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 Ignjatoviȁ, 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 in order 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 generalised 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 etc. [17], [27] etc.) are common in WSNs, many WSN designers choose to deploy redundant sensors so that neighbouring 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 neighbouring 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.

Figure 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 Figure 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

- *C.T Chou and A. Ignjatoviȁ are with the School of Computer Science and Engineering, University of New South Wales, Sydney, Australia. E-mail: ctchou@cse.unsw.edu.au, ignjat@cse.unsw.edu.au.*
- *W. Hu is with the Autonomous Systems Laboratory, CSIRO ICT Centre, Brisbane, Australia. E-mail: Wen.Hu@csiro.au*

with 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. Firstly, we work directly with the compressed data, whose dimension is only a fraction of that of the original sensor readings. Secondly, 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 organised as follows. In Section 2, we define the set-up 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.

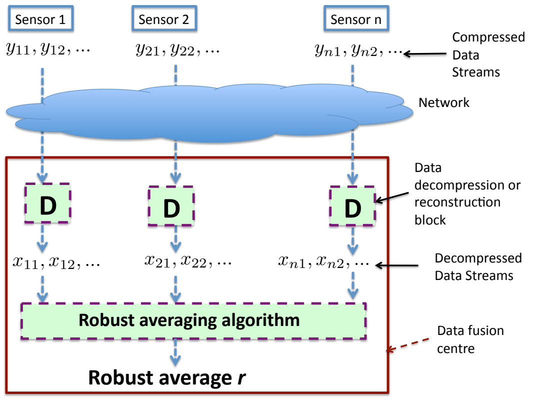

Fig. 1. This method determines the robust average after decompressing all the signals. Because of single processor environment, the time needed to decompress n compressed data streams (n = number of sensors) equals to n times of the time needed to decompress one data stream.

2 Models

2.1 Problem setting: Basic

We consider a WSN with n sensors indexed by $s = 1, ..., n$. We assume that the sensors are time synchronised 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 $s = 1, .., n$ and $t = 1, ..., m$. Note: Table 1 contains a summary of notation used in this paper.

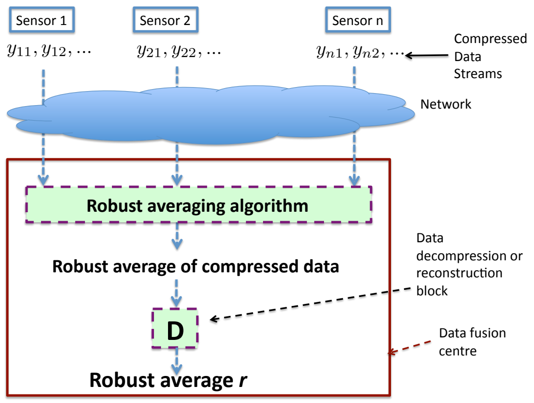

Fig. 2. The new method proposed in this paper. Note that this method requires only one decompression.

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)		

TABLE 1
Notation used in this paper.

As depicted in Figure 2, the sensors will not send their readings x_{st} directly to the data fusion centre in order 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} $(t = 1, ..., m)$ can be expressed in matrix form, as follows:

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

$$\Leftrightarrow \mathbf{y}_s = \mathbf{\Phi} \mathbf{x}_s \quad (2)$$

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 synchronised, this can be achieved by using a time dependent seed for the pseudo-random 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 $k = 1, ..., p$) 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 analogue-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 equation (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 analysed 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:

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

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 equation (3) by this expression, it can be shown that r_t can be recovered from the following optimisation problem

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

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

$$\mathbf{r} = \sum_{s=1}^{n} w_s \mathbf{x}_s \quad (5)$$

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

$$v_s = \|\mathbf{x_s} - \sum_{i=1}^{n} w_i \mathbf{x_i}\|_2^2 \quad \text{for } s = 1, ..., n \quad (6)$$

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

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 since $w_s \geq 0$ (for $s = 1, ..., n$) and $\sum_{s=1}^{n} w_s = 1$. Intuitively, v_s measures the deviation of sensor s's measurements $\mathbf{x}_s$ 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 since a sensor may only display faults over a limited period of time.

In order 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 equation (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) that, for stuck-at faults, a small value of λ will only create a small bias in the estimation of the weights w_s.

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].

