2018 IEEE 38th International Conference on Distributed Computing Systems

I(TS,CS): Detecting Faulty Location Data in Mobile Crowdsensing

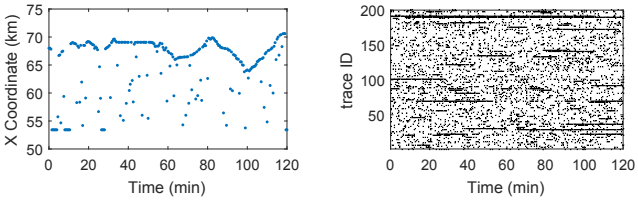
Bowen Wang*, Linghe Kong*, Liang He†, Fan Wu*, Jiadi Yu*, Guihai Chen*
*Shanghai Key Laboratory of Scalable Computing and Systems, Shanghai Jiao Tong University, China,
Email: {the_bright, linghe.kong}@sjtu.edu.cn, fwu@cs.sjtu.edu.cn, jiadiyu@sjtu.edu.cn, gchen@cs.sjtu.edu.cn
†University of Colorado Denver, USA, Email: lianghe@ucdenver.edu

***Abstract*—Mobile Crowdsensing (MCS) is a promising paradigm that utilizes ubiquitous mobile devices to collect environmental data. Specially, location data is critical among all kinds of data because most MCS applications are location-based. Faulty data and missing values, however, may exist in the collected location data due to various reasons. This brings forth an important issue of detecting faulty location data in the presence of missing values. To address this issue, we propose I(TS,CS), a joint faulty data detection framework that combines Time-Series and Compressive Sensing techniques. The framework adopts a DETECT-and-CORRECT approach to iteratively detect faulty data and reconstruct the dataset, which bypasses the tradeoff between false positive ratio (Type-I error) and false negative ratio (Type-II error), and thus detects more faulty data without increasing False Positive Rate. We have evaluated the proposed I(TS,CS) framework based on a real trace consisting of the trajectories of $2,000$ taxies, showing I(TS,CS) dramatically improve the performance of both faulty data detection and data reconstruction.**

I. Introduction

The popularity of smartphones and other mobile devices (e.g. in-vehicle sensing devices such as GPS) enables the pervasive collection of a large volume of data. Mobile Crowdsensing (MCS) [1] exploits such data collection opportunities by leveraging individuals to collect and share sensory data using their mobile devices. Furthermore, MCS is also a special and effective scheme for Internet of Things (IoT) [2], which is a rising concept that enables the environmental information to be collected and shared across platforms. Many MCS applications have emerged in recent years, including urban transportation monitoring [3], urban noise monitoring [4], indoor floorplan construction [5] and image sensing [6]. Most of the MCS applications are location-based, *i.e.,* the collected data makes sense only when we know where it is collected [4, 6]. Moreover, location is almost the only collected data in some traffic monitoring systems such as [3].

MCS, however, suffers from faulty data and missing values due to its openness, especially for location data. In MCS, the sensing devices are owned by end users instead of settled by authority — anyone with a mobile device can participant the sensing tasks, and contribute his data. This exposes the MCS system to malicious and erroneous users, who are likely to upload faulty or biased data [7]. In addition, sensor failures can also cause faulty data and missing values [8]. Also, missing values are common in MCS due to its user-centric property [9].



(a) Faulty data (b) Missing values

Fig. 1. Example of faulty data and missing values

Figure 1 illustrates an example of faulty data and missing values in MCS location data obtained from a real trace that records the trajectories of over 2,000 taxies in the urban area of Shanghai. Figure 1(a) illustrates a 2-hour trace of a taxi. The location is collected every 30 seconds. We can easily infer the route of the taxi from the trace, and the points that deviates from the route are faulty data. By statistics, 28% of the trace are faulty. Figure 1(b) illustrates missing values of the dataset that consists of traces of 200 taxies in 2 hours. The black point indicates the corresponding data is missing. By statistics, 11% of the total data are missing. Such faulty data and missing values can severely deteriorate the performance of MCS applications, thus it is crucial to filter out faculty data in the presence of missing values.

On one hand, much effort has been denoted to truth discovery and data integrity in MCS [10–13], with twofold core idea: taking advantages of user reputation and designing incentive-mechanism. These approaches assume that multiple observations for a same object are uploaded by different participants at the same time. However, location data is unique among participants, thus multiple observations may not be available. As a result, reputation-based error monitoring approaches are not applicable. Incentive mechanisms focus on ensuring the user to report correct data. However, they cannot avoid unintentional false data caused by sensor errors or transmission errors. Traditional outlier detection techniques [14] are alternatives for faulty data detection. However, these techniques do not take missing values into consideration.

On the other hand, compressive sensing [15, 16] is an effective way to reconstruct incomplete dataset. However, it does not work well when faulty data exists. Efforts have been denoted to compensate this. [17] utilizes Compressive Sensing (CS) technique [15, 16] to deal with missing values and users'

2575-8411/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDCS.2018.00083

reputation to filter out faulty data. However, that work assumes that "trusted" participants will never contribute faulty data. The assumption is impractical in most scenarios. [18] attempts to decompose a noisy and not low-rank matrix into several components including a low-rank and an error component. However, the framework cannot automatically detect faulty data.

To address these, we focus on the problem of faulty location data detection in the presence of missing values. The major challenges are twofold: (i) existing solutions require multiple observations, which are not available for location data because it is user-specific; (ii) false positive and false negative are coupled and the tradeoff between them limits the performance of faulty data detection.

In this paper, we propose an **I**terative **T**ime-**S**eries and **C**ompressive **S**ensing **I(TS,CS)** framework to detect and correct faulty data in MCS, which does not demand multiple observations. Specifically, I(TS,CS) (i) detects and filters the faulty data via **TS**-based approaches, producing extremely-low false negative rate (*i.e.*, high recall), (ii) reconstructs the dataset by **CS** to fill the missing values, and (iii) repeats the process until convergence. The iterative execution of **TS** and **CS** facilitates to bypass the tradeoff between false negative and false positive, thus minimizing the negative impact of missing values on faulty data detection. Although I(TS,CS) is designed to deal with location data, it can be easily extended to other kind of sensory data in MCS.

Through real trace based experiments, we show that I(TS,CS) can produce over 95% recall and precision in faulty data detection even when 40% of the data is missed and 40% is faulty, and the reconstruction error remains about 200m when 30% of the data is missed and 20% is faulty.

Our major contributions are listed as following.

- We explore the faulty data detection in MCS in a different scenario from existing works, where the sensing is sparse and multiple observations on the same object are not available.
- We design an Iterative Time-Series and Compressive Sensing framework, which iteratively applies outlier detection and compressive sensing until convergence. The framework bypasses the tradeoff between false positive and false negative, achieving high recall and precision at the same time.
- We evaluate I(TS,CS) based on the SUVnet dataset, showing its effectiveness even when 40% of the data are missing and 40% of the data are erroneous.

The rest of the paper is organized as follows. We first model the problem in Section II. The I(TS,CS) framework is proposed in Section III. In Section IV, we evaluate our framework through real data based experiment. We briefly review the related work in Section V. Finally, we conclude this paper in Section VI.

II. Preliminaries

Here we present the problem formulation and introduce the SUVnet dataset [19] used in the design and evaluation.

