Robust Calibration for Localization in Clustered Wireless Sensor Networks

Jung Jin Cho, Yu Ding, *Member, IEEE*, Yong Chen, *Member, IEEE*, and Jiong Tang, *Member, IEEE*

***Abstract*—This paper presents a robust calibration procedure for clustered wireless sensor networks. Accurate calibration of between-node distances is one crucial step in localizing sensor nodes in an ad-hoc sensor network. The calibration problem is formulated as a parameter estimation problem using a linear calibration model. For reducing or eliminating the unwanted influence of measurement corruptions or outliers on parameter estimation, which may be caused by sensor or communication failures, a robust regression estimator such as the least-trimmed squares (LTS) estimator is a natural choice. Despite the availability of the FAST-LTS routine in several statistical packages (e.g., R, S-PLUS, SAS), applying it to the sensor network calibration is not a simple task. To use the FAST-LTS, one needs to input a trimming parameter, which is a function of the sensor redundancy in a network. Computing the redundancy degree and subsequently solving the LTS estimation both turn out to be computationally demanding. Our research aims at utilizing some cluster structure in a network configuration in order to do robust estimation more efficiently. We present two algorithms that compute the exact value and a lower bound of the redundancy degree, respectively, and an algorithm that computes the LTS estimation. Two examples are presented to illustrate how the proposed methods help alleviate the computational demands associated with robust estimation and thus facilitate robust calibration in a sensor network.**

***Note for Practitioners*—Wireless sensor network is an emerging technology that finds numerous civilian and military applications lately. When utilizing the information from a sensor network for decision making, a statistical estimation procedure is often a necessary intermediate step in order to know about critical system parameters or state variables. For the estimation purpose, the commonly used least-squares estimation (LSE) mechanism is not robust at all against data corruptions or outliers caused by sensor and communication failures. Any single corrupted data point may cause considerable deterioration in an LSE. Applying the robust estimators available from robust statistics research to a wireless sensor network, however, faces a number of computational challenges. This paper offers several useful algorithms to address these challenges, making the application of a robust estimator easier and more efficient.**

***Index Terms*—Ad-hoc sensor network, least-trimmed squares estimator, robust regression, sensor localization.**

Manuscript received April 10, 2008; revised November 13, 2008. First published May 02, 2009; current version published January 08, 2010. This paper was recommended for publication by Associate Editor H. Tanner and Editor M. Wang upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation under Grants CMMI-0540132, CMMI-0726939, CMMI-0727305, and CMMI-0758178.

J. J. Cho is with Baker Hughes, Inc., Houston, TX 77019-2118 USA (e-mail: cho.jungjin@gmail.com).

Y. Ding is with the Department of Industrial and Systems Engineering, Texas A&M University, College Station, TX 77843-3131 USA (e-mail: yuding@iemail.tamu.edu).

Y. Chen is with the Department of Mechanical and Industrial Engineering, The University of Iowa, Iowa City, IA 52242 USA (e-mail: yongchen@engineering.uiowa.edu).

J. Tang is with the Department of Mechanical Engineering, University of Connecticut, Storrs, CT 06269 USA (e-mail: jtang@engr.uconn.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2009.2013475

I. Introduction

A. Background

WIRELESS technologies have changed the design and operation of sensor networks. Equipped with microelectromechanical systems (MEMS), a wireless sensor node becomes small, mobile, and multifunctional. One of the most significant changes caused by the wireless technologies is the implementation of an *ad-hoc* networking, referring to those having a network topology and node positions that are not fixed *a priori* [1]. This naturally calls for a solution to the *localization* or *location tracking* problem because knowing the positions of individual sensors is often the prerequisite to many subsequent decision makings. Installing a global positioning system (GPS) [2] could be a solution but the heavy power consumption and high equipment cost associated with a GPS deem it impractical to install it on every microsensor node. In practice, GPS receivers may be used only on a small portion of sensor nodes, known as *anchor nodes*, in a network [3]. The location of a non-anchor node can be decided and tracked relative to the anchor nodes in the following manner: 1) measure the distances between itself and several anchor nodes and 2) compute its location based on certain geometry principle (such as hyperbolic trilateration, triangulation, and multilateration [4]).

One method of measuring the between-node distance is to use two types of signals, a radio frequency (RF) one and an acoustic one, which travel at different speeds. The time difference of arrival (TDOA) between the two signals is then used to calculate the between-node distance [5]. One problem associated with this distance measuring approach is its inaccuracy. For example, RF signals are attenuated by metal objects [6] and the speed of acoustic signals are highly influenced by temperature and moisture [5]. The experiments performed by Whitehouse and Culler [7] showed that the error of a between-node distance measured using acoustic time of flight could be as large as 300% of the true distance. To tackle this issue, Whitehouse and Culler [7] recommended using a *calibration* procedure as follows.

- In an *offline* setting (or during a periodical maintenance time), the true distance between sensor nodes can be measured by independent and accurate means.
- Then, establish a mathematical model mapping the distance measured by TDOA to the true distance.

- During the service of sensor nodes, the mapping model established above adjusts the TDOA-based measurements to a more accurate estimation of the true distances.

Denote by $d_{u,v}$ the TDOA-measured distance between transmitter u and receiver v, by $y_{u,v}$ the true distance, and by e the random noise. Whitehouse and Culler [7] discussed several common sources of hardware variations: 1) Bias—the time for a receiver or a transmitter to start sending or receiving a signal; 2) Gain—the sensitivity of a receiver or a transmitter; 3) Frequency—the frequency difference between a transmitter and a receiver; 4) Orientation—the relative orientations of a transmitter and a receiver. With these hardware variations included, the calibration model, which maps a measured distance to a true distance, becomes

$$y_{u,v} = \alpha_u + \beta_v + \gamma_u d_{u,v} + \delta_v d_{u,v} + |F_u - F_v| d_{u,v} + f_O(O_u, O_v) d_{u,v} + e \quad (1)$$

where α_u and β_v are the bias of a transmitter u and a receiver v, γ_u and δ_v are the gain of u and v, F_u and F_v are related to the frequencies used by u and v, O_u and O_v are the orientations of u and v, respectively. Here, f_O is a nonlinear function. A common treatment in the calibration suggested by [7] is to further simplify the calibration model (1) by merging the usually less significant nonlinear terms about frequency and orientation into e so that a linear model structure is used as

$$y_{u,v} = \alpha_u + \beta_v + \gamma_u d_{u,v} + \delta_v d_{u,v} + e. \quad (2)$$

Model (2) is only for a single pair of sensor nodes. For a sensor network with a number of sensor nodes, we have an aggregated version of model (2) expressed in a matrix format. Suppose we have n sensor nodes and each sensor can work as both a transmitter and a receiver. Indexing sensor nodes from 1 to n, the calibration parameters become $(\alpha_1, \ldots, \alpha_n, \ldots, \delta_n)$. The number of the pairwise distances among n sensors is $n(n-1)/2$. Suppose m true distances out of the $n(n-1)/2$ pairwise distances are available. In a matrix form, the calibration model for all sensor nodes becomes

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \mathbf{e} \quad (3)$$

where $\mathbf{y}$ is of dimension $2m \times 1$ because each distance is used twice for a sensor node serving as a transmitter and as a receiver, $\boldsymbol{\theta}$ is a $p \times 1$ vector of unknown calibration parameters, $\mathbf{X} = (\mathbf{x}_1^T, \ldots, \mathbf{x}_{2m}^T)^T$ is a $2m \times p$ matrix having the TDOA measured distances as some of its entries, and $\mathbf{e}$ is the vector of noises. More details are provided in the Appendix regarding the construction of $\mathbf{X}$. During a calibration phase when the measurements in $\mathbf{y}$ and $\mathbf{X}$ are known, one estimates the unknown parameters in $\boldsymbol{\theta}$; while during the in-field service time, one will use the estimated parameter $\hat{\boldsymbol{\theta}}$ to predict the between-node distance.

Given the linear model structure in (3), it comes as no surprise that the least-squares (LS) estimation is the most popular method used for estimating $\boldsymbol{\theta}$; i.e.,

$$\hat{\boldsymbol{\theta}}_{\text{LS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \quad (4)$$

In fact, this is what Whitehouse and Culler [7] used in their procedure. For an LS estimator to be optimal in the sense of minimum variance or maximum likelihood, one needs certain assumptions on the noise term—$\mathbf{e}$ should follow the distribution of $N(\mathbf{0}, \sigma^2\mathbf{I})$. Statistical research [8] has come to the conclusion that the performance of an LS estimator is sensitive to, and will deteriorate remarkably in the presence of, model uncertainties and outliers. Absorbing the nonlinear terms in the complete calibration model into $\mathbf{e}$ will surely make the model assumption less likely to be true. Another source of concern comes from the corrupted measurements caused by sensor and communication failures, the presence of which are very likely in a wireless network since a wireless connection is generally weak and less reliable, and the TDOA distance-measuring mechanism is inherently inaccurate. In other words, when a distance measurement $d_{u,v}$ is accidentally corrupted, the corresponding data point $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ could result in a poor estimation of the calibration parameters.

It is worth mentioning that sensor localization is a topic under intensive studies in recent years ([9]–[12] among others); please see especially [9] for a comprehensive survey. The term "calibration" is commonly used in the literature but oftentimes it entails different meanings. In most of the contexts, calibration means almost the same thing as "localization," namely that once a sensor position is calibrated, it is localized. In this paper, however, we follow the specific meaning of calibration as defined by Whitehouse and Culler's [7] procedure, as outlined above. Simply put, "calibration" in this article means that the distance-determining parameters associated with sensor nodes need to be fine-tuned in order to produce better between-node distance results. We believe that this calibration problem raised by Whitehouse and Culler [7] has not been extensively studied in the sensor localization literature.

B. Robust Estimation and Challenges

Considering the critical role of the calibration parameters in an *ad-hoc* network's service, it is highly desirable to improve its estimation accuracy and to make the calibration process more robust with respect to environmental disturbances. For this reason, robust regression estimators [13] attracted our attention. Prior research [8] demonstrates that, if properly employed, a robust estimator can perform very closely to an LS estimation when the assumptions on $\mathbf{e}$ strictly hold, and can outperform an LS estimator otherwise. We are particularly interested in a class of the so-called *high-breakdown point* robust estimators. The concept of a breakdown point was introduced by [14] to characterize the robustness of an estimator. The breakdown point is defined as the smallest fraction of data corruptions in model (3) that can ruin an estimation. Intuitively, the higher the breakdown point value, the more outliers an estimator can tolerate.

Applying a high breakdown point estimator (a popular choice is the least trimmed squares (LTS) estimator [15]) to the sensor network calibration is, however, not a simple task. There are two major challenges, both related to some unique features of the wireless sensor networks. The first is the scale of the system and the resulting computation cost. A wireless sensor network could easily have hundreds of sensor nodes, which results in a mathematical model having hundreds even thousands of columns and rows in the $\mathbf{X}$ matrix. The sheer scale of the sensor network causes the application of an LTS estimator to be computationally demanding. In the later section of this paper, we present an example where the FAST-LTS routine in R [16] fails to estimate the parameters in a moderately large sensor network, even when a rather long period of time is given.

