Proceedings of the 3rd Annual
IEEE Conference on Automation Science and Engineering
Scottsdale, AZ, USA, Sept 22-25, 2007

Robust Calibration for Localization in Clustered Wireless Sensor Networks

Jung Jin Cho, Yu Ding, Yong Chen, and Jiong Tang

Abstract— **This paper presents a robust calibration procedure for a clustered wireless sensor network. The calibration problem is often formulated as a parameter estimation problem using a linear calibration model. For reducing or eliminating unwanted influences of measurement corruptions or outliers on parameter estimation, a robust regression estimator is a natural choice. In order to solve a robust estimation problem more efficiently, we utilize cluster structure in a network configuration and decompose a large network into smaller subsystems that can be solved much faster. To this end, we present two algorithms for a robust calibration procedure. Two examples are presented to illustrate how the proposed methods enable robust calibration in a sensor network.**

I. Introduction

Wireless technologies have changed the design and operation of sensor networks. Equipped with micro-electro-mechanical systems (MEMS), a wireless sensor node becomes small, mobile, and multi-functional. One of the most significant changes caused by the wireless technologies is the implementation of an *ad-hoc* networking, which refers to those having a network topology that can change frequently [1]. The frequent changes in the topology of an ad-hoc network naturally call for a solution to the *localization* or *location tracking* problem because knowing the positions of individual sensors is often the pre-requisite 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 on every micro-sensor 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; first, measure the distances between itself and several anchor nodes, and then, compute its location based on certain geometry principle (e.g., hyperbolic trilateration, triangulation, and multilateration).

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 [4]. One problem associated with this distance measuring approach is its inaccuracy. For example, RF signals are attenuated by metal objects, and the speed of acoustic signals are highly influenced by temperature and moisture [4]. The experiments performed by Whitehouse and Culler [5] showed that the error of a between-node distance measurement using acoustic time of flight could be as large as 300% of the true distance. To tackle this issue, Whitehouse and Culler [5] recommended using a *calibration* procedure as follows. In an off-line setting, the true distance between sensor nodes can be measured by an independent and accurate means; then, a mathematical model mapping the distance to the true distance is established. The mapping model established adjusts the distance measurements, during the service of sensor nodes, to a more accurate estimation of the true distances.

Denote by $d_{u,v}$ the measured distance between transmitter u and receiver v, by $y_{u,v}$ the true distance, and by e the random noise. Whitehouse and Culler [5] proposed a linear calibration model such as

$$y_{u,v} = \alpha_u + \beta_v + \gamma_u d_{u,v} + \delta_v d_{u,v} + e, \tag{1}$$

where α_u and β_v are the bias of a transmitter u and a receiver v, and γ_u and δ_v are the gain of u and v, respectively. Model (1) 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 (1) 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 and p sensor nodes need to be calibrated. In a matrix form, the calibration model for all sensor nodes becomes

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \mathbf{e}, \tag{2}$$

where the elements in $\mathbf{y}$ are the true distances but $\mathbf{y}$ 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 measured distances as some of its entries, and $\mathbf{e}$ is the vector of noises. During a calibration phase when $\mathbf{y}$ 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 $\boldsymbol{\theta}$ to predict the between-node distances.

Given the linear model structure in (2), it comes as no surprise that the least-squares (LS) estimation is the most

J. J. Cho and Y. Ding are with the Department of Industrial and Systems Engineering, Texas A&M University, College Station, TX 77840, USA jjcho@tamu.edu, yuding@iemail.tamu.edu

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

J. Tang is with the Department of Mechanical Engineering, University of Connecticut, Storrs, CT 06269, USA jtang@engr.uconn.edu

1-4244-1154-8/07/$25.00 ©2007 IEEE.

popular method used for estimating $\boldsymbol{\theta}$; i.e.,

$$\hat{\boldsymbol{\theta}}_{LS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \quad (3)$$

In fact, this is what Whitehouse and Culler [5] used in their procedure. However, statistical research [6] 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. That is, when a distance measurement $d_{u,v}$ is accidentally corrupted, LS estimation of the calibration parameters becomes far deviated from true values.

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 [7] attracted our attention.

Applying a robust estimator (a popular choice is the least trimmed squares (LTS) estimator [8] due to its high robustness) to the sensor network calibration is, however, not a simple task. There are two major challenges, both related to some unique features of 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.

