Localization in Wireless Sensor Networks with Gradient Descent

Dapeng Qiao, Grantham K.H. Pang
Department of Electrical and Electronic Engineering
The University of Hong Kong
{dpqiao, gpang}@eee.hku.hk

Abstract

In this article, we present two distance-based sensor network localization algorithms. The location of the sensors is unknown initially and we can estimate the relative locations of sensors by using knowledge of inter-sensor distance measurements. Together with the knowledge of the absolute locations of three or more sensors, we can also determine the locations of all the sensors in the wireless network. The proposed algorithms make use of gradient descent to achieve excellent localization accuracy. The two gradient descent algorithms are iterative in nature and result is obtained when there is no further improvement on the accuracy. Simulation results have shown that the proposed algorithms have better performance than existing localization algorithms. A comparison of different methods is given in the paper.

1. Introduction

Localization in wireless sensor networks is a hot topic since the position information is needed in applications such as wide life monitoring, habitat monitoring and military surveillance. This paper gives two new methods based on the gradient descent approach to locate the sensors based on the inter-sensor distance measurements.

In the above location-aware systems, there are usually hundreds of or even thousands of low-cost sensor nodes. In addition, based on the signals received from other nodes, it would know its distance from these nodes. Estimation on the location of these nodes is an important issue for any sensor network. It is necessary to accurately localize the sensors in order to measure data which is geographically meaningful. This localization issue has been studied by many researchers and there are many different methods and algorithms [1-5] dealing with this situation.

Generally speaking, the situation considered in this paper is the following. There are lots of nodes in the whole region, which can communicate with each other. The absolute positions of some of the nodes may be known, but most of them are with unknown positions. The localization of all the nodes depends not only on the known positions of some nodes, but also on the type of measured information among the nearby nodes, which leads to localization methods based on distance measurement based on TOA (time-of-arrival), or RSS (received signal strength).

Some algorithms can first estimate the relative coordinates of all the nodes (including known nodes and unknown nodes) based on the inter-sensor distance measurements. Then, the absolute coordinates of the unknown nodes can be found by arranging the relative coordinates to fit the coordinates of known nodes. In the 2D situation, only three known nodes can determine the whole plane in which all the relative coordinates are placed.

1.1. Related work

Multidimensional scaling (MDS) [1,2] can recover the positions of the unknown nodes, but it must know distances between any two nodes in the whole network. Some researchers employ MDS as the core step in their algorithms. MDS-MAP [1] utilizes classical MDS and obtains a success on positioning with small variation of distance measurement. First, based on the connectivity and distance information between nodes, a rough estimate of relative node distance is made. Then relative positions are obtained by using a Singular Value Decomposition on the estimated distance information matrix. Finally absolute positions of the unknown nodes are estimated based on the relative positions and the positions of the known nodes. The computation complexity of this method is about $O(n^3)$ time for a sensor network of n nodes. SMACOF (Scaling by Majorizing A COmplicated Function) uses an iterative method to tackle the multidimensional scaling problem. Xiang and Hongyuan [2] apply this algorithm on the issue of sensor network positioning, but un-convergence is common with SMACOF algorithm.

978-1-4577-0253-2/11/$26.00 ©2011 IEEE

Doherty et al [3] model range and angular constraints in sensor network localization as convex constraints. The resulting minimization problem can be solved efficiently using semi-definite program (SDP). SDP in some particular conditions can become linear programming (LP) which is much more computational efficient. The aim is to minimize a linear function over a polyhedron.

Triangulation and multi-lateration greatly depends on the density of the anchors. If there are not enough anchors, the error of the estimation of unknown nodes will be accumulated and become very inaccurate.

Proximity distance matrix (PDM) is used in [4], which aims to build a transformation between the proximity and the distance. It first estimates the distance to at least three anchors of every unknown node using PDM. Then it still needs to use multi-lateration or triangulation to change the result of PDM into the position of the unknown nodes.