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 $\ell$-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.
```

3.2 Computing robust average from $\mathbf{y}_s$

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

In order 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 equation (5) from $\mathbf{y}_s$. (We will explain in section 3.2.2 how w_s can be computed from $\mathbf{y}_s$.) By pre-multiplying both sides of equation (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 equation (2). The importance of equation (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 realised by synchronising 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 realised 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 $\mathbf{y}_s$*

In order 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 $\mathbf{d_1}, \mathbf{d_2}, ..., \mathbf{d_q}$. 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

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

□

Since the projection matrices that are commonly used in compressive sensing obey the JL Lemma, this means that v_s in Eq. (6) can be approximately computed from $\mathbf{y}_s$, as follows:

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

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 $\{y_{sk}\}$ is 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} is used instead, except that the variance becomes larger. The following derivation will also show the trade-off 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:

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

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

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

$$\mathbb{E}[\tilde{e}_{s_1 k_1} \tilde{e}_{s_2 k_2}] = \begin{cases} \frac{m}{p}\sigma_s^2 & \text{for } s_1 = s_2 \text{ and } k_1 = k_2 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

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 $\{y_{sk}\}$ is 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 trade-off 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 s-th element is v_s and w_s. In order 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 equation (6) and $\mathbf{T_{vw}}$ denote the operator that maps $\mathbf{v}$ to $\mathbf{w}$ by using equation (7). If the fixed point of equations (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 $\mathbf{y_s}$ is used instead. In this case, equation (6) is replaced by

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

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 $\mathbf{y_s}$ is used, the fixed point algorithm iterates between equations (15) and (7) instead of equations (6) and (7). Let $\hat{\mathbf{T}}_{\mathbf{wv}}$ denote the operator that maps $\mathbf{w}$ to $\mathbf{v}$ defined by equation (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 equations (13) and (16) to obtain:

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

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

By using equations (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 equations (19) and (20), it can be shown that:

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

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. E.g., for $m = 200$ and $p = 80$, we use Monte-Carlo simulation to find that there is a 95% 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

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

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 3 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 5 sensors measuring humidity and a block of data for each sensor consists of 400 points. Figure 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 Figure 4 shows the convergence of the weights w_s, when compressed data is used, in 4 iterations. Similar convergence rate is observed when uncompressed data is used, see [9].

The lower figure in Figure 4 shows the weights w_s for the sensors, where $s = 1, ..., 5$, 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 figure 15 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.

Figure 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 neighbourhood with different number of closest neighbours (which is known as the $MinPts$ parameter in LOF), and compares the distance of the points within and outside the neighbourhood 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 normalised reputation at each time as the weight to compute the weighted average. We call this method as *Reputation*.

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 3 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. Figure 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 4 sensors (3 working and 1 faulty) and 3 sensors (2 working and 1 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. Figure 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 4 working sensors, we repeated the experiment with other possible combinations and the results were similar.

We now consider the case with 2 faulty sensors. We retain the faulty sensor in the dataset 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 6 sensors (4 working and 2 faulty) and 5 sensors (3 working and 2 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 shows 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 Figure 11 and 12 (online supplemental material).

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

TABLE 2
RMS error of the average computed by four methods.

