8. KNN variants [Temperature]. (a) Error vs. *k* (loss prob. = 0.8). (b) Error vs. loss probability (*k* = 4).

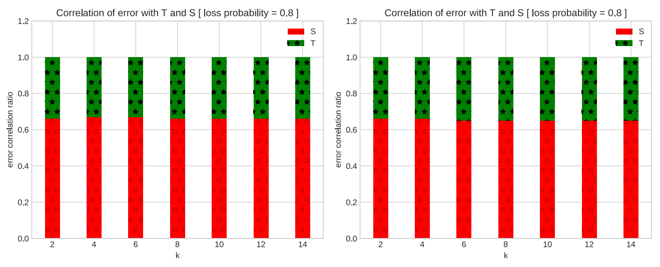

(a) *naive; Temperature* (b) *weighted; Temperature*

Fig. 9. Error correlation ratios vs. k.

KNN-T and KNN-S. It is noteworthy that even when the loss probability is 0.9, the error obtained is within 0.14 for KNN-ST.

1) Setting the Value of α: From the above results, we can conclude that judicious setting of the α value (refer equation (1)) is critical for obtaining optimal performance. Given a dataset, we can use it to estimate the optimal value of α. For this, we calculate a metric called the *error correlation ratio* of the temporal (T) and spatial (S) components. Let $ratioS$ and $ratioT$ represent the respective errors. Then, ratioS=error(KNN-S)/(error(KNN-S)+error(KNN-T)). Here, error(KNN-S) and error(KNN-T) represent the error arrived at when KNN-S and KNN-T are applied respectively. Similarly, ratioT=error(KNN-T)/(error(KNN-S)+error(KNN-T)). The error ratios for varying values of k when loss probability is 0.8 for *naive* and *weighted* are illustrated in Figs. 9a and 9b respectively. The higher the error ratio, we conclude that the lesser is the correlation. Thus, we observe that temporal correlation is higher than spatial correlation. Since, error ratio is indirectly proportional to the measure of correlation, we can use it for setting the value of α. We set $\alpha = 1 - ratioS$ since α is the weight assigned to the spatial component. So, based on

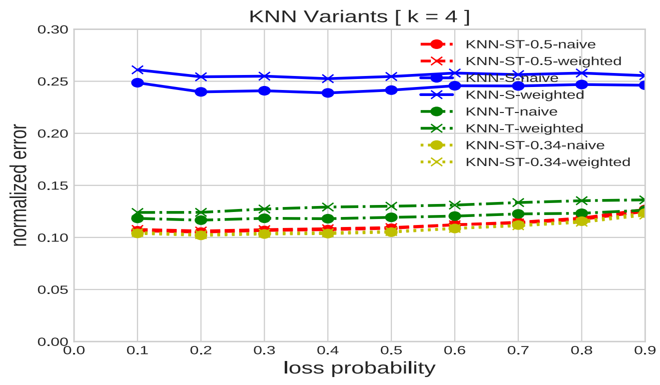


Fig. 10. Error vs. loss probability [*KNN variants; k=4; Temperature*].

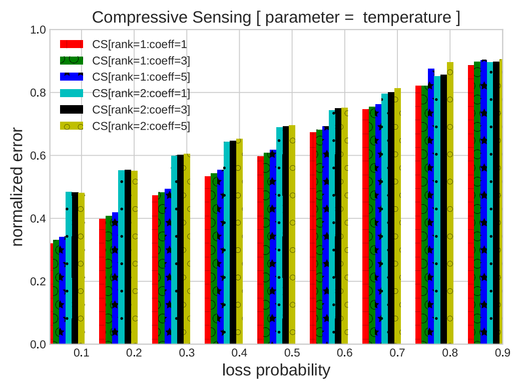


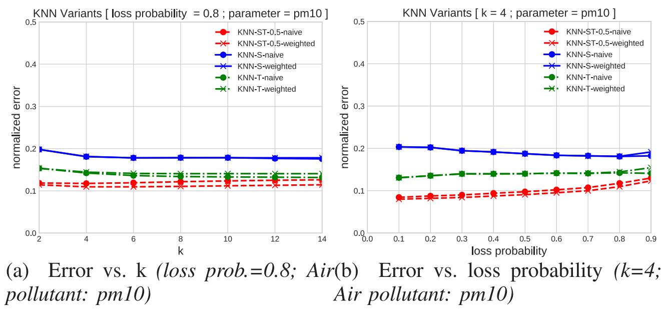
Fig. 11. Error vs. loss probability *[Compressive sensing (Temperature)]*.

both Figs. 9a and 9b, we set $\alpha = 0.34$ when $k = 4$ (for other values of k also, there is no remarkable change). To illustrate the point, we compare the results when $\alpha = 0.34$ with the others (refer Fig. 10). We find in the figure that $\alpha = 0.34$ gives the best performance when loss probability is 0.8.