The second challenge is related to the structure of a wireless sensor network. Due to the limited power available on individual sensor nodes, sensor nodes rarely communicate with all other sensors in a network. Instead, the whole network is usually grouped into a few 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. Mili and Coakley [9] provided the condition under which an LTS estimator can achieve the highest robustness. In fact, Mili and Coakley's condition is actually a function of a previously defined *degree of sensor redundancy* of a network (defined in [10]). Obtaining the redundancy degree for a large network once again runs into a computation issue. For a large sensor network, it is almost impossible to compute the redundancy degree by using the enumerative algorithm proposed in [10].

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 more efficient. In fact, the decomposition algorithm presented in [11] can compute the redundancy degree faster for a structured linear model. Our first objective is to show how the decomposition algorithm can be applied to a clustered wireless sensor network.

During our investigation, we notice that for a large sensor network, even the decomposition algorithm in [11] may fall short of computing the degree of redundancy for the application of an LTS estimator. 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 robust statistics tells us that using an overestimated redundancy degree will cause a robust estimator to lose its robustness, while using an underestimated redundancy degree could still retain a certain level of robustness [9] . Hence, our second objective is to compute a lower bound of the redundancy degree by utilizing the cluster structure in a network configuration. We will show that computing the lower bound can be done much faster and the lower bound provides considerable benefit in achieving robustness.

The remainder of the paper is organized as follows. Section 2 provides introduction to the necessary concepts regarding robust estimation. In Section 3, we show how the decomposition algorithm is applied to a clustered sensor network and how the lower bound of sensor redundancy can be computed. Section 4 presents two examples of the robust calibration procedure and compares the performances of different approaches. Finally, we conclude the paper in Section 5.

II. Calibration using robust regression on linear calibration model

The popular LS estimator, which is given by (3), minimizes the sum of squared residual $w_i^2 = (y_i - \mathbf{x}_i\hat{\boldsymbol{\theta}}_{LS})^2$ for $i = 1, \ldots, 2m$. Suppose there exist corrupted distance measurements, which are deviated far from the true distances; the corrupted distance measurements may have substantial influence on the parameter estimation since the residuals are "squared". In other words, the LS estimator may overemphasize the corrupted distance measurements and fail to obtain an accurate estimation of the parameters. To *reduce* the influences of corrupted measurements, statisticians proposed several estimators that minimize the sum of $\rho(w_i)$, a symmetric function that has the global minimum at zero. These estimators minimizing $\rho(w_i)$ are called *M-estimators*.

The M-estimators have certain limitation in its robustness. If a corruption occurs at the so-called *leverage* point, the corrupted data could still have considerable influence on the M-estimators [6]. In order to achieve a higher level of robustness, Rousseeuw [7] proposed the LTS estimator, which can *eliminate* the influence of some corrupted measurements. The LTS estimator is given by

$$\min \sum_{i=1}^{h} w_{(i)}^2, \quad (4)$$

where h is called the trimming parameter, and $w_{(1)}^2 \leq w_{(2)}^2 \leq \ldots \leq w_{(2m)}^2$ are the squared residuals $(y_i - \mathbf{x}_i\hat{\boldsymbol{\theta}})^2$ for

$i = 1, \ldots, 2m$ 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 [12] introduced the concept of finite sample breakdown point, which is defined in terms of the data points $\mathcal{Z}$ and the estimator $T(\mathcal{Z})$, where

- $\mathcal{Z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{2m}, y_{2m})\}$ (note that $\mathbf{X} = (\mathbf{x}_1^T, \ldots, \mathbf{x}_{2m}^T)^T$ and $\mathbf{y} = (y_1, \ldots, y_{2m})^T$ in model (2)), and
- $T(\mathcal{Z})$ denotes the estimator of $\boldsymbol{\theta}$; i.e., $T(\mathcal{Z}) = \hat{\boldsymbol{\theta}}$.

The breakdown point, which measures the robustness of the estimator T for a given $\mathcal{Z}$, is mathematically defined as

$$\epsilon(T, \mathcal{Z}) = \min\{\frac{q}{2m} : \sup_{\mathcal{Z}'} ||T(\mathcal{Z}') - T(\mathcal{Z})|| < \infty\},$$

where $\mathcal{Z}'$ is the set of data points, of which q data points in $\mathcal{Z}$ are arbitrarily replaced. This mathematical definition says that an estimator T is considered *broken down* when the difference between the estimator based on a corrupted data set $\mathcal{Z}'$ (i.e., $T(\mathcal{Z}')$) and the estimator based on the correct data set $\mathcal{Z}$ (i.e., $T(\mathcal{Z})$) can be infinite. In other words, q data corruptions in $\mathcal{Z}'$ can have an arbitrarily large effect on T if $q/2m$ is greater than the breakdown point ϵ.

