Efficient Computation of Robust Average of Compressive Sensing Data in Wireless Sensor Networks in the Presence of Sensor Faults

Chun Tung Chou, *Member*, *IEEE*, Aleksandar Ignjatovic, and Wen Hu, *Senior Member*, *IEEE*

Abstract—Wireless sensor networks (WSNs) enable the collection of physical measurements over a large geographic area. It is often the case that we are interested in computing and tracking the spatial-average of the sensor measurements over a region of the WSN. Unfortunately, the standard average operation is not robust because it is highly susceptible to sensor faults and heterogeneous measurement noise. In this paper, we propose a computational efficient method to compute a weighted average (which we will call robust average) of sensor measurements, which appropriately takes sensor faults and sensor noise into consideration. We assume that the sensors in the WSN use random projections to compress the data and send the compressed data to the data fusion centre. Computational efficiency of our method is achieved by having the data fusion centre work directly with the compressed data streams. The key advantage of our proposed method is that the data fusion centre only needs to perform decompression once to compute the robust average, thus greatly reducing the computational requirements. We apply our proposed method to the data collected from two WSN deployments to demonstrate its efficiency and accuracy.

Index Terms—Wireless sensor networks, compressive sensing, distributed compressive sensing, fault tolerance, data fusion, robust averaging

1 Introduction

This paper considers the fusion of distributed *compressive sensing* [7], [15], [16] data in a wireless sensor network (WSN) [5]. Compressive sensing is a collection of recently proposed sampling and signal reconstruction methods. A promise of compressive sensing is that it can obtain a good approximation of an unknown signal by performing a small number of generalized measurements, called *projections*, provided that the unknown signal is compressible. For WSNs, it means that compressive sensing can be used to reduce the bandwidth requirement and lower the energy consumption.

Given that sensor faults (e.g., offset, stuck-at errors, and variation of sensor measurement noises, and so on [17], [27]) are common in WSNs, many WSN designers choose to deploy redundant sensors so that neighboring sensors should return the same reading if they are noise-free and not faulty. For these WSNs, the users will be interested to compute the average of the data from neighboring sensor nodes. Unfortunately, the standard average operation is *not robust* when there are sensor faults; therefore, it is important to appropriately modify the averaging process to take into account the presence of faults. This paper proposes a method to compute a *robust average* of sensor measurements, which appropriately takes sensor faults and sensor noise into consideration, in a computational-efficient manner. Our proposed method achieves computational efficiency by working on the compressed data, which has a smaller dimension compared with the original data.

Fig. 1 depicts a "standard" method in which distributed compressive sensing [16] can be used to compute a robust average in a WSN. Instead of sending the original sensor measurements, each sensor performs projections on its sensor measurements to produce a lower bandwidth compressed data stream. These compressed data streams are then transmitted over the WSN to reach the data fusion centre, where these data streams are decompressed to retrieve the original signals (or more precisely, an accurate approximation of the original signals because the compression is lossy). Based on the decompressed data streams, the data fusion centre can examine which of the signals are faulty. Assuming that we are interested in calculating the average of the data, we can now use the decompressed data streams to determine suitable weights for this averaging operation, e.g., by weighting noisy signals less than the clean signals. The key advantage of this method is that bandwidth will be saved by sending compressed data streams over the network. This method is suggested in [16].

In this paper, we examine the alternative method depicted in Fig. 2. For this method, the sensors again send compressed data streams to the data fusion centre. The main difference is the sequence of operations to be performed at the data fusion centre. Instead of first decompressing the compressed data to obtain the original data stream, our proposed method will work directly with

- *C.T. Chou and A. Ignjatovic are with the School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia. E-mail: {ctchou, ignjat}@cse.unsw.edu.au.*
- *W. Hu is with the Autonomous Systems Laboratory, CSIRO ICT Centre, Queensland Centre for Advanced Technologies (QCAT), 1 Technology Court, Pullenvale, PO Box 883 Kenmore, QLD 4069, Australia. E-mail: Wen.Hu@csiro.au.*

Manuscript received 7 Mar. 2012; revised 24 Aug. 2012; accepted 25 Aug. 2012; published online 4 Sept. 2012.
Recommended for acceptance by V. Misic.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2012-03-0261.
Digital Object Identifier no. 10.1109/TPDS.2012.260.

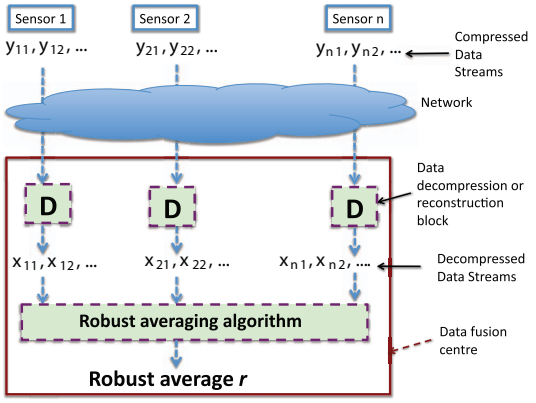

Fig. 1. This method determines the robust average after decompressing all the signals. Because of single processor environment, the time needed to decompress n compressed data streams ($n =$ number of sensors) equals to n times of the time needed to decompress one data stream.

the compressed data without decompressing them. Our proposed method determines a weight for each of the compressed data streams, which reflects whether the sensor which produces the compressed data stream may be faulty or not. We then apply these weights to compute a robust average of the compressed data streams. A nice property of this robust average of the compressed data streams is that, upon decompressing (or applying the compressive sensing reconstruction method to) this stream, we will obtain an approximation of the robust average of the original sensor readings. The key advantage of our proposed method is a great reduction of computation requirement at the data fusion centre. First, we work directly with the compressed data, whose dimension is only a fraction of that of the original sensor readings. Second, we only need to perform decompression (or compressive sensing reconstruction) once, this represents a huge saving in computation requirement because each decompression requires a linear programming problem to be solved. In addition, we show that our fusion algorithm can be implemented on resource-limited wireless sensor nodes.

The rest of this paper is organized as follows: In Section 2, we define the setup of WSN and the data models. We then present our robust averaging method in Section 3. In Section 4, we apply our proposed method to data obtained from two outdoor WSN deployments. Section 5 discusses related work and Section 6 concludes the paper.

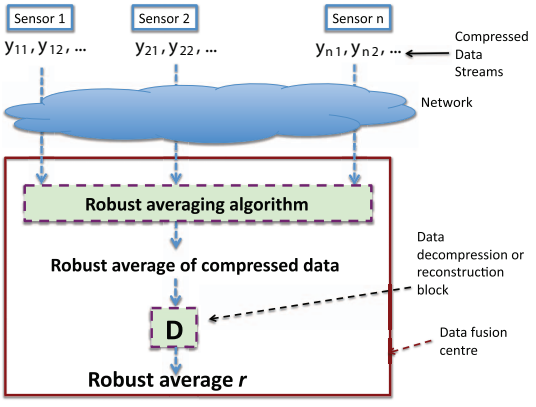

Fig. 2. The new method proposed in this paper. Note that this method requires only one decompression.

TABLE 1
Notation Used in This Paper

n	number of sensors	m	amount of data in a block
$\mathbf{x}_s$	uncompressed data vector from sensor s		
x_{st}	data from sensor s at time t		
$\mathbf{y}_s$	projected (compressed) data vector from sensor s		
y_{sk}	k-th projection from sensor s		
Φ	projection matrix	$\mathbf{r}$	weighted average
w_s	weight on sensor s	p	number of projections
v_s	deviation from weighted mean, see (6)		

2 Models

2.1 Problem Setting: Basic

We consider a WSN with n sensors indexed by $s = 1, \ldots, n$. We assume that the sensors are time synchronized and at each time slot t, each sensor performs a measurement. We will use x_{st} to denote the sensor reading by sensor s at time slot t. We assume that the network works on one block of data with m consecutive data points in a block; therefore, a block of data consists of $\{x_{st}\}$ with $s = 1, \ldots, n$ and $t = 1, \ldots, m$. Note: Table 1 contains a summary of notation used in this paper.