Next, we look at the performance of CS method as implemented in [17]. Fig. 11 shows the normalized error for varying loss probability. The number of iterations used in the alternating least squares algorithm for CS is 100. The trade-off coefficient is denoted by *coeff* in the graph. After hit and trial with different values of rank and *coeff*, we have shown only the best results. When we compare the results of CS method with KNN-ST (refer Fig. 10), we find that KNN-ST outperforms the CS method. One reason is that the data distribution in the *Temperature* dataset is quite sparse as seen in Fig. 18a. This results in the measurement matrix generated for applying the Compressing sensing method to be sparse. Hence, the poor result as compared to KNN-ST. It is noteworthy that KNN-ST (refer Fig. 10) is more robust than CS method (refer Fig. 11) in that as the loss probability increases the performance of CS method decreases whereas it is not so for KNN-ST.

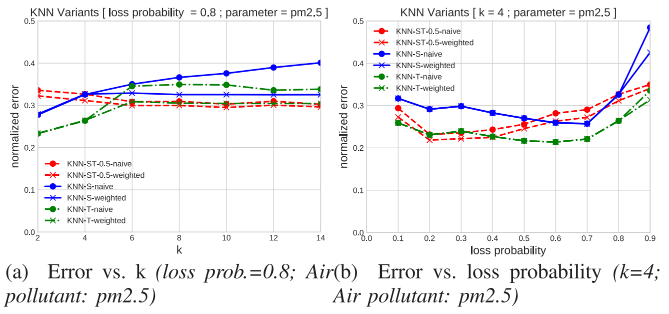
B. Air Pollution Dataset

In this subsection, results for the air-pollution dataset [28] are shown. Due to space constraints, only a few results are shown. Fig. 12a shows the error for varying values of k when loss probability is 0.8 (i.e., when 80% of data is missing) for $pm10$ measurement. We observe that KNN-ST (with $\alpha = 0.5$) gives the best performance (the least error). Fig. 12b illustrates the error for varying loss probability when $k = 4$. In this case also, KNN-ST outperforms KNN-T and KNN-S.



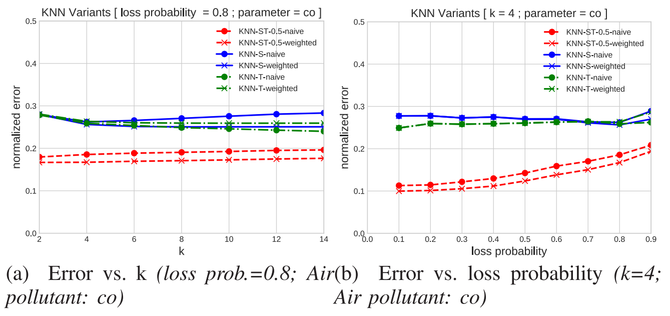
(a) Error vs. k *(loss prob.=0.8; Air pollutant: pm10)* (b) Error vs. loss probability *(k=4; Air pollutant: pm10)*

Fig. 12. KNN variants [Air pollution: *pm*10]. (a) Error vs. *k* (loss prob. = 0.8). (b) Error vs. loss probability ($k = 4$).



(a) Error vs. k *(loss prob.=0.8; Air pollutant: pm2.5)* (b) Error vs. loss probability *(k=4; Air pollutant: pm2.5)*

Fig. 13. KNN variants [Air pollution: *pm*2.5]. (a) Error vs. *k* (loss prob. = 0.8). (b) Error vs. loss probability ($k = 4$).



(a) Error vs. k *(loss prob.=0.8; Air pollutant: co)* (b) Error vs. loss probability *(k=4; Air pollutant: co)*

Fig. 14. KNN variants [Air pollution: *co*]. (a) Error vs. *k* (loss prob. = 0.8). (b) Error vs. loss probability ($k = 4$).