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 set-up in Figure 1 to Figure 2. For $p = 160$ projections, the method in Figure 1, which requires the reconstruction of 5 time series and one run of the robust average algorithm, took 2.29s to complete, while the method in

Figure 2, which requires only one reconstruction and one run of the robust average algorithm, took 0.48s 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 analogue-to-digital converter. If compressive sending is not used, then each sensor will need to send $mb = 4000$ (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 $p(b + 1 + \lceil \log_2(m) \rceil)$ =3200 bits. This represents a 20% saving. Note that the bandwidth consumption for the two methods in Figures 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 multi-hop topology.

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 is used. Table 3 shows the results for 40 to 160 projections per sensors. If we are willing to accept a 5% 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 8MHz Atmel Amega1281 micro-controller with 8KB Random Access Memory (RAM) and 128 KB flash programmable Read-Only Memory (ROM), as our hardware test environment.

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

TABLE 3
This table shows the effect of the number of projections p per sensor on four performance metrics.

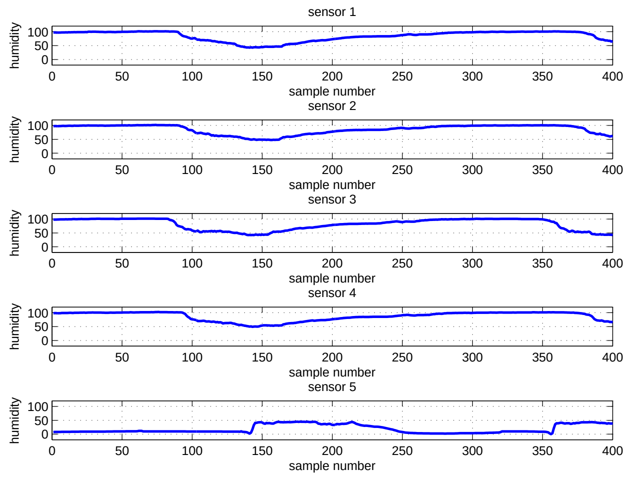

Fig. 3. Humidity reading from the QCAT deployment. Note that the readings from sensor 5 is very different from the first 4 sensors. Note: All plots use the same scale.

We use Fleck OS (FOS) [11] as our software test environment. FOS is a C-based cooperative multi-threaded operating system for WSNs. 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 5 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

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

TABLE 4
The resource requirements in Fleck3b sensor nodes.

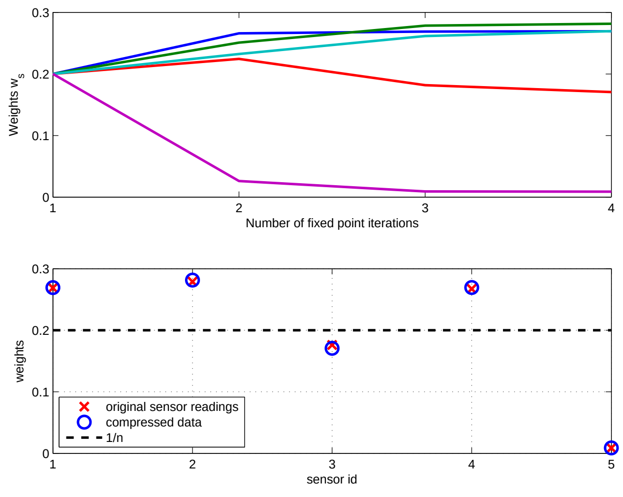

Fig. 4. The upper figure shows that the convergence of the weights w_s, when the robust averaging algorithm is applied to the compressed. The lower figure shows the final weights w_s given by the robust averaging algorithm when applied to uncompressed data (circles) and compressed data (crosses).

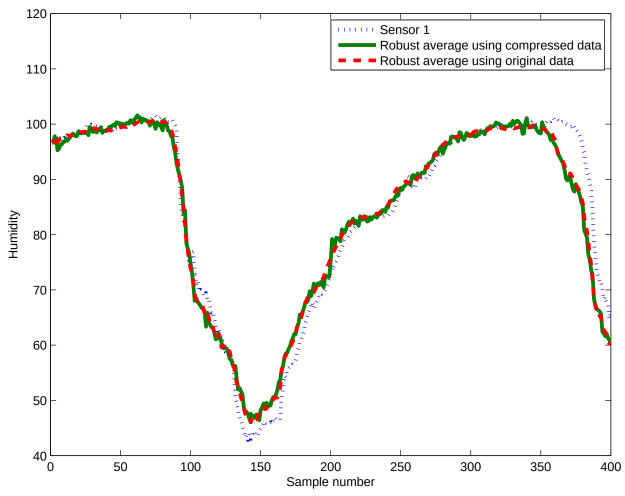

Fig. 5. The figure shows the robust averages against a working sensor (short dashes). The line with long dashes (resp. solid line) shows the robust average computed by applying the fixed point iteration to the original data (resp. compressed data followed by reconstruction).

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 micro-controller. 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 890ms to do the computation, so the duty cycle is less than 0.3%, which is very affordable. Table 4 also shows the corresponding energy consumption; note that Flecks use a voltage of 3V.

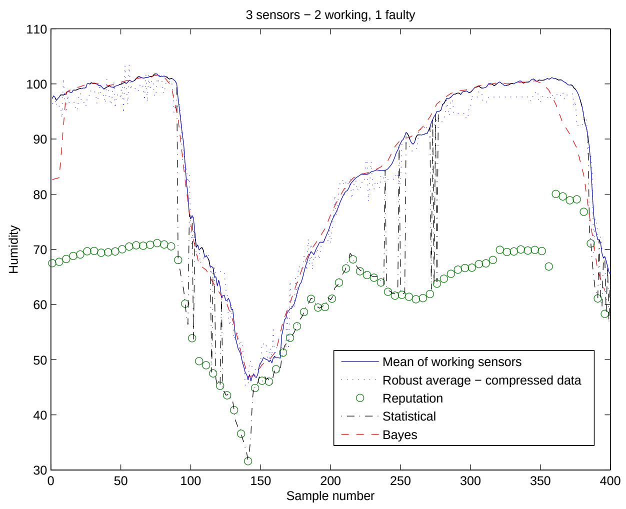

Fig. 6. Comparison of *Statistical*, *Reputation*, *Bayes* and our robust averaging method for the QCAT data set. Data from three sensors (2 working and 1 faulty) are used.

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 realised based on the framework discussed in Figure 1 where all signals are reconstructed at the data fusion centre. However, this paper adopts the framework in Figure 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] realises 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 Figure 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. *Journal of Computer and System Sciences*, Jan 2003.

[2] K. J. Åström and R. M. Murray. *Feedback systems: an introduction for scientists and engineers*. Princeton University Press, 2008.

[3] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak. Joint source–channel communication for distributed estimation in sensor networks. *Information Theory, IEEE Transactions on*, 53(10):3629 – 3653, Oct 2007.

[4] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying Density-Based Local Outliers. *SIGMOD*, pages 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/.

[7] E. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489 – 509, Feb 2006.

[8] E. Candes and T. Tao. Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies? *Information Theory, IEEE Transactions on*, 52(12):5406–5425, Dec. 2006.

[9] C. T. Chou, A. Ignjatovic, 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. *IEEE 34th Conference on Local Computer Networks (LCN 2009)*, 2009.

[11] P. Corke and P. Sikka. Demo abstract: FOS – a new operating system for sensor networks. In *Proceedings of Fifth European conference on 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 Journal of Selected Topics in Signal Processing*, 4(2):445–460, 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. *Local Computer Networks, Annual IEEE Conference on*, pages 799–806, 2007.

[15] D. Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289 – 1306, Apr 2006.

[16] M. Duarte, M. Wakin, D. Baron, and R. Baraniuk. Universal distributed sensing via random projections. *IPSN '06: Proceedings of the 5th international conference on Information processing in sensor networks*, Apr 2006.

[17] S. Ganeriwal, L. Balzano, and M. Srivastava. Reputation-based framework for high integrity sensor networks. *Transactions on Sensor Networks*, 4(3), May 2008.

[18] F. Grubbs. Procedures for Detecting Outlying Observations in Samples. *Technometrics*, 11(1):1–21, 1969.

[19] R. A. Horn and C. R. Johnson. *Matrix analysis*. CUP, 1990.

[20] A. Ignjatovic, C. T. Lee, P. Compton, C. Cutay, and H. Guo. Computing marks from multiple assessors using adaptive averaging. In *International Conference on Engineering Education (ICEE)*, 2009.