The second challenge is related to the structure of a wireless sensor network. Due to the limited power available on individual nodes, sensor nodes rarely communicate with all other sensors in a network. Instead, the whole network is usually grouped into a number of clusters. A sensor mainly communicates with the sensors that belong to the same cluster. The between-cluster communications are limited to a few more powerful cluster heads or a few nodes that are close to another cluster. The structure in the network configuration typically causes the resulting $\mathbf{X}$ matrix to have structures as well. The structure in the linear model must be considered when devising a robust estimator (including the LTS estimator); otherwise, the estimator may lose its supposed robustness [17]. Mili and Coakley's condition is actually a function of a previously defined *degree of sensor redundancy* of a network (defined in [18] and [19]). Obtaining the redundancy degree for a large network once again runs into computation issues. For a large sensor network, it is almost impossible to compute the redundancy degree by using the enumerative algorithm proposed in [18] and [19].

It turns out that the existence of a cluster structure in a sensor network configuration actually provides the opportunity to overcome the challenges faced by the application of a robust estimator. Simply put, the existence of a cluster structure allows us to decompose the whole network into smaller subsystems, of which the computation becomes much less demanding. In fact, the decomposition algorithm presented in [20] can compute the redundancy degree faster for a structured linear model. Our first objective is to establish a model that enables the decomposition algorithm in [20] to be applied to clustered wireless sensor networks.

During our investigation, we notice that for a large sensor network, even the decomposition algorithm in [20] may fall short of computing the degree of redundancy. Then, the strategy for a large scale system is that, instead of computing the exact redundancy degree, one may want to compute a lower bound. Research in the robust statistics [15] tells us that using an underestimated redundancy degree could still retain certain robustness at a suboptimal level. Hence, our second objective is to compute a lower bound of the redundancy degree by utilizing the cluster structure in a network configuration. Finally, we also devise a subcalibration procedure that utilizes the clustering information when computing the LTS estimation. Doing so appears to speed up the computation of LTS estimators a great deal.

C. Contribution and Organization of the Paper

Our contribution of this paper can be summarized in the following: 1) we establish a model for clustered sensor networks for the purpose of parameter calibration (Sections II-A and II-B). This model is essential for the decomposition algorithm in [20] to be applied; 2) we propose a new algorithm for computing the lower bound of the sensor redundancy (Sections II-C and II-D); 3) we propose a new way to compute the LTS estimator based on individual clusters instead of the entire network (Section II-E). We noticed that clustering was used for the general localization purpose (for example, in [10]), but to our knowledge, clustering has not been used in a calibration problem. In fact, the robustness impact on estimation due to sensor or communication failures appears to be relatively a new issue; the estimation methods cited in the survey paper [9] are mainly LS, weighted LS, or maximum likelihood-based, which are not robust against the existence of measurement corruptions or outliers.

The remainder of the paper is organized as follows. Section II presents a decomposition algorithm for clustered sensor networks. It also shows how a lower bound of sensor redundancy can be computed and how the LTS can be computed on subcalibration models. Section III presents two examples of the robust calibration procedure and compares the performances of different approaches. Finally, we conclude the paper in Section IV.

II. Calibration Redundancy and Its Lower Bound

Our research development is based on model (3). We include in the Appendix the details of how such a model can be established. Based on this linear model, we will use the LTS estimator to eliminate the influence of some corrupted measurements. The LTS estimator is given by

$$\min \sum_{i=1}^{h} w_{(i)}^2 \tag{5}$$

where h is called the trimming parameter, and $w_{(1)}^2 \leq w_{(2)}^2 \leq \dots \leq w_{(2m)}^2$ are the squared residuals $w_i^2 = (y_i - \mathbf{x}_i\hat{\boldsymbol{\theta}}_{\mathrm{LS}})^2$ for $i = 1, \dots, 2m$ arranged in ascending order. Essentially, the LTS estimator chooses a subset of measurements that are likely not corrupted to fit the parameters, and the trimming parameter h decides the size of the subset of measurements.

In order to evaluate the robustness of a robust estimator, Donoho and Huber [14] introduced the concept of finite sample breakdown point. Based on the theoretical results presented in Mili and Coakley [17], the maximum attainable breakdown point $\epsilon_{\max}^*$ can be expressed

$$\epsilon_{\max}^* = \frac{[\eta(\mathbf{X})/2] + 1}{2m} \tag{6}$$

where $[a]$ denotes the largest integer smaller than or equal to a and $\eta(\mathbf{X})$ is the minimum number of the row vectors in $\mathbf{X}$, whose removal from $\mathbf{X}$ makes the remaining matrix rank deficient. This $\eta(\mathbf{X})$ is in fact the degree of sensor redundancy as defined in [18] and [19]. The redundancy degree $\eta(\mathbf{X})$ is also called *calibration redundancy* in this paper. Mathematically, it was defined as

$$\eta(\mathbf{X}) = \min\left\{d - 1 : \text{there exists } \mathbf{X}_{(-d)} \text{s.t. } r\left(\mathbf{X}_{(-d)}\right) < p\right\}.$$

where $r(\cdot)$ represents the rank function of a matrix and $\mathbf{X}_{(-d)}$ is the reduced matrix after deleting d rows in $\mathbf{X}$. The physical interpretation of the calibration redundancy is as follows. The

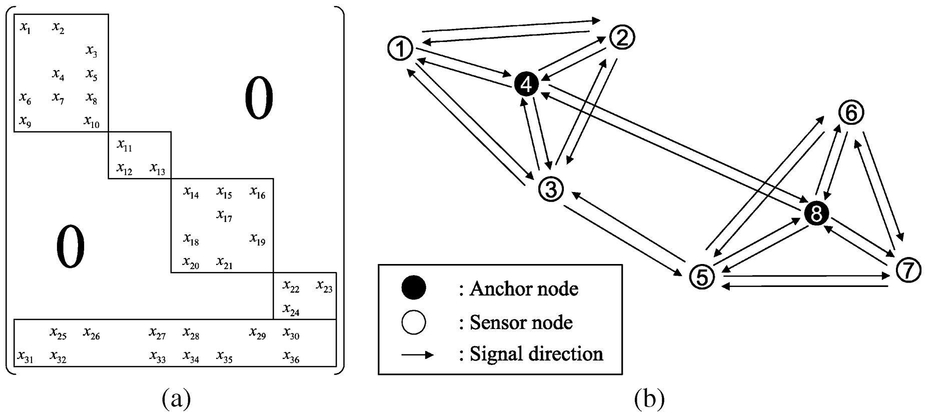

Fig. 1. Bordered block form and wireless sensor network. Here and also in Fig. 3, each node cluster is a clique, but this simple structure is constructed for the sake of illustration. For the proposed algorithms to work, each node cluster is not required to be a clique. (a) Bordered block form. (b) Wireless sensor network with clusters.

rows in $\mathbf{X}$ is a vector function of a distance measurement. Particularly, one can see from (18) (in the Appendix) that the row vectors in $\mathbf{X}$ take one of the following vector forms:

- $(\mathbf{a}_1\ \mathbf{a}_2\ d\mathbf{a}_1\ d\mathbf{a}_2)$,
- $(\mathbf{a}_1\ \mathbf{0}\ d\mathbf{a}_1\ \mathbf{0})$, or
- $(\mathbf{0}\ \mathbf{a}_2\ \mathbf{0}\ d\mathbf{a}_2)$,

where $\mathbf{a}_1$, and $\mathbf{a}_2$ are unit row vectors, $\mathbf{0}$ is a zero row vector, and d is a distance measurement. The distance measurement d in each row vector in $\mathbf{X}$ is unique, so deleting a row vector in $\mathbf{X}$ implies that we disregard the distance measurement corresponding to the deleted row. To that extent, the calibration redundancy $\eta(\mathbf{X})$ indicates the number of distance measurements that a sensor network can disregard while still uniquely estimating the unknown parameters $\boldsymbol{\theta}$.

For a better engineering interpretation, $\epsilon^*_{\max}$ can be transformed into an integer as

$$\tau_{\max}(\mathbf{X}) = 2m \cdot \epsilon^*_{\max} - 1 = [\eta(\mathbf{X})/2] \tag{7}$$

and $\tau_{\max}(\mathbf{X})$ is labeled as the *fault tolerance capability*. The $\tau_{\max}(\mathbf{X})$ benchmarks how many corrupted measurements an estimator can tolerate before breaking down. Apparently, $\tau_{\max}(\mathbf{X})$ is decided by the sensor network configuration that is modeled by $\mathbf{X}$.

The commonly used algorithm to compute this redundancy degree is the enumerative rank testing algorithm [18], [19], which literally follows the definition of $\eta(\mathbf{X})$ and tests the ranks of all the reduced matrices $\mathbf{X}_{(-d)}$. The computation of the enumerative rank testing is proportional to $\sum_{d=1}^{\eta(\mathbf{X})+1}\binom{2m}{d}$ so that the computation time increases rapidly when $\eta(\mathbf{X})$ or $2m$ increases.

As mentioned in Section I, a decomposition algorithm developed in [20] can remarkably improve the computation efficiency for evaluating the redundancy degree. In the rest of this section, we first show how such a decomposition algorithm can be applied to a clustered wireless sensor network (through a matrix transformation in Sections II-A and II-B). Then, we devise a recursive procedure to obtain a lower bound of the calibration redundancy (Section II-C) and show how the lower bound should be used (Section II-D). Finally, a subcalibration-based LTS computation procedure is developed (Section II-E).

A. *Application of the Decomposition Algorithm*

Our research finds that in order to use the decomposition algorithm in [20], one need to transform the model matrix into a *bordered block form* (BBF) [refer to Fig. 1(a)]. In a BBF matrix, the nonzero submatrices along the diagonal are called blocks and the nonzero submatrix at the bottom is called a border. A BBF matrix is defined so that the rows of the border submatrix has nonzero elements in the columns of at least two blocks (for more discussions on BBF, please refer to [21]).

Suppose that such a transformation can be done (and we will discuss how to do it in Section II-B). The search for $\eta(\mathbf{X})$ can be performed in two stages. First, perform the rank testings of the original $\mathbf{X}$ matrix until the number of the deleted rows d reaches a bound, and then, perform the rank testings on the submatrices consisting of the individual blocks and the rows in the border, until the redundancy is found.

The bound that allows the switch from a full-matrix rank testing to a submatrix rank testing is decided primarily by the size of the border rows. A detailed procedure of this algorithm will be presented for our calibration problem shortly. This decomposition is much more efficient than the enumerative testing algorithm because the sizes of submatrices are usually much smaller than that of the original matrix.

A clustered sensor network, shown in Fig. 1(b), can actually be modeled, quite ideally, by a BBF matrix. The essence of a clustered network is that there are abundant communication channels among the sensor nodes *within* a cluster, while there are relatively fewer communication channels *between* clusters. The between-cluster communications are generally conducted by the sensor nodes serving as the cluster heads or those closest to their neighboring clusters. Using the terms of a BBF matrix, the communication links between sensor nodes within the same cluster are modeled as the blocks, and the between-cluster communication links correspond to the border rows.