Figs. 13a and 13b give the corresponding plots for $pm2.5$ measurement. From the plots, we can conclude that spatio-temporal correlation is weak for $pm2.5$ measurement. The reasons being that the errors obtained are too high and the plots are haphazard. For pollutant, co, the corresponding plots are shown in Figs. 14a and 14b. From both the graphs, we observe that KNN-ST (with $\alpha = 0.5$) outperforms KNN-S and KNN-T. For a judicious setting of α value, we can follow the guidelines in section IV-A.1

The results of CS method obtained using *Air-pollution* dataset are show in Figs. 15 to 17. We observe that KNN-ST outperforms CS method (compare Fig. 12b with Fig. 15, Fig. 13b with Fig. 16 and Fig. 14b with Fig. 17). In KNN-ST, performance in case of parameter *pm10* is better than the other two, viz., *pm2.5* and *co* illustrating a relatively better spatio-temporal correlation in the case of *pm10*. The same behavior is seen also in the results of CS method. This confirms that both KNN-ST and CS method capture the correlation between the sensed data.

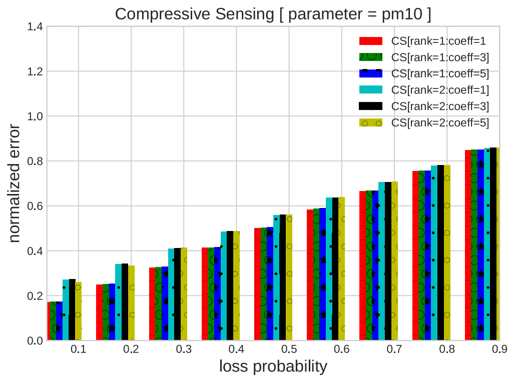


Fig. 15. Error vs. loss probability *[Compressive sensing (Air Pollution: pm10)]*.

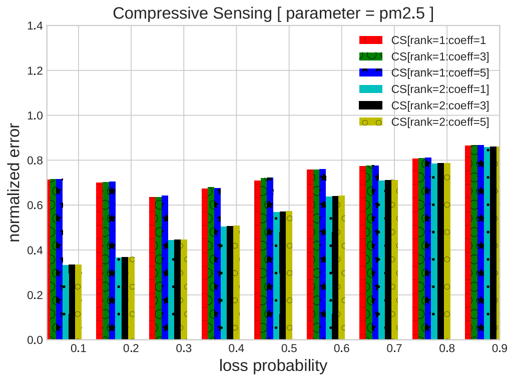


Fig. 16. Error vs. loss probability *[Compressive sensing (Air Pollution: pm2.5)]*.

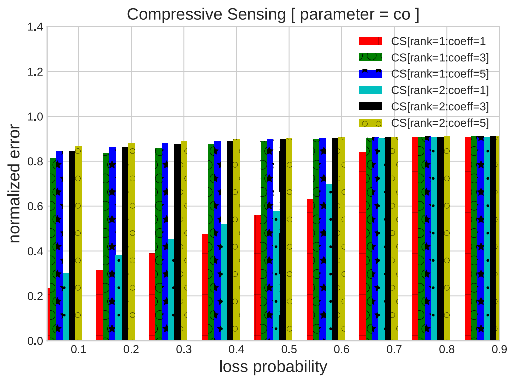
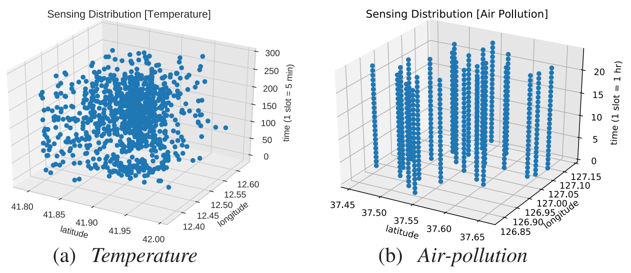


Fig. 17. Error vs. loss probability *[Compressive sensing (Air Pollution: co)]*.



(a) *Temperature* (b) *Air-pollution*

Fig. 18. Sensed data distribution.

C. Sensed Data Distribution

The spatio-temporal distribution of the sensed values for *Temperature* and *Air-pollution* datasets are shown in Figs. 18a and 18b respectively. We can clearly observe that the distribution is more even in Fig. 18a than in Fig. 18b. Accordingly, better results are observed for *Temperature* than for *Air-pollution*.