[21] B. Krishnamachari and S. Iyengar. Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. *Computers, IEEE Transactions on*, 53(3):241 – 250, Jan 2004.

[22] P. Laureti, L. Moret, Y. Zhang, and Y. Yu. Information filtering via iterative refinement. *Europhysics Letters*, Jan 2006.

[23] S. Lee, S. Pattem, and M. Sathiamoorthy. Spatially-localized compressed sensing and routing in multi-hop sensor networks. *3rd International Conference on Geosensor Networks (GSN)*, 2009.

[24] P. Li, T. Hastie, and K. Church. Very sparse random projections. *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, Aug 2006.

[25] C. Luo, F. Wu, J. Sun, and C. Chen. Compressive data gathering for large-scale wireless sensor networks. *MobiCom '09: Proceedings of the 15th annual international conference on Mobile computing and networking*, Sep 2009.

[26] K. Ni and G. Pottie. Bayesian Selection of Non-Faulty Sensors. In *2007 IEEE International Symposium on Information Theory*, pages 616–620. IEEE, 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. *Transactions on Sensor Networks*, 5(3), May 2009.

[28] O. Obst. Poster abstract: Distributed fault detection using a recurrent neural network. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN 2009)*, pages 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. *Information Theory and Applications Workshop (ITA 2009)*, 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. In *EWSN'11: Proceedings of the 8th European conference on Wireless sensor networks*. Springer-Verlag, 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. In *IPSN '10: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM Request Permissions, Apr. 2010.

[32] Y. Shen, W. Hu, R. Rana, and C. T. Chou. Non-uniform compressive sensing in wireless sensor networks: Feasibility and application. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2011 Seventh International Conference on*, pages 271–276, 2011.

[33] H. White. Maximum likelihood estimation of misspecified models. *Econometrica: Journal of the Econometric Society*, 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. *Automatic Control, IEEE Transactions on*, 11(2):228 – 238, Jan 1966.

Chun Tung Chou Chun Tung Chou is an Associate Professor at the School of Computer Science and Engineering, University of New South Wales, Sydney, Australia. He received his BA in Engineering Science from the University of Oxford, UK and his Ph.D. in Control Engineering from the University of Cambridge, UK. His current research interests are wireless networks, participatory sensing, compressive sensing, nano-communication and network optimisation.

Aleksandar Ignjatoviċ Aleks got his Bachelor's and Master's degrees in Mathematics at the University of Belgrade, former Yugoslavia, and Ph.D. in Mathematical Logic at 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" Aleks 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 Wen Hu 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 are low-power communications, compressive sensing, and security issues in sensor networks. Hu has a PhD in sensor networks from the University of New South Wales (UNSW). 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 IEEE.

Supplementary materials to "Efficient computation of robust average of compressive sensing data in wireless sensor networks in the presence of sensor faults"

Appendix A
Computation of $\rho(F)$ and $\|G\|_\infty$ using compressed data

Note that the perturbation analysis in Section 3.4.1 assumes that we evaluate the Jacobian matrices at the fixed point $\mathbf{w^0}$ and $\mathbf{v^0}$ determined by the original data $\{\mathbf{x_s}\}$. Since the original data is assumed to be not available, we will estimate the perturbation by using the fixed point given by the compressed data instead. This is straightforward for the Jacobian matrix $\frac{\partial \mathbf{T_{vw}}}{\partial \mathbf{v}}$ (whose expressions are given in section A.1). However, the Jacobian matrix $\frac{\partial \mathbf{T_{wv}}}{\partial \mathbf{w}}$ depends on the data; in fact, the (i,j)-element of the matrix is:

$$\left[\frac{\partial \mathbf{T_{wv}}}{\partial \mathbf{w}}\right]_{ij} = 2\mathbf{x_j}^T(\sum_{q=1}^{n} w_q \mathbf{x_q} - \mathbf{x_i}) \tag{24}$$

Since the projection matrix $\mathbf{\Phi}$ approximately preserves the ℓ_2-norm with a high probability, we expect that the projection matrix $\mathbf{\Phi}$ will also approximately preserve the inner product because the inner product of two vectors $\mathbf{u}$ and $\mathbf{v}$ can be computed by:

$$\mathbf{u}^T\mathbf{v} = \frac{\|\mathbf{u}+\mathbf{v}\|_2^2 - \|\mathbf{u}-\mathbf{v}\|_2^2}{4} \tag{25}$$

In other words, we will use:

$$\left[\frac{\partial \mathbf{T_{wv}}}{\partial \mathbf{w}}\right]_{ij} \approx 2\mathbf{y_j}^T(\sum_{q=1}^{n} w_q \mathbf{y_q} - \mathbf{y_i}) \tag{26}$$

A.1 Expressions of Jocobian matrices

The Jacobian matrix $\frac{\partial \mathbf{T_{vw}}}{\partial \mathbf{v}}$ is given by $\mathbf{J_1 J_2}$ where

$$[\mathbf{J_1}]_{ij} = \frac{\delta_{ij}}{\sum_{k=1}^{n}\tilde{w}_k} - \frac{\tilde{w}_i}{(\sum_{k=1}^{n}\tilde{w}_k)^2} \tag{27}$$

$$[\mathbf{J_2}]_{ij} = \frac{\frac{v_i}{(\sum_{k=1}^{n} v_k)^2} - \frac{\delta_{ij}}{\sum_{k=1}^{n} v_k}}{(\frac{1}{\sum_{k=1}^{n} v_k} + \lambda)^2} \tag{28}$$

where

$$\tilde{w}_s = \frac{1}{\frac{1}{\sum_{k=1}^{n} v_k} + \lambda}, \tag{29}$$

$[\bullet]_{ij}$ denotes the (i,j)-element of a matrix and $\delta_{ij} = 1$ if $i = j$ and is zero otherwise.