Nonlinear dimensionality reduction is also used in sensor network localization. Chengqun et al. [5] employ the geodesic distances to measure the dissimilarity between sensors, and propose a centralized algorithm based on isomap technique. In specific, they first construct the neighborhood graph by using the sensors and their pair-wise distance, and then compute the geodesic distance of each pair of sensors. Finally, the 2D embedding is constructed and the relative coordinate system of the sensors using MDS is obtained. Since the performance of isomap is sensitive to the given parameter, in order to alleviate the influence of the parameter, they also propose an adaptive parameter selection procedure based on the true locations of the anchors and their transformed locations.

Patwari et al [6] utilize the manifold learning techniques (including isomap) for localization problem under the spatially correlated sensor model. Generally, this algorithm is similar to the classic algorithm MDS-MAP [1], because isomap can be considered as a geodesic distance version of the MDS. Instead of using the Euclidean distance for embedding, isomap considers the geodesic distance on a weighted neighbouring graph.

1.2. Algorithms based on Gradient descent

The problem of localization on sensor network can be thought as an optimization problem which is to find the most suitable positions of the sensors to fit the known inter-sensor distance measurements. Ravi Garg, et al [7] set the target function as the distance error between the known distances and the estimated distances based on the estimated positions of sensors. The derivative of the function with respect to the positions of a sensor is obtained. However, their method is used to update the positions of only one node for each iteration.

Similar to gradient descent, some other methods also use the gradient to find the best direction for iteration. Patwari, et al [8] use Polka-Ribiere updating to find the direction of an localization issue and reach good results. Biswas et al. [10] use gradient descent to localize the unknown nodes based on the maximum likelihood estimation. However, it sometimes stays in the local minima, which affects the accuracy of the estimation.

Gradient descent is also used in combination with other methods stated before. Biswas et al. [9] apply SDP first, and use gradient descent to refine the result of SDP. In their simulation, the accuracy of estimated positions is improved significantly.

In this paper, two gradient descent algorithms for sensor network localization are given. At the end of the iteration in each method, the sum of the squared errors between the given distance measurements and the solution is minimized. The locations of all the unknown nodes are then obtained.

2. Gradient descent method A (GDA)

The idea and the detail of the first gradient descent algorithm are explained next. For a sensor network with n nodes, the inter-sensor distances $d_{k,l}$ ($1 \le k < l \le n$, k, l are the indexes of the kth and lth node) are known. Let node 1 be one of the unknown nodes with location (x_1, y_1). In this gradient descent algorithm, it is based on the idea that a change to the value of x_1 would affect all the distances related to it. These distances are $d_{1,2}, d_{1,3}, \ldots, d_{1,n}$. Hence,

$$x_1 = w_{1,2}d_{1,2} + w_{1,3}d_{1,3} + \ldots + w_{1,n}d_{1,n} \quad (1)$$

x_1 is expressed as a linear function of $d_{1,2}, d_{1,3}, \ldots, d_{1,n}$ with weights $w_{1,2}, w_{1,3}, \ldots, w_{1,n}$.

Similarly,

$$y_1 = v_{1,2}d_{1,2} + v_{1,3}d_{1,3} + \ldots + v_{1,n}d_{1,n} \quad (2)$$

and the weights are $v_{1,2}, v_{1,3}, \ldots, v_{1,n}$. For node i, similar weights ($w_{i,j}$ and $v_{i,j}$ $1 \le j \le n, j \ne i$) are also defined.

These relationships are shown in the lower part of Fig. 1.

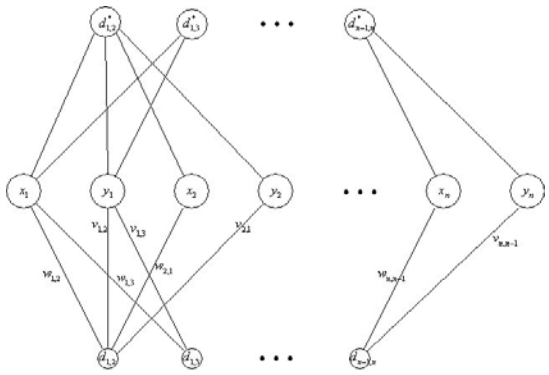

Fig. 1 The weights defined to connect the unknown coordinates with the known distances in n nodes situation

When the locations of the unknown nodes have been estimated, the following equation would calculate the distance between each pair of nodes (e.g. node *k* and *l*):

$$d^*_{k,l} = \sqrt{(x_k - x_l)^2 + (y_k - y_l)^2} \tag{3}$$

The target function is the sum of the squared errors between the given distance measurements ($d_{k,l}$) and the distance calculated based on the estimated location of the nodes ($d^*_{k,l}$).

$$E = 0.5 \times \sum_{\substack{k,l=1 \\ k<l}}^{n} \left(d_{k,l} - d^*_{k,l} \right)^2 \tag{4}$$

The idea of the algorithm is to minimize the target function with successive iteration of the algorithm. The partial derivatives of the target function (*E*) with respect to each weight ($w_{i,j}$ and $v_{i,j}$) need to be computed for each iteration.

Gradient descent is used to find the best weights $w_{i,j}$ $v_{i,j}$ which can minimize the target function. At the end of the iteration, the estimated positions of the sensors can be calculated by equations like equation (1) and (2).

The detail of the algorithm is given below:

1. There are n nodes with unknown locations, $(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_n, y_n)$.

2. The known information is the distances between any pair of nodes. The distances can be written in a distance matrix with definition of:

$$\mathrm{D} \triangleq \begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,n} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,n} \\ \vdots & \vdots & & \vdots \\ d_{n,1} & d_{2,n} & \cdots & d_{n,n} \end{pmatrix} \tag{5}$$

D is symmetric. The element $d_{k,l}$ represents the distance between node *k* and node *l*. $d_{k,l}$ is zero when $k = l$.

The target is to find the positions (x_1, y_1), (x_2, y_2), (x_3, y_3), …, (x_n, y_n).

Note that the results obtained are of relative coordinates.

We define the target function as (4).

The relationship between (x_k, y_k) $k = 1, 2, 3, ..., n$ and $d_{k,l}$ $1 \le k < l \le n$ is expressed by

$$\mathbf{x} = diag(\mathbf{w} \times D^T) \tag{6}$$

$$\mathbf{y} = diag(\mathbf{v} \times D^T) \tag{7}$$

where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \ \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix};$$

$diag(A)$ returns the main diagonal of A;

D^T is the transpose of D. It equals to D because D is symmetric.

$$\mathbf{w} \triangleq \begin{bmatrix} 0 & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & 0 & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \cdots & 0 \end{bmatrix}$$

is the matrix containing all weights between *x* coordinate and the distances. For example, $w_{k,l}$ is the weight connecting *x* coordinate of node *k*: x_k and the distance $d_{k,l}$ if $k < l$ or the distance $d_{l,k}$ if $k > l$.

$$\mathbf{v} \triangleq \begin{bmatrix} 0 & v_{1,2} & \cdots & v_{1,n} \\ v_{2,1} & 0 & \cdots & v_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n,1} & v_{n,2} & \cdots & 0 \end{bmatrix}$$

is the matrix containing all weights between *y* coordinate and the distances.

Because of the definition of $d^*_{k,l}$ in (3) $d^*_{k,l}$ is only connected to x_k, y_k, x_l, y_l. Measurement $d_{k,l}$ is related to x_k, y_k, x_l, y_l by weights in (6) (7).

E in (4) is decreased for each iteration by updating $\mathbf{w}$, $\mathbf{v}$ towards the direction of its derivative.

$$\mathbf{w}^{i+1} = \mathbf{w}^i - K \frac{dE}{d\mathbf{w}^i} \tag{8}$$

where

$$\frac{dE}{d\mathbf{w}}=\begin{bmatrix} 0 & dE/dw_{1,2} & \cdots & dE/dw_{1,n} \\ dE/dw_{2,1} & 0 & \cdots & dE/dw_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ dE/dw_{n,1} & dE/dw_{n,2} & \cdots & 0 \end{bmatrix}$$

K is step size.

$$\mathbf{v}^{i+1}=\mathbf{v}^{i}-K\frac{dE}{d\mathbf{v}^{i}} \tag{9}$$

where

$$\frac{dE}{d\mathbf{v}}=\begin{bmatrix} 0 & dE/dv_{1,2} & \cdots & dE/dv_{1,n} \\ dE/dv_{2,1} & 0 & \cdots & dE/dv_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ dE/dv_{n,1} & dE/dv_{n,2} & \cdots & 0 \end{bmatrix}$$