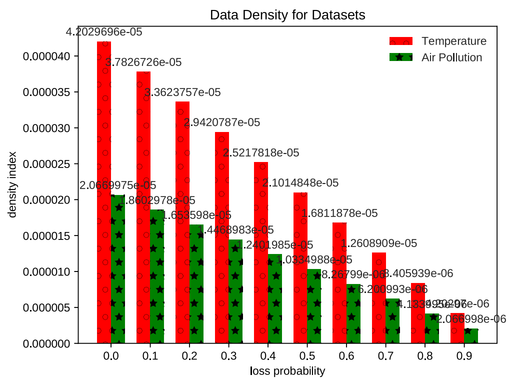


Fig. 19. Density index vs. loss probability.

Finally, we use a metric, density index to estimate the sparsity of the sensed data. Density index is the ratio of the number of voxels with sensed values to the total number of voxels in the 3D target volume. Fig. 19 illustrates the density index for varying values of loss probability. We clearly see that the density index is very low. Yet, we find that KNN-ST works well even so. Moreover, since the density distribution of *Temperature* dataset is denser that of *Air-pollution* dataset, the data inference results are relatively better for the *Temperature* dataset.

V. Conclusion

This work presents the development of a spatio-temporal K Nearest Neighbor scheme. First, we show how incorporating the spatial (KNN-S) and temporal (KNN-T) correlation independently in KNN does not work well. Based on this observation, we incorporate both the spatial and temporal correlation resulting in KNN-ST, which is in fact a weighted combination of KNN-S and KNN-T. Thus, the degree of the temporal and spatial correlation components in KNN-ST can be made to vary by adjusting the weighting parameter (α) based on the requirement of the application. We also provide guidelines on how to set the same (α). Numerical simulation results after applying on two real-life datasets show the validity of the proposed scheme. Moreover, comparison with another deduction method, Compressing sensing is also presented. Interestingly, we find KNN-ST indeed shows good performance (normalized absolute error of about 0.1) even when the loss probability is as high as 0.9. Thus, we confirm that even though K-Nearest Neighbors method is generally known to perform poorly when data is sparse, the proposed KNN-ST scheme gives good enough performance when spatio-temporal properties are exploited even under such a scenario.

References

[1] R. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: Current state and future challenges," *IEEE Commun. Mag.*, vol. 49, no. 11, pp. 32–39, Nov. 2011.

[2] L. Wang, D. Zhang, Y. Wang, C. Chen, X. Han, and A. M'hamed, "Sparse mobile crowdsensing: Challenges and opportunities," *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 161–167, Jul. 2016.

[3] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[4] L. Kong, D. Jiang, and M.-Y. Wu, "Optimizing the spatio-temporal distribution of cyber-physical systems for environment abstraction," in *Proc. IEEE 30th Int. Conf. Distrib. Comput. Syst.*, Jun. 2010, pp. 179–188.

[5] H. Zhu, Y. Zhu, M. Li, and L. M. Ni, "SEER: Metropolitan-scale traffic perception based on lossy sensory data," in *Proc. IEEE INFOCOM-28th Conf. Comput. Commun.*, Apr. 2009, pp. 217–225.

[6] R. Baraniuk, "Compressing sensing," *IEEE Signal Process. Mag.*, vol. 24, no. 4, pp. 118–121, Jul. 2007.

[7] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[8] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[9] L. Kong *et al.*, "Data loss and reconstruction in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 2818–2828, Nov. 2014, doi: 10.1109/TPDS.2013.269.

[10] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu, "Spatio-temporal compressive sensing and Internet traffic matrices," in *Proc. ACM SIGCOMM Conf. Data Commun. (SIGCOMM)*, 2009, pp. 267–278.

[11] L. Wang *et al.*, "CCS-TA: Quality-guaranteed online task allocation in compressive crowdsensing," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, Osaka, Japan, 2015, pp. 683–694.

[12] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Found. Comput. Math.*, vol. 9, no. 6, pp. 717–772, Dec. 2009.

[13] X. Liu, X. Wang, L. Zou, J. Xia, and W. Pang, "Spatial imputation for air pollutants data sets via low rank matrix completion algorithm," *Environ. Int.*, vol. 139, Jun. 2020, Art. no. 105713.

[14] Z. Wang, M.-J. Lai, Z. Lu, W. Fan, H. Davulcu, and J. Ye, "Rank-one matrix pursuit for matrix completion," in *Proc. Int. Conf. Mach. Learn.*, Beijing, China, 2014, pp. 91–99.