Researchers further proved that there exists an upper bound of the breakdown point for the regression equivariant estimators [7] [9]. Based on the theoretical results presented in Mili and Coakley [9], the maximum attainable breakdown point $\epsilon^*_{\max}$ can be expressed

$$\epsilon^*_{\max} = \frac{[\eta(\mathbf{X})/2] + 1}{2m}, \tag{5}$$

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 also the degree of sensor redundancy as defined in [10]. Note that $\epsilon^*_{\max}(\mathcal{Z})$ does not depend on the estimator T. It is actually regarded as the maximum value over all possible regression equivariant estimators for a given $\mathcal{Z}$. On the other hand, an estimator T that can attain the maximum breakdown point $\epsilon^*_{\max}(\mathcal{Z})$ is called a *high* breakdown point estimator.

For a better engineering interpretation, $\epsilon^*_{\max}$ is transformed into an integer;

$$\tau_{\max}(\mathbf{X}) = 2m \cdot \epsilon^*_{\max}(\mathcal{Z}) - 1 = [\eta(\mathbf{X})/2], \tag{6}$$

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 $\mathbf{X}$.

A LTS estimator is a high breakdown point estimator and attains the fault tolerance capability; thus, we choose to use LTS estimation for the calibration problem. 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 [9] 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 $h_U = [(4m - \eta(\mathbf{X}) + 1)/2]$. Obtaining $\eta(\mathbf{X})$ is critical for properly using the LTS estimator and determining its breakdown point. Incorrect $\eta(\mathbf{X})$ could lead to an improper choice of h and may cause a robust estimator to lose its robustness. The next section discusses the method to compute $\eta(\mathbf{X})$ or a lower bound of it for calibrating a clustered sensor network.

III. Calibration redundancy and its lower bound

The redundancy degree $\eta(\mathbf{X})$ is also called *calibration redundancy* in this paper. Mathematically, it was defined in [10] as

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

where $r(\cdot)$ represents the rank function of a matrix and $\mathbf{X}_{(-d)}$ is the reduced matrix after deleting d rows in $\mathbf{X}$. 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 $\boldsymbol{\theta}$.

The commonly used algorithm to compute this redundancy degree is the enumerative rank testing algorithm [10], which literally follows the definition of $\eta(\mathbf{X})$ and tests the ranks of all the reduced matrix $\mathbf{X}_{(-d)}$. 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 1, a decomposition algorithm developed in [11] 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, and then we also devise a recursive procedure to obtain a lower bound of the calibration redundancy for the even larger systems of which the exact redundancy degree is too expensive to compute.

A. Application of the decomposition algorithm

The basic idea of the decomposition algorithm in [11] is as follows. The search for $\eta(\mathbf{X})$ can be performed in two stages if $\mathbf{X}$ can be transformed into a *bordered block form* (BBF) such as

$$\begin{pmatrix} \mathbf{A}_1 & & \\ & \ddots & \\ & & \mathbf{A}_k \\ \mathbf{B}_1 & \ldots & \mathbf{B}_k \end{pmatrix}. \tag{7}$$

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 $\mathbf{A}_i$ and $\mathbf{B}_i$ for $i = 1, \ldots, k$, until the redundancy is found.

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 $r(\mathbf{X}^{(i)}_{(-d)}) < r(\mathbf{X}^{(i)})$
      $\eta(\mathbf{X}) = d - 1$ and stop;
    Set $d = d + 1$;

A clustered sensor network 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 $\mathbf{A}_i$, and the between-cluster communication links correspond to the border rows $\mathbf{B}$, which is $(\mathbf{B}_1, \ldots, \mathbf{B}_k)$.

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 $R_1, \ldots, R_k$ 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 $\mathbf{X}[R - B, C]$. 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 [11] can be rewritten using the cluster matrices $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(k)}$. Theorem 4 in [11] 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

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

Once a linear calibration model is established, one can follow the procedure established in [11] to decompose the calibration matrix into blocks and borders so that Algorithm 1 can be applied. For details about finding blocks and border rows in $\mathbf{X}$, please refer to hypergraph based algorithm presented in [11].

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) {
  If $|B| \geq |C|$ or $|B| = \min(\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)})$
    Run the enumerative algorithm from d, and return $\eta(\mathbf{Z})$;
  Else if $\binom{|R|}{2|B|-2} \geq K$
    Set $l_1 = $ Lowerbound($\mathbf{Z}^{(1)}, d$);
    Set $l_2 = $ Lowerbound($\mathbf{Z}^{(2)}, d$);
    Return $\min\{l_1, l_2,$
      $\max\{l_1 - |B|, 0\} + \max\{l_2 - |B|, 0\} + 1\}$;
  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($\mathbf{Z}^{(2)}, d$)$\}$;
}

Run Lowerbound($\mathbf{X}$, 1)

B. 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 parameters h of an LTS estimator.

Searching for a lower bound of the calibration redundancy is usually much 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 tests merely the ranks of cluster matrices and avoids testing the original matrix altogether. Algorithm 2 is based on Theorem 2, which suggests a lower bound of $\eta(\mathbf{X})$.