A. Problem Statement

We consider a location-focused MCS system consisting of n participants. Each participant uploads their location to the centralized server periodically. Time is divided into slots of duration τ (e.g., 30 seconds per unit), and participants upload their locations to the centralized server at each time slot, in the form of $(x(i,j), y(i,j))$ for the i-th participant at the j-th time slot. Considering a system operating period of t time slots, $i \in [1,n]$ and $j \in [1,t]$. Also, we assume $x(i,j)$ and $y(i,j)$ will be accurate/inaccurate/lost together.

We make the following definitions to mathematically formulate the problem.

Definition 1. Coordinate Matrices (CM): describe the real locations of the participants in each time slot. We use two matrices $X_{n\times t}$ and $Y_{n\times t}$ to denote x and y coordinates, respectively:

$$X = [x(i,j)]_{n\times t}, \quad (1)$$

$$Y = [y(i,j)]_{n\times t}. \quad (2)$$

This way, each line of X and Y represent a time series of location data of one participant. Note that X and Y are participants' true locations, thus containing no missing value or faulty data. Also note that all the following definitions besides **Definition 3, 5** and **7** have two copies for X and Y. We will only explain the X-version due to space limit.

Definition 2. Sensory Matrices (SM): contain location data uploaded by participants, where faulty data or missing values may exist. SM are denoted by S_X and S_Y.

According to [20], if there is no faulty data in **SM**, it can be represented as a linear combination of **CM**:

$$S_X = \mathcal{A}(X) \quad (3)$$

$$= \mathcal{E} \circ X, \quad (4)$$

where $\mathcal{A}(\cdot)$ is a linear operator, and $\mathcal{E}$ is **Existence Matrix** defined as following:

Definition 3. Existence Matrix (EM): is an $n \times t$ matrix, denoting if a data point in **CM** is actually collected in **SM**:

$$\mathcal{E}(i,j) = \begin{cases} 0 & \text{, if } X(i,j) \text{ and } Y(i,j) \text{ are missing in SM,} \\ 1 & \text{, otherwise.} \end{cases} \quad (5)$$

However, when faulty data is present in the dataset, the problem is no longer linear. To address this, we make the following definitions:

Definition 4. Faulty Data: A data point $S_X(i,j)$ is faulty if the difference between $S_X(i,j)$ and $X(i,j)$, denoted by $\epsilon^X_{i,j}$, satisfies $|\epsilon^X_{i,j}| > \mathcal{T}$, where $\mathcal{T}$ is a pre-defined threshold. The optimal value of $\mathcal{T}$ is system-specific, which will be elaborated in Section III-B.

This way, the SM can be mathematically defined as:

$$S_X(i,j) = \begin{cases} 0 & \text{, if } X(i,j) \text{ is missing,} \\ X(i,j) + \epsilon^X_{i,j} & \text{, otherwise,} \end{cases} \quad (6)$$

where $\epsilon^X_{i,j}$s satisfy: (i) $\epsilon^X_{i,j}$ is small for normal data and large

for faulty data, and (ii) the expectation of sum of $\epsilon^X_{i,j}$s in normal data $\mathbb{E}(\sum \epsilon^X_{i,j}) = 0$.

Definition 5. Faulty Matrix (FM): is a matrix marking if $X(i,j)$ and $Y(i,j)$ are faulty. It is denoted by $\mathcal{F}$ and defined as:

$$\mathcal{F}(i,j) = \begin{cases} 1 & \text{, if } S_X(i,j) \text{ and } S_Y(i,j) \text{ are faulty} \\ 0 & \text{, otherwise.} \end{cases} \tag{7}$$

The task of identifying the faulty data in S_X and S_Y — can be formulated as following.

Problem 1. Faulty Data Detection (FDD): Given S_X, S_Y and $\mathcal{E}$, find the **Detection Matrices(DM)**, denoted as $\mathcal{D}$, that is as close to $\mathcal{F}$ as possible, *i.e.*,

$$\begin{aligned} &\text{Objective:} \quad \min \|\mathcal{D} - \mathcal{F}\|_F \\ &\text{Subject to:} \quad S_X,\ S_Y \text{ and } \mathcal{E}, \end{aligned} \tag{8}$$

where $\|\cdot\|$ is the Frobenius norm quantifying the difference between $\mathcal{D}$ and $\mathcal{F}$.

To formulate the task of data reconstruction, we further make the following definitions.

Definition 6. Reconstructed Matrices (RM): is generated by removing faulty data and reconstructing the missing values in S_X and S_Y. They are denoted by $\hat{X} = [\hat{x}(i,j)]_{n\times t}$ and $\hat{Y} = [\hat{y}(i,j)]_{n\times t}$.

Definition 7. Generalized Binary Index Matrix (GBIM): is the combination of Existence Matrix $\mathcal{E}$ and Detection Matrix $\mathcal{D}$. It is denoted as $\mathcal{B}$ and defined as:

$$\mathcal{B}(i,j) = \begin{cases} 1 & \text{, if } \mathcal{E}(i,j) = 1 \text{ and } \mathcal{D}(i,j) = 0, \\ 0 & \text{, otherwise.} \end{cases} \tag{9}$$

This way, the task of data reconstruction can be formulated as following:

Problem 2. Data Reconstruction (DR): Given S_X, S_Y and $\mathcal{B}$, determine the optimal $\hat{X}$ and $\hat{Y}$ that best approximate X and Y, *i.e.*,

$$\begin{aligned} &\text{Objective:} \quad \min \|X - \hat{X}\|_F \\ &\text{Subject to:} \quad S_X \text{ and } \mathcal{B}. \end{aligned} \tag{10}$$

B. Real-World Mobility Traces

We evaluate our design with a publicly available dataset of SUVnet [19]. SUVnet records the trajectories of over $2,000$ taxies in the urban area of Shanghai, covering a period of 28 days during 2007-02-01 – 2007-03-01. The raw data of SUVnet, however, is not suitable for evaluation because of significant missing values and corrupted data (which has been illustrated in Section I), leading to the lack of ground truth. Thus, we perform preprocessing on the raw data and select complete subset. The selected subset contain 158 participants $\times$ 240 slots, with a slot duration of $\tau = 30$ second.

III. Iterative Time-Series and Compressive Sensing

Below we explain Iterative Time-Series and Compressive Sensing framework **I(TS,CS)** in detail.

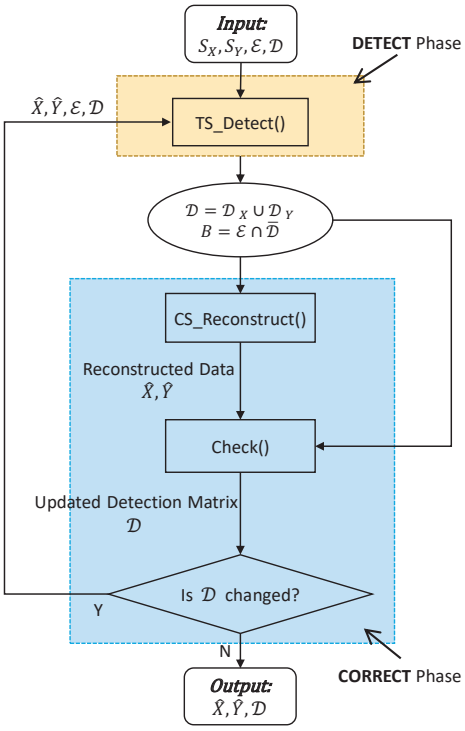


Fig. 2. Logic flow of I(TS,CS).