[15] J. Y. Gotoh, A. Takeda, and K. Tono, "DC formulations and algorithms for sparse optimization problems," *Math. Program.*, vol. 169, pp. 141–176, May 2018.

[16] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5115–5124.

[17] Y. Zhu, Z. Li, H. Zhu, M. Li, and Q. Zhang, "A compressive sensing approach to urban traffic estimation with probe vehicles," *IEEE Trans. Mobile Comput.*, vol. 12, no. 11, pp. 2289–2302, Nov. 2013.

[18] A. M. Durán-Rosal, C. Hervás-Martínez, and A. J. Tallón-Ballesteros, "Massive missing data reconstruction in ocean buoys with evolutionary product unit neural networks," *Ocean Eng.*, vol. 117, pp. 292–301, May 2016.

[19] S. Tak, S. Woo, and H. Yeo, "Data-driven imputation method for traffic data in sectional units of road links," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 6, pp. 1762–1771, Jun. 2016.

[20] F. Tonini, W. W. Dillon, and E. S. Money, "Spatio-temporal reconstruction of missing forest microclimate measurements," *Agricult. Forest Meteorol.*, vols. 218–219, pp. 1–10, Mar. 2016.

[21] S. Londhe, P. Dixit, S. Shah, and S. Narkhede, "Infilling of missing daily rainfall records using artificial neural network," *ISH J. Hydraulic Eng.*, vol. 21, no. 3, pp. 255–264, Sep. 2015.

[22] J. Tipton, M. Hooten, and S. Goring, "Reconstruction of spatio-temporal temperature from sparse historical records using robust probabilistic principal component regression," *Adv. Stat. Climatol., Meteorol. Oceanogr.*, vol. 3, no. 1, pp. 1–16, 2017.

[23] W. Ruan, P. Xu, and Q. Z. Sheng, "Recovering missing values from corrupted spatio-temporal sensory data via robust low-rank tensor completion," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, Cham, Switzerland: Springer, 2017, pp. 607–622.

[24] S. Cheng, F. Lu, P. Peng, and S. Wu, "A spatiotemporal multi-view-based learning method for short-term traffic forecasting," *ISPRS Int. J. Geo-Inf.*, vol. 7, no. 6, p. 218, 2018, doi: 10.3390/ijgi7060218.

[25] X. Luo, D. Li, Y. Yang, and S. Zhang, "Spatiotemporal traffic flow prediction with KNN and LSTM," *J. Adv. Transp.*, vol. 2019, pp. 1–10, Feb. 2019, doi: 10.1155/2019/4145353.

[26] S. Griffioen, "A voxel-based methodology to detect (clustered) outliers in aerial LIDAR point clouds," M.S. thesis, Dept. Urbanism, Dept. Archit. Built Environ., Delft Univ. Technol., Delft, The Netherlands, Nov. 2018.

[27] M. A. Alswailim, H. S. Hassanein, and M. Zulkernine. (Nov. 2015). *CRAWDAD Dataset Queensu/Crowd Temperature (V. 20151120): Derived From Roma/Taxi (V. 20140717)*. [Online]. Available: https://crawdad.org/queensu/crowd_temperature/20151120, doi: 10.15783/C7CG65.

[28] (Apr. 27, 2020). *Air Pollution in Seoul*. Accessed: Apr. 3, 2020. [Online]. Available: https://www.kaggle.com/bappekim/air-pollution-in-seoul

Ningrinla Marchang (Senior Member, IEEE) received the B.Tech. degree in computer science and engineering from the North Eastern Regional Institute of Science and Technology, Itanagar, India, in 1993, the M.Tech. degree in computer science and engineering from the Indian Institute of Technology Delhi (IIT Delhi), India, in 1995, and the Ph.D. degree in computer science and engineering from NERIST in 2010. From 1995 to 1996, she was a Research Engineer with the Department of Computer Science and Engineering, IIT Delhi. From 1996 to 2001, she taught at the Department of Computer Applications, Sathyabama Engineering College, Chennai, India. Since 2001, she has been a Faculty Member of NERIST, where she is an Associate Professor with the Department of Computer Science and Engineering. Her research interests include mobile ad hoc networks, cognitive radio networks, and mobile crowdsensing.

Rakesh Tripathi (Member, IEEE) received the Ph.D. degree from the Indian Institute of Technology Guwahati in 2018. He is an Assistant Professor with the Department of Information Technology, NIT Raipur. His research interests include blockchain, data center networks, and intrusion detection.