To see this more specifically, we use a graph representation. Denote by $G(V, E)$ a graph representation for a wireless sensor network, where V is a set of vertices corresponding to sensor nodes, and E is the set of edges corresponding to the communications between sensor nodes. The communications between the anchor nodes are not included in E since

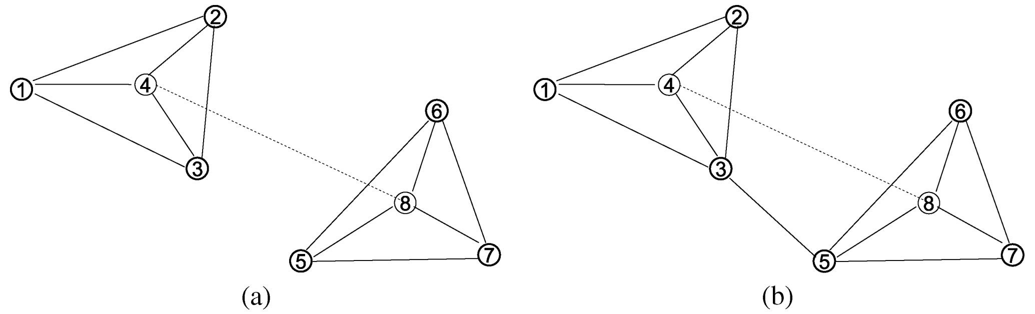

Fig. 2. Graph representations of the wireless sensor network in Fig. 1(b). Here the dashed line represents the communication link between two anchor nodes and the solid lines represent the communication links between regular nodes or between a regular node and an anchor node. (a) Disconnected clusters. (b) Connected clusters.

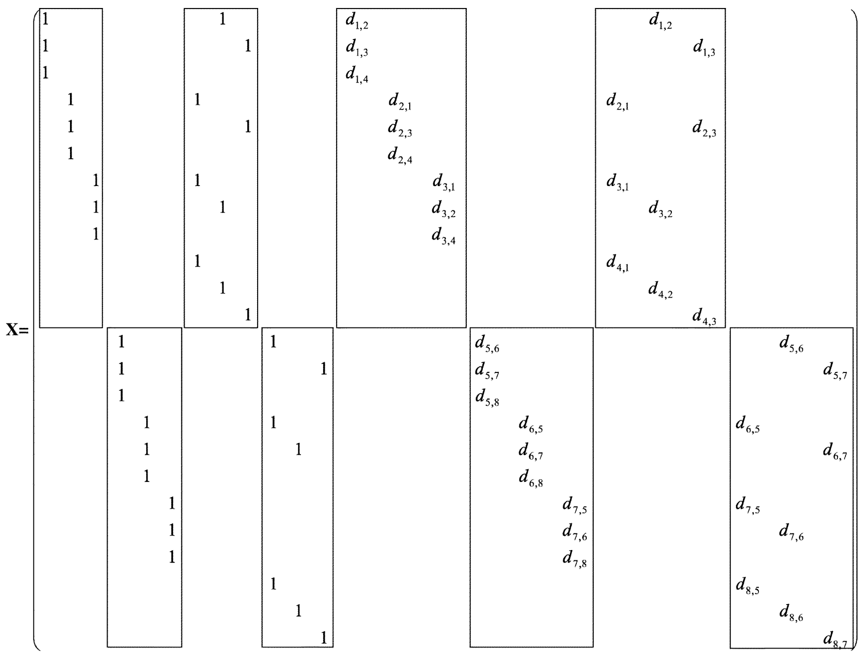

Fig. 3. The design matrix consists of disjoint submatrices. This is corresponding to the case of disconnected clusters in Fig 3(a). $d_{u,v}$ is the TDOA-measuring distance between nodes u and v.

the calibration parameters associated with those nodes are already known. For the wireless sensor network in Fig. 1(b), suppose that the between-cluster communication occurs only between the anchor nodes (which are micro-calibrated), and the communication link $\{(3,5)\}$ does not exist. The graph representation of the sensor network of this two-cluster network is shown in Fig. 2(a), where the communication between the microcalibrated sensors is depicted as a dashed line. Since the two subgraphs in $G(V,E)$ is connected by a dashed line, G is considered disconnected. The calibration matrix $\mathbf{X}$ of this sensor network is a composition of disjoint submatrices as illustrated in Fig. 3. This calibration matrix form is a special BBF, known as a *block form*, where no border row exists. For such a calibration matrix, the calibration redundancy $\eta(\mathbf{X})$ is simply the smallest value of the calibration redundancies associated with disjoint submatrices.

If both between-cluster communication channels in Fig. 1(b) are working, then, its graph representation is shown in Fig. 2(b), where $G(V,E)$ is not disconnected since there is a solid line connecting vertex 3 in one subgraph to vertex 5 in the second one. As such, the corresponding $\mathbf{X}$ is not in a block form but a bordered block form and is shown in Fig. 4. The tenth and fourteenth rows of the matrix in Fig. 4 are the *border rows*, which are associated with $d_{3,5}$ and $d_{5,3}$ in the wireless sensor network, or associated with edge (3,5) of G in Fig. 2(b). Apparently, by removing edge (3,5), G would become disconnected. Accordingly, if we eliminate the border rows associated with edge (3,5), the corresponding calibration matrix would be in a block form.

In order to denote the blocks and the border rows, we use the set of row labels and column labels. Denote by $\mathbf{X}[I,J]$ the submatrix of $\mathbf{X}$ with the row set I and the column set J, i.e, $\mathbf{X}[I,J] = (x_{ij}|i \in I, j \in J)$; also let $R = \text{Row}(\mathbf{X})$ and $C = \text{Col}(\mathbf{X})$. Denote by B the set of row labels associated with the border rows. The notation $\mathbf{X}[R-B,C]$ represents the rest of the original $\mathbf{X}$ matrix after removing its border rows so that $\mathbf{X}[R-B,C]$ is in a block form. Let k be the number of blocks in the BBF of the calibration matrix $\mathbf{X}$. Denote by

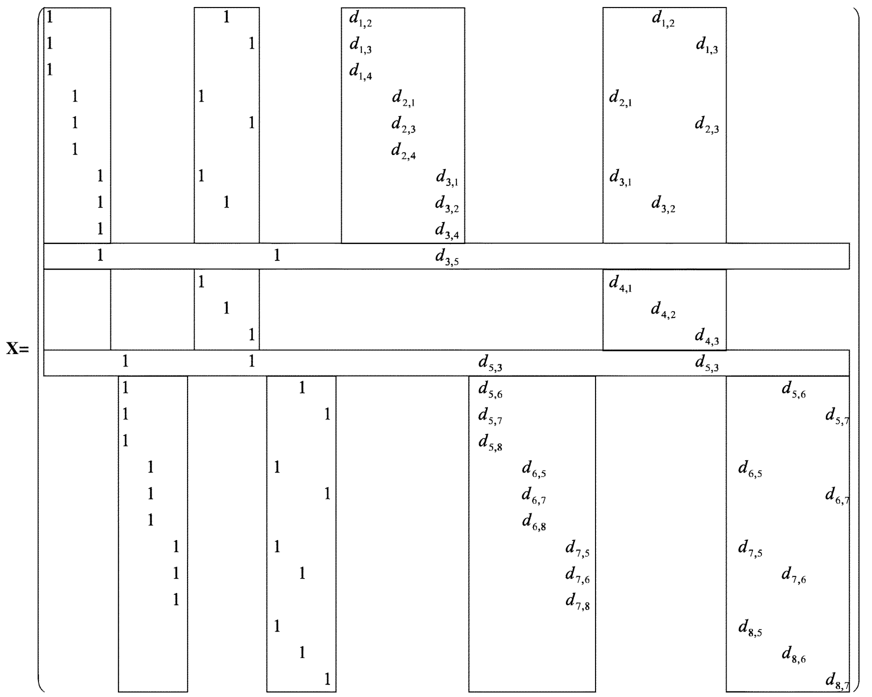

Fig. 4. The design matrix in a bordered block form. This is corresponding to the case of connected clusters in Fig. 3(b). $d_{u,v}$ is the TDOA-measuring distance between nodes u and v.

$R_1, \ldots, R_k$ the row sets of the blocks of $\mathbf{X}[R - B, C]$, and by $C_1, \ldots, C_k$ the column sets of the blocks of $\mathbf{X}[R - B, C]$; i.e., $\mathbf{X}[R_1, C_1], \ldots, \mathbf{X}[R_k, C_k]$ are the blocks of $\mathbf{X}[R - B, C]$. Furthermore, the ith *cluster* of a sensor network is defined as the set of the sensor nodes, of which unknown calibration parameters are $\boldsymbol{\theta}[C_i]$, i.e., $(\theta_j | j \in C_i)$ for $i = 1, \ldots, k$. The ith *cluster matrix*, denoted by $\mathbf{X}^{(i)}$, is defined as $\mathbf{X}[R_i \cup B, C_i]$ for $i = 1, \ldots, k$.

The decomposition algorithm for computing $\eta(\mathbf{X})$ proposed in [20] can be rewritten using the cluster matrices $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(k)}$. Theorem 4 in [20] that enables the decomposition of $\mathbf{X}$ into submatrices is presented below as Theorem 1 using the notations defined in this paper (the proof is omitted). The decomposition algorithm is summarized as Algorithm 1.

Theorem 1: If $\eta(\mathbf{X}) \geq 2|B| - 2$, then

$$\eta(\mathbf{X}) = \min_{i \in \{1, \ldots, k\}} \eta\left(\mathbf{X}^{(i)}\right).$$

The computational benefit and scalability of the decomposition algorithm depends on the network topology. It is not difficult to see that our approach will demonstrate computational advantages (or being scalable) under the following settings: 1) there exist relatively many clusters (i.e., many blocks in the model matrix); and 2) the number of between-cluster links is much smaller than the total number of communication links (i.e., the size of the boarder $|B|$ is much smaller than n). We feel that these settings are not restrictive since a wireless network is likely to be arranged in small (yet many) clusters with relatively few communications between clusters, due to the distributing nature of targets under detection, as well as the requirements on communication congestion and for power management.

Algorithm 1: Computing the calibration redundancy η of a matrix $\mathbf{X}$.

```
Parameters: integer $d \geq 1$.
Input: a calibration matrix $\mathbf{X} \in \mathbb{R}^{2m \times p}$, the border rows $B$ of $\mathbf{X}$, and the row set and column set of blocks $(R_1, R_2, \ldots, R_k)$ and $(C_1, C_2, \ldots, C_k)$.
Set $d = 1$;
Loop
    While $d \leq 2|B| - 2$
        If there exist $\mathbf{X}_{(-d)}$ such that $r(\mathbf{X}_{(-d)}) < r(\mathbf{X})$
            $\eta(\mathbf{X}) = d - 1$ and stop;
        Set $d = d + 1$;
Loop
    While $d \leq 2m - p + 1$
        If there exists $\mathbf{X}^{(i)}_{(-d)}$ such that $r(\mathbf{X}^{(i)}_{(-d)}) < r(\mathbf{X}^{(i)})$ for $i \in \{1, 2, \ldots, k\}$
            $\eta(\mathbf{X}) = d - 1$ and stop;
        Set $d = d + 1$;
```

B. Finding Blocks and Border in $\mathbf{X}$

Once a linear calibration model is established, one could follow a procedure established in [20] to decompose the calibration matrix into blocks and borders so that Algorithm 1 can be applied. The method in [20] to find blocks and borders took an additional step to represent a matrix with a hyper-graph, however. In the wireless sensor network problem, because the original network can be easily represented by a graph as illustrated in Section II-A, this additional step becomes unnecessary. In fact, we are better off in decomposing the calibration matrix by directly utilizing the graph representation of a sensor network. Given a sensor network having n nodes, when the network is modeled by a matrix, the matrix could have up to $n(n-1)$ rows, which is going to be the number of vertices of the graph used for decomposition in [20]. By contrast, in our graph representation of the same network, the number of vertices is only n, which certainly makes the decomposition a lot easier.