A. Overview

Intuitively, false negative and false positive are two closely coupled challenges in faulty data detection. I(TS,CS) mitigates such coupling with a **DETECT-and-CORRECT** procedure. In the DETECT phase, I(TS,CS) aims to find as much faulty data as possible, even at the cost of misjudged normal data — minimizing the false negative rate at the cost of false positive. Such sacrifice on false positive is then compensated at the CORRECT phase by reconstructing the data with compressive sensing. The application of compressive sensing here is justified by (i) compressive sensing is known to reconstruct missing values effectively, and comparing to classical interpolation methods, it is relatively insensitive to the ratio of missing values [21], and (ii) the major limitation of compressive sensing — suffering from faulty data with large deviation [22] — has been mitigated during the DETECT phase.

Figure 2 presents an overview of the I(TS,CS) framework, taking **Sensory Matrices (SM)** S_X, S_Y, **Existence Matrix (EM)** $\mathcal{E}$ and **Detection Matrix (DM)** $\mathcal{D}$ as the input. In DETECT phase, I(TS,CS) processes the data with $TS\_Detect()$, a time-series based outlier detection algorithm, after which "suspicious" data will be detected and marked in $\mathcal{D}$. In CORRECT phase, I(TS,CS) treats the "suspicious" data as missing values, marks them in **Generalized Binary Index Matrix (GBIM)** B, and reconstructs them with $CS\_Reconstruct()$. Then, I(TS,CS) uses the **Reconstructed Matrices(RM)** $\hat{X}, \hat{Y}$ as ground truth to check the result of $TS\_Detect()$ and update

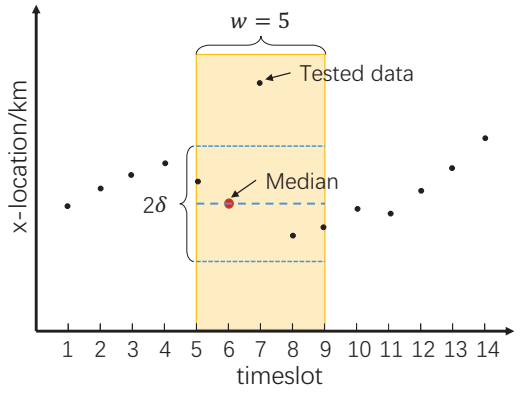


Fig. 3. Local Median Method

$\mathcal{D}$. If $\mathcal{D}$ changes, I(TS,CS) sends the **RM**s, together with updated **DM**, **EM** and original **SM**s, back to $TS\_Detect()$. The procedure continues until $\mathcal{D}$ never changes again. The final **DM** is the set of detected faulty data.

In the rest of this section, we will introduce the detailed design of $TS\_Detect()$, $CS\_Reconstruct()$ and $Check()$. Specifically, we designed optimized local median method for faulty data detecting and checking, and modified Compressive Sensing for missing value reconstruction.

B. Optimized Local Median Method

Optimized Local Median Method is a variant of outlier detection method in time-series. Here we use outliers and faulty data interchangeably — neither of them can be used by the MCS system and thus are *faulty* in our focus.

Figure 3 illustrates the main idea of the Local Median Method. We select the i-th row of the sensory matrix S_X (or S_Y), denoted as $\mathbf{x_i} = [x_i^{(1)}, x_i^{(2)}, \cdots, x_i^{(t)}]$, and test the data points one by one. Define an odd window size w. When testing $x_i^{(k)}$, we consider the $(w-1)$ points around $x_i^{(k)}$, *i.e.,* $[x_i^{(l)}, \cdots, x_i^{(k)}, \cdots, x_i^{(l+w-1)}]$, and calculate the median of these w values, denoted by $m^{(k)}$. Note that l is the index of the first timeslot in the window, as defined in Equation (12). The tested data $x_i^{(k)}$ is judged as faulty if $|x_i^{(k)} - m^{(k)}| > \delta$, where δ is a tolerance threshold.

Pre-defining a fixed δ for the Local Median Method does not work well for location data. For example, vehicles on a highway may run at $100km/h$, but only at $20km/h$ on local road. Assuming $\tau = 30s$, the maximum distance a vehicle can run in a time slot is $833m$ in the former case, but is only $167m$ in the latter — a data point with $300m$ deviation from median will be normal in the highway scenario but is likely faulty on local road.

We use velocity, i.e., the changing rate of location, to optimize the setting of δ, which is readily-available on many mobile systems such as vehicles and smartphones. The velocities are represented by two matrices $V_x = [v_x(i,j)]_{n\times t}$ and $V_y = [v_y(i,j)]_{n\times t}$, where $v_x(i,j)$ and $v_y(i,j)$ represent the velocity component in X and Y direction of the ith participant in the jth timeslot, respectively.

We estimate how far a participant could travel within one timeslot based on his velocity. However, $v_x(i,j)$s and $v_y(i,j)$s are instant velocities at exactly the time when location is collected, and thus may not capture the average velocity within a time slot accurately. As a remedy, we define Average Velocity Matrices $\overline{V}_x$ and $\overline{V}_y$ as:

$$\overline{V}_x = \begin{bmatrix} v_x(1,1) & \frac{v_x(1,1)+v_x(1,2)}{2} & \cdots & \frac{v_x(1,t-1)+v_x(1,t)}{2} \\ v_x(2,1) & \frac{v_x(2,1)+v_x(2,2)}{2} & \cdots & \frac{v_x(2,t-1)+v_x(2,t)}{2} \\ \vdots & \vdots & & \vdots \\ v_x(n,1) & \frac{v_x(n,1)+v_x(n,2)}{2} & \cdots & \frac{v_x(n,t-1)+v_x(n,t)}{2} \end{bmatrix}. \tag{11}$$

and $\overline{V}_y$ is defined similarly. We then apply linear interpolation to estimate the average velocity based on $\overline{V}_x$ and $\overline{V}_y$. Denote $\overline{V}_x(i,j)$ as the average X direction velocity component of the ith participant from the $(j-1)$th to jth timeslot. For simplicity, we assume $v_x(i,0) = v_x(i,1)$ and $v_y(i,0) = v_y(i,1)$, i.e., the average velocity from the 0th and the 1st timeslot is instant velocity of the 1st timeslot.

We define the dynamic tolerance δ based on the maximum distance a participant can travel within w timeslots. Specifically, the tolerance of the ith participant, jth timeslot, in X direction is defined as

$$\begin{aligned} \delta_i^{(j)} &= \xi \times \max_{k=1,2,\cdots,w} \{\sum_{p=l}^{l+k} \overline{V}_x(i,j) \times \tau\}, \\ l &= \min\{\max\{1, j-(w-1)/2\}, t-w+1\}, \end{aligned} \tag{12}$$

where l is the index of the first timeslot in the window, τ is the length of each timeslot, and ξ is a parameter to fine-tune the tradeoff between false negative and false positive errors.

Note that we ignore missing values in the Local Median Method — if there are n missing values in the window, we calculate median $m^{(j)}$ and $\delta_i^{(j)}$ based on the remaining $(w-n)$ data points. Such simplification may reduce the reliability of the faulty data detection method, which can be compensated in the CHECK phase, as we will explain later.

Algorithm 1 shows the pseudo code of the Optimized Local Median Method. First, the first three input of $TS\_Detect()$ can be either X-Version or Y-Version. The returned $\mathcal{D}$ is also either $\mathcal{D}_X$ or $\mathcal{D}_Y$. The final **DM** is $\mathcal{D} = \mathcal{D}_X \cup \mathcal{D}_Y$ (see Figure 2). Second, $\mathcal{D}$ is set to all ones when $TS\_Detect()$ is executed for the first time, and its corresponding element will be set to 0 if a data point is concluded to be normal. This is because $TS\_Detect()$ is designed to find as much faulty data as possible, regardless of how many normal data are misjudged. Moreover, this ensures the convergence of the I(TS,CS) framework. Third, missing values appear in $TS\_Detect()$ only when it is first executed, and will be replaced by the reconstructed value after reconstructing the dataset.