The steps of the methods are:

1. Initialize $\begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_n & y_n \end{pmatrix}$.

2. Compute $\frac{dE}{dw_{k,l}}$ and $\frac{dE}{dv_{k,l}}$ in (8) and (9), $k,l=1,2,3,...,n$

$\frac{dE}{dw_{k,l}}$ can be calculated by:

$$\frac{dE}{dw_{k,l}}=\sum_{\substack{q=1\\q<k}}^{n}\frac{dE}{d(d^{*}_{k,q})}\frac{d(d^{*}_{k,q})}{dx_k}\frac{dx_k}{dw_{k,l}}=\sum_{\substack{q=1\\q<k}}^{n}(d^{*}_{k,q}-d_{k,q})\frac{x_k-x_q}{d^{*}_{k,q}}d_{k,l}$$

Similarly,

$$\frac{dE}{dv_{k,l}}=\sum_{\substack{q=1\\q<k}}^{n}\frac{dE}{d(d^{*}_{k,q})}\frac{d(d^{*}_{k,q})}{dy_k}\frac{dy_k}{dv_{k,l}}=\sum_{\substack{q=1\\q<k}}^{n}(d^{*}_{k,q}-d_{k,q})\frac{y_k-y_q}{d^{*}_{k,q}}d_{k,l}$$

3. Update $\mathbf{w}^{i+1}=\mathbf{w}^{i}-K\frac{dE}{d\mathbf{w}^{i}}$

and $\mathbf{v}^{i+1}=\mathbf{v}^{i}-K\frac{dE}{d\mathbf{v}^{i}}$

(*i* means the *i*th iteration)

4. Go to step 2, till $\begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_n & y_n \end{pmatrix}$ becomes convergent.

3. Gradient descent method B (GDB)

This gradient descent method is based on finding the partial derivative of the target function with respect to the nodes' coordinates.

Assume:

1. There are *n* unknown nodes.
2. The distances between any pair of nodes are collected in the distance matrix below.

$$\mathrm{D}\triangleq\begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,n} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,n} \\ \vdots & \vdots & & \vdots \\ d_{n,1} & d_{2,n} & \cdots & d_{n,n} \end{pmatrix}$$

which is same as the statement in GDA.

The positions of n nodes (x_1,y_1), (x_2,y_2), (x_3,y_3), …, (x_n,y_n) are also the target that we want to find.

We also use gradient descent to minimize the target function, which is defined as below

$$E=0.5\times\sum_{\substack{k,l=1\\k<l}}^{n}\left(d_{k,l}-d^{*}_{k,l}\right)^2$$

where $d_{k,l}$ is the known distance between node *k* to node *l*,

$$d^{*}_{k,l}\triangleq\sqrt{(x_k-x_l)^2+(y_k-y_l)^2}$$ and

$(x_k,y_k),(x_l,y_l)$ are unknown.

Gradient descent needs to update coordinates instead of weights for each iteration such as:

$$(x_k,y_k)=(x_k,y_k)-K(dE/dx_k,dE/dy_k)$$

where $k=1,2,3,...,n$ K is step size.

The steps of this method are:

1. Initialize $\begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_n & y_n \end{pmatrix}$.

2. Compute $\frac{dE}{dx_k}$ and $\frac{dE}{dy_k}$ $k=1,2,3,...,n$

$\frac{dE}{dx_k}$ can be calculated by:

$$\frac{dE}{dx_k}=\sum_{\substack{l=1\\l<k}}^{n}\frac{dE}{d(d^{*}_{k,l})}\frac{d(d^{*}_{k,l})}{dx_k}=\sum_{\substack{l=1\\l<k}}^{n}(d^{*}_{k,l}-d_{k,l})\frac{x_k-x_l}{d^{*}_{k,l}}$$

Similarly,

$$\frac{dE}{dy_k}=\sum_{\substack{l=1\\l<k}}^{n}\frac{dE}{d(d^{*}_{k,l})}\frac{d(d^{*}_{k,l})}{dy_k}=\sum_{\substack{l=1\\l<k}}^{n}(d^{*}_{k,l}-d_{k,l})\frac{y_k-y_l}{d^{*}_{k,l}}$$

3. Update (x_k,y_k) $k=1,2,3,...,n$ by

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix}=\begin{bmatrix} x_k \\ y_k \end{bmatrix}-K\begin{bmatrix} \frac{dE}{dx_k} \\ \frac{dE}{dy_k} \end{bmatrix}$$

K is the step size.

4. Go to step 2, till (x_k, y_k) $k = 1, 2, 3, ..., n$ becomes convergent.

4. Simulations

In the first simulation, the two gradient descent methods presented in this paper are compared with MDS, SMACOF, PDM and another gradient descent method GD_Ravi [7]. The situation is as follows:

- There are totally 50 nodes randomly located in a square of [0, 10] by [0, 10]; in which, 10 nodes are anchors.
- The distances between any two nodes are known.
- The noise of the distance measurement is generated by a Gaussian distribution generator in Matlab. The elements in the distance matrix of equation (5) is generated by the MATLAB function *normrnd*.

The true locations of the 50 nodes are shown in Fig. 2. With the noisy distance measurements, the locations of all the nodes are estimated by several localization methods and their results are shown in Fig. 3. The abscissa shows different noise levels. Standard derivation $\sqrt{\sigma_N}$ is set to 0.3, 0.6, 0.9, … , 2.7, 3 in the evaluation. The accuracy can be judged by the error per node, which means the average distance for each node between the true position and estimated position. For different standard derivation value, the error per node for MDS, GDA, SMACOF, GDB, GD_Ravi and PDM are given. The figure shows that GDA and GDB obtained the same result and are better than other methods.

Fig.4 shows the computation time required by the various methods in the Matlab environment. The time used by GDA and GDB are much less than the time used by SMACOF and the GD_Ravi gradient descent method. The accuracy of the methods for different percentages of anchors is shown in Fig. 5. GDA and GDB also perform best for different percentages of nodes as anchors. In the simulation, the step size *K* is an important parameter in an iterative algorithm. This is because the parameter is related to the speed of convergence in the gradient descend algorithms. A suitable value has been used, and this value can be made adaptive in the iterative process so as to optimize the performance of computation.

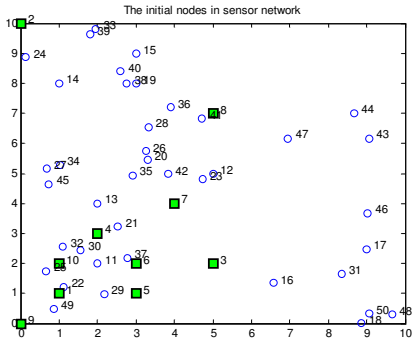

Fig. 2 A sensor network with 50 nodes. The true positions of the 10 anchors are marked as small squares, while the rest nodes are marked as small circles

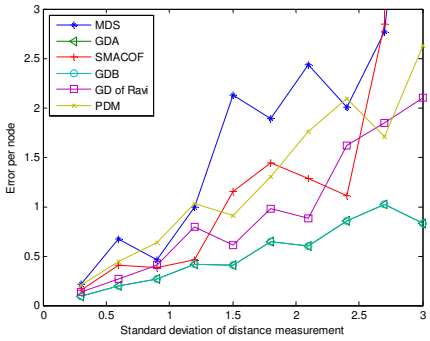

Fig. 3 Accuracy of the methods for increasing distance measurement noise

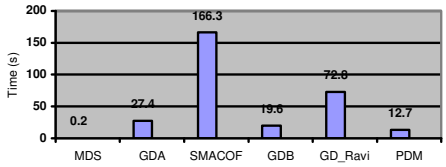

Fig. 4 Comparison on the computation time of the methods

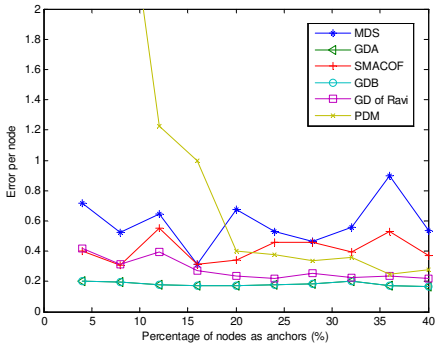

Fig. 5 Error for different percentages of anchors

The density of the nodes in a network is not always uniform. Some researchers try to find suitable algorithm in anisotropic networks. The next simulation result gives the performance of the methods in an anisotropic network. Fig.6 shows a network of 'C' shape. There is no nodes located in the area 3<x-axis and 3<y-axis<7 in the 'C' shape network. With 10 anchors, the accuracy of the methods is given in Fig.7. The figure shows GDA, GDB outperform the other methods in accuracy. It should be noted that both algorithms arrive at the same result at the end of the iterations.