In order to decompose the graph representing a wireless sensor network, we need to find the *minimum cut* S of a graph G. A cut is defined as the set of edges of G if there exist two vertices $u, v \in V$ such that all paths between u and v pass through at least one edge of the cut. The minimum cut is the one of minimum cardinality. Removing a cut will turn the original graph into disconnected subgraphs. From the definition of a cut, it is apparent that the rows in $\mathbf{X}$ corresponding to the minimum cut S in G are the border rows of interest. The blocks can be easily isolated out from the rows in $\mathbf{X}$ corresponding to the subgraphs after the removal of B. For example, the minimum cut of the graph in Fig. 2 is {(3,5)} so the border rows are the rows in $\mathbf{X}$ containing $d_{3,5}$ and $d_{5,3}$, which are the tenth and fourteenth rows of $\mathbf{X}$ in Fig. 4.

A number of efficient algorithms to find the minimum cut are available in graph theory [22]. Reference [23, p. 131] introduces such an algorithm with the complexity $O(|E||V|\min\{|E|^{1/2}, |V|^{2/3}\})$. For the details of other faster algorithms, please refer to [24] and [25].

C. Lower Bound of the Calibration Redundancy

Algorithm 1 works very efficiently for a matrix having a small $|B|$ and small-sized blocks but is not so efficient when the size of $\mathbf{X}$ or $|B|$ is large. The problem is that the first loop of Algorithm 1 may take too much computation time since it tests the ranks of the original matrix. Under that circumstance, computing the exact degree of calibration redundancy may become unaffordable.

It turns out a lower bound of the calibration redundancy is valuable for robust estimation. Using an underestimated redundancy degree to construct an LTS estimator can achieve certain degree of robustness though it may not reach the highest attainable robustness level. It is definitely better than using an ordinary LS estimator or arbitrarily choosing a redundancy degree for deciding the trimming parameter h of an LTS estimator.

Searching for a lower bound of the calibration redundancy is usually easier than for the exact redundancy degree. Algorithm 2 presented in the latter part of this section can compute a lower bound of the calibration redundancy for a large-sized $\mathbf{X}$, a large $|B|$, or both. The computation benefit of Algorithm 2 comes from that it avoids testing the original matrix as much as it could.

Before presenting Algorithm 2, we introduce the following theorem and corollaries that allow it. First, define the operator $\oplus$ as

$$\mathbf{X}^{(i)} \oplus \mathbf{X}^{(j)} = \mathbf{X}[R_i \cup R_j \cup B, C_i \cup C_j]$$

where $\mathbf{X}[R_i, C_i]$ are $\mathbf{X}[R_j, C_j]$ are the blocks of $\mathbf{X}$. This notation indicates that $\mathbf{X}$ is constituted by two cluster matrices. Under that circumstance, Theorem 2 suggests a lower bound of $\eta(\mathbf{X})$ when $\mathbf{X} = \mathbf{X}^{(1)} \oplus \mathbf{X}^{(2)}$. This theorem was proven in a general form (called matroid, which includes matrix as a special case) in [20] as Lemma 2. We restate the result using the terms of matrix but omit the proof here.

Theorem 2: Suppose B is the set of row labels of the border rows in $\mathbf{X}$, and $\mathbf{X} = \mathbf{X}^{(1)} \oplus \mathbf{X}^{(2)}$. Define $l(\mathbf{X}) = \max\{\eta(\mathbf{X}^{(1)}) - |B|, 0\} + \max\{\eta(\mathbf{X}^{(2)}) - |B|, 0\} + 1$.

1) If $l(\mathbf{X}) \geq \min\{\eta(\mathbf{X}^{(1)}), \eta(\mathbf{X}^{(2)})\}$,

$$\eta(\mathbf{X}) = \min\left\{\eta\left(\mathbf{X}^{(1)}\right), \eta\left(\mathbf{X}^{(2)}\right)\right\}.$$

2) If $l(\mathbf{X}) < \min\{\eta(\mathbf{X}^{(1)}), \eta(\mathbf{X}^{(2)})\}$,

$$\eta(\mathbf{X}) \geq l(\mathbf{X}).$$

In order to better reflect the lower bound aspect, the following corollary is stated as a direct result from Theorem 2. Note that computing $\min\{\eta(\mathbf{X}^{(1)}), \eta(\mathbf{X}^{(2)}), l(\mathbf{X})\}$ does not involve $\mathbf{X}$ directly but involves the cluster matrices $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$. So computing the lower bound becomes easier.

Corollary 1: $\eta(\mathbf{X}) \geq \min\{\eta(\mathbf{X}^{(1)}), \eta(\mathbf{X}^{(2)}), l(\mathbf{X})\}$.

When it comes to decomposing the original matrix, removing each minimum cut can result in two disjoint blocks (since there will be only two disconnected subgraphs as the result of such removal), but there could exist multiple minimum cuts having the same cardinality, of which the removal could result in more than two blocks. Algorithm 1 could utilize more than two blocks all at once, while Algorithm 2 will utilize only two blocks at each time. However, owing to the structural result stated in Theorem 2, Algorithm 2 can recursively decompose the remaining matrix. Thus, if there is more than one minimum cuts, one can choose any one of them to decompose the matrix into two blocks. What we mean to do next is to present the results that allow for a recursive procedure.

Denote by $\nu(\mathbf{X})$ such that $\nu(\mathbf{X}) \leq \eta(\mathbf{X})$; i.e., $\nu(\mathbf{X})$ is a lower bound of $\eta(\mathbf{X})$. A lower bound $\nu(\mathbf{X})$ can be obtained using Theorem 2. Apparently,

$$\eta\left(\mathbf{X}^{(1)}\right) \geq \nu\left(\mathbf{X}^{(1)}\right), \text{ and } \eta\left(\mathbf{X}^{(2)}\right) \geq \nu\left(\mathbf{X}^{(2)}\right). \quad (8)$$

In addition

$$l(\mathbf{X}) \geq \max\left\{\nu\left(\mathbf{X}^{(1)}\right) - |B|, 0\right\} + \max\left\{\nu\left(\mathbf{X}^{(2)}\right) - |B|, 0\right\} + 1. \quad (9)$$

Using (8) and (9), it is straightforward to get the following result from Corollary 1.

Corollary 2: Define $l_\nu(\mathbf{X}) = \max\{\nu(\mathbf{X}^{(1)}) - |B|, 0\} + \max\{\nu(\mathbf{X}^{(2)}) - |B|, 0\}+1$; then

$$\eta(\mathbf{X}) \geq \min\left\{\nu\left(\mathbf{X}^{(1)}\right), \nu\left(\mathbf{X}^{(2)}\right), l_\nu(\mathbf{X})\right\}.$$

Corollary 2 benefits the computation because $\nu(\mathbf{X}^{(1)})$ and $\nu(\mathbf{X}^{(2)})$ only evaluate a second-layer (and thus smaller) cluster matrices embodied in a first-layer cluster matrix. This action can be applied to the next layer until a cluster matrix can no longer be decomposed. As the decomposition goes deep, the computation becomes less and less demanding; on the other hand, the lower bound found also becomes looser, which is obvious by noting the two inequalities used in (8) and (9), but given the scale of typical sensor networks, the battle is principally on the computation side, suggesting that one will be better off to go as deep as possible in doing the decomposition.

We also make an extension from Theorem 1 in Corollary 3, which returns a lower bound instead of the exact value when the redundancy degree is relatively large. Corollary 3 is a straightforward result from (8) and Theorem 1.

Corollary 3: If $\eta(\mathbf{X}) \geq 2|B| - 2$, then

$$\eta(\mathbf{X}) \geq \min\left\{\nu\left(\mathbf{X}^{(1)}\right), \nu\left(\mathbf{X}^{(2)}\right)\right\}.$$

Finally, we present the following recursive Algorithm 2 based on Corollary 2 and 3. In addition to $\mathbf{X}$, executing Algorithm 2 needs a constant K, which sets a threshold of computation load for Algorithm 2. Recall that the lower-bound algorithm aims at reducing the computation associated with the rank testings before the bound condition in Theorem 1 is satisfied, under which circumstance the rank testings are performed in an exhaustive manner and could thus be expensive for a large $|B|$. The constant K sets an upper bound for the exhaustive rank testings before the bound condition. This K should be set as large as possible because the danger of using a small K is that it may switch the algorithm to the lower-bound portion too soon and thus results in a poor lower bound. For a given hardware setting, suppose that one rank testing of the model matrix takes s seconds. So a total of K rank testings take $K \times s$ seconds. If this computation is tolerable for one's application, then use it; otherwise decrease K. In examples in Section III, we select K to be one million for a computer with 3.6-GHz Pentium CPU and 4-GB memory.

Algorithm 2: Computing the lower bound of $\eta(\mathbf{X})$

```
Parameters: a matrix $\mathbf{Z}$ and integers $a, d, i, l_1, l_2 \geq 1$
Input: a calibration matrix $\mathbf{X} \in \mathbb{R}^{2m\times p}$ and a constant $K$.
Function: Lowerbound($\mathbf{Z}, d$) {
    Find the border rows $B$ from a minimum cut and the corresponding cluster matrices $\mathbf{Z}^{(1)}$ and $\mathbf{Z}^{(2)}$.
    If $|B| \geq |C|$ or $|B| = \min(\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)})$ %If a network is densely connected
        Run the enumerative rank testing algorithm starting from $d$, and return $\eta(\mathbf{Z})$;
    Else if $\binom{|R|}{2|B|-2} \geq K$ %If $|R|$ and/or $|B|$ are large
        Set $l_1 =$ Lowerbound($\mathbf{Z}^{(1)}, d$) %Finding $\nu(\mathbf{Z}^{(1)})$
        Set $l_2 =$ Lowerbound($\mathbf{Z}^{(2)}, d$) %Finding $\nu(\mathbf{Z}^{(2)})$
        Return $\min\{l_1, l_2, \max\{l_1 - |B|, 0\} + \max\{l_2 - |B|, 0\} + 1\}$; %Corollary 2
    Else
        Loop
            While $d \leq 2|B| - 2$
                If there exist $\mathbf{Z}_{(-d)}$ such that $r(\mathbf{Z}_{(-d)}) < r(\mathbf{Z})$
                    Return $d - 1$;
                Set $d = d + 1$;
        Return $\min\{$Lowerbound($\mathbf{Z}^{(1)}, d$), Lowerbound($\mathbf{Z}^{(2)}, d$)$\}$; %Corollary 3
}
Run Lowerbound($\mathbf{X}, 1$)
```