C. Improved Compressive Sensing

Compressive Sensing (CS) [15, 16] is originally designed to compress transmitted data to reduce data transmission cost. It

Algorithm 1: TS_Detect(S,$\hat{S}$,$\overline{V}$,D,$\mathcal{E}$,w,ξ)

Input : Sensory Matrix S; Reconstructed Matrix $\hat{S}$; Average Velocity Matrix $\overline{V}$; Detection Matrix $\mathcal{D}$; Existence Matrix $\mathcal{E}$; Window Size w; Tradeoff Coefficient ξ

Output: Faulty data detection result $\mathcal{D}$

if *Not first executed* **then**
  foreach $S(i,j)$ **do**
    if $\mathcal{E}(i,j) = 0$ **then**
      $S(i,j) \leftarrow \hat{S}(i,j)$;
  $\mathcal{E} \leftarrow ones(n,t)$;
for $i \leftarrow 1$ *to* n **do**
  for $j \leftarrow 1$ *to* t **do**
    if $\mathcal{E}(i,j) = 0$ **then**
      continue;
    Calculate l and δ according to Equation (12);
    $N \leftarrow \emptyset$;
    for $k \leftarrow l$ *to* $l + w - 1$ **do**
      if $\mathcal{E}(i,k) = 1$ **then**
        Add $S(i,j)$ to N;
    $m \leftarrow$ median of N;
    if $|S(i,j) - m| < \delta$ **then**
      $\mathcal{D}(i,j) \leftarrow 0$;
return $\mathcal{D}$;

is also proved to be a powerful tool for data reconstruction, and is tolerable to high data loss. Essentially, CS-based data reconstruction is a matrix completion method, which only applies to low-rank (sparse) matrix. Next, we first analyze the structure of location dataset to show its sparseness, and then design an optimized compressive sensing approach for location data reconstruction.

1) Low-rank Feature of Sensory Data: The low-rank property of location dataset can be proved analytically. Consider a Coordinate Matrix X (we just discuss X here as the analysis on Y is identical) with n rows (n participants) and t columns (t timeslots), and denote its first column as $\alpha_{\mathbf{1}} = [x(1,1), x(2,1), \cdots, x(n,1)]^T$. If every participant move at constant velocity, i.e., $\mathbf{v} = [v_1, v_2, \cdots, v_n]^T$, then X can be presented by

$$X = [\alpha_{\mathbf{1}}, \alpha_{\mathbf{1}} + \tau \cdot \mathbf{v}, \cdots, \alpha_{\mathbf{1}} + (t-1)\tau \cdot \mathbf{v}]. \quad (13)$$

Through elementary transformation, X can be transformed to

$$X' = [\alpha_{\mathbf{1}}, \tau \cdot \mathbf{v}, 0, 0, \cdots, 0], \quad (14)$$

with a rank of 2. The rank of X is not exactly 2 in practice, because participants do not always move at constant velocity. However, the Coordinate Matrix is most likely to show low-rank (sparse) property because the velocity of is stable in most cases [23].

Actually, according to [21], besides location data, most sensory data is low-rank. We further use Principal Component Analysis (PCA) to show the sparseness of location data. According to PCA, a matrix X can be approximately represent by

$$\widetilde{X} = \sum_{i=1}^{r} \sigma_i u_i v_i^T, \quad (15)$$

where σ_i is the i-th largest singular value of X, meaning X can be approximately represented by its top-r singular values. Clearly, X is low-rank if $r \ll \min(n,t)$. Figure 4(a) shows the distribution of singular values in Coordinate Matrices, based on SUVnet dataset described in Section II-B. The X-axis represents the ith singular values, and the Y-axis represents the CDF of the first i singular values, i.e., $\sum_{k=1}^{i} \sigma_k$. We normalize X-axis for uniformity, *i.e.,* $\min(n,t)$ is normalized to 1. This figure suggests that the total energy of Coordinate Matrices is always occupied by the top few singular values. For instance, top 9% singular values of X and top 11% singular values of Y occupy 95% of the total energy, verifying the low-rank property of the Coordinate Matrices.

2) Compressive Sensing Based Method Design: The goal of Data Reconstruction is to estimate $\hat{X}$ and $\hat{Y}$ based on S_X and S_Y. Again, because the reconstruction of S_X and S_Y are identical, we only discuss the reconstruction of S_X in this section for simplicity. According to Equation (15), a sparse matrix $X_{n\times t}$ can be represented by $\widetilde{X} = \sum_{i=1}^{r} \sigma_i u_i v_i^T$. Conversely, given an incomplete matrix S_X, we can construct a sparse matrix

$$\hat{X} = \sum_{i=1}^{\hat{r}} \hat{\sigma}_i \hat{u}_i \hat{v}_i \quad (16)$$

to approximate the origin matrix X. The challenges, however, lie in estimating $\hat{r}$, $\hat{u}_i$ and $\hat{v}_i$.

With the low-rank property of X, we can find $\hat{X}$ by solving the following problem:

$$\begin{aligned} \text{Objective:} \quad & \min(rank(\hat{X})) \\ \text{Subject to:} \quad & \hat{X} \circ \mathcal{B}_X = S_X \end{aligned} \quad (17)$$

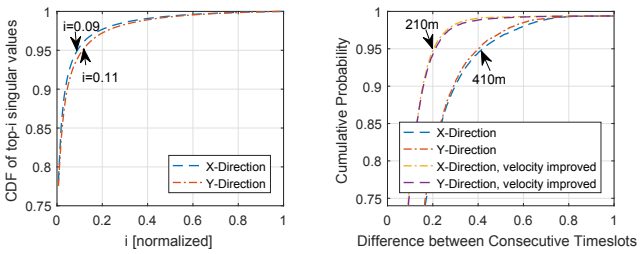
The problem, however, is non-convex, and thus difficult to solve. As a remedy, we take advantage of the SVD-like factorization of $\hat{X}$:

$$\hat{X} = \hat{U}\hat{\Sigma}\hat{V}^T = LR^T, \quad (18)$$

where $\hat{\Sigma}$ is an $\hat{r} \times \hat{r}$ diagonal matrix containing the top $\hat{r}$ singular values $\hat{\sigma}_i, i = 1, 2, \cdots, \hat{r}$, and $L = \hat{U}\hat{\Sigma}^{1/2}$, $R = \hat{V}\hat{\Sigma}^{1/2}$.

According to the theory of Compressive Sensing [15, 16], if the restricted isometry property holds, rank minimization problem (17) can be transformed into the nuclear norm minimization problem of low rank matrix LR^T. Thus, we only need to minimize the sum of L's and R's Frobenius norms:

$$\begin{aligned} \text{Objective:} \quad & \min(\|L\|_F^2 + \|R\|_F^2) \\ \text{Subject to:} \quad & (LR^T) \circ \mathcal{B}_X = S_X \end{aligned} \quad (19)$$



(a) Low-rank feature (b) Temporal stability

Fig. 4. Features extracted from selected dataset

In practice, it is difficult to find L and R that strictly satisfy (19), because (i) collected location data always contains noise; (ii) as shown in Figure 4, X is only approximately low-rank. Therefore, we apply Lagrange multiplier method to relax the constraint:

$$\min(\|(LR^T)\circ\mathcal{B}_X - S_X\|_F^2 + \lambda_1(\|L\|_F^2 + \|R\|_F^2)), \quad (20)$$

where λ_1 is a parameter for tuning the tradeoff between rank minimization and fitting accuracy.