The Jacobian matrix $\frac{\partial \mathbf{T_{wv}}}{\partial \mathbf{w}}$ is

$$\left[\frac{\partial \mathbf{T_{wv}}}{\partial \mathbf{w}}\right]_{ij} = 2\mathbf{x_j}^T(\mathbf{r} - \mathbf{x_i}) \tag{30}$$

Appendix B
Proof of Proposition 1

By using equations (16) and (17), we have

$$\mathbf{w^1} = \mathbf{T_{vw}}(\mathbf{\hat{T}_{wv}}(\mathbf{w^1})) \tag{31}$$

We will "center" this equation by using the following operations:

$$\begin{aligned}
& \mathbf{w^1} &=& \mathbf{T_{vw}}(\mathbf{\hat{T}_{wv}}(\mathbf{w^1})) \\
\Leftrightarrow\ & \mathbf{T_{vw}^{-1}}(\mathbf{w^1}) &=& \mathbf{\hat{T}_{wv}}(\mathbf{w^1}) \\
\Leftrightarrow\ & \mathbf{T_{vw}^{-1}}(\mathbf{w^1}) - \mathbf{T_{wv}}(\mathbf{w^1}) &=& \mathbf{\hat{T}_{wv}}(\mathbf{w^1}) - \mathbf{T_{wv}}(\mathbf{w^1}) \\
\Leftrightarrow\ & (\mathbf{T_{vw}^{-1}} - \mathbf{T_{wv}})(\mathbf{w^1}) &=& (\mathbf{\hat{T}_{wv}} - \mathbf{T_{wv}})(\mathbf{w^1})
\end{aligned}$$

$$\begin{aligned}
\Leftrightarrow\ \mathbf{w^1} &= (\mathbf{T_{vw}^{-1}} - \mathbf{T_{wv}})^{-1}(\mathbf{\hat{T}_{wv}} - \mathbf{T_{wv}})(\mathbf{w^1}) \\
\Leftrightarrow\ \mathbf{w^1} &= \mathbf{T}((\mathbf{\hat{T}_{wv}} - \mathbf{T_{wv}})(\mathbf{w^1}))
\end{aligned}$$

Note that the Johnson-Lindenstrauss Lemma says that $\mathbf{\hat{T}_{wv}}(\mathbf{w^1})$ appears as a perturbation around $\mathbf{T_{wv}}(\mathbf{w^1})$ (see equation (10)), therefore the aim of the above operation is to "centre" $\mathbf{\hat{T}_{wv}}(\mathbf{w^1})$ around $\mathbf{T_{wv}}(\mathbf{w^1})$. The "centering" method has also been used in [34] to obtain tighter stability bound in the presence of conic sector-bound non-linearity.

Consider the norm of $\mathbf{w^1} - \mathbf{w^0}$, we have

$$\begin{aligned}
& \|\mathbf{w^1} - \mathbf{w^0}\| \\
=\ & \|\mathbf{T}((\mathbf{\hat{T}_{wv}} - \mathbf{T_{wv}})(\mathbf{w^1})) - \mathbf{w^0}\| && (32) \\
=\ & \|\mathbf{T}((\mathbf{\hat{T}_{wv}} - \mathbf{T_{wv}})(\mathbf{w^1})) - \mathbf{T}((\mathbf{\hat{T}_{wv}} - \mathbf{T_{wv}})(\mathbf{w^0})) + \\
& \mathbf{T}((\mathbf{\hat{T}_{wv}} - \mathbf{T_{wv}})(\mathbf{w^0})) - \mathbf{w^0})\| && (33) \\
\leq\ & \|\mathbf{T}((\mathbf{\hat{T}_{wv}} - \mathbf{T_{wv}})(\mathbf{w^1})) - \mathbf{T}((\mathbf{\hat{T}_{wv}} - \mathbf{T_{wv}})(\mathbf{w^0}))\| + \\
& \|\mathbf{T}((\mathbf{\hat{T}_{wv}} - \mathbf{T_{wv}})(\mathbf{w^0})) - \mathbf{w^0})\| && (34) \\
\leq\ & \|\mathbf{T}(\mathbf{\hat{T}_{wv}} - \mathbf{T_{wv}})\|_L \|\mathbf{w^1} - \mathbf{w^0}\| + \\
& \|\mathbf{T}(\mathbf{\hat{T}_{wv}}\mathbf{T_{wv}}) - \mathbf{I}\|\|\mathbf{w^0}\| && (35)
\end{aligned}$$

where $\mathbf{I}$ in equation (35) denotes the identity map. In the above derivation, equation (33) is obtained by adding and subtracting the term $\mathbf{T}((\mathbf{\hat{T}_{wv}} - \mathbf{T_{wv}})(\mathbf{w^0}))$ within the norm. Equation (34) is obtained by using the triangle inequality, and equation (35) is obtained from using the definition of the Lipschitz operator norm and operator norm.

Proposition 1 is now obtained by re-arranging the last inequality.

Appendix C
Behaviour under commonly encountered fault models in WSNs

In this section, we will show that our robust averaging algorithm can deal with three commonly encountered faults, namely stuck-at faults, offset and variance degradation fault [17], in WSNs. Two pieces of results will be presented here. Firstly, for the case of stuck-at fault, we present an analytical study in Section C.1 to show that our robust averaging algorithm can be designed to give an arbitrarily small bias in the robust average. Secondly, we present simulation results in Section C.2 to show that our robust averaging algorithm works even all the three aforementioned faults are present in the network. Note that the results in Section C.2 are obtained via simulation. We will present results on applying our robust averaging algorithm to real data in Section 4.

We will first give precise definitions of the three aforementioned faults.

- The *stuck-at fault* occurs when the sensor readings are being stuck at a value which is not related to the true sensor measurement. It can be modelled as
$$x_{st} \quad = \quad c \;\; \forall t$$
where c is a constant.
- The *offset fault* occurs when the reported sensor reading is corrupted by an additive offset. Let $\tilde{x}_{st}$ be the true sensor reading that sensor s should report at time t, then sensor s is said to suffer from the offset fault if the sensor reading reported by sensor s is:
$$x_{st} \quad = \quad \begin{cases} \tilde{x}_{st} + c & \text{with probability } p_o \\ \tilde{x}_{st} & \text{with probability } 1 - p_o \end{cases}$$
where c is a constant additive offset with corrupts the data with a probability of p_o.
- The *variance degradation fault* occurs when the noise variance becomes larger over time. The variance degradation fault can be modelled as an additive Gaussian distributed noise of larger variance compared with the other sensors. Let $\tilde{x}_{st}$ be the true sensor reading that sensor s should report at time t, then the actual reading that will be reported by sensor s at time t is:
$$x_{st} \quad = \quad \tilde{x}_{st} + e_{st}$$
where e_{st} is a Gaussian distributed random variable of zero mean and variance σ_s^2.

C.1 Effect of λ on the robust averaging algorithm

In this section, we use analytical tools to investigate the property of our robust averaging algorithm in the presence of stuck-at faults. Our goal is to show that the positive constant λ in our robust averaging algorithm will only produce a small change in the weights and a small bias in the robust average.