D. Robust Estimation Using the Lower Bound

As we pointed out in Section I, the construction of an LTS estimator needs to consider the redundancy degree $\eta(\mathbf{X})$. In order to attain the maximum breakdown point (or equivalently, the fault tolerance capability), Mili and Coakley [17] stated that the trimming parameter h of an LTS estimator should be in the range $h_L \leq h \leq h_U$, where $h_L = [(4m - \eta(\mathbf{X}))/2]$ and $h_U = [(4m - \eta(\mathbf{X}) + 1)/2]$. When only the lower bound $\nu(\mathbf{X})$ of the calibration redundancy is available, the exact range for the optimal h cannot be computed. Instead, using $\nu(\mathbf{X})$ to replace $\eta(\mathbf{X})$ shifts both h_L and h_U to a larger value. Denote by h'_L and h'_U the new range based on $\nu(\mathbf{X})$ such that

$$h'_L = \left[\left(4m - \nu(\mathbf{X})\right)/2\right], \text{ and } h'_U = \left[\left(4m - \nu(\mathbf{X}) + 1\right)/2\right]. \quad (10)$$

If we choose h from $[h'_L, h'_U]$, it is likely that the chosen h is greater than or equal to h_U. When $h \geq h_U$, the breakdown point of an LTS estimator becomes [17]

$$\epsilon^* \left(T_{\mathrm{LTS}(h)}\right) = \frac{2m - h + 1}{2m} \tag{11}$$

where $T_{\mathrm{LTS}(h)}$ is an LTS estimator whose trimming parameter is chosen from the new range. Note that the larger the h, the worse off the breakdown point. In other words, the consequence of using the lower bound is that the resulting LTS estimator will reach a robustness level lower than the optimally devised LTS estimator.

Even though on the surface a smaller h may give us a better robustness in the resulting robust estimator, this is actually only true for $h \geq h_L$. When $h \leq h_L$, the breakdown point of an LTS estimator is in a complicated form and depends on $\mathbf{y}$ as well as $\mathbf{X}$ [17]. That is the reason why an LTS estimator loses its supposed robustness altogether and may perform worse than an LS estimator when using an overestimated redundancy degree.

Under the circumstances when h_L is not known (because $\eta(\mathbf{X})$ is unknown) but only h'_L and h'_U are known, the safest choice is to let $h = h'_L$. We denote this value as h^*, but the maximum breakdown point of the LTS estimator with $h = h^*$ depends on where h'_L lies.

When $\nu(\mathbf{X})$ is an even number, $h'_L = h'_U \geq h_U$ so that (11) is valid. Then, the maximum breakdown point of the LTS estimator is

$$\epsilon^* \left(T_{\mathrm{LTS}(h^*)}\right) = \frac{[(\nu(\mathbf{X}))/2] + 1}{2m}. \tag{12}$$

From (7) and (11), the fault tolerance capability using h^* is

$$\tau \left(T_{\mathrm{LTS}(h^*)}, \mathbf{X}\right) = [\nu(\mathbf{X})/2] \tag{13}$$

When $\nu(\mathbf{X})$ is an odd number, $h'_U = h'_L + 1$ and the situation becomes complicated. If h'_L is still greater than or equal to h_U, the maximum breakdown point of the LTS estimator becomes

$$\epsilon^* \left(T_{\mathrm{LTS}(h^*)}\right) = \frac{[(\nu(\mathbf{X}) + 1)/2] + 1}{2m} \tag{14}$$

and the corresponding fault tolerance capability is

$$\tau \left(T_{\mathrm{LTS}(h^*)}, \mathbf{X}\right) = [(\nu(\mathbf{X}) + 1)/2]. \tag{15}$$

If h'_L is less than h_U, it implies that $h'_L = h_L$. This is because h_L and h_U are both integers that are apart by at most one, and so are h'_L and h'_U. The situation that $h'_L = h_L$ could happen only when the lower bound $\nu(\mathbf{X})$ is the same as $\eta(\mathbf{X})$. So the maximum breakdown point of the LTS estimator and the fault tolerance capability are the same as those in (6) and (7), respectively.

The difficulty associated with the case where $\nu(\mathbf{X})$ is an odd number is that one does not know the relationship between h'_L and h_U since $\eta(\mathbf{X})$ is unknown, which is the reason why $\nu(\mathbf{X})$ is calculated in the first place. If we always use (14) and (15), it will give an overestimated breakdown point or fault tolerance capability when $h'_L < h_U$, or equivalently, when $\nu(\mathbf{X}) = \eta(\mathbf{X})$. This happens because (15) is derived from (11), which is only valid for an $h \geq h_U$ (and this condition is difficult to verify). Given this difficulty, we suggest using (12) and (13) to calculate the breakdown point and the fault tolerance capability at all times but acknowledge that the actual LTS estimator can probably perform slightly better than what the calculated value suggests.

E. Subcalibration Model in Computing LTS Estimators

The LTS estimator, as defined in (5), can be computed using a FAST-LTS routine [26] in several statistical software packages including R [16], S-Plus [27], and SAS [28]. In order to use the FAST-LTS routine, the user needs to input the trimming parameter h to ensure the resulting estimation to retain an appropriate level of robustness. Our analysis in the previous sections essentially provides the proper way of determining the h to be used.

Once h is determined, computing an LTS estimator using the original $\mathbf{X}$ could still be computationally demanding because the algorithm needs to solve for the minimum sum of h squared residuals out of all the possible combinations. For a large matrix $\mathbf{X}$, we propose a subcalibration model, which makes a few approximations but can greatly speed up the computation.

The subcalibration applies an LTS estimator to the cluster matrix $\mathbf{X}^{(i)}$, instead of the original matrix $\mathbf{X}$, to estimate the parameters associated with sensor nodes in that particular cluster. In this approach, the calibration parameters are estimated separately for each cluster in the network. In so doing, we assume that the sensor nodes that are not in the ith cluster but communicate with the sensor nodes in the ith cluster are microcalibrated. This assumption could possibly reduce the accuracy of the estimation, but the nice aspect of using a robust estimator is that the estimation result is not very sensitive to this assumption. Since the LTS estimator eliminates suspicious data points, inaccurate values resulting from the assumption will be disregarded. As one will see in Section III, doing the calibration for each cluster separately causes little difference as compared to calibrating the whole network using the FAST-LTS routine.

More specifics of the subcalibration model is given as follows. Denote by $\boldsymbol{\theta}^{(i)}$ the parameters associated with the sensor nodes in the ith cluster, by $\mathbf{y}^{(i)}$ the true distances associated with the row labels R_i of $\mathbf{X}^{(i)}$, and by $\mathbf{e}^{(i)}$ the corresponding random noises. Because we assume that the sensor nodes connecting to the ith cluster are microcalibrated, we deem that the parameters associated with those are constant. Then, we can write the following subcalibration model for the ith cluster:

$$\mathbf{y}^{(i)} - \mathbf{X}[R_i, C - C_i](*\,0 \ldots 0\,0.5 \ldots 0.5\,*)^T = \mathbf{X}^{(i)}\boldsymbol{\theta}^{(i)} + \mathbf{e}^{(i)}. \tag{16}$$

In the above model, $(*\ 0\ \ldots\ 0\ .5\ \ldots\ .5\ *)^T$ has the same dimension with the subvector of $\boldsymbol{\theta}$ after $\boldsymbol{\theta}^{(i)}$ is excluded, in which the constants 0's and .5's are the parameters associated with the sensor nodes communicating with sensor nodes in the ith cluster, and the "$*$," meaning that an arbitrary value can be chosen, corresponds to the parameters of the nodes that are neither in the cluster nor communicating with the ith cluster. The computation benefit of using the subcalibration model will become apparent in the numerical examples shown in Section III.

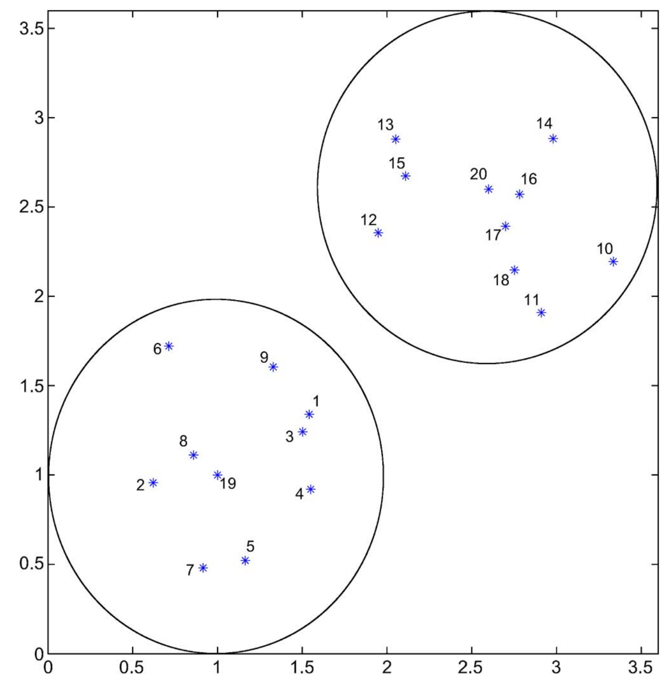

Fig. 5. *Ad-hoc* sensor network example. There are a total of 20 sensors, forming two clusters. The communication limit is one normalized unit of distance.

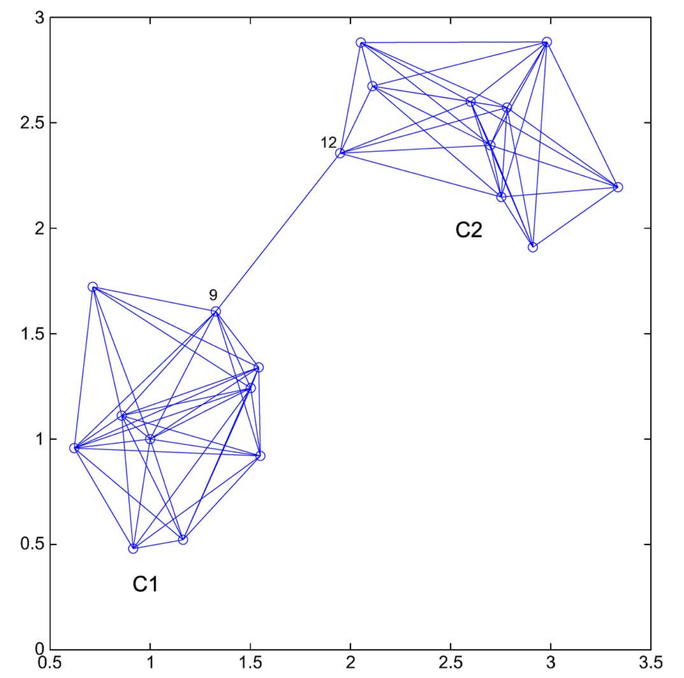

Fig. 6. Graph representation of *ad-hoc* sensor network example. The edge {9,12} is the minimum cut in the graph, corresponding to the border rows in the model matrix.