3) Temporal Stability and Velocity Improvement: The above section describes the basic form of compressive sensing. In this section, to improve the reliability of compressive sensing against faulty data, we introduce temporal stability and velocity improvement into compressive sensing.

In real world, most environmental value tends to be stable between adjacent timeslots when the interval is not too large. We measure the temporal stability of participant i at timeslot j by calculating the difference between adjacent timeslots

$$\triangle x(i,j) = |x(i,j) - x(i,j-1)|, \quad (21)$$

and $\triangle y(i,j)$ is defined similarly.

We can introduce temporal stability into compressive sensing by adding the summation of $\triangle x(i,j)$ into object function (20). Theoretically, the smaller $\triangle x(i,j)$ is, the more accurate the result is. Thus, we incorporate velocity to further extract temporal stability property. The rationale of the design is that the estimated average velocity times the timeslot duration τ should be approximately equal to the distance travelled in the timeslot. We measure "velocity improved temporal stability" of participant i at timeslot j by

$$\triangle_v x(i,j) = |x(i,j) - x(i,j-1)| - \overline{V}_x(i,j)\times\tau, \quad (22)$$

where $\overline{V}_x$ is Average Velocity Matrix defined by Equation (11) and τ is timeslot duration.

Figure 4(b) shows the CDF of $\triangle x$, $\triangle y$, $\triangle_v x$ and $\triangle_v y$. The X-axis represents the normalized difference between two consecutive timeslots in Coordinate Matrices, and the Y-axis represents the cumulative probability. In the figure, we see that before incorporating velocity, 95% of $\triangle x(i,j)$s are less than 410m, which is reduced to 210m by incorporating velocity.

After incorporating velocity-improved temporal constraint, the minimization problem (20) changes into:

$$\begin{aligned}\min(\|(LR^T)\circ\mathcal{B}_X - S_X\|_F^2 + \lambda_1(\|L\|_F^2 + \|R\|_F^2) \\ +\lambda_2\|LR^T\mathbb{T} - \tau\overline{V}_x\|_F^2,\end{aligned} \quad (23)$$

where $\overline{V}_x$ is average velocity matrix defined in Equation (11), τ is timeslot duration, and

$$\mathbb{T} = \begin{bmatrix} 1 & -1 & 0 & \dots & 0 \\ 0 & 1 & -1 & \ddots & \vdots \\ 0 & 0 & 1 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & -1 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}_{t\times t}. \quad (24)$$

The matrix $LR^T\mathbb{T}$ captures time stability of the dataset, and $\tau\overline{V}_X$ incorporates velocity. Note that temporal stability is an intrinsic property of location data, and velocity is closely related to location, but collected from another dimension. Thereby, the additional constraint provides more information and can filter out more noises.

4) Modified CS Algorithm: The Modified CS algorithm finds the appropriate L and R that minimize (23). We define the object function

$$f(L,R) = f_1(L,R) + f_2(L,R) + f_3(L,R), \quad (25)$$

where

$$f_1(L,R) = \|(LR^T)\circ\mathcal{B}_X - S_X\|_F^2, \quad (26)$$
$$f_2(L,R) = \lambda_1(\|L\|_F^2 + \|R\|_F^2), \quad (27)$$
$$f_3(L,R) = \lambda_2\|LR^T\mathbb{T} - \tau\overline{V}_x\|_F^2. \quad (28)$$

Obviously, (25) is non-convex. However, if we fix L or R, the other would be convex. Thus, we apply Alternating Steepest Descent (ASD) algorithm [24, 25], which has been proved to be efficient for minimization problem. The main idea of ASD is to alternatively perform steepest gradient descend on L and R. First, L and R are randomly initialized. Note the parameter r is the estimated rank of X, which can be determined by experiment. Next, we iteratively fix L and perform gradient descent on R, and fix R and perform gradient descent on L. The steepest descent along R and L are selected to minimize the updated value of f along the direction ∇_r and ∇_l, respectively. This can be solved by calculating the differential of f and set it to zero, *i.e.*, assume $g_r(\alpha) = f(R, R-\alpha\nabla_r), g_l(\alpha) = f(L, L-\alpha\nabla_l)$, let $g_r'(\alpha) = g_l'(\alpha) = 0$, and solve α.

ASD might suffer from local-optimal in practice. To mitigate it, we optimize the initial value of R and L by: (i) letting $S' = S$ and the missing values in S' be its nearest existing value; intuitively, S' is approximate to the original **Coordinate Matrix (CM)**; (ii) applying SVD to S' and computing R and L. This way, starting points of R and L are close to the optimal one, and thus alleviates the potential local-optimal problem. The pseudo code of modified CS is shown in Algorithm 2.

$Check()$ compares the original **SM**s to **RM**s: if the difference is less than a lower threshold while the corresponding

Algorithm 2: CS_Reconstruct(S,$\mathcal{B}$,$\overline{V}$,λ_1,λ_2,r,$ratio$)

Input : Sensory Matrix S; Generalized Binary Index Matrix $\mathcal{B}$; Average Velocity Matrix $\overline{V}$; Tradeoff Coefficient λ_1, λ_2; Rank bound r; Terminate Ratio $ratio$

Output: Reconstructed Matrix $\hat{S}$

$m, n \leftarrow size(S)$;
$S' \leftarrow S$;
foreach $S'(i,j)$ *in* S' **do**
  if $\mathcal{B}(i,j) = 0$ **then**
    $S'(i,j) \leftarrow$ the nearest existing value;
$[U, \Sigma, V] \leftarrow svd(S')$;
$L \leftarrow U * \Sigma_r^{1/2}$; /* The first r cols of Σ */
$R \leftarrow V * (\Sigma_r^T)^{1/2}$ /* The first r rows of Σ */
repeat
  $\mu_1 \leftarrow f(L, R)$;
  $\nabla_r \leftarrow \frac{\partial f(L,R)}{\partial R}$;
  $\alpha_r \leftarrow \arg\min_\alpha f(L, R - \alpha\nabla_r)$;
  $R \leftarrow R - \alpha_r \nabla_r$;
  $\nabla_l \leftarrow \frac{\partial f(L,R)}{\partial L}$;
  $\alpha_l \leftarrow \arg\min_\alpha f(L, L - \alpha\nabla_l)$;
  $L \leftarrow L - \alpha_l \nabla_r$;
  $\mu_2 \leftarrow f(L, R)$;
until $\frac{\mu_2 - \mu_1}{\mu_1} < ratio$;
$\hat{S} \leftarrow L * R^T$;
return $\hat{S}$;

element in D is 1, turn it to 0; if the difference is larger than a upper threshold while the corresponding element in D is 0, turn it to 1. Similar to Algorithm 1, this algorithm also has X-Version and Y-Version, whose pseudo code is summarized in Algorithm 3.

D. Discussion

The overall time complexity of I(TS,CS) framework is $N(O(TS\_Detect) + O(CS\_Reconstruct) + O(Check))$, where N is the number of iterations. $O(TS\_Detect) = O(ntw) = O(nt)$ and $O(Check) = O(nt)$. The computational cost of $O(CS\_Reconstruct)$ is dominated by a series of matrix multiplications, which is $O(mnt)$, where m is the iteration time in ASD algorithm. As a result, the total complexity of I(TS,CS) is $O(Nmnt)$.