We consider a network of n sensors. The sensor readings for these sensors are given by:

$$x_{st} \quad = \quad \begin{cases} X & \text{for sensors } s = 1, ..., n_0; t = 1, .., m \\ X + D & \text{for sensors } s = n_0 + 1, ...n; t = 1, .., m \end{cases}$$

For this model, we assume that n_0 sensors are functioning correctly and they all register the correct reading X. The other $n_1 = n - n_0$ sensors are stuck-at the same wrong value which is $X + D$. We further assume that there are more working sensors than faulty sensors, i.e. $n_0 > n_1$. Under these assumptions, all the n_0 working sensors will have the same weight (denoted by α_0) and all the n_1 faulty sensors also have the same weights (denoted by α_1).

By using equation (6), we have

$$\begin{aligned} & v_i \\ = \; & \sum_{t=1}^{m} (X - \sum_{i=1}^{n} x_{st})^2 \\ = \; & \begin{cases} m n_1^2 \alpha_1^2 D^2 & \text{for sensors } s = 1, ..., n_0, \forall t = 1, .., m \\ m n_0^2 \alpha_0^2 D^2 & \text{for sensors } s = n_0 + 1, ...n, \forall t = 1, .., m \end{cases} \end{aligned} \quad (36)$$

By using equation (7), it can be shown that

$$\alpha_0 (v_1 + \lambda \sum_{s=1}^{n} v_s) \quad = \quad \alpha_1 (v_n + \lambda \sum_{s=1}^{n} v_s) \qquad (37)$$

By substituting the expressions of v_i in equation (36) in the above equation, we have

$$\begin{aligned} & \alpha_0 (n_1^2 \alpha_1^2 + \lambda n_0 n_1 (n_1 \alpha_1^2 + n_0 \alpha_0^2)) \\ = \; & \alpha_1 (n_0^2 \alpha_0^2 + \lambda n_0 n_1 (n_1 \alpha_1^2 + n_0 \alpha_0^2)) \end{aligned} \qquad (38)$$

Since all the weights must sum up to unity, we also have

$$n_0 \alpha_0 + n_1 \alpha_1 \quad = \quad 1 \qquad (39)$$

By using equations (38) and (39), we can solve for α_1 and α_2. Also, the robust average is

$$r_t \quad = \quad X + n_1 \alpha_1 D \;\; \forall t \qquad (40)$$

This also means that the bias in the robust average is $n_1 \alpha_1 D$. Therefore, the smaller value of α_1 (= weight of the faulty sensor), the smaller the bias in the robust average.

For $\lambda = 0$, it can be shown that the admissible solution is $\alpha_0 = \frac{1}{n_0}$ and $\alpha_1 = 0$. Therefore the weights for the faulty sensors are zero. This also means that the robust average is exactly X which is the correct value.

We will now show that, if the parameter λ if chosen sufficiently small, it will only produce a small bias. We first divide both sides of equation (38) by α_0. After substituting $\beta = \frac{\alpha_1}{\alpha_0}$ and re-arrange the result as a polynomial in λ, we have that β is the root of this following equation:

$$f(\beta) = \lambda \underbrace{[n_0 n_1 (n_1 \beta^2 + n_0)(\beta - 1)]}_{f_\lambda(\beta)} + \underbrace{\beta (n_0^2 - n_1^2 \beta)}_{f_0(\beta)} \; = \; 0 \quad (41)$$

First note that $f(0) = -\lambda n_0^2 n_1 < 0$, i.e. $f(0)$ is strictly negative. With the assumption that $n_0 > n_1$, we have $f_0(\beta)$ is strictly positive for all $0 < \beta < 1$ and $f_\lambda(\beta)$ is strictly negative for all $0 < \beta < 1$. Therefore, for sufficiently small λ, $f(\beta_0)$ is strictly positive for some $\beta_0 \in (0, 1)$. Therefore, there is a root for equation (41) in the range $(0, \beta_0)$. In fact, β_0 can be made arbitrarily small (by having a sufficiently small λ), therefore $\beta = \frac{\alpha_1}{\alpha_0}$ can be made arbitrarily small, i.e. the weights of the faulty sensors α_1 can be made as close to 0 as possible. Therefore, with a small enough λ, the weights of the sensors will only change by a small amount. Since the bias in the robust average is proportional to α_1, therefore the small positive constant will result in only a small bias.

C.2 Mixture of different types of faults in a network

In this section, we consider a simulation where all the three sensor faults described earlier appear in the same wireless sensor network. Our simulation consists of 10 sensors and a block of data consists of 100 data points. The true signal is a sinusoid. The data is plotted in Figure 7. Sensors 1, 2 and 3 are faulty. Sensor 1 has a higher noise variance of 25 while Sensors 4-10 have a smaller variance between 1 and 4. Sensor 2 suffers the stuck-at fault and the sensor reading stays at 50 all the time. Sensor 3 has an offset of 40 with a probability of 0.75.

We use Bernoulli distributed projection matrices and 40 projections. We apply our robust averaging algorithm to both the original sensor measurements as well as to the compressed data. Figure 8 show the weights computed from using these two sets of data. It can be seen that the weights computed by these two sets of data are almost identical. Furthermore, Sensors 2-3 have a smaller weights, thus indicating that they are likely to be faulty. Note that Sensor 1 has a weight which is smaller than Sensors 4-10 but a larger weight compared with Sensors 2-3. This is reasonable since the stuck-at fault (Sensor 2) and offset fault (Sensor 3) produce data that does not resemble the average behaviour. Therefore, the weights for Sensors 2 and 3 are lower. Although Sensor 1 has larger variance, the true data hidden behind the noise is similar to what is found in Sensors 4-10, therefore Sensor 1 has a weight which is in-between.

Finally, we show the robust average computed in Figure 9. The dashed blue line show the readings from sensor 10, which is a working sensor, as the reference. The thick red line shows the robust average computed using the original sensor reading. The thick green line shows the robust averaging computed using the compressed data. We see that there is almost no loss in fidelity in using the compressed data to compute the robust average.

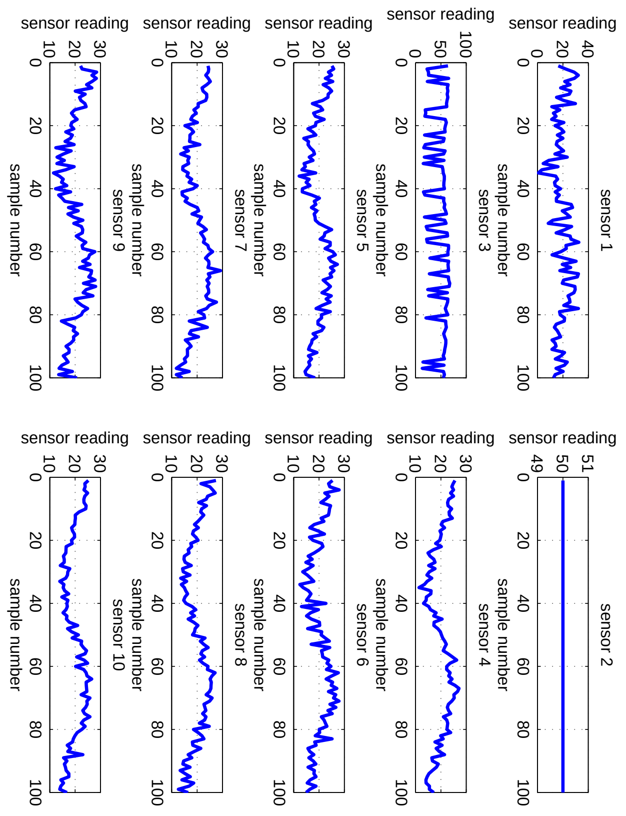

Fig. 7. Simulated data with three different types of sensor faults. Sensor 1 has a higher noise variance. Sensor 2 has stuck-at fault. Sensor 3 has an offset fault. Noise of different variances are added to the other 7 sensor readings.

Appendix D
Additional results from the QCAT deployment

This appendix contains 3 additional figures (Figures 10, 11 and 12) on the results from the QCAT deployment. In addition, this appendix contains the results on applying the perturbation analysis (discussed in Section 3.4) on the QCAT data set and results on comparing our robust averaging method against random sampling.

D.0.1 Additional figures

In Figure 10, we compare the performance of our robust averaging algorithm against that of *Statistical* and *Reputation* and *Bayes* to the QCAT dataset with 4 working and 1 faulty sensors. The robust averaging algorithm uses compressed data while the other methods use uncompressed data.

In Figures 11 and 12), we compare the performance of our robust averaging algorithm against that of *Statistical* and *Reputation* and *Bayes* when there are 5 sensors (3 working, 2 faulty with one stuck at the level of 20) and 6 sensors (4 working and 2 faulty with one stuck at the level of 110).

D.0.2 Perturbation analysis

Table 5 shows the values of the spectral norm $\rho(\mathbf{F})$ (which determines the stability of the fixed point) and $\|\mathbf{G}\|_\infty$ (which gives the size of perturbation caused by using the compressed data). One pair of values is calculated from using the original sensor measurements. The other pair of values is calculated from using the compressed data. It can readily be seen that both the original data and the compressed data give similar results. Since the spectral norm $\rho(\mathbf{F})$ is less than 1, the fixed point is stable. In addition, $\|\mathbf{G}\|_\infty$ is small, which means that the perturbation on the weights in using the compressed data is small. This is also confirmed earlier in Figure 4.

We use the results of Proposition 1 to study the effect of large perturbation on the weights w_s. We first use the particular projection matrix that we have used in our experiment to generate a probability distribution of ϵ_s. By assuming that each ϵ_s is independently distributed, we compute the ∞-norm relative perturbation of the

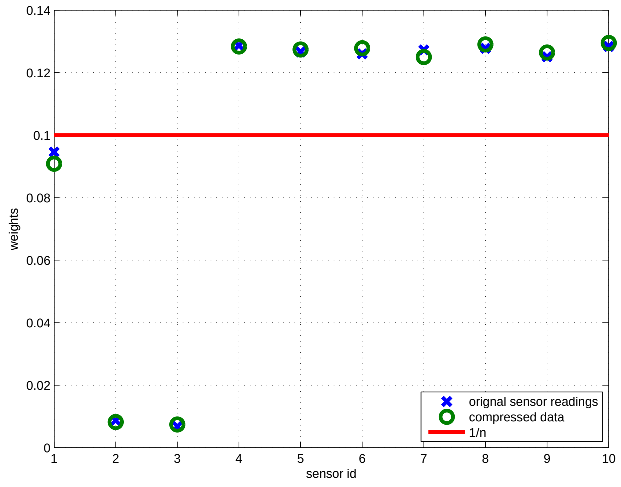

Fig. 8. The weighs w_s from our robust averaging algorithm. The weights computed from the original sensor readings are shown in circles and those computed from the compressed data are shown in crosses. The line is at the level of $\frac{1}{10}$ which is the expected weight when no sensors are faulty.

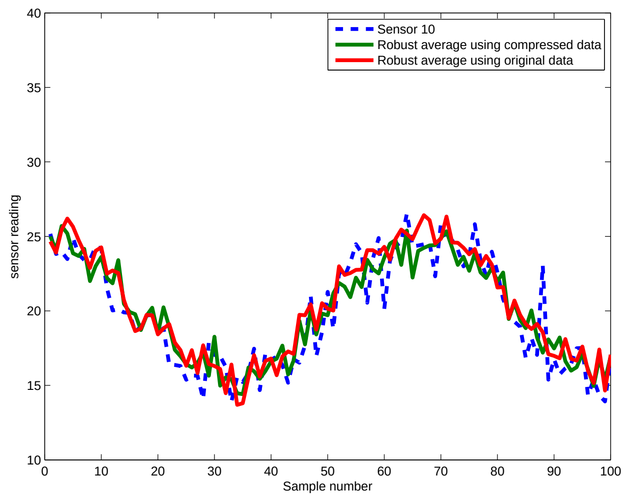

Fig. 9. The dashed blue line show the readings from sensor 10, which is a working sensor. The red line shows the robust average computed using the original sensor reading. The green line shows the robust averaging computed using the compressed data.

weights $\mathbf{w}$ given by the right-hand side of equation (23) for 10,000 sets of $\{\epsilon_s\}$ generated. Figure 13 shows the cumulative probability distribution of the bound of relative perturbation. It shows that there is a high probability that the relative perturbation is less than 0.02.

D.0.3 Comparison with random sampling

We compare the performance of compressive sensing and random sampling for a given percentage of bandwidth saving. From the above calculation, we know that 100 projections will give a bandwidth saving of 50%.

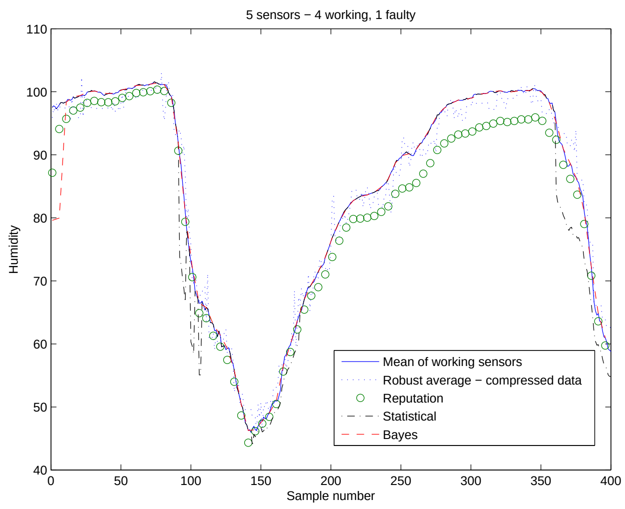

Fig. 10. Comparison of *Statistical*, *Reputation*, *Bayes* and our robust averaging method for the QCAT data set. All five sensors (4 working and 1 faulty) are used.

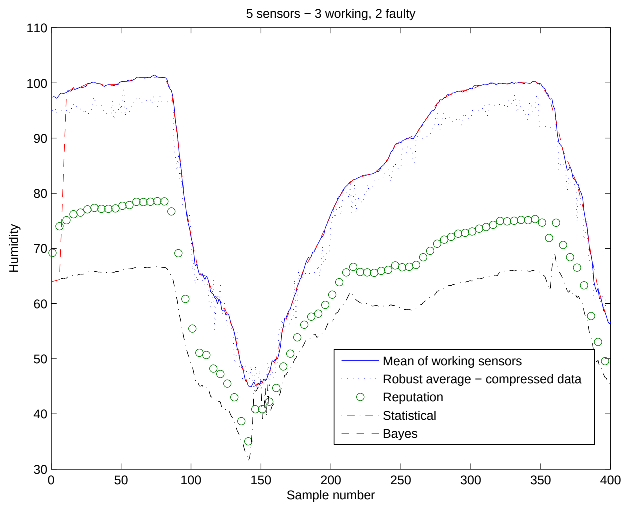

Fig. 11. Comparison of *Statistical*, *Reputation*,*Bayes* and our robust averaging method. Data from 5 sensors (3 working and 2 faulty) are used.

This amount of bandwidth saving can also be realised by having the sensors performing random sampling so that only half of the samples (in this case 200 samples per sensor) are sent from the sensors to the fusion centre. Given these random samples, we can follow our robust averaging procedure to compute the robust average at each sampling instance. Finally, the complete time series of the robust average (of which 200 samples are not known because they are not sampled at these instances) can be reconstructed by using compressive sensing reconstruction method. We find that compressive sensing gives a relative error in reconstructing the robust average (the first metric in the previous paragraph) of 2.73% while the random sampling method gives 8.02%. We repeat the same experiment with 50 projections (which

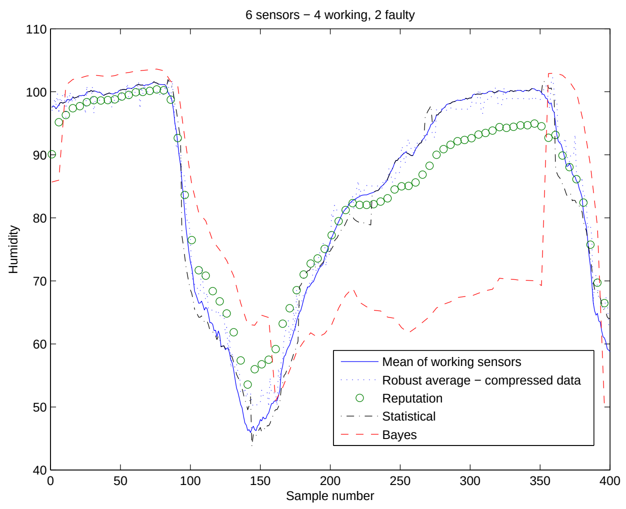

Fig. 12. Comparison of *Statistical*, *Reputation*,*Bayes* and the proposed robust averaging method. Data from 6 sensors (4 working and 2 faulty) are used.

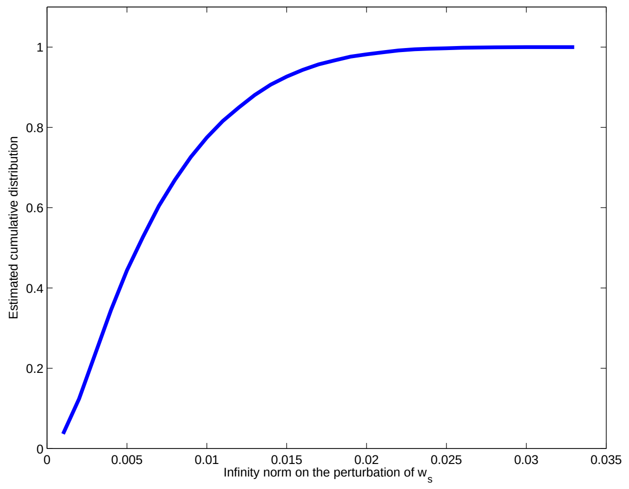

Fig. 13. This figure shows the cumulative probability distribution of the bound of relative perturbation on the weight vector $\mathbf{w}$ obtained by Monte Carlo simulation.

	$\rho(\mathbf{F})$	$\|\mathbf{G}\|_\infty$
Using original sensor readings	0.1358	0.1894
Using compressed data	0.1427	0.1965

TABLE 5
This table compares the values of $\rho(\mathbf{F})$ and $\|\mathbf{G}\|_\infty$ evaluated by using the original sensor readings against those computed by using the compressed data. This table is for the QCAT deployment.