III. Numerical Examples and Discussions

A. Examples and Comparisons

This section presents two examples of wireless sensor network with different scales and complexities. Fig. 5 shows the configuration of a wireless sensor network, where "$*$" denotes the location of a sensor node. Note that the coordinates in Fig. 5 have been normalized for convenience and thus the unit has no physical meaning. There are a total of 20 sensors, among which the sensors at locations (1.0, 1.0) and (2.6, 2.6) are microcalibrated so that their calibration parameters are set to be constants, and the remaining 18 sensor nodes are ordinary ones whose parameters are to be estimated. The sensors are assigned an index from 1 to 20, where the numbers of 19 and 20 are reserved for the two anchor nodes.

The limit of the communication distance between a pair of sensor nodes is 1 (that is what we chose as the basic unit to scale the sensor network), meaning that the sensor nodes that are apart farther than this limit cannot communicate with each other. Applying this rule to the sensor network in Fig. 5, we can obtain a graph representation of the network in Fig. 6, where one can easily observe two clusters and the between-cluster communication is through the pair {(9,12)}.

The linear calibration matrix can be established following the procedure outlined in the Appendix, or a more detailed procedure in [7]. Given that there are four parameters associated with each sensor node to be calibrated, the dimension of $\boldsymbol{\theta}$ is $p = 72$. From Fig. 6, we count $m = 77$ edges in the graph so that the size of $\mathbf{y}$ is $2m = 154$. This means the calibration matrix $\mathbf{X}$ is a 154×72 matrix. We choose to omit $\mathbf{X}$ here to save space.

Finding the calibration redundancy of a 154×72 matrix is computationally difficult. The enumerative rank testing method will run up to $\sum_{d=1}^{\eta(\mathbf{X})+1} \binom{154}{d}$ rank testings. Since we know $\eta(\mathbf{X}) = 4$ in this case from our latter analysis, $\sum_{d=1}^{5} \binom{154}{d} \simeq 7 \times 10^9$. Even for the best case scenario, where the enumerative algorithm finds $\eta(\mathbf{X})$ at its first rank test right after d reaches 5, it still needs to go through $\sum_{d=1}^{4} \binom{154}{d} + 1 \simeq 2.3 \times 10^8$ rank testings. Implementing the enumerative rank testing in C++, we ran it on a computer equipped with a 3.6-GHz Pentium CPU and 4-GB memory. The program is terminated after a day of computing since it has already taken far more time than the decomposition algorithm.

In order to use the algorithms presented in this paper, we need to identify the BBF of $\mathbf{X}$ first. One can find the minimum cut of the graph in Fig. 6 using the algorithm in [23], or because of the simple network configuration here, one can actually tell which set of edges is the minimum cut by simply observing the graph. The minimum cut is {(9,12)} so the border rows B are associated with $d_{9,12}$ and $d_{12,9}$ ($|B| = 2$). Subsequently, we can identify two cluster matrices $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ corresponding to the sensors contained in each circle in Fig. 5, respectively. The $\mathbf{X}^{(1)}$ is a 80×36 matrix, and $\mathbf{X}^{(2)}$ is a 76×36 matrix.

Since $\binom{|R|}{2|B|-2} = \binom{154}{2}$ is less than K, the constant used in Algorithm 2 (recall we set $K = 10^6$), Algorithm 2 tests the rank of $\mathbf{X}_{(-d)}$ for $d = 1$ to $2|B| - 2 (= 2)$ and finds that $\eta(\mathbf{X}) \geq 2|B| - 2$. Then, we can apply Theorem 1 so that $\eta(\mathbf{X})$ should be the smaller one of the calibration redundancies of the two cluster matrices. Testing the ranks of $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ is much faster, and we get $\eta(\mathbf{X}^{(1)}) = 4$ and $\eta(\mathbf{X}^{(2)}) = 4$. By Theorem 1, $\eta(\mathbf{X}) = 4$. The computation of $\eta(\mathbf{X})$ by the decomposition algorithm took less than two hours on the same computer as mentioned earlier. The number of rank-testing operations is only about 1% of that in the best case scenario for the enumerative algorithm.

TABLE I
MSE AND COMPUTATION TIME OF THE EXAMPLE IN FIG. 6

Number of data corruptions	LS	FAST-LTS	FAST-LTS using sub-calibration model
0	0.0681	0.0817.	0.0814
1	0.4277	0.0792	0.0764
2	0.5067	0.1094	0.0804
Average time for one iteration (in second)	0.06	263.51	13.13

To illustrate the robustness of the LTS estimator, we simulate $M = 100$ instances of the calibration process using the above sensor network and compare the mean of squared errors (MSE) of the parameter estimation with an ordinary LS estimator. We use two methods to compute the LTS estimator; one is to run the FAST-LTS routine [26] in R on model (3) and the second is to run the same routine on the subcalibration models in (16). We assume that a distance measurement in $\mathbf{y}$ is contaminated by a small measurement device error, normally distributed with zero mean and a standard deviation of .002, and a distance measurement in $\mathbf{X}$ is contaminated by a relatively large measurement error with zero mean and a standard deviation of .01. We simulate the corrupted distance measurements due to sensor or communication failures by adding a substantial deviation (up to 100%) to some of the measurements in $\mathbf{X}$. The MSE of the parameter estimation is calculated as

$$\text{MSE} = \frac{1}{M}\sum_{t=1}^{M}(\boldsymbol{\theta}_{\text{true}} - \hat{\boldsymbol{\theta}}_t)^T(\boldsymbol{\theta}_{\text{true}} - \hat{\boldsymbol{\theta}}_t).$$

Since $\tau_{\max} = 2$, we simulate the cases with no, one, and two corrupted measurements, respectively. Table I summarizes the MSEs and the computation time of the LTS estimators and the LS estimator. With the presence of data corruptions, the LTS estimator is more robust than the LS estimator, as indicated by its relatively flat MSE value, whereas the MSE values of the LS estimator escalate rapidly. The former's MSE is about one-fifth of the latter's. Regarding computation, an LTS estimation is obviously much more expensive than an LS estimation, but it is worth noting that utilizing the cluster structure in the network can remarkably reduce the computation of an LTS estimator—the LTS estimation using the subcalibration model consumes about only 5% of the time using the original model (3), while the supposed robustness of an LTS estimator is by and large maintained.

One alternative method engineers sometimes use to handle outliers is a self-diagnosis procedure to identify and eliminate the outliers. It starts with an LS estimation using all measurements, and then, obtain a new LS estimation by hypothetically eliminating one measurement, and do so for every individual measurement. Comparing the new LS estimates with the one obtained without eliminating any measurements could potentially lead to the identification of an outlier. If applying this procedure to a system where there are two outliers, the self-diagnosis procedure needs to run until three measurements are eliminated, implying a total of $154 \times 3 = 462$ executions of the LS estimation. Then, the total time spent using this self-diagnosis procedure is twice more than using the subcalibration based LTS. Furthermore, this sequential way of performing self-diagnosis also suffers from a shortcoming that the existence of multiple outliers makes it difficult to identify the outliers correctly. By comparison, the proposed method is more robust and technically sound than the self-diagnosis mechanism.

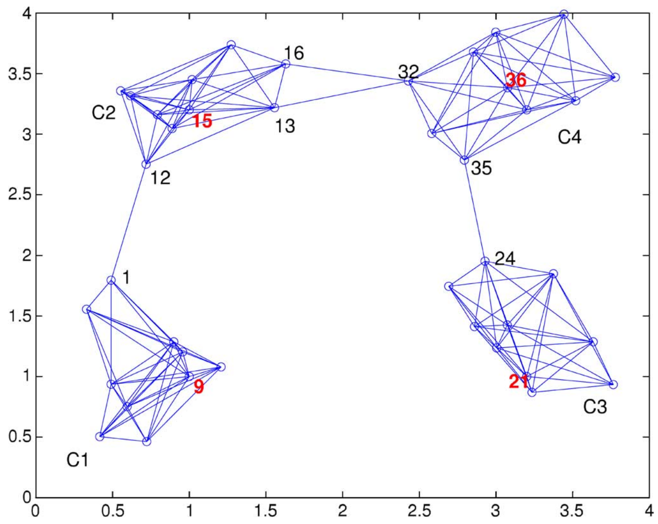

Fig. 7. Graph representation of the second *ad-hoc* sensor network example. There are 40 sensor nodes, forming four clusters. The anchor nodes are {9, 15, 21, 36}.

The second example concerns a network twice larger than (comprising $n = 40$ sensors) that in the first example. The communication limit is the same as before. The graph representation is shown in Fig. 7, where $m = 159$ edges are counted. There are four microcalibrated anchor nodes in this network so that $p = 144$ parameters are associated with the remaining 36 ordinary sensors. The calibration matrix $\mathbf{X}$ is thus of 318 × 144. The size of this matrix makes it almost impossible to use the enumerative rank testing algorithm for computing the calibration redundancy.

It is equally difficult to use Algorithm 1 to compute the exact calibration redundancy in this case. To illustrate this, consider the following. There are two minimum cuts of equal cardinality: {(1,12)} and {(24,35)}, and their cardinality is one. We thus partition $\mathbf{X}$ into $k = 3$ blocks and $|B| = 4$ border rows (since each edge in the graph corresponds to two rows). Algorithm 1 would have to test a total of $\sum_{d=1}^{2|B|-2}\binom{318}{d} \simeq 1.4 \times 10^{12}$ ranks of reduced matrices of $\mathbf{X}$ before testing any cluster matrices.

For a system of this size, it is safer to start with Algorithm 2, which decomposes the original matrix recursively. The first step is to choose either one of the two minimum cuts, say {(1,12)}, to partition the graph and the corresponding matrix. We end up with two border rows ($|B| = 2$) and two cluster matrices, $\mathbf{X}^{(1)}$ of 80 × 36 and $\mathbf{X}^{(2)}$ of 240 × 108, which correspond to C1 and the collection of C2, C3, and C4 in Fig. 7. Since $\binom{|R|}{2|B|-2} = \binom{318}{2} = 50,403 < K$, we test the rank of $\mathbf{X}_{(-d)}$

TABLE II
MSE AND COMPUTATION TIME OF THE EXAMPLE IN FIG. 7

Number of data corruptions	LS	FAST-LTS	FAST-LTS using sub-calibration model
0	0.1066	fail	0.1215
1	0.9569	fail	0.1247
2	1.3928	fail	0.8348
Average time for one iteration (in second)	0.18	> two weeks	28.02

for $d = 1,\ 2$ and find that $\eta(\mathbf{X}) \geq 2$. After that, we need to find the minimum of $\nu(\mathbf{X}^{(1)})$ and $\nu(\mathbf{X}^{(2)})$ as stated in the last line of the function Lowerbound$(\mathbf{X}, d)$ in Algorithm 2.