Theorem 2: Let $l(\mathbf{X}) = \max\{\eta(\mathbf{X}^{(1)}) - |B|, 0\} + \max\{\eta(\mathbf{X}^{(2)}) - |B|, 0\} + 1$. If $k = 2$,

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

Theorem 2 is direct result of Lemma 2 in [11], which is proven in a general form (called matroid, which includes matrix as a special case). Executing Algorithm 2 needs a constant K, which decides the condition of that which corollary to invoke. This constant K sets a threshold of computation load for Algorithm 2. Using a small K may

reduce the computation time, but may result in a poor lower bound that is far from the actual $\eta(\mathbf{X})$. On the contrary, using a large K could lose the computation benefit of Algorithm 2 even though the lower bound found may be close to the actual $\eta(\mathbf{X})$. To that extent, K should be chosen appropriately meeting the computing requirement of a problem. We select K to be one million for a computer with 3.6GHz Pentium CPU and 4G memory in examples in Section 4.

C. Robust estimation using the lower bound

According to [9], the LTS estimator attains the maximum breakdown point when the trimming parameter h is appropriately chosen to be between h_L and h_U. 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, we can compute h'_U the new range based on $\nu(\mathbf{X})$ such that

$$h'_U = [(4m - \nu(\mathbf{X}) + 1)/2]. \tag{8}$$

If we choose h to be h'_U, then h is greater than or equal to h_U. Then, the breakdown point of an LTS estimator becomes [9]

$$\epsilon^*(T_{LTS(h'_U)}, \mathcal{Z}) = \frac{[(\nu(\mathbf{X}))/2] + 1}{2m}. \tag{9}$$

where $T_{LTS(h'_U)}$ is an LTS estimator whose trimming parameter is chosen to be h'_U. 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.

From (6) and (9), the fault tolerance capability using h'_U is

$$\tau(T_{LTS(h'_U)}, \mathbf{X}) = [\nu(\mathbf{X})/2] \tag{10}$$

IV. Examples

This section presents two examples of wireless sensor network with different scales and complexities. Figure 1 shows the graph representation of a wireless sensor network, where one can easily observe two clusters and the between-cluster communication is through the pair $\{(9, 12)\}$.

From Figure 1, we count $m = 72$ 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 (18 out of 20 sensors are needed to be calibrated). We choose to omit $\mathbf{X}$ here to save space.

In order to use the algorithms presented in this paper, we need to identify the BBF of $\mathbf{X}$ first. Because of the simple network configuration here, one can actually tell which set of edges is the minimum cut and correspond to border rows by simply observing the graph. The minimum cut is $\{(9, 12)\}$ so the border rows B are associated with $d_{9,12}$ and $d_{12,9}$ ($|B| = 2$).

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. The computation

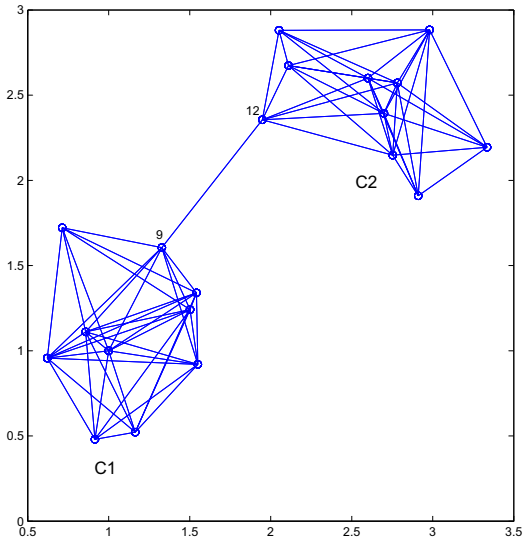

Fig. 1. Graph represnetation of ad-hoc sensor network example

of $\eta(\mathbf{X})$ by the decomposition algorithm took less than two hours on the same computer as mentioned earlier, going through about three millions rank-testings. The number of rank-testing operations is only about 1% of that in the best case scenario for the enumerative algorithm.

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 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 $\mathbf{X}$. Table I summarizes the MSE's 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.