In the design of I(TS,CS), velocity plays an important part. It seems that if the velocity data itself is faulty and contains missing values, the performance of I(TS,CS) framework will be impacted. In fact, this impact is negligible. This is proved in Section IV-D.

IV. Evaluation

In this section, we evaluate I(TS,CS) in terms of faulty data detection performance and data reconstruction accuracy.

Algorithm 3: Check(S,$\hat{S}$,$\mathcal{D}$,$thres_l$,$thres_h$)

Input : Sensory Matrix S; Reconstructed Matrix $\hat{S}$; Detection Matrix $\mathcal{D}$; Threshold $thres_l$, $thres_u$;

Output: Updated Faulty Data Detection Matrix $\mathcal{D}$

foreach $S(i,j)$ *in* S **do**
  if $|S(i,j) - \hat{S}(i,j)| < thres_l$ *and* $\mathcal{D}(i,j) = 1$ **then**
    $\mathcal{D}(i,j) \leftarrow 0$;
  if $|S(i,j) - \hat{S}(i,j)| > thres_u$ *and* $\mathcal{D}(i,j) = 0$ **then**
    $\mathcal{D}(i,j) \leftarrow 1$;
return $\mathcal{D}$;

A. Evaluation Settings

We evaluate I(TS,CS) based on the taxi data in SUVnet [19]. We preprocess the data using the same method with Section II-B. The selected traces contain 158 participants $\times$ 240 timeslots, with slot duration $\tau = 30$s. The original data is stored in two matrices $X_{n\times t}$ and $Y_{n\times t}$. The spatial size of the taxi data is 110×140km, covering major area of Shanghai.

In the trace-driven evaluation, the Existence Matrix $\mathcal{E}$ is randomly generated with a control parameter α specifying the ratio of 0s (i.e., missing values) in $\mathcal{E}$. Also, we randomly select a fraction of the dataset as faulty data. These points are marked in the Faulty Matrix F by setting the corresponding element to be 1. The ratio of faulty data is controlled by β. For the faulty data, we add a random bias $\epsilon_{i,j}$ to the original data, i.e., the generated Sensory Matrices are $S_X = X \circ \mathcal{E} + \mathcal{F} \circ [\epsilon_{i,j}]_{n\times t}$ and $S_Y = Y \circ \mathcal{E} + \mathcal{F} \circ [\epsilon_{i,j}]_{n\times t}$.

We evaluate the performance in two aspects: performance in faulty data detection and performance in reconstruction error. The performance in faulty data detection is judged by Precision and Recall, which are defined as $Precision = \frac{\#TP}{\#TP+\#FP}$ and $Recall = \frac{\#TP}{\#TP+\#FN}$, where $\#TP$, $\#FP$ and $\#FN$ represent True Positive (concluded as faulty data and is indeed faulty data), False Positive (concluded as faulty data while actually is not faulty data) and False Negative (not concluded as faulty data while actually is faulty data), respectively.

The accuracy of missing value reconstruction is measured by **Mean Absolute Error (MAE)**:

$$err = \frac{\sum_{i,j:\mathcal{E}(i,j)=0 \text{ or } \mathcal{D}(i,j)=1} \sqrt{err_x(i,j)^2 + err_y(i,j)^2}}{\sum_{i,j:\mathcal{E}(i,j)=0 \text{ or } \mathcal{D}(i,j)=1} 1}, \tag{29}$$

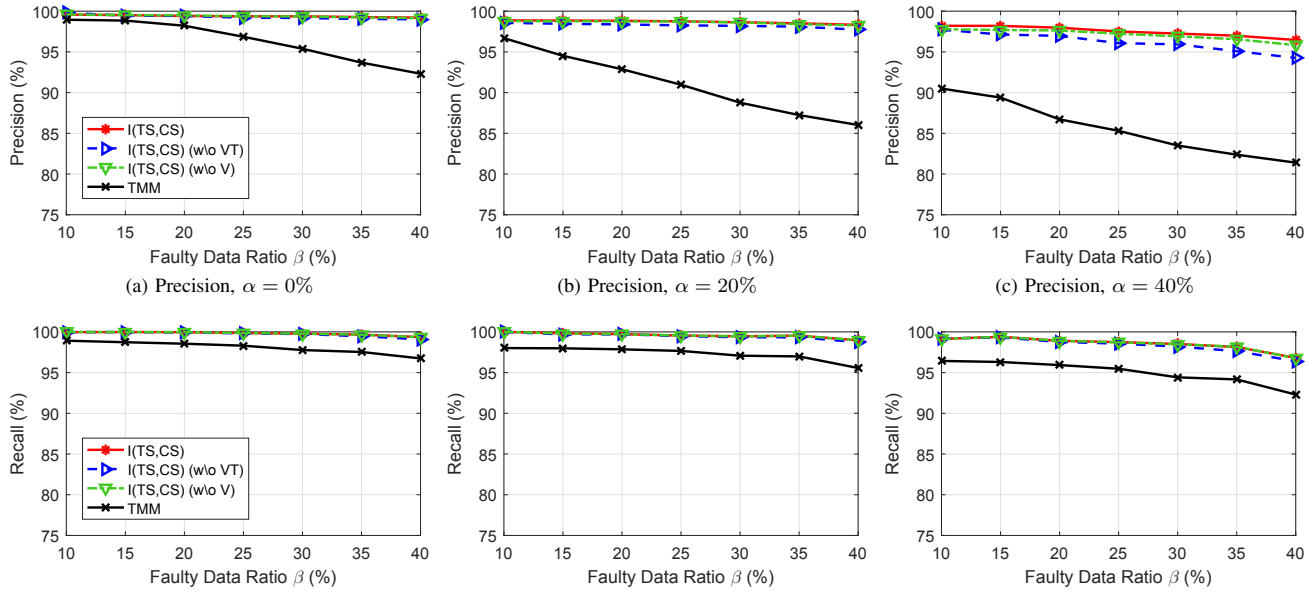
where

$$err_x(i,j) = |X(i,j) - \hat{X}(i,j)|, \tag{30}$$

$$err_y(i,j) = |Y(i,j) - \hat{Y}(i,j)|. \tag{31}$$

We use the following 3 methods as benchmarks when evaluating the performance in faulty data detection:

- **Two-sided median method(TMM)**: A time-series based outlier detection algorithm proposed in [26], which also



(a) Precision, $\alpha = 0\%$ (b) Precision, $\alpha = 20\%$ (c) Precision, $\alpha = 40\%$

(d) Recall, $\alpha = 0\%$ (e) Recall, $\alpha = 20\%$ (f) Recall, $\alpha = 40\%$

Fig. 5. Performance of faulty data detection

compares each data points with the median in the window, but the outlier range is predefined;

- **I(TS,CS) without VT**: Similar to I(TS,CS), but the compressive sensing is not temporal and velocity improved;
- **I(TS,CS) without V**: Similar to I(TS,CS), the compressive sensing is temporal improved but not velocity improved.

Because TMM does not provide data reconstruction, when evaluating the reconstruction error, we replace it with the following method:

- **Modified compressive sensing**: The algorithm is described in Section III-C.

B. Performance in Faulty Data Detection

We first evaluate the performance of faulty data detection in terms of precision and recall. The experiment is conducted when missing value ratio $\alpha = 0\%, 20\%$ and 40%, and in each experiment the faulty data ratio β varies from 10% to 40%. The results are shown in Figure 5.