Second, for the first cluster matrix $\mathbf{X}^{(1)}$, it is no longer beneficial to decompose further since the sensors within the cluster are densely connected; its redundancy degree is computed by simply testing the ranks of the reduced matrices $\mathbf{X}^{(1)}_{(-d)}$. Given its much smaller size, the computation associated with $\mathbf{X}^{(1)}$ is very much affordable. We find that $\eta(\mathbf{X}^{(1)}) = 4$. The cluster matrix $\mathbf{X}^{(2)}$ can, and should, be decomposed. The minimum cut for the subgraph associated with $\mathbf{X}^{(2)}$ is {(24,35)}. After decomposing the second cluster matrix, we obtain two border rows (i.e., $|B| = 2$) for $\mathbf{X}^{(2)}$ and two second-layer cluster matrices $\mathbf{X}^{(2,1)}$ of 160×72 and $\mathbf{X}^{(2,2)}$ of 82×36. Then, we need to find the minimum of $\nu(\mathbf{X}^{(2,1)})$ and $\nu(\mathbf{X}^{(2,2)})$, as we did in the first iteration.

Third, it turns out that $\mathbf{X}^{(2,2)}$, which corresponds to C3, needs no decomposition but $\mathbf{X}^{(2,1)}$ does. Testing on $\mathbf{X}^{(2,2)}$, we find $\eta(\mathbf{X}^{(2,2)}) = 4$. Continue carrying out the decomposition on $\mathbf{X}^{(2,1)}$ similar to what was done above. We find that the minimum cut is {13,32} and {16,32}, so there are four border rows (i.e., $|B| = 4$). The third-layer cluster matrices are $\mathbf{X}^{(2,1,1)}$ of size 86×36 and $\mathbf{X}^{(2,1,2)}$ of size 78×36, which correspond to C2 and C4 in Fig. 7, respectively. Now, for $\mathbf{X}^{(2,1)}$, $\binom{|R|}{2|B|-2} = \binom{160}{6} \simeq 2 \times 10^{10} \geq K$, so we proceed to find $\nu(\mathbf{X}^{(2,1)})$ using Corollary 1. For that, we obtain $\eta(\mathbf{X}^{(2,1,1)}) = 5$ and $\eta(\mathbf{X}^{(2,1,2)}) = 4$, so $l(\mathbf{X}^{(2,1)}) = \max\{4 - 4, 0\} + \max\{5 - 4, 0\} + 1 = 2$. By Corollary 1, $\nu(\mathbf{X}^{(2,1)}) = \min\{\eta(\mathbf{X}^{(2,1,1)}), \eta(\mathbf{X}^{(2,1,1)}), l(\mathbf{X}^{(2,1)})\} = 2$.

Finally, we can get a lower bound by $\eta(\mathbf{X})$ by combining the results given above. We have $\nu(\mathbf{X}^{(2,1)}) = 2$ and $\nu(\mathbf{X}^{(2,2)}) = \eta(\mathbf{X}^{(2,2)}) = 4$. By Corollary 3, a lower bound of $\eta(\mathbf{X}^{(2)})$ is $\nu(\mathbf{X}^{(2)}) = 2$. Using Corollary 3 one more time (with $\nu(\mathbf{X}^{(1)}) = \eta(\mathbf{X}^{(1)}) = 4$), one can get a lower bound of $\eta(\mathbf{X})$ as $\nu(\mathbf{X}) = 2$.

Given this lower bound $\nu(\mathbf{X})$, the trimming parameter in LTS estimation is $h^* = 317$ according to (10). Using this h^* to construct an LTS estimator leads to a robust calibration estimate with the fault tolerance capability of $\tau^*(T_{LTS(h^*)}, \mathbf{X}) = 1$. The simulation results of the second example, performed under the same setup of the previous example, are summarized in Table II. We report a "fail" under the column of FAST-LTS because it failed to compute the LTS estimation after continuously running for two weeks. By comparison, the LTS estimator using the sub-calibration model is much faster and finishes in about half a minute. The MSE of the LTS estimator is considerably lower than that of the LS estimator when there is one sensor fault or one corrupted measurement but not so much better when the number of data corruptions becomes two. This numerical result is consistent with the theoretical analysis of the fault tolerance capability associated with this LTS estimator (which indicates $\tau^*(T_{LTS(h^*)}, \mathbf{X}) = 1$). The actual redundancy could be higher but without knowing the exact redundancy degree it is safer to use the lower bound value that leads to certain robustness at a suboptimal level.

B. Discussions on Scalability

We made comments about the scalability issue at the end of Section II-A and outlined the conditions under which the decomposition algorithm can work well. Here, we would like to provide some further elaborations.

The time consumption of using the proposed method comes from two aspects: the time to decompose the whole system into subsystems (i.e., decompose a big model matrix into smaller submatrices) and the time to compute the LTS based on the subcalibration models (using individual submatrices). The latter part, i.e., the computational complexity of LTS algorithm is a subject studied in statistical literatures. Our method does not directly improve the LTS routine itself. Our method can do better because it is much faster to conduct the LTS computations on smaller submatrices. So our focus of discussions here is regarding the decomposition.

From Algorithm 1, the computation time of the decomposition is determined by how many times one needs to evaluate $r(\mathbf{X}_{(-d)})$ or $r(\mathbf{X}^{(i)}_{(-d)})$. We call such an evaluation of the rank of a matrix a *rank test*. The total number of rank tests required for a system is a function of the number of rows, its border size $|B|$, and its calibration redundancy η. Suppose that there are k blocks in a matrix, each of which has R number of rows, and the bound condition in Algorithm 1 is indeed satisfied (namely $\eta \geq 2|B| - 2$). Then, the total number of rows in $\mathbf{X}$ is $kR + |B|$. When using our decomposition algorithm, we can calculate the total number of rank tests that are needed to find η as

$$\sum_{d=1}^{2|B|-2} \binom{kR + |B|}{d} + \sum_{d=2|B|-1}^{\eta+1} \binom{|B| + R}{d}.$$

When using the exhaustive search algorithm, the total number of rank tests is

$$\sum_{d=1}^{\eta+1} \binom{kR + |B|}{d}.$$

Tables III and IV compare the numbers of rank tests from some selected systems of different sizes when using both methods. The first author has also performed a numerical study in his dissertation [29]. It is worth to note that the first example in [29, Table III], which uses a matrix of 26 rows, matches the first example in Table IV. The ratio of the number of rank tests between the two methods is 78, very close to the ratio of the CPU times in [29, Table III] (which is 80). This verifies that the CPU time is indeed proportional to the number of rank tests.

TABLE III
RANK TEST RESULTS WHEN $|B| = 1, R = 12, \eta = 6$

Design matrix		Total number of rank tests	
# Rows	k	Exhaustive search	Decompose
49	4	1.02×10^8	23,244
97	8	1.39×10^{10}	46,488
193	16	1.84×10^{12}	92,976
385	32	2.40×10^{14}	185,952
769	64	3.10×10^{16}	371,904
1,537	128	3.98×10^{18}	743,808

TABLE IV
RANK TEST RESULTS WHEN $|B| = 2, R = 4, \eta = 4$

Design matrix		Total number of rank tests	
# Rows	k	Exhaustive search	Decompose
26	4	83,681	1,079
50	8	2,369,935	2,731
98	16	7.17×10^7	7,763
194	32	2.23×10^9	24,739
386	64	7.05×10^{10}	86,339
770	128	2.24×10^{12}	320,131
1,538	256	7.15×10^{13}	1,230,083

The rank-test results (and the computational results in [29, Table III] as well) show that the decomposition algorithm has a reasonably good scalability when the model matrix is properly structured.

IV. CONCLUSION

This paper discusses the computation aspect of using a robust regression estimator for a robust calibration of an *ad-hoc* wireless sensor network. When an LTS estimator is used, one can gain certain robustness against outliers, measurement corruptions, and/or violation of model assumptions, but the down side is its expensive computation, especially for a large scale network. Our research presents one model that enables a decomposition procedure, an algorithm of computing the lower bound of calibration redundancy, and one subcalibration model that can greatly reduce the computation of solving for the LTS estimation. The computational benefit and the gain in robustness are illustrated using two sensor network examples. In our examples in Section III, we only demonstrate the robustness of an LTS estimator against data corruptions. In fact, an LTS estimator is also able to provide robustness in estimation against other types of disturbances such as violation of normality and model uncertainty; please refer to [8] for more details.

Our method provides more accurate estimates of the distance-determining parameters associated with sensor nodes. The benefit is that the more accurate the parameters are used during the service time, the more accurate between-node distances an TDOA mechanism can produce, and then, the more accurate a sensor node can be localized. Even though our approach helps improve the computation aspects for robust calibration, the current version is still not fast enough in real time. The current version is more suitable for offline applications or for periodically calibrating a sensor network during a downtime maintenance mode.

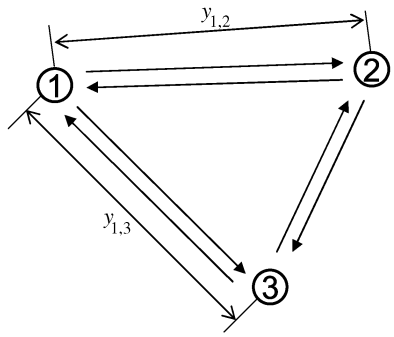

Fig. 8. Wireless sensor network example. There are three sensor nodes. y denotes the true between-sensor distance.

Please note that in this paper we only perform an *analysis* (rather than *synthesis*) of redundancy and fault tolerance degree for the calibration problem. In the examples presented in Section III, one can observe that the levels of fault tolerance capability are not high, suggesting that the LTS can only provide a safeguard against a few sensor failures. The low fault tolerance capability observed in the examples points to the necessity of developing optimization methods that can improve the robustness level of a sensor network. Maximizing the fault tolerance capability of a sensor network is certainly computationally expensive, and solving it is more challenging than the analysis work presented here. Answering this question is outside the scope of this paper but that is actually where our continual research efforts are going.

APPENDIX
LINEAR CALIBRATION MODEL

This appendix provides additional steps showing how model (3) is aggregated from model (2).

In model (2), we set four parameters $(\alpha, \beta, \gamma, \delta)$ for a single wireless sensor node because a sensor node is assumed to work as both a transmitter and a receiver. Denote by Λ the set of the available true distances. In Fig. 8, for example, suppose that the available true distances are $y_{1,2}$ and $y_{1,3}$; then, $\Lambda = \{y_{1,2}, y_{1,3}\}$. The actual number of the true distances used in the model doubles the cardinality of Λ since the between-node communications are two-way. We need to duplicate the elements in Λ such that $y_{u,v} = y_{v,u}$ and include them in a new vector $\boldsymbol{\lambda}$. For the example in Fig. 8,

$$\boldsymbol{\lambda} = (y_{1,2}\; y_{2,1}\; y_{1,3}\; y_{3,1})^T.$$

Denote by $\mathbf{b}$ the $4n \times 1$ vector of all the calibration parameters

$$\mathbf{b} = (\alpha_1 \ldots \alpha_n\; \beta_1 \ldots \beta_n\; \gamma_1 \ldots \gamma_n\; \delta_1 \ldots \delta_n)^T.$$

In Fig. 8, we have three sensor nodes, so $\mathbf{b}$ is a 12×1 vector. For a given $\boldsymbol{\lambda}$ and $\mathbf{b}$, it is straightforward to obtain the following matrix expression from (2)

$$\boldsymbol{\lambda} = \mathbf{G}\mathbf{b} + \mathbf{e} \tag{17}$$

where $\mathbf{G}$ is a $2m \times 4n$ matrix function. For the sensor network in Fig. 8, the $\mathbf{G}$ matrix is

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & d_{1,2} & 0 & 0 & 0 & d_{1,2} & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & d_{2,1} & 0 & d_{2,1} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & d_{1,3} & 0 & 0 & 0 & 0 & d_{1,3} \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & d_{3,1} & d_{3,1} & 0 & 0 \end{pmatrix}. \tag{18}$$

In order to uniquely estimate $\mathbf{b}$ in (17), $\mathbf{G}$ should be of full column rank. A necessary condition is to have $2m \geq 4n$. This condition can be achieved if we include more sensors in a network. The number of pairwise distances $n(n-1)/2$ increases faster than $4n$ the number of parameters.