The second example concerns a network twice larger (comprising $n = 40$ sensors) than that in the first example. The graph representation is shown in Figure 2, where $m = 159$ edges can be observed. There are four micro-calibrated 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 ma-

Number of data corruptions	LS	LTS
0	0.0681	0.0814
1	0.4277	0.0764
2	0.5067	0.0804
Average time for one iteration (in second)	0.06	13.13

TABLE I
MSE AND COMPUTATION TIME OF THE EXAMPLE IN FIGURE 1

Number of data corruptions	LS	LTS
0	0.1066	0.1215
1	0.9569	0.1247
2	1.3928	0.8348
Average time for one iteration (in second)	0.18	28.02

TABLE II
MSE AND COMPUTATION TIME OF THE EXAMPLE IN FIGURE 2

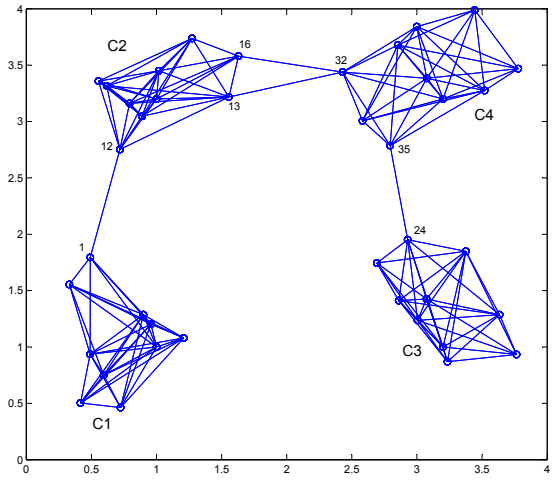

Fig. 2. Graph representation of the second ad-hoc sensor network example

trix 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. For a system of this size, it is safer to start with Algorithm 2, which decomposes the original matrix recursively. By applying Algorithm 2, 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 (8). 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. 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.

V. CONCLUSION

This paper discusses the computation aspect of using a robust regression estimator such as the LTS 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. The down side is the expensive computation for a large scale network. This paper presents two decomposition algorithms that can remarkably reduce the computation demand for the calibration redundancy.

Robust fusion of sensor data is an interesting extension of the analysis in this paper. For example, we may be able to reduce false alarm rates of a wireless surveillance system by trimming some suspicious surveillance observations.

The low fault tolerance capability observed in the examples raises another research issue: maximizing the fault tolerance capability of a sensor network. However, it is more computationally demanding than evaluating the fault tolerance capability. That is where our continual research efforts are going.

REFERENCES

[1] I. F. Akyildiz, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor netowrks: a survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.
[2] P. Enge and P. Misra, "Special issue on global positioning system," *Proceedings of the IEEE*, vol. 87, pp. 3–172, 1999.
[3] C. Shen, C. Srisathapornphat, and C. Jaikaeo, "Sensor information networking architecture and applications," *IEEE Personal Communications*, vol. 8, pp. 52–59, 2001.
[4] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proceedings of Mobile computing and Networking*, 2000.
[5] K. Whitehouse and D. Culler, "Macro-calibration in sensor/actuator networks," *Mobile Networks and Applications*, vol. 8, pp. 463–472, 2003.
[6] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*. Hoboken, New Jersey: John Wiley & Sons, 2003.
[7] P. J. Rousseeuw, "Least median of squares regression," *Journal of the American Statistical Association*, vol. 79, pp. 871–880, 1984.
[8] L. Mili, M. G. Cheniae, and P. J. Rousseeuw, "Robust state estimation of electric power systems," *IEEE Transactions on Circuits and Systems*, vol. 41, pp. 349–358, 1994.
[9] L. Mili and C. W. Coakley, "Robust estimation in structured linear regression," *Annals of Statistics*, vol. 24, pp. 2593–2607, 1996.
[10] M. Staroswiecki, G. Hoblos, and A. Aïtouche, "Sensor network design for fault tolerant estimation," *International Journal of Adaptive Control and Signal Processing*, vol. 18, pp. 55–72, 2004.
[11] J. J. Cho, Y. Chen, and Y. Ding, "On the (co)girth of connected matroids," Accepted, a technical report version is available at http://ie.tamu.edu/people/faculty/Ding/Technical.Report.DAM.pdf, 2006.
[12] D. L. Donoho and P. J. Huber, "The notion of breakdown point," in *A Festschrift for Erich L. Lehman*, P. Bickel, K. Doksum, and J. L. H. Jr., Eds. Belmont, California: Wadsworth, 1983, pp. 157–184.