As depicted in Fig. 2, the sensors will not send their readings x_{st} directly to the data fusion centre to conserve bandwidth and energy. Instead, each sensor will perform projections [8] on its sensor readings and send only the results of the projections, which we call either the compressed data stream or the compressed data, to the data fusion centre. The action of, say sensor s, on projecting its data sequence x_{st} $(t = 1, \ldots, m)$ can be expressed in matrix form, as follows:

$$\underbrace{\begin{bmatrix} y_{s1} \\ y_{s2} \\ \vdots \\ y_{sp} \end{bmatrix}}_{\mathbf{y}_s} = \underbrace{\frac{1}{\sqrt{p}} \begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1m} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ \phi_{p1} & \phi_{p2} & \cdots & \phi_{pm} \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} x_{s1} \\ x_{s2} \\ \vdots \\ x_{sm} \end{bmatrix}}_{\mathbf{x}_s} \tag{1}$$

$$\Leftrightarrow \mathbf{y}_s = \mathbf{\Phi} \mathbf{x}_s. \tag{2}$$

The matrix $\mathbf{\Phi}$ is the projection matrix, and the vector $\mathbf{y}_s$ is the result of the projection. (Note that all vector and matrix quantities are typeset in boldface in this paper.) The theory of compressive sensing says that the elements ϕ_{ij} can be drawn from one of these distributions: 1) Standard Gaussian distribution; 2) Symmetric Bernoulli distribution of random numbers $\{+1, -1\}$; or 3) Categorical distribution whose outcomes $\sqrt{3}$, 0, and $-\sqrt{3}$ occur with probability $\frac{1}{6}$, $\frac{2}{3}$, and $\frac{1}{6}$, respectively. We assume that all the sensors use the same projection matrix for each block of data. Since the network is synchronized, this can be achieved by using a time dependent seed for the pseudorandom number generators. This also means that the sink knows the projection matrix that is being used, and the sensors do not have to send this information. The parameter p $(< m)$

here is the number of projections to be used, and we assume that all sensors use the same value of p.

The compressed data stream $\{y_{sk}\}$ (for $k = 1, \ldots, p$) is to be transmitted by sensor s to the data fusion centre of the WSN. For practical implementation, it is easier to use either Bernoulli distribution or Categorical distribution mentioned earlier. For Bernoulli distribution, it will be easier for the sensors to send $\sqrt{p}y_{sk}$ to the data fusion centre because only integer addition and subtraction are needed to perform by the sensor nodes. Furthermore, for suitable values of p, the transmission of the sequence $\{y_{sk}\}$ will require less number of bits than $\{x_{st}\}$. If the analog-to-digital (A/D) convertor on the sensor uses b bits, then sending the original sequence will need mb bits. However, sending the compressed sequence (assuming the aforementioned practical implementation) will require $p(b + 1 + \lceil \log_2 m \rceil)$ bits (where $\lceil u \rceil$ denote the smallest integer larger than or equal to u) because the range of y_{sk} is $[-m(2^b - 1), m(2^b - 1)]$. Thus, bandwidth is saved if $p(b + 1 + \lceil \log_2 m \rceil) < mb$ (assuming that no special coding scheme is used) and we will show that using data from WSN deployments in Section 4. With suitable implementation of Categorical distribution, the sensors again only have to perform integer arithmetic. Also, sampling energy can be saved if a zero is drawn from the Categorical distribution because such samples are not needed. For the rest of this paper, we will assume that either the Bernoulli or Categorical distribution is used. Note that we will continue to write projection using the convention in (2), which is commonly used in compressive sensing literature but noting that the practical implementation may be slightly different.

2.2 Data and Fault Models for Robust Averaging

We assume that all the sensors in WSN are measuring the same physical value at any given time but some of the sensors can be faulty. The type of faults can be bias, stuck-at fault or heterogeneous measurement noise [17]. We model that by assuming that the sensor reading of sensor s at time t, x_{st}, is generated from the Gaussian distribution with mean r_t and variance σ_s^2, i.e., $x_{st} \sim \mathcal{N}(r_t, \sigma_s^2)$. In other words, the model assumes that all sensors should have the same mean reading r_t at time t but the measurement noise of different sensors can be different. Our goal is, therefore, to recover the value of r_t from the sensor readings. We will refer to this process of recovering r_t in the presence of faults as *robust averaging*.

Remark 1. The assumption that all sensors have the same mean reading is a common assumption made in sensor fault detection literature in WSNs. For example, such an assumption is made in [21], [17].

3 Robust Averaging

We will show in Section 3.1 how we can recover r_t if the sensor readings $\{\mathbf{x}_s\}$ are available. After that, in Section 3.2, we show how the same algorithm can be used for recovering r_t from $\{\mathbf{y}_s\}$. Properties of the proposed algorithm are then analyzed in Sections 3.3 and 3.4.

3.1 Computing Robust Average from $\mathbf{x}_s$

Given $\{\mathbf{x}_s\}$ where $x_{st} \sim \mathcal{N}(r_t, \sigma_s^2)$, we propose to recover r_t by using maximum-likelihood estimation. The log-likelihood function for the data sequences $\{x_{st}\}$ with r_t and σ_s as the unknown parameters is

$$L = \mathrm{C} - m\sum_{s=1}^{n}\log(\sigma_s) - \sum_{s=1}^{n}\sum_{t=1}^{m}\frac{(x_{st} - r_t)^2}{2\sigma_s^2}, \tag{3}$$

where C is a constant. By differentiating the log-likelihood function L with respect to σ_s, we have at the maximum of L, $\sigma_s^2 = \frac{1}{m}\sum_{t=1}^{m}(x_{st} - r_t)^2$. After replacing σ_s^2 in (3) by this expression, it can be shown that r_t can be recovered from the following optimization problem:

$$\max_{r_1, r_2, \ldots, r_m} \sum_{s=1}^{n} -\log\left(\sum_{t=1}^{m}(x_{st} - r_t)^2\right). \tag{4}$$

Let $\mathbf{r}$ denote the column vector $[r_1\ r_2\ \ldots\ r_m]^T$ (where ^{T} denotes matrix transpose). It can be shown that the optimal $\mathbf{r}$ can be computed by

$$\mathbf{r} = \sum_{s=1}^{n} w_s \mathbf{x}_s, \tag{5}$$

where $w_1, w_2, \ldots, w_n$ are given by the fixed point of the following two set of equations:

$$v_s = \left\|\mathbf{x_s} - \sum_{i=1}^{n} w_i \mathbf{x_i}\right\|_2^2 \quad \text{for } s = 1, \ldots, n \tag{6}$$

$$w_s = \frac{\frac{1}{\frac{v_s}{\sum_{j=1}^{n} v_j} + \lambda}}{\sum_{i=1}^{n}\frac{1}{\frac{v_i}{\sum_{j=1}^{n} v_j} + \lambda}} \quad \text{for } s = 1, \ldots, n, \tag{7}$$

with the parameter λ set to zero. (Note: The role of λ will be explained shortly.) Note that the estimated $\mathbf{r}$ is a weighted average of the signal because $w_s \geq 0$ (for $s = 1, \ldots, n$) and $\sum_{s=1}^{n} w_s = 1$. Intuitively, v_s measures the deviation of sensor s's measurement $\mathbf{x}_s$ from $\mathbf{r}$. This deviation will be large for faulty sensor and vice versa. Consequently, the weight w_s should be small for those sensors that are faulty or have a large noise variance. Note that for each block of data, a weight is assigned to each sensor. The weight for each sensor can change from a block of data to another because a sensor may only display faults over a limited period of time.