results in a bandwidth saving of 75%) for compressive sensing and compare it against random sampling where each sensor returns 100 random samples (out of 400 samples, which also gives a bandwidth saving of 75%) to the fusion centre. In this case, we find that compressive sensing gives a relative error in reconstructing the robust average of 7.90% while the random sampling method gives 11.92%. Thus, compressive sensing gives better estimation of robust average compared with other methods for the same amount of bandwidth.

Appendix E
The Belmont deployment

This appendix describes the results of applying our robust averaging algorithm to the data obtained from an outdoor wireless sensor network testbed operated by CSIRO in Belmont, central Queensland, Australia. The WSN consists of 32 sensors measuring temperature and a block of data for each sensor consists of 221 points. Figure 14 shows the temperature measurement of the 8 selected sensors over the measurement period. The correct trend is that the temperature readings should drop from (about) 31°C to (about) 19°C over the measurement period, which is the behaviour of sensors 1 and 32. Some of the sensors are faulty. For example, sensor 3 is stuck at 31°C throughout, sensor 10 is stuck at 28 °C and sensor 11 is stuck at 12 °C . Some sensors are only faulty for part of the time. For example, sensor 9 is stuck at 30°C in the first half of the measurement period but works in the second half; both sensors 13 and 14 are stuck in the second half of the measurement period.

We apply our fault detection algorithm to the original sensor measurements, as well as to the compressed data with 88 $(= p)$ projections with a Bernoulli distributed projection matrix. Figure 15 shows the weights w_s for the sensors, where $s = 1, ..., 32$, 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 figure 15 is at the level of $\frac{1}{32}$ which is the weight to use when all sensors are working. Figure 15 shows that sensors 3, 10 and 11 have very low weights, and they correspond to those three sensors that are stuck throughout the measurement period. Sensors 13 and 14 also receive low weights because they are stuck for a good part of the measurement period. The same observation apply to sensors 5 and 9, which also have low weightings.

Figure 16 shows the average temperatures given by our fixed point iteration algorithm. (We use the ℓ_1-magic toolbox [6] for compressive sensing reconstruction.) The curve with short dashes shows the behaviour of sensor 1, which is a working sensor. 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 no loss in fidelity in using the compressed data. It is hard to distinguish the latter two curves because they are almost on top of each other. The figure also shows the robust average captures the trend of the working sensor.

We compare the computation time of the two meth-

	$\rho(\mathbf{F})$	$\|\mathbf{G}\|_\infty$
Using original sensor readings	0.0982	0.0186
Using compressed data	0.0967	0.0185

TABLE 6
This table compares the values of $\rho(\mathbf{F})$ and $\|\mathbf{G}\|_\infty$ evaluated by using the original sensor readings against those computed by using the compressed data.

ods shown in Figures 1 and 2. The computation was performed on a MacBook running Matlab 2009b. The method in Figure 1, which requires the reconstruction of 32 time series and one run of the robust average algorithm, took 10.47s to complete, while the method in Figure 2, which requires only one reconstruction and one run of the robust average algorithm, took 0.37s and is therefore 28.1 times faster.

Table 6 shows the values of the spectral norm $\rho(\mathbf{F})$ (which determines the stability of the fixed point) and $\|\mathbf{G}\|_\infty$ (which gives the size of perturbation caused by using the compressed data). One pair of values is calculated from using the original sensor measurements. The other pair of values is calculated from using the compressed data. It can readily be seen that both the original data and the compressed data give similar results. Since the spectral norm $\rho(\mathbf{F})$ is less than 1, the fixed point is stable. In addition, $\|\mathbf{G}\|_\infty$ is very small, which means that the perturbation on the weights in using the compressed data is minimal. This is also confirmed earlier in Figure 15.

The bandwidth saving in using compressive sensing can be computed. Since the sensors use 10-bit A/D converter, sending all the sensor readings to the data fusion centre will mean each sensor needs to transmit 2210 bits of data. With compressive sensing, the number of bits of data that has to be sent by each sensor is $p(b + 1 + \lceil \log_2(m) \rceil)$ =1408 bits. This represents a 25% saving.

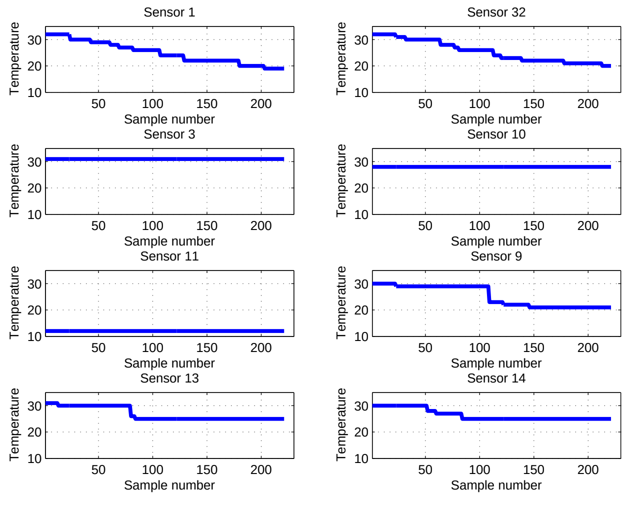

Fig. 14. Sensor readings of 8 selected sensors.

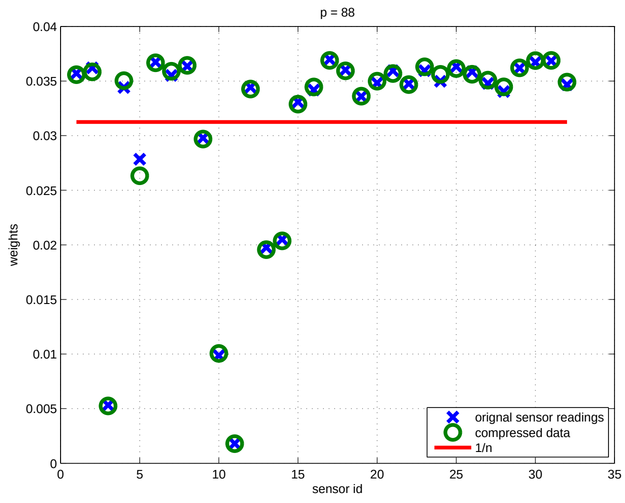

Fig. 15. The weights w_s are shown in circles (from original sensor readings) and crosses (from compressed data). The line is at the level of $\frac{1}{32}$ which is the expected weight when no sensors are faulty.

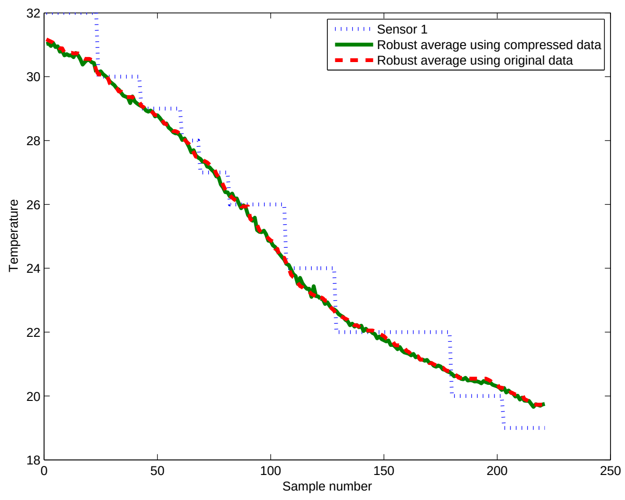

Fig. 16. The figure shows the averages against the trend of one working sensor (short dashes). The line with long dashes shows the robust average computed by applying the fixed point iteration to the original data. The solid curve shows the robust average by applying the fixed point iteration to the compressed data followed by reconstruction.