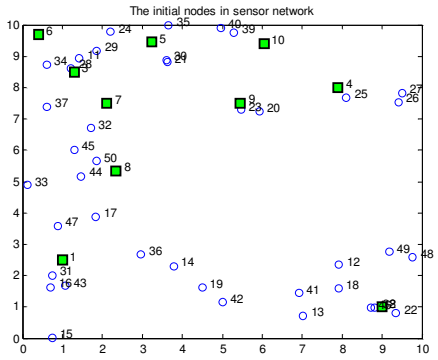

Fig. 6 A 'C' shape network with anchors being marked as small squares

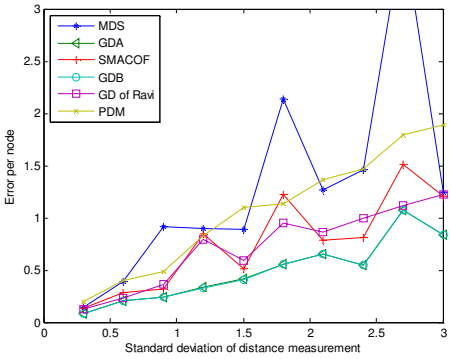

Fig. 7 Accuracy of the methods for increasing distance measurement noise in a 'C' shape network

5. Conclusion

In this paper, two localization methods based on gradient descent are given. The gradient descent methods would minimize the difference between the measured distances and the distances from estimated locations. From the comparison with other well-known localization methods, the two newly developed gradient descend algorithms can reach better accuracy at the expense of computation complexity. This is not surprising as the proposed algorithms are iterative in nature, as opposed to the MDS algorithm which provides an estimate to the solution using the SVD computation.

The new gradient descend algorithms have similar improvements when the problem has more anchors. In future work, the parameter K (step size) can be made adaptive in the convergence process to speed up the computation.

6. References

[1] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz, "Localization from mere connectivity," in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking* Annapolis, Maryland, USA: ACM, 2003.

[2] J. Xiang and Z. Hongyuan, "Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, 2004, pp. 2652-2661 vol.4.

[3] L. Doherty, K. S. J. pister, and L. El Ghaoui, "Convex position estimation in wireless sensor networks," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2001, pp. 1655-1663 vol.3.

[4] L. Hyuk and J. C. Hou, "Localization for anisotropic sensor networks," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 2005, pp. 138-149 vol. 1.

[5] W. Chengqun, C. Jiming, S. Youxian, and S. Xuemin, "Wireless Sensor Networks Localization with Isomap," in *Communications, 2009. ICC '09. IEEE International Conference on*, 2009, pp. 1-5.

[6] N. Patwari and A. O. Hero, III, "Manifold learning algorithms for localization in wireless sensor networks," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, 2004, pp. iii-857-60 vol.3.

[7] R. Garg, A. L. Varna, and W. Min, "Gradient descent approach for secure localization in resource constrained wireless sensor networks," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pp. 1854-1857.

[8] N. Patwari, A. O. Hero, III, M. Perkins, N. S. Correal, and R. J. O'Dea, "Relative location estimation in wireless sensor networks," *IEEE Transactions on Signal Processing,* vol. 51, pp. 2137-2148, 2003.

[9] P. Biswas, T.-c. Liang, K.-c. Toh, T.-c. Wang, and Y. Ye, "Semidefinite programming approaches for sensor network localization with noisy distance measurements," *IEEE Transactions on Automation Science and Engineering,* vol. 3, 2006.

[10] R. Biswas and S. Thrun, "A passive approach to sensor network localization," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 2004, pp. 1544-1549 vol.2.