To speed up the convergence of computing $\mathbf{r}$, we use the fixed-point iteration in Algorithm 1. Since the objective function (4) can be unbounded, therefore a small positive constant λ is needed to improve the algorithm's numerical properties. For a small λ, the fixed-point iteration algorithm can, therefore, be interpreted as an approximate maximum-likelihood estimator. Note that if λ is a large number, then the weight w_s in (7) is almost equal to $\frac{1}{n}$; therefore, it is not recommended to choose a large λ. We will show in Appendix C.1 (online supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.260) that,

for stuck-at faults, a small value of λ will only create a small bias in the estimation of the weights w_s.

Algorithm 1. Robust averaging fixed-point iteration
1: Let $w_s^{[\ell]}$ and $v_s^{[\ell]}$ be, respectively, the values of w_s and v_s at the ℓ-th iteration. Perform the following:
2: Initialise $\ell = 0$. $w_s^{[\ell]} = \frac{1}{n}$.
3: $\ell \leftarrow \ell + 1$
4: Compute $v_s^{[\ell]}$ from $w_s^{[\ell-1]}$ using equation (6).
5: Compute $w_s^{[\ell]}$ from $v_s^{[\ell]}$ using equation (7).
6: If the iteration has converged, output $\mathbf{r} = \sum_{s=1}^{n} w_s^{[\ell]} \mathbf{x}_s$; otherwise, go back to Step 3.

We will study the convergence of the fixed-point iteration using data collected from various outdoor WSN deployments in Section 4. We find that the fixed-point iteration converges quickly and this makes it ideal for implementation in wireless sensor nodes, which have only limited computation power.

Note that the above maximum-likelihood interpretation assumes that the fault is a Gaussian noise. When this assumption does not hold, maximum-likelihood estimator can be viewed as a minimiser of the Kullback-Leibler divergence between the true and assumed distributions [33].

3.2 Computing Robust Average from $\mathbf{y}_s$

3.2.1 Computing $\mathbf{r}$ from w_s and $\mathbf{y}_s$

To explain how the robust average $\mathbf{r}$ can be computed from the compressed data $\mathbf{y}_s$, let us, for the time being, assume that we have a method to determine the weights w_s in (5) from $\mathbf{y}_s$. (We will explain in Section 3.2.2 how w_s can be computed from $\mathbf{y}_s$.) By premultiplying both sides of (5) by the projection matrix $\mathbf{\Phi}$, we have

$$\mathbf{\Phi r} = \sum_{s=1}^{n} w_s \mathbf{\Phi x}_s \; = \; \sum_{s=1}^{n} w_s \mathbf{y}_s, \tag{8}$$

where we have used the definition of the projected data streams $\mathbf{y}_s$ given in (2). The importance of (8) is that $\mathbf{\Phi r}$ is in fact the compressed version of the robust average $\mathbf{r}$. Therefore, if the weights w_s are known, then one can obtain the compressed version of the robust average by applying the same weights to the compressed data streams $\mathbf{y}_s$. This means that one can readily obtain $\mathbf{r}$ by decompressing $\sum_{s=1}^{n} w_s \mathbf{y}_s$. This derivation also shows three of the ingredients which are needed for our scheme to work (note: There are two more, for the estimation of w_s, which will be explained later): 1) The averaging operation must be linear. 2) The compression operation must be linear, which is the case for compressive sensing. 3) All sensors must use the same projection matrix, which can be realized by synchronizing the sensor nodes.

Before explaining how the weights w_s can be computed from the compressed data $\{y_{sk}\}$, we will first explain how decompression (or reconstruction) can be done. For our case, decompression can be realized by using any compressive sensing reconstruction algorithm, e.g., basis pursuit [8]. For example, if we know that the robust average is sparse in the basis $\mathbf{\Psi} \in \mathbb{R}^{m \times m}$ (where the columns are the basis vectors), then we can obtain an approximation of $\mathbf{r}$ by

$$\hat{\mathbf{r}} = \mathbf{\Psi}\hat{\mathbf{z}} \text{ where } \hat{\mathbf{z}} = \arg\min_{\mathbf{z} \in \mathbb{R}^m} \|\mathbf{z}\|_1 \text{ s.t. } \mathbf{\Phi\Psi z} = \sum_{s=1}^{n} w_s \mathbf{y}_s.$$

3.2.2 Computing w_s from $\mathbf{y}_s$

To understand how w_s can be computed from $\mathbf{y}_s$, we first state the Johnson-Lindenstrauss (JL) Lemma.

Lemma 1 (JL Lemma [1]). *Let Q be an arbitrary set of q points in $\mathbb{R}^m$, represented by the vectors $\mathbf{d_1}, \mathbf{d_2}, \ldots, \mathbf{d_q}$. Given $\epsilon, \beta > 0$, let*

$$p_0 = \frac{4 + 2\beta}{\frac{\epsilon^2}{2} - \frac{\epsilon^3}{3}} \log q. \tag{9}$$

For integer $p \geq p_0$, let $\mathbf{\Phi}$ be a random $p \times m$ matrix whose elements are generated from either: 1) a symmetric Bernoulli distribution of random numbers $\{+1, -1\}$, or 2) Categorical distribution whose outcomes $\sqrt{3}$, 0, and $-\sqrt{3}$ occur with probability $\frac{1}{6}$, $\frac{2}{3}$, and $\frac{1}{6}$, then with probability at least $1 - q^{-\beta}$, the following holds:

$$\begin{aligned}(1-\epsilon)\|\mathbf{d_i} - \mathbf{d_j}\|_2^2 &\leq \left\| \frac{1}{\sqrt{p}} \mathbf{\Phi}(\mathbf{d_i} - \mathbf{d_j}) \right\|_2^2 \\ &\leq (1+\epsilon)\|\mathbf{d_i} - \mathbf{d_j}\|_2^2 \; \forall \mathbf{d_i}, \mathbf{d_j} \in Q.\end{aligned} \tag{10}$$

Since the projection matrices that are commonly used in compressive sensing obey the JL Lemma, this means that v_s in (6) can be approximately computed from y_s, as follows:

$$\begin{aligned} v_s &= \left\| \mathbf{x_s} - \sum_{i=1}^{n} w_i \mathbf{x_i} \right\|_2^2 \\ &\approx \left\| \Phi\left(\mathbf{x_s} - \sum_{i=1}^{n} w_i \mathbf{x_i} \right) \right\|_2^2 \text{(by JL Lemma)} \\ &= \left\| \mathbf{y_s} - \sum_{i=1}^{n} w_i \mathbf{y_i} \right\|_2^2. \end{aligned}$$

This derivation shows that one can simply replace $\mathbf{x}_s$ by $\mathbf{y}_s$ in Algorithm 1 to compute the robust average from the compressed data $\mathbf{y}_s$. For the remainder of this section, we will first provide a maximum-likelihood interpretation of using compressed data to compute the robust average and then study the perturbation on the weights w_s when the compressed data $\{y_{sk}\}$ are used instead of the original measurements $\{x_{st}\}$.

Based on the above discussion, we see that there are two properties that are needed in order that the weights can be obtained from the compressed data: 1) The projection matrix needs to satisfy the JL Lemma. 2) The algorithm that determines the weights from the data can only use ℓ_2-norm of differences in $\mathbb{R}^m$, where m is the dimension of the original data vectors.

3.3 Maximum-Likelihood Interpretation when Compressed Data Is Used

We argued earlier that, when $\lambda = 0$, the fixed-point iteration for robust averaging can be interpreted as a maximum-likelihood estimator, where the original sensor measurements x_{st} are generated from $\mathcal{N}(r_t, \sigma_s^2)$ with unknown parameters r_t and σ_s^2. The maximum-likelihood interpretation continues to hold when the compressed data y_{sk} are

used instead, except that the variance becomes larger. The following derivation will also show the tradeoff between efficiency and accuracy in using compressed data. Let us decompose x_{st} as the sum of r_t and a noise term e_{st}, as follows: $x_{st} = r_t + e_{st}$, where $e_{st} \sim \mathcal{N}(0, \sigma_s^2)$. The compressed data y_{sk} can be written as follows:

$$y_{sk} = \frac{1}{\sqrt{p}} \sum_{t=1}^{m} \phi_{kt} x_{st} = \underbrace{\sum_{t=1}^{m} \frac{1}{\sqrt{p}} \phi_{kt} r_t}_{\tilde{r}_k} + \underbrace{\sum_{t=1}^{m} \frac{1}{\sqrt{p}} \phi_{kt} e_{st}}_{\tilde{e}_{sk}}.$$

Note that the noise term $\tilde{e}_{sk}$ is a sum of Gaussian distributed random variables; therefore, $\tilde{e}_{sk}$ is also Gaussian distributed. By using the facts that

1. ϕ_{kt} is a random variable with zero mean and unit variance;
2. $\phi_{k_1 t_1}$ is independent of $\phi_{k_2 t_2}$ if either $k_1 \neq k_2$ or $t_1 \neq t_2$;
3. e_{st_1} is independent of e_{st_2};
4. ϕ_{kt} is independent of e_{st};

it can be shown that

$$\mathbb{E}[\tilde{e}_{sk}] = 0 \quad \forall s = 1, \ldots, n, k = 1, \ldots, p \tag{11}$$

$$\mathbb{E}\left[\tilde{e}_{s_1 k_1} \tilde{e}_{s_2 k_2}\right] = \begin{cases} \frac{m}{p}\sigma_s^2 & \text{for } s_1 = s_2 \text{ and } k_1 = k_2 \\ 0 & \text{otherwise,} \end{cases}$$

where $\mathbb{E}$ denotes expectation. Therefore, if the compressed data $\{y_{sk}\}$ is used to determine the robust average instead, the assumption that the noise affecting each compressed datum y_{sk} is corrupted by an independent Gaussian noise continues to hold. This means that the maximum-likelihood interpretation continues to hold even if the compressed data $\{y_{sk}\}$ are used instead. Note that the variance of the noise affecting the compressed datum y_{sk} is $\frac{m}{p}\sigma_s^2$. Since $m > p$, this noise variance is larger than that affecting the original sensor measurement. This derivation also shows the price that is being paid by using the compressed data for robust averaging is an increase in variance. This also shows that there is a tradeoff between computation efficiency (which is achieved by using a small p) and accuracy (which is by using a large p).

3.4 Effect of Using Compressed Data

The derivation of the fixed-point iteration in Section 3.1 assumes that the sensor readings $\{x_{st}\}$ are available to compute the weights w_s. Since our goal is to compute these weights from the compressed data $\{y_{sk}\}$ and use the JL approximation, the aim of this section is to study the perturbation on the weights w_s due to the use of compressed data. We first set up the framework for performing the perturbation analysis.

Let $\mathbf{v}$ and $\mathbf{w}$ denote, respectively, the vectors whose sth element is v_s and w_s. To facilitate the perturbation analysis, we define two operators. Let $\mathbf{T_{wv}}$ denote the operator that maps $\mathbf{w}$ to $\mathbf{v}$ defined by (6), and $\mathbf{T_{vw}}$ denote the operator that maps $\mathbf{v}$ to $\mathbf{w}$ by using (7). If the fixed point of (6) and (7) are given by $\mathbf{w^0}$ and $\mathbf{v^0}$, then

$$\mathbf{v^0} = \mathbf{T_{wv}}(\mathbf{w^0}) \tag{13}$$

$$\mathbf{w^0} = \mathbf{T_{vw}}(\mathbf{v^0}). \tag{14}$$

Let us consider the situation if compressed data $\mathbf{y_s}$ are used instead. In this case, (6) is replaced by

$$v_s = \left\| \mathbf{y_s} - \sum_{i=1}^{n} w_i \mathbf{y_i} \right\|_2^2 = \left\| \mathbf{\Phi}\left(\mathbf{x_s} - \sum_{i=1}^{n} w_i \mathbf{x_i} \right) \right\|_2^2. \tag{15}$$

Note that the rightmost expression of (15) is an approximation of the right-hand-side of (6) based on the JL approximation. If compressed data $\mathbf{y_s}$ are used, the fixed-point algorithm iterates between (15) and (7) instead of (6) and (7). Let $\hat{\mathbf{T}}_{\mathbf{wv}}$ denote the operator that maps $\mathbf{w}$ to $\mathbf{v}$ defined by (15). Let $\mathbf{w^1}$ and $\mathbf{v^1}$ be the solution of the fixed-point iterations using compressed data, then we have

$$\mathbf{v^1} = \hat{\mathbf{T}}_{\mathbf{wv}}(\mathbf{w^1}) \tag{16}$$

$$\mathbf{w^1} = \mathbf{T_{vw}}(\mathbf{v^1}). \tag{17}$$

Our goal is to derive the perturbation on the weights $\Delta\mathbf{w} = \mathbf{w^1} - \mathbf{w^0}$. In addition, we let $\Delta\mathbf{v} = \mathbf{v^1} - \mathbf{v^0}$ and define the relative error ϵ_s evaluated at $\mathbf{w^1}$ by

$$\epsilon_s = \left. \frac{\|\mathbf{\Phi}(\mathbf{x_s} - \sum_{i=1}^{n} w_i \mathbf{x_i})\|_2^2 - \|\mathbf{x_s} - \sum_{i=1}^{n} w_i \mathbf{x_i}\|_2^2}{\|\mathbf{x_s} - \sum_{i=1}^{n} w_i \mathbf{x_i}\|_2^2} \right|_{\mathbf{w}=\mathbf{w^1}}. \tag{18}$$

3.4.1 Local Perturbation Analysis

In this section, we derive the first order approximation for the perturbation $\Delta\mathbf{w}$ assuming that ϵ_s is small. First, we use (13) and (16) to obtain

$$\begin{aligned} \Delta\mathbf{v} &= \hat{\mathbf{T}}_{\mathbf{wv}}(\mathbf{w^1}) - \mathbf{T_{wv}}(\mathbf{w^0}) \\ &= (\hat{\mathbf{T}}_{\mathbf{wv}}(\mathbf{w^1}) - \mathbf{T_{wv}}(\mathbf{w^1})) + (\mathbf{T_{wv}}(\mathbf{w^1}) - \mathbf{T_{wv}}(\mathbf{w^0})) \\ &\approx \mathbf{V^1 e} + \left. \frac{\partial \mathbf{T_{wv}}}{\partial \mathbf{w}} \right|_{\mathbf{w^0}} \Delta\mathbf{w}, \end{aligned} \tag{19}$$

where $\mathbf{V^1}$ is a diagonal matrix whose sth diagonal element is the sth element of $\mathbf{v^1}$, $\mathbf{e}$ is a vector whose sth element is ϵ_s, and $\frac{\partial \mathbf{T_{wv}}}{\partial \mathbf{w}}$ is a Jacobian matrix. Note that $\hat{\mathbf{T}}_{\mathbf{wv}}(\mathbf{w^1}) - \mathbf{T_{wv}}(\mathbf{w^1})$ equals to $\mathbf{V^1 e}$ because of the definition of ϵ_s in (18). Lastly, the second term in (19) comes from Taylor series expansion. (Note: Expressions of Jacobian matrices are given in the appendix, online supplemental material.)

By using (14) and (17), and Taylor series expansion, it can be shown that

$$\Delta\mathbf{w} \approx \left. \frac{\partial \mathbf{T_{vw}}}{\partial \mathbf{v}} \right|_{\mathbf{v^0}} \Delta\mathbf{v}. \tag{20}$$

Given (19) and (20), it can be shown that

$$\Delta\mathbf{w} = \underbrace{\left(\mathbf{I} - \underbrace{\left. \frac{\partial \mathbf{T_{vw}}}{\partial \mathbf{v}} \right|_{\mathbf{v^0}} \left. \frac{\partial \mathbf{T_{wv}}}{\partial \mathbf{w}} \right|_{\mathbf{w^0}}}_{\mathbf{F}} \right)^{-1} \left. \frac{\partial \mathbf{T_{vw}}}{\partial \mathbf{v}} \right|_{\mathbf{v^0}}}_{\mathbf{G}} \mathbf{V^1 e}, \tag{21}$$

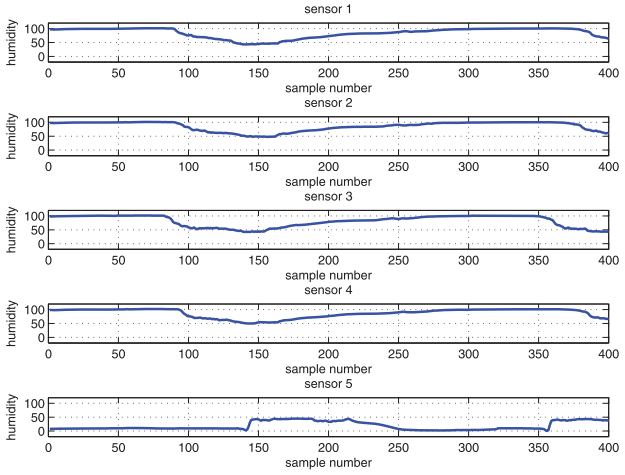

Fig. 3. Humidity reading from the QCAT deployment. Note that the readings from sensor 5 is very different from the first four sensors. Note: All plots use the same scale.

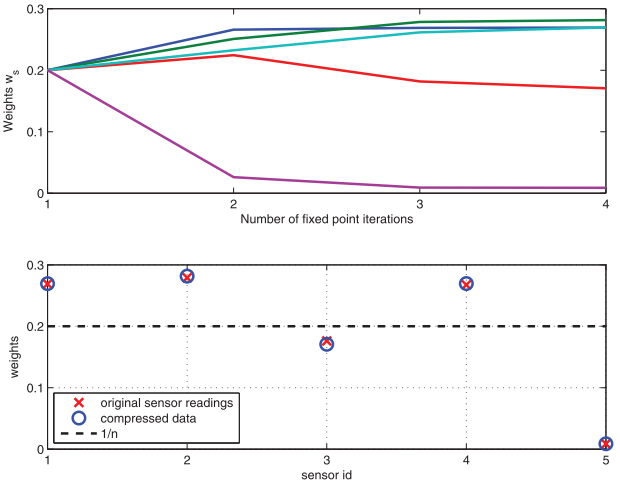

Fig. 4. The upper figure shows that the convergence of the weights w_s, when the robust averaging algorithm is applied to the compressed. The lower figure shows the final weights w_s given by the robust averaging algorithm when applied to uncompressed data (circles) and compressed data (crosses).

where $\mathbf{I}$ denotes the identity matrix. If the perturbation ϵ_s obeys $|\epsilon_s| \leq \epsilon$, i.e., $\|\mathbf{e}\|_\infty \leq \epsilon$, then by using standard result on the induced ∞-norm [19], the largest possible perturbation in the weights w_s is given by

$$\|\Delta \mathbf{w}\|_\infty = \|\mathbf{G}\|_\infty \epsilon. \tag{22}$$

Therefore, if $\|\mathbf{G}\|_\infty$ is small, then the use of $\mathbf{y}_s$ will cause only a small change in w_s. Also, if the spectral norm $\rho(F)$ of the matrix $\mathbf{F}$ is bounded by unity, then the fixed point is locally stable [2]. Unfortunately, the exact computation of $\rho(F)$ and $\|\mathbf{G}\|_\infty$ requires the uncompressed data. We show in Appendix A (online supplemental material), how we can approximate them using compressed data. In the Appendix (online supplemental material), we will evaluate the size of perturbation and local stability of the fixed-point iteration by using data collected from two WSN deployments.

3.4.2 Large Perturbation Analysis

The local perturbation analysis is Section 3.4.1 applies when the value of ϵ_s is small. However, for the practical situation considered in this paper, this is generally not true. For example, for $m = 200$ and $p = 80$, we use Monte-Carlo simulation to find that there is a 95 percent probability that ϵ_s is in the range of $[-0.3, 0.3]$. Therefore, we derive an expression for the effect on w_s when the perturbation ϵ_s is large.

Proposition 1. *The perturbation of* $\|\mathbf{w^1} - \mathbf{w^0}\|$ *obeys*

$$\|\mathbf{w^1} - \mathbf{w^0}\| \leq \frac{\|\mathbf{T}(\hat{\mathbf{T}}_{\mathbf{wv}} - \mathbf{T}_{\mathbf{wv}})\|}{1 - \|\mathbf{T}(\hat{\mathbf{T}}_{\mathbf{wv}} - \mathbf{T}_{\mathbf{wv}})\|_L} \|\mathbf{w^0}\|, \tag{23}$$

where $\mathbf{T} = (\mathbf{T_{vw}}^{-1} - \mathbf{T_{wv}})^{-1}$ *and* $\|\bullet\|_L$ *denotes the Lipschitz operator norm. Note that the norm can be* $1-$, $2-$ *or* ∞*-norm, as long as the same type of norm is used.*

The proof of this proposition can be found in the Appendix B (online supplemental material). We will use this result to study the effect of the perturbation on the weights using data from an outdoor WSN in Section 4.

4 Application to Data Collected from Outdoor WSN Deployments

We have studied the behaviour of our robust averaging algorithm when it is applied to three commonly found sensor faults in WSNs, namely stuck-at-faults, offset, and variance degradation fault [17]. Our study shows that our robust averaging algorithm can deal with these faults, even when a mixture of them appear in a WSNs, see Appendix C (online supplemental material).

We have applied our robust averaging algorithm to data obtained from two outdoor WSN deployments. The results for the Belmont deployment (with 32 temperature sensors) are presented in Appendix E (online supplemental material).

Note that the results presented in this section are obtained from using the symmetric Bernoulli distribution to form the projection matrix. The results from using Categorical distribution to form the projection matrix are very similar and are not shown here.

4.1 The QCAT Deployment

This section describes the results of applying our robust averaging algorithm to the data obtained from an outdoor WSN testbed operated by CSIRO in their QCAT research facility in Brisbane, Australia. The WSN consists of five sensors measuring humidity and a block of data for each sensor consists of 400 points. Fig. 3 shows the sensor measurements. The first four sensors have the same trend, but the fifth sensor shows completely erroneous measurements.

4.1.1 Accuracy of Robust Averaging Using Compressed Data

We apply our robust averaging algorithm to the original sensor measurements as well as to the compressed data with $160 (= p)$ projections using a Bernoulli distributed projection matrix. The fixed-point iteration converges quickly. The upper plot in Fig. 4 shows the convergence of the weights w_s, when compressed data is used, in four

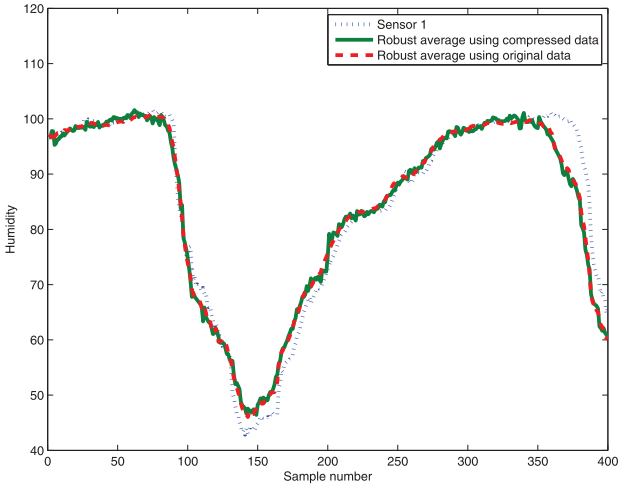

Fig. 5. The figure shows the robust averages against a working sensor (short dashes). The line with long dashes (resp. solid line) shows the robust average computed by applying the fixed-point iteration to the original data (resp. compressed data followed by reconstruction).

iterations. Similar convergence rate is observed when uncompressed data is used, see [9].

The lower figure in Fig. 4 shows the weights w_s for the sensors, where $s = 1, \ldots, 5$, obtained from using the original sensor readings as well as the compressed data. It can be seen that the two sets of weights are very close to each other. The solid line in Fig. 4 is at the level of $\frac{1}{5}$ which is the weight to use when all sensors are working. The figure shows that sensor 5, whose measurements are erroneous, has a very low weight.

Fig. 5 shows the average humidity given by our fixed-point iteration algorithm. The curve with long dashes shows the result obtained from applying the fixed-point iteration to the original sensor measurements. The solid curve shows the result obtained from applying the fixed-point iteration to the compressed data followed by reconstruction. It can be seem that there is not much loss in fidelity in using the compressed data as the two curves are almost on top of each other. The figure also shows the robust average captures the trend of the working sensor, which is given by the curve with short dashes.

4.1.2 Comparison with Other Fault Detection Techniques

We compare our proposed robust averaging method against three other fault detection methods. The first method is based on a classical statistical method—the Grubb's method [18]—for detecting the outlier in a set of univariate data. At each time t, we consider the readings from the n sensors as the data set. The Grubb's method computes the absolute standard score of each data point in the data set and compares it against a critical value (which depends on the sample size, significance level and t-distribution) to determine whether it is an outlier. If an outlier is detected, we remove it from the data set. We then compute the average of the data points remaining in the data set. This step is repeated for each time t. We call this method *Statistical*.

The second method is a fault detection technique for WSNs [17] based on using the local outlining factor (LOF) [4], which is a recent method to detect outliers without making any assumption on statistical distribution of data. This method uses: 1) LOF to give a probability that a data point at time t from sensor s is an outlier. LOF looks at the neighborhood with different number of closest neighbors (which is known as the $MinPts$ parameter in LOF), and compares the distance of the points within and outside the neighborhood to decide whether a point can be an outlier. 2) Recursive Bayesian update of the reputation of a node at time t based on the output of LOF. A node which is believed to be faulty at time t will have a low reputation at that time. We use the normalized reputation at each time as the weight to compute the weighted average. We call this method as *Reputation*.

TABLE 2
RMS Error of the Average Computed by Four Methods

n_w working + n_f faulty sensors	Robust average	*Statistical*	*Reputation*	*Bayes*
$n_w = 4$ and $n_f = 1$	2.3	3.4	3.3	2.8
$n_w = 3$ and $n_f = 1$	1.9	5.7	3.9	3.6
$n_w = 2$ and $n_f = 1$	2.6	9.8	24.4	4.7
$n_w = 4$ and $n_f = 2$	2.9±0.3	11.1±9.1	10.3±4.4	3.4±7.9
$n_w = 3$ and $n_f = 2$	3.8±0.4	14.8±10	12.5±5.3	4.0±0.9

The third method [26] determines the posteriori probability that a sensor in a WSN is faulty. It makes use of the fact that a group of working sensors should show similar trend and value. It fits a linear regression model over a short time interval to determine whether sensors have similar trend. In addition, a recursive Bayesian update where the posteriori probability of the combination of working/faulty sensors at time t becomes the prior at time $t+1$. The weighted average is computed using a weight proportional to the probability that a sensor is working at time t. We call this method *Bayes*. Note that both *Statistical* and *Reputation* treat the data at each time as independent while *Bayes* takes temporal correlation into consideration.

We apply *Statistical*, *Reputation*, and *Bayes* to the *uncompressed* data streams (Note: These three methods are designed for uncompressed data.), and our robust averaging to the compressed data with $p = 160$. For *Statistical*, the significance level is 0.05. For *Reputation*, the $MinPts$ parameter varies from 2 to the number of sensors minus 1. We use the standard average of the first four sensors (the working sensors) as the reference and compute the root-mean-square (RMS) error of the average obtained by the four methods. The results are shown in the first row of Table 2. It can be seem that the results are comparable though robust averaging has a slight edge over the other methods. Fig. 10 (online supplemental material) plots the time series of the average obtained by the four methods.

We next apply these four methods to data from four sensors (three working and one faulty) and three sensors (two working and one faulty). The reference is the average of the working sensors. The second and third rows of Table 2 show the RMS error of the four methods. It can be seem that our robust averaging method performs better than the other three methods. Fig. 6 plots the time series of the average given by the four methods against the reference. Note that the results in Table 2 are obtained from a particular choice of 2 (resp. 3) sensors from four working

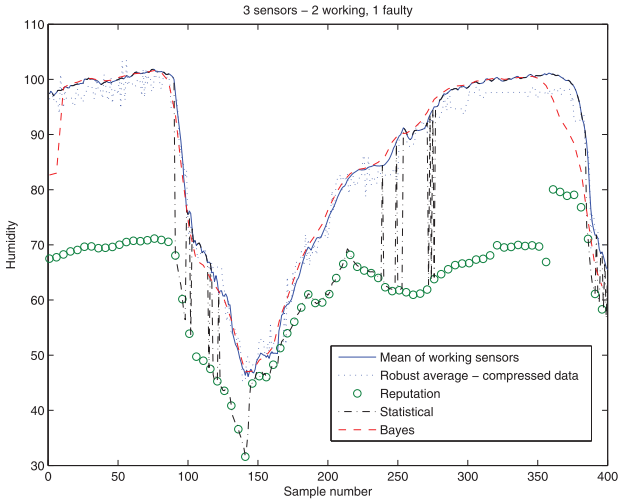

Fig. 6. Comparison of *Statistical*, *Reputation*, *Bayes*, and our robust averaging method for the QCAT data set. Data from three sensors (two working and one faulty) are used.

sensors, we repeated the experiment with other possible combinations and the results were similar.

We now consider the case with two faulty sensors. We retain the faulty sensor in the data set and introduce an artificial faulty sensor with stuck-at fault [17], which is a commonly observed fault in WSNs. The readings of this faulty sensor is stuck at the same value all the time. We apply the methods to data from six sensors (four working and two faulty) and five sensors (three working and two faulty). For a given number of sensors, we perform 10 experiments, where in each experiment, the sensor with stuck-at fault is stuck at a different value in the range $[20, 110]$. The last two rows of Table 2 show the average and standard deviation (computed over 10 experiments) of the RMS errors. The proposed robust averaging algorithm performs better than the other methods. Also, the performance of the robust averaging is stable while that of the other three algorithms fluctuates with the "stuck-at value" being used. Comparison of time series for the 5 and 6 sensor cases are shown in Figs. 11 and 12 (online supplemental material).

4.1.3 Effect of p on Accuracy and Resource Requirements

The key advantage of the proposed robust averaging method is a reduction in computation time in going from the setup in Figs. 1 and 2. For $p = 160$ projections, the method in Fig. 1, which requires the reconstruction of five time series and one run of the robust average algorithm, took 2.29 s to complete, while the method in Fig. 2, which requires only one reconstruction and one run of the robust average algorithm, took 0.48 s and is, therefore, 4.8 times faster. These results were obtained from running Matlab 2009b on a MacBook.

The bandwidth saving in using compressive sensing (compared with the case where compressive sensing is not used) can also be computed. The QCAT deployment uses a 10-bit A/D converter. If compressive sending is not used, then each sensor will need to send $mb = 4,000$ (where $b = 10$) bits of data to the data fusion centre. If compressive sensing is used, and assuming $p = 160$ projections are used, the number of bits of data that has to be sent by each sensor is $p(b + 1 + \lceil \log_2(m) \rceil) = 3,200$ bits. This represents a 20 percent saving. Note that the bandwidth consumption for the two methods in Figs. 1 and 2 are identical. This bandwidth saving is with respect to methods that do not use compressive sensing. Note also that the above calculation assumes that the sensor is one hop away from the data fusion centre; however, the percentage saving is identical even for a multihop topology.

TABLE 3
This Table Shows the Effect of the Number of Projections p per Sensor on Four Performance Metrics

p	40	60	80	100	120	140	160
Metric 1	80	80	80	72	79	78	80
Metric 2	80	70	60	50	40	30	20
Metric 3	9.20	8.27	4.10	4.48	1.70	2.64	1.11
Metric 4	5.43	10.90	8.12	10.58	3.96	14.80	2.01

The above results are obtained by using 160 projections per sensor. We investigate the effect of the number of projections per sensor on performance. We use four different performance metrics:

1. The percentage reduction in computation time;
2. The percentage of bandwidth saving when compressive sensing is used;
3. Relative error in reconstructing the robust average $\mathbf{r}$, expressed as a percentage;
4. Relative error in the estimation of the weights w_s, expressed as a percentage.

Metrics 1 and 2 are calculated in the same way in the previous paragraphs. For 3 and 4, the "true" robust average and weights are computed when uncompressed data are used. Table 3 shows the results for 40 to 160 projections per sensors. If we are willing to accept a 5 percent error in reconstruction of robust average, then we can reduce the number of projections further.

A possible method to choose the parameter p is to build a predictive model of p based on historical data. For example, [30] uses support vector machine to predict the number of projections needed for a tracking problem. An alternative approach is to adaptively adjust the number of projections based on the data, see [10], using a Bayesian approach. We will not tackle the problem here and will leave it as future work.

We have also performed perturbation analysis (Section 3.4) on this data set and compared our robust averaging method against random sampling. The results can be found in Appendix D (online supplemental material).

4.2 Implementation on Sensor Nodes

We implemented the fixed-point iteration of Algorithm 1 on a real-world sensor platform to understand its resource requirements. We use the Fleck3b sensor node platform [14], which features an 8-MHz Atmel Amega1281 microcontroller with 8-KB Random Access Memory (RAM) and 128-KB flash programmable Read-Only Memory (ROM), as our hardware test environment. We use Fleck OS (FOS) [11] as our software test environment. FOS is a C-based cooperative multithreaded operating system for WSNs.

TABLE 4
The Resource Requirements in Fleck3b Sensor Nodes

p	RAM (byte)	ROM (byte)	Computation time (ms)	Energy consumption (mJ)
50	1,338	6,092	616	19.7
60	1,538	6,292	734	23.5
70	1,738	6,492	852	27.3
80	1,938	6,692	890	28.8
90	2,138	6,892	1,088	34.8

The fixed-point iteration is implemented as an application thread in FOS, and is waken up whenever the projection buffer is full.

We implemented our fixed-point iteration algorithm in C and ran it on the Fleck3b platform using different number of projections per sensor with the data set in Section 4.1. Table 4 shows the RAM and ROM usage, together with computation time with different number of projections p per sensor. Note that there are five sensors in this data set; therefore, if each sensor uses p projections, then the robust averaging algorithm works with $5p$ projections. As expected, the resource usage increases with the number of projections. Note that the increase is almost linear and this makes the algorithm a scalable solution in WSNs. Table 4 also shows that the proposed algorithm is fairly affordable in resource-impoverished sensor platforms. Let us compute the duty cycle of the microcontroller. A block of data contains $m = 400$ data points obtained at a sampling rate of 1 per 10 seconds. If $p = 80$ projections are used per sensor, it takes a sensor 890 ms to do the computation, so the duty cycle is less than 0.3 percent, which is very affordable. Table 4 also shows the corresponding energy consumption; note that Flecks use a voltage of 3V.

5 Related Work

There are a number of papers investigating how compressive sensing can be applied in WSNs. We can classify them according to whether projections are computed "temporally" or "spatially". In [16], "temporal" projections are used, where each sensor computes projections of its own data and sends it to the data fusion centre. It also discusses how fault tolerance can be realized based on the framework discussed in Fig. 1, where all signals are reconstructed at the data fusion centre. However, this paper adopts the framework in Fig. 2, where only a minimal number of signals have to be reconstructed to achieve computation efficiency at the data fusion centre.

Other works in applying compressive sensing in WSNs view the spatial data from the sensors at each time instance as an image or snapshot, and compute projections of the sensor data for each snapshot. All these works aim at obtaining a sufficient accurate snapshot of the sensor field by using as little energy as possible. The work [3] realizes this goal by using an additive radio channel to compute projections. The works in [29], [23], [10], [25] compute projections by passing messages between sensors, while the work in [32], [31] consider random samples as projections.

There is a rich literature in fault tolerance in WSNs. The work [27] discusses the types of faults that commonly appear in WSNs. The papers [21], [17], [26] use a Bayesian approach to detect whether sensors are faulty. The work [28] proposes to detect sensor faults by using recurrent neural networks.

The problem of computing a robust average also appears in other fields, e.g., reputation ranking and fair assessment of students' work in education [20]. The paper [22] studies the statistical properties of using a maximum-likelihood estimator for reputation ranking. However, this estimator experiences convergence problem [13], a fact which we also pointed out in Section 3.1.

The use of projection matrix as a dimensionality reduction method is fairly well known in the data mining and the machine learning communities, see [24]. The framework depicted in Fig. 2 opens up the possibility of using some of the algorithms developed in these communities to WSNs, e.g., performing clustering using the compressed data. A feature of our algorithm is that it works directly with the compressed data without decompressing them. There is also some recent effort in using compressed data directly for classification and detection, see [12].

6 Conclusions

In this paper, we propose a method to compute a robust average of sensor measurements in a wireless sensor network in a computationally efficient manner. Our method exploits compressive sensing for efficient data transmission. Furthermore, our method integrates norm-preserving property of random projection matrices and compressive sensing so that the data fusion centre can work directly with compressed data without first decompressing them. This means the data fusion centre will only need to perform decompression once and this represents a great reduction in computation requirements. We have applied our algorithm to data obtained from two WSN deployments.

References

[1] D. Achlioptas, "Database-Friendly Random Projections: Johnson-Lindenstrauss with Binary Coins," *J. Computer and System Sciences*, vol. 66, pp. 671-687, Jan. 2003.

[2] K.J. Åström and R.M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton Univ. Press, 2008.

[3] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak, "Joint Source-Channel Communication for Distributed Estimation in Sensor Networks," *IEEE Trans. Information Theory*, vol. 53, no. 10, pp. 3629-3653, Oct. 2007.

[4] M.M. Breunig, H.-P. Kriegel, R.T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, pp. 93-104, 2000.

[5] N. Bulusu and S. Jha, *Wireless Sensor Network Systems*. Artech, 2005.

[6] E. Candes and J. Romberg, *ℓ_1-Magic : Recovery of Sparse Signals via Convex Programming*, http://www.acm.caltech.edu/l1magic/, 2013.

[7] E. Candes, J. Romberg, and T. Tao, "Robust Uncertainty Principles: Exact Signal Reconstruction from Highly Incomplete Frequency Information," *IEEE Trans. Information Theory*, vol. 52, no. 2, pp. 489-509, Feb. 2006.

[8] E. Candes and T. Tao, "Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies?" *IEEE Trans. Information Theory*, vol. 52, no. 12, pp. 5406-5425, Dec. 2006.

[9] C.T. Chou, A. Ignjatovič, and W. Hu, "Efficient Computation of Robust Average in Wireless Sensor Networks using Compressive Sensing," technical report ftp://ftp.cse.unsw.edu.au/pub/doc/papers/UNSW/0915.pdf, UNSW, 2009.

[10] C.T. Chou, R. Rana, and W. Hu, "Energy Efficient Information Collection in Wireless Sensor Networks Using Adaptive Compressive Sensing," *Proc IEEE 34th Conf. Local Computer Networks (LCN '09),* 2009.

[11] P. Corke and P. Sikka, "Demo Abstract: FOS - A New Operating System for Sensor Networks," *Proc. Fifth European Conf. Wireless Sensor Networks (EWSN),* 2008.

[12] M.A. Davenport, P.T. Boufounos, M.B. Wakin, and R.G. Baraniuk, "Signal Processing with Compressive Measurements," *IEEE J. Selected Topics in Signal Processing,* vol. 4, no. 2, pp. 445-460, Apr. 2010.

[13] C. de Kerchove and P.V. Dooren, "Iterative Filtering for a Dynamical Reputation System," *Arxiv preprint arXiv:0711.3964,* Jan. 2007.

[14] T.L. Dinh, W. Hu, P. Sikka, P. Corke, L. Overs, and S. Brosnan, "Design and Deployment of a Remote Robust Sensor Network: Experiences from an Outdoor Water Quality Monitoring Network," *Proc. IEEE Ann. Conf. Local Computer Networks,* pp. 799-806, 2007.

[15] D. Donoho, "Compressed Sensing," *IEEE Trans. Information Theory,* vol. 52, no. 4, pp. 1289-1306, Apr. 2006.

[16] M. Duarte, M. Wakin, D. Baron, and R. Baraniuk, "Universal Distributed Sensing via Random Projections," *Proc. Fifth Int'l Conf. Information Processing in Sensor Networks (IPSN '06),* Apr. 2006.

[17] S. Ganeriwal, L. Balzano, and M. Srivastava, "Reputation-Based Framework for High Integrity Sensor Networks," *Trans. Sensor Networks,* vol. 4, no. 3, article 15, May 2008.

[18] F. Grubbs, "Procedures for Detecting Outlying Observations in Samples," *Technometrics,* vol. 11, no. 1, pp. 1-21, 1969.

[19] R.A. Horn and C.R. Johnson, *Matrix Analysis.* Cambridge Univ. Press, 1990.

[20] A. Ignjatovic, C.T. Lee, P. Compton, C. Cutay, and H. Guo, "Computing Marks from Multiple Assessors using Adaptive Averaging," *Proc. Int'l Conf. Eng. Education (ICEE),* 2009.

[21] B. Krishnamachari and S. Iyengar, "Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks," *IEEE Trans. Computers,* vol. 53, no. 3, pp. 241-250, Jan. 2004.

[22] P. Laureti, L. moret, Y. Zhang, and Y. Yu, "Information Filtering via Iterative Refinement," *Europhysics Letters,* vol. 75, Jan. 2006.

[23] S. Lee, S. Pattem, and M. Sathiamoorthy, "Spatially-Localized Compressed Sensing and Routing in Multi-Hop Sensor Networks," *Proc. Third Int'l Conf. Geosensor Networks (GSN),* 2009.

[24] P. Li, T. Hastie, and K. Church, "Very Sparse Random Projections," *Proc. 12th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '06),* Aug. 2006.

[25] C. Luo, F. Wu, J. Sun, and C. Chen, "Compressive Data Gathering for Large-Scale Wireless Sensor Networks," *Proc. ACM Mobicom '09,* Sept. 2009.

[26] K. Ni and G. Pottie, "Bayesian Selection of Non-Faulty Sensors," *Proc. IEEE Int'l Symp. Information Theory,* pp. 616-620, 2007.

[27] K. Ni, N. Ramanathan, M. Chehade, L. Balzano, S. Nair, S. Zahedi, E. Kohler, G. Pottie, M. Hansen, and M. Srivastava, "Sensor Network Data Fault Types," *Trans. Sensor Networks,* vol. 5, no. 3, article 25, May 2009.

[28] O. Obst, "Poster Abstract: Distributed Fault Detection Using a Recurrent Neural Network," *Proc. Int'l Conf. Information Processing in Sensor Networks (IPSN '09),* pp. 373-374, 2009.

[29] G. Quer, R. Masiero, D. Munaretto, M. Rossi, J. Widmer, and M. Zorzi, "On the Interplay between Routing and Signal Representation for Compressive Sensing in Wireless Sensor Networks," *Proc. Information Theory and Applications Workshop (ITA '09),* 2009.

[30] R. Rana, W. Hu, T. Wark, and C.T. Chou, "An Adaptive Algorithm for Compressive Approximation of Trajectory (AACAT) for Delay Tolerant Networks," *Proc. Eighth European Conf. Wireless Sensor Networks (EWSN '11),* Feb. 2011.

[31] R.K. Rana, C.T. Chou, S.S. Kanhere, N. Bulusu, and W. Hu, "Ear-Phone: An End-to-End Participatory Urban Noise Mapping System," *Proc. ACM/IEEE Ninth Int'l Conf. Information Processing in Sensor Networks (IPSN '10),* Apr. 2010.

[32] Y. Shen, W. Hu, R. Rana, and C.T. Chou, "Non-Uniform Compressive Sensing in Wireless Sensor Networks: Feasibility and Application," *Proc. Seventh Int'l Conf. Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP),* pp. 271-276, 2011.

[33] H. White, "Maximum Likelihood Estimation of Misspecified Models," *Econometrica: J. Econometric Soc.,* vol. 50, pp. 1-25, Jan. 1982.

[34] G. Zames, "On the Input-Output Stability of Time-Varying Nonlinear Feedback Systems Part One: Conditions Derived Using Concepts of Loop Gain, Conicity, and Positivity," *IEEE Trans. Automatic Control,* vol. AC-11, no. 2, pp. 228-238, Jan. 1966.

Chun Tung Chou received the BA degree in engineering science from the University of Oxford, United Kingdom, and the PhD degree in control engineering from the University of Cambridge, United Kingdom. He is an associate professor at the School of Computer Science and Engineering, University of New South Wales, Sydney, Australia. His current research interests include wireless networks, participatory sensing, compressive sensing, nanocommunication, and network optimization. He is a member of the IEEE.

Aleksandar Ignjatovic received the bachelor's and master's degrees in mathematics from the University of Belgrade, former Yugoslavia, and the PhD degree in mathematical logic from the University of California at Berkeley, where he had University of California Regents' Fellowship. His thesis "Fragments of Arithmetic and Lengths of Proofs" was supervised by Professor Jack Silver, one of the foremost set theorists. After graduation, he got a tenure track position as an assistant professor at the Carnegie Mellon University, where he taught for 5 years at the Department of Philosophy and the CMU's Program for Pure and Applied Logic. He left CMU to found with his business partner and attorney Nick Carlin their start up "Kromos Technology." The company's CEO was Raj Parekh, former CTO of Sun Microsystems and among their investors and Board members were former President and COO of AMD Atiq Raza, the former CEO of Fiberlane, Cerent and Siara Raj Singh, as well as Redwood Venture Partners. After the company was acquired by "Comstellar Technologies," he joined in 2002 the School of Computer Science and Engineering at UNSW, where he is teaching algorithms. His research interests include sampling theory and signal processing, applications of mathematical logic to computational complexity theory, algorithms for embedded systems design as well as educational use of puzzles for teaching serious problem solving techniques.

Wen Hu received the PhD degree in sensor networks from the University of New South Wales (UNSW). He is a senior research scientist and research team leader at the Autonomous Systems Laboratory in the Commonwealth Scientific and Industrial research Organisation's Information and Communication Technologies (CSIRO ICT) Centre. His current research interests include low-power communications, compressive sensing, and security issues in sensor networks. He is also an adjunct associate professor at Queensland university of Technology and holds a visiting research position at UNSW. He is a senior member of the IEEE.