When missing value ratio $\alpha = 0\%$ (Figure 5(a) – 5(c)) and faulty data ratio β is low ($\leq 20\%$), all these four methods produce similar precision and recall ($> 98\%$). However, with β continuing raising, the precision and recall of TMM drops while the that of the other three methods remains high. When $\alpha = 0\%$, precision and recall of time-series based method drops to 91% and 96%, respectively, while that of the other three methods is still more than 98%. This indicates that I(TS,CS) improves faulty data detection even in case of no missing value.

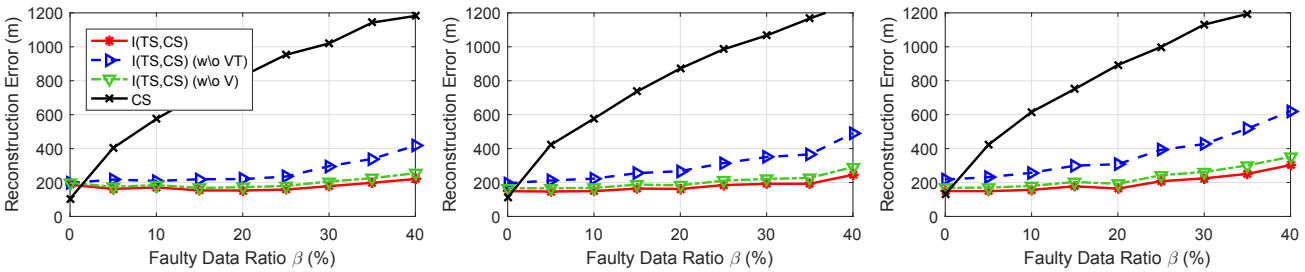
When $\alpha > 0\%$, even if β is low, there is still a distinct gap between the performance of TMM and the other three methods. We can observe that the performance of the three I(TS,CS)-like methods is very stable. Even when the data quality is quite bad, i.e., with $\alpha = 40\%$ and $\beta = 40\%$, the precision and recall is still more than 95%.

Note that the precision and recall of the three I(TS,CS)-like methods is almost indistinguishable. This is because the faulty data is typically at least kilometers away from the normal data. However, as we will see later, the reconstruction error of all these three I(TS,CS)-like methods is less than 1 km in most cases. Consequently, the difference in reconstruction error among the three I(TS,CS)-like methods can hardly influence the performance of faulty data detection.

C. Performance in Missing Value Reconstruction

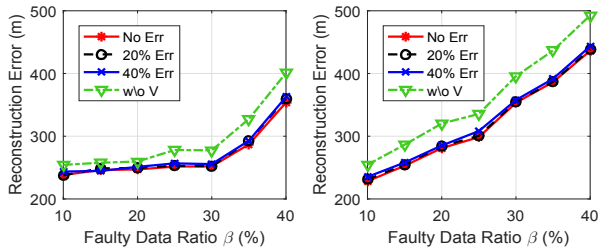
In this section, we evaluate I(TS,CS)'s reconstruction error. The experiment is conducted when missing value ratio $\alpha = 10\%, 20\%$ and 30%, and in each experiment the faulty data ratio β varies from 0% to 40%. The results are shown in Figure 6. When there is no faulty data ($\beta = 0\%$), the reconstruction error of CS is slightly less than those of the three I(TS,CS)-like method. That is because in I(TS,CS), the DETECT phase will introduce misjudged "faulty data", and increase the overall missing value ratio. However, the difference between reconstruction of CS and other three I(TS,CS)-like methods decreases when more values are missing.

Faulty data, even if only 5% of the total volume, increases CS's reconstruction error dramatically. In contrast, the reconstruction error of the other three I(TS,CS)-like methods does not change a lot. When $\beta = 40\%$, CS's reconstruction error increases to over 1200m, while that of I(TS,CS) remains at about 200m. I(TS,CS) performs the best among the three I(TS,CS)-like methods. Specifically, the reconstruction error



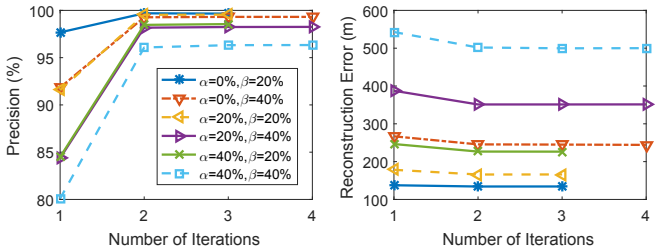
(a) Reconstruction Error, $\alpha = 10\%$ (b) Reconstruction Error, $\alpha = 20\%$ (c) Reconstruction Error, $\alpha = 30\%$

Fig. 6. Performance of missing value reconstruction



(a) $\alpha = 20\%$ (b) $\alpha = 40\%$

Fig. 7. Impact of velocity faulty data



(a) Precision (b) Reconstruct error

Fig. 8. Converging rate of I(TS,CS).

of I(TS,CS) is only half of that with I(TS,CS)-without-VT, and about $10 - 18\%$ better than I(TS,CS)-without-V. In most cases ($\beta \leq 30\%$, $\alpha \leq 20\%$), the reconstruction error of I(TS,CS) is less than 200m. Even when there 30% of the data is missing and 40% of the data is faulty, the reconstruction error is still less than 300m. Compared to spatial size of the traces (110×140km), the error is acceptable. Such errors can be further reduced via map matching [27].

D. Impact of Faulty and Missing Data in Velocity

In this section, we evaluate the impact of faulty data in velocity. In the experiment, we randomly select a fraction (noted by γ) of velocity dataset, and artificially add 100% error onto it (*i.e.,* suppose the original velocity is v, the modified velocity with error is randomly selected between 0 to $2v$). From Section 5 we know, that the Recall and Precision do not show much difference between the three I(TS,CS)-like methods. Thus, we only compare the reconstruction error between I(TS,CS)-with-faulty-velocity and I(TS,CS)-without-V.

The results are shown in Figure 7. The experiment is conducted when missing value ratio $\alpha = 20\%$ and 40%, and the faulty data ratio varies from 10% to 40%. We can observe that when 20% of the velocity is faulty, the reconstruction error is almost the same as that when no faulty data exists in velocity. Even when 40% of the velocity is faulty, the reconstruction error only slightly increases. In contrast, if velocity is not utilized, the reconstruction error dramatically increases.

E. Convergence of I(TS,CS)

Last but not the least, we evaluate I(TS,CS)'s converging rate. The precision and reconstruction error in each iterations are shown in Figure 8. Under all circumstances, the improvement between iteration 1 and iteration 2 is significant, and the later iterations only contribute tiny improvement. Moreover, I(TS,CS) converges in no more than 4 iterations even when $\alpha = 40\%$ and $\beta = 40\%$.

V. Related Work

Faulty data and missing values are common in mobile crowdsensing applications [9]. Works have been done to maintain data integrity. Most of them are based on user reputations and focus on designing incentive mechanism to ensure users to report correct data [10–13]. [28] considers the influence of context and propose a context aware data quality estimation scheme. However, all the above approaches rely on multiple observations on the same sensing target, which is impractical for some applications, especially for location data. Moreover, all these approaches did not take missing values into consideration. [17] utilizes Compressive Sensing (CS) technique to deal with missing values and users' reputation to filter out fault data. However, it does not propose an adequate approach to select faulty data.

A time-series [29] is a series of data points indexed (or listed or graphed) in time order. Correspondingly, outlier detection in time-series [14] is distinctive from time-independent dataset. Various of approaches for time-series outlier detection have been developed, including unsupervised approaches [30], supervised approaches [31] and window based approaches [32].

These approaches cannot be directly applied to faulty location data detection in MSC because they didn't take missing values into consideration. Moreover, they didn't utilize intrinsic structure of location data.

Compressive Sensing (CS) [15, 16] is a technique that originally designed to compress transmitted data in order to reduce data transmission cost. It is also proved to be a powerful tool for data reconstruction, which can tolerate high data loss [21]. CS is able to reconstruct the whole dataset from only a small fraction of data, if only the dataset is low-rank. However, according to the theory of compressive sensing, faulty data is not allowed in the reconstruction matrix [22]. [18] compensates this by decomposing a noisy and not low-rank matrix into several components including a low-rank and a error component. However, the framework cannot automatically detect faulty data.

VI. Conclusion

In this paper, we focused on location data in mobile crowdsensing and considered the problem of faulty data detection in the presence of missing values. We proposed I(TS,CS) framework in which time-series based outlier detection method and compressive sensing are iteratively conducted. We further improve the performance of I(TS,CS) by modifying compressive sensing and having the outlier region of time-series based outlier detection method dynamically calculated. Finally, we conduct a real trace based evaluation, showing the proposed I(TS,CS) framework can drastically improve the performance of faulty data detection and missing value reconstruction.

VII. Acknowledgement

This work was supported in part by the National Key Research and Development Program (2016YFE0100600), in part by China NSF grant 61672349, 61672353, 61472252, 61672348 and 61772338, and in part by the State Key Development Program for Basic Research of China (973 project 2014CB340303).

References

[1] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, 2011.

[2] F. Xia, L. T. Yang, L. Wang, and A. Vinel, "Internet of things," *International Journal of Communication Systems*, vol. 25, no. 9, p. 1101, 2012.

[3] J. Yoon, B. Noble, and M. Liu, "Surface street traffic estimation," in *Mobisys*, 2007, pp. 220–232.

[4] Y. Zheng, T. Liu, Y. Wang, Y. Zhu, Y. Liu, and E. Chang, "Diagnosing New York city's noises with ubiquitous data," in *ACM UbiComp*, 2014, pp. 715–725.

[5] M. Elhamshary, M. Youssef, A. Uchiyama, H. Yamaguchi, and T. Higashino, "Transitlabel: A crowd-sensing system for automatic labeling of transit stations semantics," in *ACM MobiSys*, 2016, pp. 193–206.

[6] Y. Wu, Y. Wang, W. Hu, and G. Cao, "Smartphoto: a resource-aware crowdsourcing approach for image sensing with smartphones," *IEEE Transactions on Mobile Computing*, vol. 15, no. 5, pp. 1249–1263, 2016.

[7] S. Saroiu and A. Wolman, "I am a sensor, and I approve this message," in *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*, 2010, pp. 37–42.

[8] J. Jun, Y. Gu, L. Cheng, B. Lu, J. Sun, T. Zhu, and J. Niu, "Social-Loc: Improving indoor localization with social sensing," in *Mobisys*, 2013, pp. 14:1–14:14.

[9] H. Kurasawa, H. Sato, A. Yamamoto, H. Kawasaki, M. Nakamura, Y. Fujii, and H. Matsumura, "Missing sensor value estimation method for participatory sensing environment," in *IEEE PerCom*, 2014, pp. 103–111.

[10] Y. Li, Q. Li, J. Gao, L. Su, B. Zhao, W. Fan, and J. Han, "On the discovery of evolving truth," in *ACM SIGKDD*, 2015, pp. 675–684.

[11] C. Meng, W. Jiang, Y. Li, J. Gao, L. Su, H. Ding, and Y. Cheng, "Truth discovery on crowd sensing of correlated entities," in *ACM SenSys*, 2015, pp. 169–182.

[12] H. Jin, L. Su, H. Xiao, and K. Nahrstedt, "Inception: incentivizing privacy-preserving data aggregation for mobile crowd sensing systems." in *MobiHoc*, 2016, pp. 341–350.

[13] S. Yang, F. Wu, S. Tang, X. Gao, B. Yang, and G. Chen, "On designing data quality-aware truth estimation and surplus sharing method for mobile crowdsensing," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 4, pp. 832–847, 2017.

[14] A. J. Fox, "Outliers in time series," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 350–363, 1972.

[15] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[16] E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?" *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5406–5425, 2006.

[17] L. Cheng, J. Niu, L. Kong, C. Luo, Y. Gu, W. He, and S. K. Das, "Compressive sensing based data quality improvement for crowdsensing applications," *Journal of Network and Computer Applications*, vol. 77, pp. 123–134, 2017.

[18] Y. Chen, L. Qiu, X. Guangtao, and Z. Hu, "Robust network compressive sensing," in *ACM MOBICOM*, 2014, pp. 545–556.

[19] "SUVnet data collected by Shanghai Jiao Tong University," http: //wirelesslab.sjtu.edu.cn/download.html.

[20] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu, "Spatio-temporal compressive sensing and internet traffic matrices," in *ACM Signcomm*, 2009, pp. 1–1.

[21] L. Kong, M. Xia, X.-Y. Liu, M.-Y. Wu, and X. Liu, "Data loss and reconstruction in sensor networks," in *IEEE INFOCOM*, 2013, pp. 1654–1662.

[22] E. J. Candes and Y. Plan, "Matrix completion with noise," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 925–936, 2010.

[23] S. Rallapalli, L. Qiu, Y. Zhang, and Y.-C. Chen, "Exploiting temporal stability and low-rank structure for localization in mobile networks," in *MOBICOM*, 2010, pp. 161–172.

[24] J. Tanner and K. Wei, "Low rank matrix completion by alternating steepest descent methods," *Applied and Computational Harmonic Analysis*, vol. 40, no. 2, pp. 417–429, 2016.

[25] F. Wu, D. Liu, Z. Wu, Y. Zhang, and G. Chen, "Cost-efficient indoor white space exploration through compressive sensing," *IEEE/ACM Transactions on Networking*, 2017.

[26] S. Basu and M. Meckesheimer, "Automatic outlier detection for time series: an application to sensor data," *Knowledge and Information Systems*, vol. 11, no. 2, pp. 137–154, 2007.

[27] C. E. White, D. Bernstein, and A. L. Kornhauser, "Some map matching algorithms for personal navigation assistants," *Transportation Research Part C: Emerging Technologies*, vol. 8, no. 1, pp. 91–108, 2000.

[28] S. Liu, Z. Zheng, F. Wu, S. Tang, and G. Chen, "Context-aware data quality estimation in mobile crowdsensing," in *IEEE INFOCOM*, 2017.

[29] J. D. Hamilton, *Time series analysis*. Princeton university press Princeton, 1994, vol. 2.

[30] V. Chandola, V. Mithal, and V. Kumar, "Comparative evaluation of anomaly detection techniques for sequence data," in *IEEE ICDM*, 2008, pp. 743–748.

[31] J. B. Cabrera, L. Lewis, and R. K. Mehra, "Detection and classification of intrusions and faults using sequences of system calls," *ACM SIGMOD Record*, vol. 30, no. 4, pp. 25–34, 2001.

[32] A. K. Ghosh, A. Schwartzbard, M. Schatz *et al.*, "Using program behavior profiles for intrusion detection," in *Proceedings of the SANS Intrusion Detection Workshop*, 1999.