In many cases, however, even if $2m \geq 4n$ holds, we may still run into a $\mathbf{G}$ that is not of full column rank. It turns out that the following linear dependence relationship exists:

$$\mathbf{g}_1 + \mathbf{g}_2 +, \ldots, +\mathbf{g}_n = \mathbf{g}_{n+1} + \mathbf{g}_{n+2} + \mathbf{g}_{2n} = (1\,1\,\ldots,\,1)^T$$

where $\mathbf{g}_i$ denotes the ith column vector of $\mathbf{G}$. This means that the original calibration formulation introduced in the wireless network literature ([7], [30]) over-parameterizes the system. In order to uniquely estimate the calibration parameters, additional constraints should be used to make the linear matrix of full column rank.

The constraint we use here comes from the small number of anchor nodes; we assume that each cluster has one anchor node, which is able to communicate with all the nodes in a cluster. This appears feasible even if the anchor node is not in the center of a cluster because an anchor node is typically more powerful in terms of communication. According to [7], the anchor nodes can be *microcalibrated*, meaning that these sensor nodes are regularly maintained and their distance-determining parameters are accurate. Then, the calibration parameters associated with the anchor nodes can be set as constants. In this research, the parameters of the anchor nodes are set as (0, 0, .5, .5), which were suggested in [7]. Through hardware adjustment, one could possibly set the parameters to other constant values.

Let $\boldsymbol{\theta}'$ be the vector of the calibration parameters of anchor nodes and $\boldsymbol{\theta}$ be the vector of the unknown calibration parameters. Permute the rows in $\mathbf{b}$ such that

$$\mathbf{b}^T = (\boldsymbol{\theta}\,\boldsymbol{\theta}')^T.$$

Likewise, denote by $\mathbf{X}$ the submatrix of $\mathbf{G}$ associated with $\boldsymbol{\theta}$ and by $\mathbf{X}'$ the submatrix of $\mathbf{G}$ associated with $\boldsymbol{\theta}'$. Permute the columns in $\mathbf{G}$ such that

$$\mathbf{G} = (\mathbf{X}\,\mathbf{X}').$$

Then, (17) can be rewritten as

$$\boldsymbol{\lambda} - \mathbf{X}'\boldsymbol{\theta}' = \mathbf{X}\boldsymbol{\theta} + \mathbf{e}. \tag{19}$$

By defining $\mathbf{y} = \boldsymbol{\lambda} - \mathbf{X}'\boldsymbol{\theta}'$, we construct the calibration model in (3), of which the number of elements in $\boldsymbol{\theta}$ is denoted by $p(< 4n)$.

In this paper, $\mathbf{X}$ is assumed to be of full column rank. We also assume that there exists a central processor that can access the global information about the network. Once the communication linkages are established in a sensor network, each node can return its connectivity information to the central processor, which can be used to establish a graph structure of the network for the procedures in Section II.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Comput. Netw.*, vol. 38, pp. 393–422, 2002.

[2] P. Enge and P. Misra, "Special issue on global positioning system," *Proc. IEEE*, vol. 87, pp. 3–172, 1999.

[3] C. Shen, C. Srisathapornphat, and C. Jaikaeo, "Sensor information networking architecture and applications," *IEEE Personal Commun.*, vol. 8, no. 4, pp. 52–59, Aug. 2001.

[4] J. Beutel, "Location management in wireless sensor networks," in *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, M. Ilyas and I. Mahgoub, Eds. Boca Raton, FL: CRC, 2005, pp. 20-1–20-23.

[5] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proc. Mobile Comput. Netw.*, 2000.

[6] J. Hightower, C. Vakili, G. Borriello, and R. Want, Design and Calibration of the Spoton Ad-Hoc Location Sensing System 2001 [Online]. Available: citeseer.ist.psu.edu/hightower01design.html

[7] K. Whitehouse and D. Culler, "Macro-calibration in sensor/actuator networks," *Mobile Netw. Applicat.*, vol. 8, pp. 463–472, 2003.

[8] R. R. Wilcox, *Introduction to Robust Estimation and Hypothesis Testing*. San Diego, CA: Academic, 2005.

[9] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, III, R. L. Moses, and S. C. Neiyer, "Locating the nodes: Cooperative localization in wireless sensor networks," *IEEE Signal Process. Mag.*, pp. 54–69, 2005.

[10] H. Chan, M. Luk, and A. Perrig, "Using clustering information for sensor network localization," in *Distrib. Comput. Sens. Syst.*. Berlin/Heidelberg, Germany: Springer, 2005, pp. 109–125.

[11] A. Ihler, J. Fisher, III, R. Moses, and A. S. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *IEEE J. Sel. Areas Commun.*, pp. 809–819, 2005.

[12] C. Taylor, A. Rahimi, J. Bachrach, H. Shrobe, and A. Grue, "Simultaneous localization, calibration, and tracking in an ad hoc sensor network," in *Proc. 5th Int. Conf. Inf. Process. Sens. Netw.*, Nashville, TN, 2006, pp. 27–33.

[13] P. J. Rousseeuw, "Least median of squares regression," *J. Amer. Statist. Assoc.*, vol. 79, pp. 871–880, 1984.

[14] D. L. Donoho and P. J. Huber, "The notion of breakdown point," in *A Festschrift for Erich L. Lehmann*, P. Bickel, K. Doksum, and J. L. Hodges, Jr., Eds. Belmont, CA: Wadsworth, 1983, pp. 157–184.

[15] L. Mili, M. G. Cheniae, and P. J. Rousseeuw, "Robust state estimation of electric power systems," *IEEE Trans. Circuits Syst.*, vol. 41, no. 5, pp. 349–358, May 1994.

[16] [Online]. Available: http://www.r-project.org

[17] L. Mili and C. W. Coakley, "Robust estimation in structured linear regression," *Ann. Statist.*, vol. 24, pp. 2593–2607, 1996.

[18] M. Luong, D. Maquin, C. T. Huynh, and J. Ragot, "Observability, redundancy, reliability and integrated design of measurement systems," in *Proc. 2nd IFAC Symp. Intell. Compon. Instrum. Control Applicat. SICICA 94*, Budapest, Hungary, 1994, pp. 8–10.

[19] M. Staroswiecki, G. Hoblos, and A. Aïtouche, "Sensor network design for fault tolerant estimation," *Int. J. Adaptive Control Signal Process.*, vol. 18, pp. 55–72, 2004.

[20] J. J. Cho, Y. Chen, and Y. Ding, "On the (co)girth of connected matroids," *Discrete Appl. Math.*, vol. 155, pp. 2456–2470, 2007.

[21] C. Aykanat, A. Pinar, and Ü. V. Çatalyürek, "Permuting sparse rectangular matrices into block-diagonal form," *SIAM J. Sci. Comput.*, vol. 25, pp. 1860–1879, 2004.

[22] O. R. Oellermann, "Connectivity and edge-connectivity in graphs: A survey," *Congressus Numerantium*, vol. 116, pp. 231–252, 1996.

[23] S. Even, *Graph Algorithms*. Potomac, MD: Computer Science Press, 1979.

[24] D. W. Matula, "Determining edge connectivity in $O(nm)$," in *28th Annu. Symp. Foundation Comput. Sci.*, 1987, pp. 249–251.

[25] H. Nagamochi and T. Ibaraki, "Computing edge-connectivity in multigraphs and capacitated graphs," *SIAM J. Discrete Math.*, vol. 5, pp. 54–66, 1992.

[26] P. J. Rousseeuw and K. V. Driessen, "Computing LTS regression for large data sets," *Data Mining and Knowledge Discovery*, vol. 12, pp. 29–45, 2006.

[27] [Online]. Available: http://www.splus.com

[28] [Online]. Available: http://www.sas.com

[29] J. J. Cho, "On the clustered sensor networks," Ph.D. dissertation, Texas A&M Univ., College Station, TX, 2007, p. 61.

[30] K. Whitehouse and D. Culler, "Calibration as parameter estimation in sensor networks," in *Proc. 1st ACM Int. Workshop Wireless Sens. Netw. Applicat.*, Atlanta, GA, 2002, pp. 59–67.

Jung Jin Cho received the M.S. and B.S. degrees from Seoul National University, Seoul, Korea, and the Ph.D. degree from Texas A&M University, College Station.

He currently works as an Actuarial Engineer with Baker Hughes, Inc, Houston, TX. His job involves predicting and maximizing the performance incentives in drilling and evaluation contracts with major oil companies by applying his research expertise in statistical data-mining and modeling

Yu Ding (M'01) received the B.S degree in precision engineering from University of Science and Technology of China, Hefei, in 1993, the M.S. degree in precision instruments from Tsinghua University, Beijing, China in 1996, the M.S. degree in mechanical engineering from the Pennsylvania State University, State College, in 1998, and the Ph.D. degree in mechanical engineering from the University of Michigan, Ann Arbor, in 2001.

He is currently an Associate Professor in the Department of Industrial and Systems Engineering at Texas A&M University, College Station. His research interests are in the area of systems informatics and control. He serves as a Department Editor of the *IIE Transactions*

Dr. Ding currently serves as an Associate Editor of IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING. He is a member of INFORMS, IIE, and ASME.

Yong Chen (M'05) received the B. E. degree in computer science from Tsinghua University, Beijing, China, in 1998, and the M.S. degree in statistics and the Ph.D. degree in industrial and operations engineering, both from the University of Michigan, Ann Arbor, in 2003.

He is currently an Assistant Professor in the Department of Mechanical and Industrial Engineering at the University of Iowa, Iowa City. His research interests include quality and reliability engineering, pattern recognition, applied statistics, and simulation optimization. He currently serves as an Associate Editor of *Naval Research Logistics*.

Dr. Chen is a member of the INFORMS, the IIE, and ASA.

Jiong Tang (M'09) received the B.S. and M.S. degrees in applied mechanics from Fudan University, Shanghai, China, in 1989 and 1992, respectively, and the Ph.D. degree in mechanical engineering from Pennsylvania State University, State College, in 2001.

He worked at the GE Global Research Center as a Mechanical Engineer from 2001 to 2002. He is currently an Associate Professor in the Mechanical Engineering Department, University of Connecticut, Storrs. His research interests include structural dynamics and system dynamics, control, and sensing and monitoring. He serves as an Associate Editor for the *ASME Journal of Vibration and Acoustics*

Dr. Tang serves as an Associate Editor of the IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT.