Passive Localization: Large Size Sensor Network Localization Based on Environmental Events

YoungMin Kwon
Microsoft Corporation
ykwon4@cs.uiuc.edu

Gul Agha
Department of Computer Science
University of Illinois at Urbana Champaign
agha@cs.uiuc.edu *

Abstract

We develop a localization algorithm based on global environmental events observed by a sensor network. Examples of such events include the sound of thunder, the shades of moving clouds, or the vibrations in seismic data. Because our localization method does not generate signals for distance measurements, it saves energy. In fact, the algorithm may use existing sensor recordings to determine the locations of nodes at which the recordings were taken. Moreover, the method does not accumulate errors, making it also effective for large and sparse sensor networks. The localization uses time synchronization; we provide an algorithm to compensate for clock synchronization errors. Versions for both two dimensional and three dimensional localization of the algorithm are presented. Simulation results suggest that the algorithm can provide a high degree of accuracy when many events are recorded.

1 Introduction

A *Wireless Sensor Network* (WSN) is a collection of sensor nodes cooperating with others through wireless communication channels. Because each node can process data locally and can collaborate with other nodes, WSNs have many successful applications such as structural health monitoring, environmental monitoring, target tracking, or shooter localization [8, 16, 9, 7]. Because many of those applications require sensor position information, *localization* has been an important problem in WSNs and a number of approaches have been proposed.

The most intuitive localization approaches are based on multilateration by measuring distances from multiple anchor nodes [10]. One popular distance measurement method is to use use the *Time Difference of Arrival* (TDOA) between two different signals. Much as we measure the distance to a lightning cloud by measuring the TDOA between the light and the thunder, the TDOA between the radio and the sound can be used to measure distances between sensor nodes [4, 12]. Although such anchor based localization methods are accurate and less vulnerable to error propagation, their drawback is that they require a high density of anchor nodes.

Without anchor nodes, we can estimate relative coordinate systems that may be rotated, translated, or flipped with respect to a physical coordinate system but still preserves the distances between nodes. If all pairs of node-to-node distances have been estimated estimated, one can use *Multi-Dimensional Scaling* (MDS) to localize [13]. In [13], the pairs of node-to-node distances are approximated by message hop counts between two nodes. However, there is often non-uniformity in radio ranges of sensor nodes. Moreover, there may be skewness in hop counts of distances in regular deployments; for example, in a grid layout, nodes in a diagonal direction have a shorter hop distance than those measured in a lateral direction. To mitigate these problems, incremental localization approaches have been proposed [14, 3]. In these incremental approaches, local MDS is applied to the nodes within certain hop count ranges and the locally localized patches are incrementally stitched together. Unfortunately, errors can accumulate during the incremental stitching process.

Another multi-dimensional scaling type of localization approach that does not require all pairs of distances and does not accumulate errors is proposed in [4]. In [4] *Least Squares Scaling* (LSS) based localization method is used to minimize the sum of squares of differences between the available distance measures and the distances calculated from their estimated positions.

Many localization algorithms are based on node-to-node distances from actively generated signals. However, because sensor nodes usually have low power actuators (for

*The authors thank Eunhee Kim and MyungJoo Ham for their helpful comments on drafts of the paper. This research has been supported in part by NSF under grants CNS 05-09321 and CMS 06-00433 and by ONR under DoD MURI award N0014-02-1-0715.

example, the beeper of Mica motes has a range of a few meters [2]), it is difficult to measure long range distances. Moreover, in energy constrained sensor network applications, generating large signals can be prohibitively expensive. With only short range distance measurements, localization algorithms can suffer from error accumulation: localization results in local neighboring nodes may be relatively accurate but they are significantly misplaced in the global configuration. Another problem is the inability to deal with a sparse measurement density. In order to uniquely determine the position of a node in 2D space, the node needs at least three independent low error measurements. Sparse distance measurements may mean some nodes fail to meet this requirement. This problem can be more severe for the nodes at the edge of a configuration.

In this paper, we propose a new localization algorithm called *Passive Localization* which addresses these problems. The proposed algorithm uses time differences of global environmental events observed by a sensor network, such as the sound of thunder, the shadows of moving cloud, or the vibrations of seismic data. Because the recording time differences are proportional to the distances from the positions of the nodes to a plane perpendicular to the event propagation direction, we call these time differences *Projected Distances*.

Consider a simple example of a localization from projected distances. Suppose we have two sets of projected distances from two global events propagating in perpendicular direction at the same speed. We can then obtain a relative coordinate system by simply pairing the two distances of each node. However, in general, we cannot get the event propagation directions from initially unlocalized sensor networks. Instead, we compute the principal axes that have the largest variances from the set of projected distances: in this paper, we show that the positions of nodes can be obtained from linear combinations of the principal axes of which coefficients can be found using anchor nodes. Some of the merits of the proposed method are as follows:

- Because the projected distances can be measured without actively generating signals, passive localization is good for the longevity of sensor networks.
- The environmental events span across the entire sensor network. Thus, the error accumulation problem in localization can be eliminated.
- The problems caused by sparse measurement density can be solved. Even the nodes at the edge generally have as many distance measurements as the nodes in the center of the network.
- We do not assume that clock synchronization is maintained but provide an algorithm to mitigate time synchronization errors.

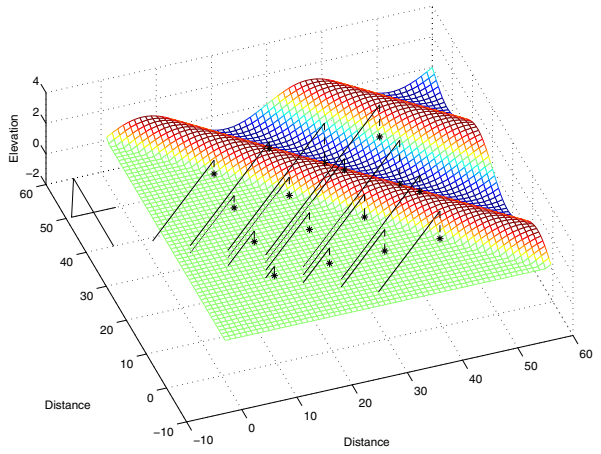

Figure 1. An example of a global event propagating over a WSN.

- The proposed method can be applied to existing recordings so that the positions of nodes can be found post-hoc.

We present simulation results which show that our passive localization algorithm can succeed in localizing nodes to a median error of $<6\%$. One weakness of passive localization appears to be the need for dozens of events to provide sufficient accuracy.

2 Distance Measurement

Passive localization is based on projected distances which can be obtained from recordings of possibly multiple types of sensors. In this section we describe how to detect the presence of global events and how to get projected distances from the event recording.

2.1 Detecting Global Events

In order to get projected distances, we need to detect the presence of global events from the raw recordings first. We detect the events by checking the covariance of recordings of all nodes in a sliding time window, instead of looking for any previously assumed patterns for the events. In this section we explain the exponential complexity of the covariance comparison and propose a frequency domain solution. For the sliding time window, we propose an incremental *spectrogram* algorithm which has linear time complexity for computing *Discrete Fourier Transform* (DFT) in terms of the window size.

We begin this section with an example of a global event. Figure 1 shows a seismic vibration of a land surface. This vibration passes through a sensor network, marked as stars,

from the north-east side of the figure. As the vibration passes the system, all the nodes will record the same sequence of vertical accelerations of the land. However, the sequence begins at different times at each node: the node at the north-east corner has the first recording and the node at the south-west corner has the last recording. The recording time differences are proportional to the projected distances from the nodes to a plane perpendicular to the event propagation direction. In Figure 1, we draw these distances with solid lines. Note that, since we do not know the speed of event propagation, we do not know the actual projected distances. We now explain how to detect the global presence of such recording sequences on the system, and in the next section we explain how to obtain the projected distances from the recordings.

We detect global events by checking the similarity of recordings of all nodes. If there are similar sequences present in all recordings then we assume that it is made from a global event. Checking the similarity between recordings can be done by measuring the correlation between them. Because the sequences are time shifted by event propagation delays, we need to find the time shift between them that will make the maximum correlation. However, because we are interested in finding projected distances of all nodes, the time shift between two nodes cannot be determined independently of other nodes. For example, let t_{ab}, t_{bc}, and t_{ac} be the time shift amounts between node pairs (a, b), (b, c), and (c, a). Then, the constraint $t_{ab} + t_{bc} = t_{ac}$ must be satisfied. In ideal case, this constraint should be satisfied trivially. However, in a noisy environment, we need to find the best set of time shifts that satisfy the constraint. Note that the whole recording of a node may have recordings of multiple events coming from different directions at different times. We can set a *time window* and limit the comparison within the window to isolate the events. We slide the window through the entire recording. In summary, what we want to find is the amount of time shift t_i for each node n_i that maximizes:

$$\mathrm{cov_t} = \sum_{r_i \in N} \sum_{r_j \in N \wedge i \neq j} \sum_{\tau=0}^{T-1} \tilde{r}_{i,t+t_i}(\tau) \cdot \tilde{r}_{j,t+t_j}(\tau),$$

where t is the beginning of the time window, for all k, $t_k \in [0, T)$ is the time shift for node n_k, $r_k : \mathbb{N} \rightarrow \mathbb{R}$ is the array of samples of node n_k, N is a set of recordings from all nodes, T is the size of the time window, and $\tilde{r}_k : \mathbb{N} \rightarrow \mathbb{R}$ is the mean deviated form of the sample: $\tilde{r}_{k,t}(\tau) = r_k(t + \tau) - (\sum_{s=0}^{T-1} r_k(t+s))/T$. We can say that there is a global event if *cov* is larger than a threshold value and the time shift information t_i's to get the maximum *cov* are the projected distances. Note that to get the maximum *cov* we need to check all different configurations of time shifts. Thus the worst case time complexity is $O(T^n)$ which we may want to avoid for large n or T. We show that the covariance can in fact be estimated in the frequency domain (without shifting the signals). Estimation of the time shift is discussed in the next section.

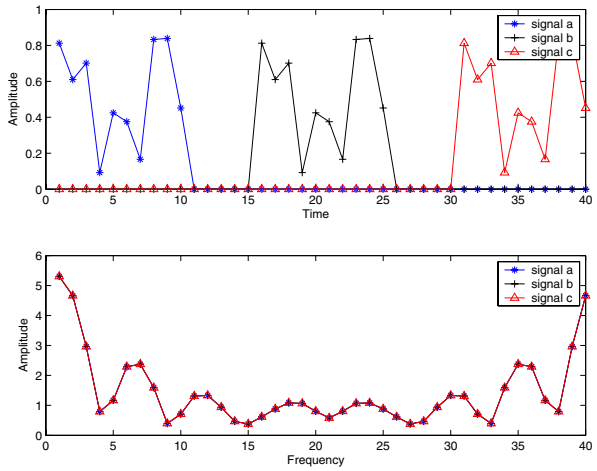

Figure 2. Time shift invariance of the amplitude of DFT of a signal.

It is well known that the amplitude spectrum of *Discrete Fourier Transform* (DFT) of a time domain signal is invariant to time shift [11]. As an example, Figure 2 demonstrates the invariance to time shift of the amplitude spectrum of DFT of the signal. The first graph shows a time domain signal and its two different time shift variations. The second graph shows their amplitude spectrum after DFT. Note that all the three amplitude spectrum agree regardless of their time shift. Because DFT eliminates the time shift factor from the recordings, we do not need to do the exponentially complex search. Instead, we compute the covariance directly by:

$$\mathrm{cov'_t} = \sum_{r_i \in N} \sum_{r_j \in N \wedge i \neq j} \sum_{f=1}^{T-1} |\tilde{R}_{i,t}(f)| \cdot |\tilde{R}_{j,t}(f)|,$$

where $|\tilde{R}_{k,t}|$ is the mean deviated form of $|R_{k,t}|$ in the window and $R_{k,t} : \mathbb{N} \rightarrow \mathbb{C}$ is DFT of r:

$$R_{k,t}(f) = \sum_{\tau=0}^{T-1} r_k(t + \tau) \cdot e^{\frac{2\pi i \tau f}{T}}, \qquad (1)$$

where $i = \sqrt{-1}$, $e^{xi} = cos(x) + i \cdot sin(x)$, $f \in [0, T)$, and t is the beginning of time window. The whole set of $R_{k,t}$ of a node n_k from each t is called *spectrogram* [6]. For simplicity, we drop the node id k or the time index t from $R_{k,t}$ when they are obvious or not necessary.

Note that *cov'* is not equal to *cov*: in computing *cov* we compare signals of length T that begin from different time

```
double Q[T], S[T], C[T];
int h=0;
void update(double d) {
   for(f=0; f<T; f++) {
      C[f] = C[f] - Q[h]*cos(2π·f·h/T);
      S[f] = S[f] - Q[h]*sin(2π·f·h/T);
      C[f] = C[f] + d*cos(2π·f·h/T);
      S[f] = S[f] + d*sin(2π·f·h/T);
   }
   Q[h] = d;
   h = (h + 1) modulo T;
}
double amplitude(int f) {
   return √(C[f]^2 + S[f]^2);
}
double angle(int f) {
   return atan2(S[f], C[f]) - 2π·h/T;
}
```

Figure 3. Incremental spectrogram algorithm: *update* function slides the DFT window a step with a new sample data d. $atan2(y,x)$ returns an angle θ such that $cos(\theta) = x/\sqrt{x^2+y^2}$ and $sin(\theta) = y/\sqrt{x^2+y^2}$.

steps, whereas in computing *cov'* we compare the signals of length *T'* begin from the same time step. In other words, for *cov*, we compare signals of length *T* within an actual window of length *2T*, and for *cov'* we compare signals of length *T* in the same window.

Note also that in *cov'* the time shift terms t_i and t_j of *cov* are eliminated and so does the need for the exponential search. Instead, we need to do DFT of Equation (1) which has $O(T^2)$ time complexity – $O(T \cdot \log(T))$ if we do *Fast Fourier Transform*. We developed an incremental spectrogram algorithm that reduces the time complexity to $O(T)$ to compute DFT for the adjacent window. Although it is a relatively straight-forward algorithm, to the best of our effort, it has not been previously published. Note that R_t of Equation (1) is for the window $[t, t+T-1]$, and the window for the next step is $[t+1, t+T]$. Let

$$R'_{t+1}(f) \quad = \quad R_t(f) - r(t) \cdot e^{\frac{2\pi i f \cdot 0}{T}} + r(t+T) \cdot e^{\frac{2\pi i f \cdot T}{T}}.$$

Then, R'_{t+1} is computed over the next time window on a phase shifted basis vectors. Using the addition formulas of *sin* and *cos*, we can easily prove that the phase shifted basis vectors do not have any effect on the magnitude: $|R'_{t+1}| = |R_{t+1}|$, and they shift the phase by $\frac{2\pi}{T}$: $\angle R'_{t+1} - \frac{2\pi}{T} = \angle R_{t+1}$, where $\angle a \cdot e^{i \cdot b}$ is b. In this paper, we also use the *phasor notation* $a\angle b$ for $a \cdot e^{i \cdot b}$. Figure 3 shows the incremental spectrogram algorithm. We maintain a circular queue of length *T* to remove the first element of the previous window from the computation for the current window. Also, the index *h* maintains the header position in the queue as well as the phase shift information of the basis vectors.

2.2 Computing Projected Distances

Once a global event is detected, the next step is to get the projected distances. In the previous section, we explained that the magnitude spectrum of DFT is invariant to time shift. The time shift information is encoded as phase shift after DFT. In this section we describe a method to get the time shift information from the phase shift and confidence value of the estimation. We also describe a measurement quality improvement method based on the confidence value and by the principal axis analysis.

Converting phase shift to time shift

Let $R_i(f) = a_i \angle \theta_i$ and $R_j(f) = a_j \angle \theta_j$, then the following equality holds between phase shift and time shift:

$$\frac{\theta_i - \theta_j}{2\pi} \quad = \quad \frac{d_{ij} \bmod (T/f)}{T/f}, \tag{2}$$

where d_{ij} is the time shift between the recordings of node n_i and node n_j, and *mod* is the remainder operator. This equality holds because a time domain signal can be represented as a linear combination of orthogonal basis of sinusoidal and time shift of the signal is equivalent to phase shift of the basis sinusoidal. Because Equation (2) is about the remainder of time shift, there can be multiple candidates for d_{ij}:

$$d_{ij}(k) \quad = \quad k \cdot \frac{T}{f} + \frac{\theta_i - \theta_j}{2\pi} \cdot \frac{T}{f} \tag{3}$$

for any integer $k \in [-\left\lceil \frac{T}{f} \right\rceil, \left\lceil \frac{T}{f} \right\rceil]$. However, if we plot the candidates of all frequencies together, the amount of real shift will be evident.

Figure 4 shows the time shift candidates of all frequencies for the signals in Figure 2. The first graph is about the time shift between signal *a* and signal *b*. Note that, the candidates of all frequencies are present at step 15. The second graph is the candidates for signal *a* and signal *c*. As expected, candidates of all frequencies are present at step 30. The last graph is about signal *b* and signal *a*. However, in this graph, the dense marks are at step 25, instead of at step 15. We will explain this later in this section.

Figure 4 is an ideal case phase shift diagram: the candidates are aligned in a single column. In real measurements, because of measurement errors, the candidates will be scattered around their actual position. Because such scattering need not be Gaussian, we choose the distance that minimizes the following *median error* function:

$$E_{ij}(d) \quad = \quad \sum_{f=1}^{T-1} \min_{k \in [-\lceil \frac{T}{f} \rceil, \lceil \frac{T}{f} \rceil]} w_{ij}(f) \cdot |d_{ij}(k) - d|, \tag{4}$$

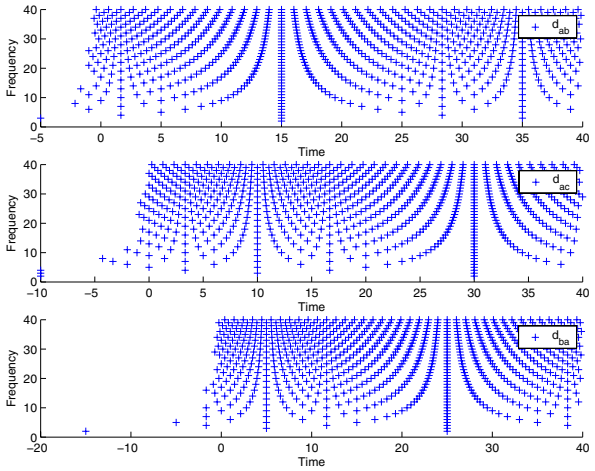

Figure 4. Time shift candidates from phase shift information.

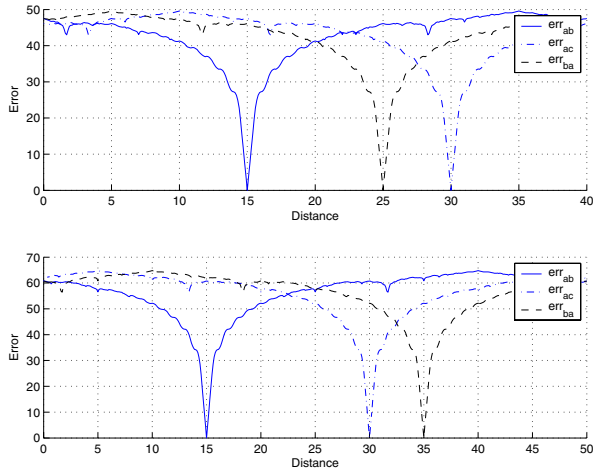

Figure 5. The error E_{ij} as a function of estimated distances d_{ij}.

where $w_{ij} : \mathbb{N} \rightarrow \mathbb{R}$ is a weight function for the error at frequency f. We use the geometric mean of the magnitudes of two signals at that frequency as the weight $(w_{ij}(f) = \sqrt{|R_i(f)| \cdot |R_j(f)|})$ so that the phase differences of the frequencies with large energy have large weights.

The last factor to consider in converting phase shift to time shift is that an angle θ can also be $-(2\pi - \theta)$. This explains why the time shift candidate column of the last graph of Figure 4 is at 25: $25 - 40 = -15$. In order to resolve this ambiguity we introduce a second time window of different size, say T_2. Let d^1_{ij} and d^2_{ij} are the measured distances from the two time windows respectively and let $\delta^f_{ij} = |d^1_{ij} - d^2_{ij}|$ and $\delta^r_{ij} = |(T - d^1_{ij}) - (T_2 - d^2_{ij})|$. Then, the distance between nodes n_i and n_j is

$$d_{ij} \quad = \quad \begin{cases} d^1_{ij} & \text{if } \delta^f_{ij} \leq \theta \wedge \delta^f_{ij} \leq \delta^r_{ij} \\ d^1_{ij} - T & \text{if } \delta^r_{ij} \leq \theta \wedge \delta^f_{ij} > \delta^r_{ij} \\ 0 & \text{otherwise} \end{cases}$$

where θ is a threshold value that prevents large error.

Figure 5 shows the error function E_{ij} of Equation (4) as a function of d_{ij}. For this graph, we use a constant weight function to show the median error. The first graph is plotted with a time window of size 40 and the second graph is plotted with a time window of size 50. Note that in both graphs the minimum of err_{ab} and err_{ac} agrees at 15 and 30 respectively, and the minimum of err_{ba} agrees at 25 and 35 $(25 - 40 = 35 - 50 = -15)$.

Measurement quality improvement

Because projected distances hold the following equality,

$$d_{ij} \quad = \quad d_{ik} + d_{kj} \quad \text{for all } k$$

a poor quality measurement can be replaced with a combination of better quality measurements.

Note that we estimate two distances from two time windows of different size. Thus, the degree of their agreement can be an indicator of a *confidence of the measurement*. Let δ_{ij} be the confidence of measurement (the smaller the better),

$$\delta_{ij} \quad = \quad \min(\delta^f_{ij}, \delta^r_{ij}).$$

Then, we can iteratively improve the quality of measurements by replacing the low confidence measurements with better ones:

- while there are nodes n_i, n_j, and n_k such that $\delta_{ij} > \max(\delta_{ik}, \delta_{jk}) + \epsilon$ update δ_{ij} and d_{ij}:

$$\begin{aligned} \delta_{ij} &= \max(\delta_{ik}, \delta_{jk}) + \epsilon \\ d_{ij} &= d_{ik} - d_{jk}, \end{aligned}$$

  where $\epsilon \geq 0$ is a parameter to prevent the set of entire measurements from being dominated by only few measurements.

Let $\Delta \in \mathbb{R}^{n \times n}$ be a matrix such that $\Delta_{ij} = d_{ij}$, then in ideal measurement Δ can be written as:

$$\Delta \quad = \quad \begin{bmatrix} d_1 & 1 \\ \vdots & \vdots \\ d_n & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & \cdots & -1 \\ d_1 & \cdots & d_n \end{bmatrix},$$

where d_i is the projected distance of node n_i to a reference plane. Because the rank of Δ is two, the ratio, σ_3/σ_1, between the first and the third singular values of Δ is an indicator of the overall quality of the measurement. We discard the whole set of measurement if this ratio is larger than a threshold value. Also, we take the mean deviated form of the first left eigenvector of Δ as our final distance measurement as it best represents the distances.

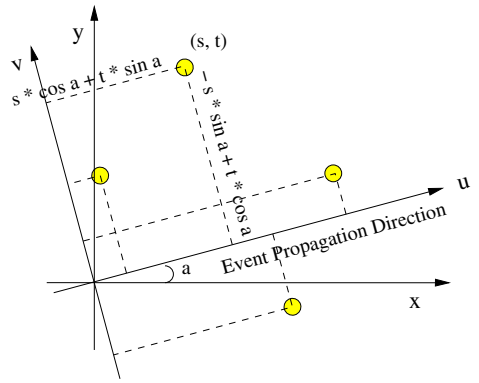

Figure 6. Node positions on the axes of event propagation direction.

3 Passive Localization

Given a sufficient number of sets of projected distances, we can compute the coordinates of the nodes. Because projected distances are different form node to node distances, we need to formulate a new way of computing node positions from projected distances. As will be explained in this section, in the ideal two (or three) dimensional localization, there are only two (resp. three) independent sets of projected distances. Thus, in noisy measurements, we need to restore the most significant two (resp. three) sets of distances. Recall the previous perpendicular event propagation example, because there are only two independent sets of measurements the two sets of projected distances can be expressed as linear combinations of the two independent measurements. We can find the linear combination coefficients using anchor nodes.

In this section we explain two dimensional and three dimensional localization algorithms as well as a compensation method for time synchronization error which will distort the projected distances.

3.1 Two Dimensional Localization

Suppose that we have *n* nodes and *m* sets of projected distances, then from the *m* column vectors of length *n* we can build a *distance matrix*:

$$D \quad = \quad [w_1 \cdot \tilde{\mathbf{u}}_1, \ldots, w_m \cdot \tilde{\mathbf{u}}_m] \,,$$

where w_i is the weight for the measurement based on its quality, for example the ratio σ_1/σ_3, and $\tilde{\mathbf{u}}_i$ is the normalized mean deviation form of $\mathbf{u}_i$. In the ideal case, the distance matrix *D* can be decomposed into a matrix of node positions and a matrix corresponding to the event propagations. In order to illustrate this, let us explain Figure 6. In Figure 6 sensor nodes are marked as yellow circles and an event propagates with angle *a* from *x* axis. If we convert the node positions in $x - y$ system into positions in $u - v$ system, where u is the event propagation direction, then their u coordinates divided by the event propagation speed, say v_e, are the time shifts or the projected distances. In this figure, a node position (s, t) is converted to $(s\cos a + t\sin a, -s\sin a + t\cos a)$. Thus, the node's projected distance with respect to this event is $\frac{1}{v_e}(s\cos a + t\sin a)$. A column of the distance matrix *D* is a set of projected distances computed in this way.

Let $D \in \mathbb{R}^{n\times m}$ be a distance matrix, let (x_i, y_i) be the position of a node n_i, let v_j be the propagation speed of the j^{th} event, and let θ_j be its propagation angle. Then, the distance matrix can be decomposed into the node positions, and a matrix of the event propagation angles and speeds, as follows:

$$D \quad = \quad \begin{bmatrix} \tilde{x}_1 & \tilde{y}_1 \\ \vdots & \vdots \\ \tilde{x}_n & \tilde{y}_n \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{v_1}\cdot\cos(\theta_1) & \ldots & \frac{1}{v_m}\cdot\cos(\theta_m) \\ \frac{1}{v_1}\cdot\sin(\theta_1) & \ldots & \frac{1}{v_m}\cdot\sin(\theta_m) \end{bmatrix},$$

where $\tilde{x}_i$ and $\tilde{y}_i$ are the i^{th} elements of the mean deviated form of position vectors $[x_1, \ldots, x_n]^T$ and $[y_1, \ldots, y_n]^T$. Note that in ideal case the rank of *D* is at most two no matter how many measurements we made. Note also that because we are using the mean deviated form for node positions, the projected distances are in the mean deviated form. That is, the sum of all elements of each column is zero. Hence, if we take the mean deviated form of measurement, then we get the mean deviated form of a coordinate system whose origin is at the center of mass of the node positions. Let $\tilde{\mathbf{x}} = [\tilde{x}_1, \ldots, \tilde{x}_n]^T$ and $\tilde{\mathbf{y}} = [\tilde{y}_1, \ldots, \tilde{y}_n]^T$ be the vectors of mean deviated *x* and *y* positions, then the two vectors span the column space of *D*. Hence, $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ can be written as linear combinations of any two independent vectors in the column space of *D*.

In real measurement, where error is present, the rank of *D* is larger than two. Thus, we need to take the two independent vectors that best describe the projected distances. We choose the two vectors from principal axis analysis on *D*: the two orthogonal vectors in the column space of *D* that will make the two largest variances if the columns of *D* are projected to them. Once we have the two orthogonal vectors, we can compute the linear combination coefficients using two anchor nodes in the mean deviated coordinate system. However, because, in general, we do not know the center of mass of the positions of the nodes, we need another column vector of ones in the linear combinations and a third anchor node.

In summary, in order to get the positions of nodes, we

first find the SVD of the distance matrix *D*:

$$D = P \cdot \Sigma \cdot Q.$$

Let $\mathbf{p}_1$ and $\mathbf{p}_2$ be the first two column vectors of *P*, then, they are the two most important principal axes of *D* we are looking for. Now suppose that node n_i is an anchor node and its real position (x_i, y_i) is given, then:

$$\begin{aligned} x_i &= a_1 \cdot \mathbf{p}_{1i} + b_1 \cdot \mathbf{p}_{2i} + c_1 \\ y_i &= a_2 \cdot \mathbf{p}_{1i} + b_2 \cdot \mathbf{p}_{2i} + c_2, \end{aligned} \tag{5}$$

where $\mathbf{p}_{ij}$ is the j^{th} element of $\mathbf{p}_i$, a_1, a_2, b_1, and b_2 are the linear combination coefficients and c_1 and c_2 are the amount of translation or the coefficient for the vector of ones. Because there are 6 variables and each anchor position gives two equations, we can localize the whole system if three anchor positions are given.

If more than three anchor positions are available then we can leverage the redundancy in computing the linear combination coefficients: we choose the coefficients such that they minimize the sum of squared distances between real anchor positions and the computed anchor positions as a function of the coefficients. That is, the error functions to minimize are:

$$\begin{aligned} \mathrm{err}_x &= \sum_{i=1}^{k} (a_1 \cdot \mathbf{p}_{1i} + b_1 \cdot \mathbf{p}_{2i} + c_1 - x_i)^2 \\ &= (A \cdot \mathbf{s} - \mathbf{x})^T \cdot (A \cdot \mathbf{s} - \mathbf{x}), \\ \mathrm{err}_y &= \sum_{i=1}^{k} (a_2 \cdot \mathbf{p}_{1i} + b_2 \cdot \mathbf{p}_{2i} + c_2 - y_i)^2 \\ &= (A \cdot \mathbf{t} - \mathbf{y})^T \cdot (A \cdot \mathbf{t} - \mathbf{y}), \end{aligned}$$

where, *k* is the number of anchor nodes [1], and

$$A = \begin{bmatrix} \mathbf{p}_{11} & \mathbf{p}_{21} & 1 \\ \vdots & \vdots & \vdots \\ \mathbf{p}_{1k} & \mathbf{p}_{2k} & 1 \end{bmatrix}, \tag{6}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix}, \mathbf{s} = \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix}, \mathbf{t} = \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix}.$$

The $\mathbf{s}$ and $\mathbf{t}$ that minimizes the error functions are

$$\mathbf{s} = A^+ \cdot \mathbf{x}, \qquad \mathbf{t} = A^+ \cdot \mathbf{y},$$

where A^+ is the pseudo inverse of *A* [15, 5]. Once we compute the coefficients, we can get the positions of a node n_i by applying Equation (5) to the i^{th} elements of $\mathbf{p}_1$ and $\mathbf{p}_2$.

[1]Without loss of generality we assume that nodes n_1 to n_k are the anchor nodes.

3.2 Compensating Time Sync. Error

The projected distances of passive localization depend mainly on the recording time differences between nodes for the same global events. Thus, errors in time synchronization can be critical in the localization result. However, in a multi-hop network, like WSNs, maintaining accurate time synchronization for a long duration requires considerable energy consumption. Furthermore, because passive localization does not require communications between nodes [2] node spacings can be larger than their radio range, making the usual time synchronization techniques infeasible. Instead of requiring difficult to achieve time synchronization, we propose a compensation method to mitigate errors in time synchronization.

First, we define a *clock model* used in this section [17]. Let $t_i : \mathbb{R} \rightarrow \mathbb{R}$ be the function that maps the imaginary global time to the local time of node n_i. Then, the clock model we assume in this section is:

- Each node may have different initial time when recording is started: $t_i(g_0) \neq t_j(g_0) \rightarrow i \neq j$, where g_0 is the initial global time.
- Clock drift rate is constant: for any global times g_a, g_b, and g_c,

$$\frac{t_i(g_a) - t_i(g_b)}{g_a - g_b} = \frac{t_i(g_a) - t_i(g_c)}{g_a - g_c}.$$

In summary, our clock model allows initial clock differences and constant clock drifts between nodes. Thus, the local clock function can be rewritten as $t_i(g) = \sigma_i + \delta_i \cdot (g - g_0)$, where σ_i corresponds to initial time difference between nodes and δ_i is the clock drift speed of node n_i from the global time. With this clock model we can choose the imaginary global time *g* such that

$$\sum_i \sigma_i = 0 \quad \text{and} \quad \sum_i \delta_i = 0. \tag{7}$$

That is, the global clock speed is the average clock speed of the system and the initial global time is the average initial local time of the system.

With the clock error, the distance matrix *D* can be rewritten as:

$$D = \begin{bmatrix} \tilde{x}_1 & \tilde{y}_1 & \delta_1 & \sigma_1 \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{x}_n & \tilde{y}_n & \delta_n & \sigma_n \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{v_1} \cdot \cos(\theta_1) & \cdots & \frac{1}{v_m} \cdot \cos(\theta_m) \\ \frac{1}{v_1} \cdot \sin(\theta_1) & \cdots & \frac{1}{v_m} \cdot \sin(\theta_m) \\ g_1 - g_0 & \cdots & g_m - g_0 \\ 1 & \cdots & 1 \end{bmatrix}.$$

[2]Requirements for the lack of communications and a localization based on correlations among recordings of local events are discussed in [1].

Note that *D* is in the mean deviated form because of Equation (7). In other words, mean deviated form of *D* will have the global time of Equation (7). In ideal case, because *D* has only four independent columns, there are four nonzero singular values in Σ and the four vectors $\mathbf{x}$, $\mathbf{y}$, σ, and δ are in the space spanned by the first four columns of *P*. Thus, by finding the linear combination coefficients for the first four columns of *P* and the column vector of ones, we can localize the whole system. That is, we need at least five anchor nodes to compensate for the clock error in two dimensional localization.

3.3 Three Dimensional Localization.

Three dimensional localization is a simple extension of the two dimensional localization of Section 3.1. We simply reinterpret the distance matrix *D* and find an extended set of linear combination coefficients corresponding to the additional dimension.

In three dimensional localization we need to consider the event propagation direction in three dimensional space. In two dimensional localization, the *z* component of the propagation direction is encoded in the event propagation speed that affects all the nodes equally. However, in three dimensional localization we cannot encode the *z* coordinate of event propagation vector into its propagation speed because it affects the event recording time differently for the nodes at different elevation. The event propagation vector can be written as $v \cdot [\cos(\theta) \cdot \cos(\phi), \cos(\theta) \cdot \sin(\phi), \sin(\theta)]$, where *v* is the event propagation speed, ϕ is the vector's angle in *x*-*y* plane, and θ is the minimum angle from the *x*-*y* plane to the vector. Then, in ideal case, the distance matrix *D* can be written as:

$$D = \begin{bmatrix} \tilde{x}_1 & \tilde{y}_1 & \tilde{z}_1 \\ \vdots & \vdots & \vdots \\ \tilde{x}_n & \tilde{y}_n & \tilde{z}_n \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{v_1} \cdot \cos(\theta_1) \cdot \cos(\phi_1) & \dots & \frac{1}{v_m} \cdot \cos(\theta_m) \cdot \cos(\phi_m) \\ \frac{1}{v_1} \cdot \cos(\theta_1) \cdot \sin(\phi_1) & \dots & \frac{1}{v_m} \cdot \cos(\theta_m) \cdot \sin(\phi_m) \\ \frac{1}{v_1} \cdot \sin(\theta_1) & \dots & \frac{1}{v_m} \cdot \sin(\theta_m) \end{bmatrix}.$$

We have three independent columns in *D*. Thus we need at least four anchor nodes to localize nodes in a three dimensional field. The compensation algorithm for time synchronization error of Section 3.2 can be applied to the three dimensional localization after an obvious extension with the additional two dimensions.

4 Simulation Study

We now describe the results of a discrete event simulation study to evaluate the performance of our localization algorithm. We experiment with both two-dimensional and three-dimensional localization, and study the effectiveness of our proposed time synchronization error compensation method. We first explain the simulation settings and then evaluate each steps of the localization algorithm.

4.1 Simulation Settings

Assume a node *n* is a structure that has its position vector $n.\mathbf{p} \in \mathbb{R}^3$ and an array $n.\mathbf{r} : \mathbb{N} \rightarrow \mathbb{R}$ for signal recording. We assume that all the nodes are time synchronized, except for an experiment to examine the effect of time synchronization error. On each sample, we disrupt the reading by a uniform random variable N_{Sensor} representing the sensor noise.

An event *e* is also a structure that has a position vector $e.\mathbf{p} \in \mathbb{R}^3$ representing its origin, a scalar value $e.v \in \mathbb{R}$ for its propagation speed, a finite array $e.\mathbf{s} : [0, L-1] \rightarrow \mathbb{R}$ for the signal it is carrying, and another finite array $e.\mathbf{d} : [0, L] \rightarrow \mathbb{R}$ representing the propagation distance of each piece of signal, where *L* is a randomly chosen length of the signal. The event propagation distance is initialized as $e.\mathbf{d}(t) = -v \cdot t$ for $t = 0, \dots, L$ and on each step of the simulation the array is updated as $e.\mathbf{d}(t) = e.\mathbf{d}(t) + v$ for $t = 0, \dots, L$. We assume that the event is propagating spherically. Thus, its signal strength is inversely proportional to the square of the propagation distance.

Let *E* be a set of active events at time *t*, the events whose signal did not completely pass through all the sensor nodes yet, then the signal recording of node *n* at time *t* is

$$n.\mathbf{r}(t) = \sum_{e \in E} \sum_{\tau=0}^{e.L-1} \frac{s_e(\tau)}{|n.\mathbf{p} - e.\mathbf{p}|^2},$$

where

$$s_e(\tau) = \begin{cases} e.\mathbf{s}(\tau) & \text{if } e.\mathbf{d}(\tau+1) \leq |n.\mathbf{p} - e.\mathbf{p}| < e.\mathbf{d}(\tau) \\ 0 & \text{otherwise} \end{cases}$$

On each step of the simulation we generate an event with a uniform probability of ρ = 0.01. Because we are generating events uniformly independent of their past, there can be more than one events present at the same time.

Figure 7 shows the configuration of 25 nodes deployed on a plane for this simulation study. We place the nodes on square grid blocks. The cross marks of Figure 7 are the node positions. We also put node IDs for only the nodes at the corner of the deployment. The node IDs for the other nodes can be easily inferred: node 1 to node 5 are deployed in the first row, node 6 to node 10 are deployed in the next row, and so on. Figure 7 also shows the origins of events generated within the first 2,000 steps of the simulation. In order to show the height of event origin, we draw a line from the event origin, marked as circle, to its projected position

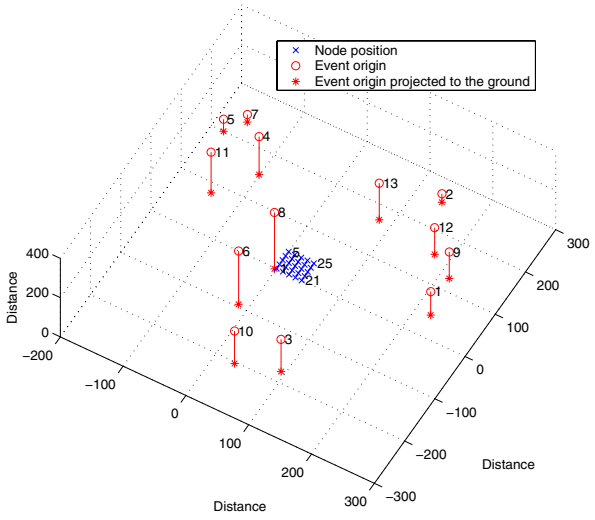

Figure 7. Node deployment and event origins during the first 2,000 steps of simulation.

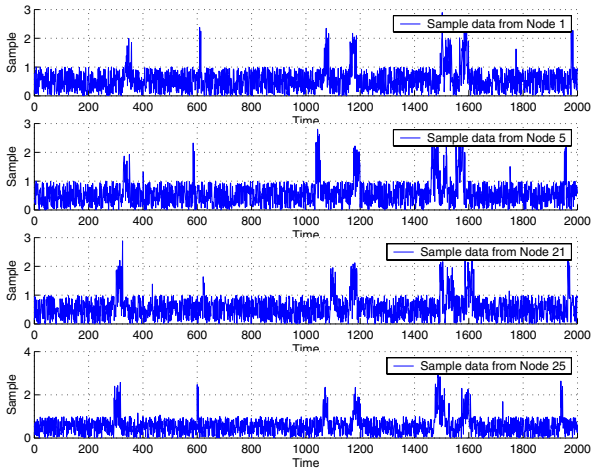

Figure 8. Raw sample data of nodes at the corner of deployment.

on the ground, marked as star. We use five anchor nodes through out this section. They are nodes 1, 5, 13, 21, and 25: the nodes at the center and the corners of the deployment.

Figure 8 shows the raw sample data of the four nodes at the corner of deployment during the first 2,000 steps. Because we are interested in the recording time difference of global events, and because the largest time difference occurs between a pair of nodes at the corner, Figure 8 shows the most distinctive situation. Figure 7 shows 13 events, but Figure 8 shows only 8 or 9 explicit ones: some of them are too weak to become obvious, and some of them overlap with others.

In Figure 8 the spikes near step 600 are from *event* 4. By checking the origin of the *event* 4 from the Figure 7, we can imagine that the event will be first detected by *node* 5, then by *node* 1 and *node* 25, and lastly by *node* 21, as can be seen in Figure 8. We will see the recordings resulting from other events in different orders.

4.2 Detection of Global Events

In this experiment, we use the time windows of size T_1 = 70 and T_2 = 100. In the simulation configuration, the longest time for an event to go through the system is 57 steps: namely, when the signal for the event passes through the diagonal of the square block that is parallel to the ground. Thus, there are at least 13 samples of overlapping recordings at the first and the last receivers of the event, although the number of overlapping recordings will be larger in practice. Note that enlarging the time window size increases the sample overlap, and thus increases the correlations for a long event; however, it also reduces the correlation for a short event in a noisy environment. Also, large overlaps have adverse effects when multiple events are present in a short interval: multiple events coming from different directions in a time window reduce the correlation.

Figure 9 shows the time when events enter and leave the system of nodes (the first graph) during the first 2,000 steps of simulation, and the correlation of the amplitude spectrum as the time window slides over the sample recording (the second graph). The first graph shows many overlapping events near step 1,500 and different duration of events: for example, event 8 has very short duration because from Figure 7, its event propagation direction is almost perpendicular to the ground.

From the second graph we can see large correlations where events present. Our global event detection algorithm starts searching for the maximum correlation once the correlation becomes larger than a threshold value, θ_1, until the correlation becomes smaller than another threshold value, θ_2. We set θ_1 larger than θ_2 in order to avoid local minima due to the jitter in correlation. In this simulation, given the jitter levels, we use θ_1=3.5 and θ_2=3.0.

4.3 Distance Measurement

During the whole 20,000 steps of simulation we generated 194 events, we detected 74 events, and out of the 74 events, we got 55 sets of projected distances. Figure 10 shows the results of measurements with and without the quality improvement. For comparison, we normalize the length of the projected distances. In this graph, the horizontal axis is the real distance and the vertical axis is the estimated distance. The solid line in this graph shows the ideal measurement. The circles in the graph are the distances ob-

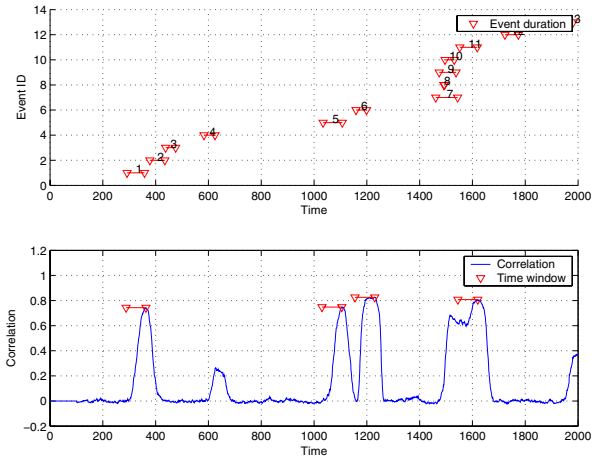

Figure 9. The duration of events (top) and the correlation among samples of nodes (bottom). In the second graph, the extent of time window at the maximum correlation is marked by two triangles connected by a solid line.

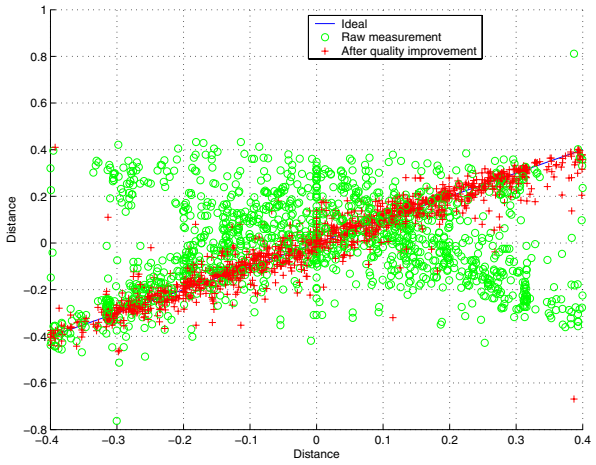

Figure 10. Measured distances with and without the quality improvement algorithm. Mean errors before applying the improvement algorithm is 19.31% of the real distance. The mean error is reduced to 3.61% after applying the improvment algorithm.

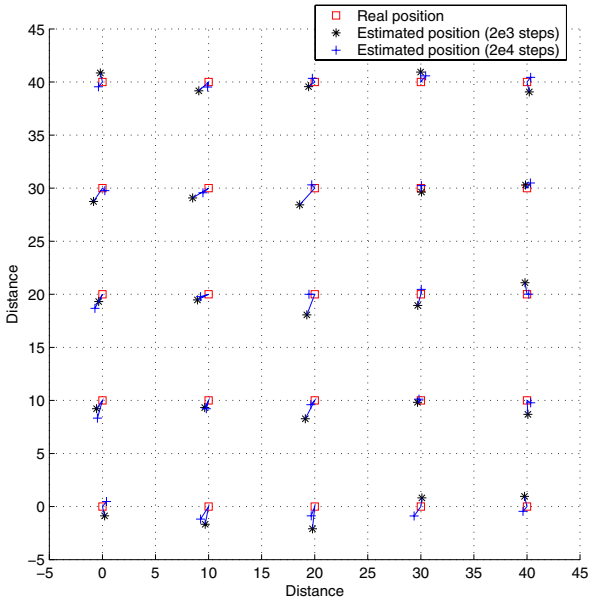

Figure 11. Localization results after 2,000 and 20,000 steps. Mean and median errors are 11.53% and 9.83% of node spacing after 2,000 steps and 6.73% and 5.80% after 20,000 steps.

tained from phase shift information. Note that this graph shows some negatively correlated measurements which is due to the error in resolving the circular ambiguity of phase shift. Moreover, the vertical distribution at distance 0 is the measurements of node 13. Because *node 13* is at the center of mass of the system, its real projected distance is always 0. The mean estimation error at this step is 19.31% of the real distance. The plus marks in the graph are the estimated distances after applying the measurement quality improvement algorithm. Note that most of the negative correlation is removed from this step. This indicates that the erroneous measurements have low confidence and they are replaced with combinations of better measurements. The improvement algorithm result in a mean estimation error of 3.61% of the real distance.

4.4 Localization

Figure 11 shows the results of localization after 2,000 steps of simulation and after 20,000 steps of simulation. During the first 2,000 steps there were 4 events actually converted to projected distances and from them we got a localization result with mean and median errors of 11.53% and 9.83% of grid spacing. The position estimate after 2,000 steps are marked as '*' in Figure 11. After 20,000 steps of simulation, we got 55 distance measurements and from

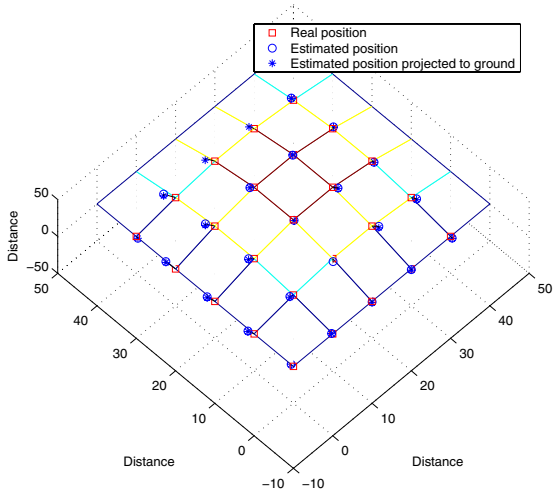

Figure 12. 3D Localization result. Mean and median errors are 9.79% and 7.82% of node spacing.

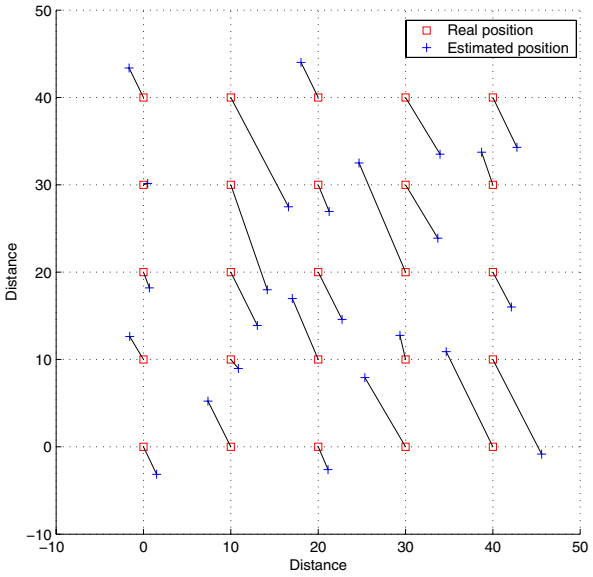

Figure 13. 2D Localization result with clock error. Mean and median errors are 63.00% and 58.45% of node spacing.

them the mean and the median localization errors are reduced to 6.73% and 5.80% of grid spacing. The estimated positions after 20,000 steps are marked as '+'. Note that in Figure 11 even anchor nodes have errors. This is because the anchors are used for finding the linear combination coefficients not for the positions of individual nodes.

Figure 12 shows a three dimensional localization result. We run the simulation with the same set of events, and the same set of anchor nodes, but we put the nodes at $[10x, 10y, 20 \sin \frac{\pi x}{5} \cdot \sin \frac{\pi y}{5}]$. In Figure 12 the estimated positions are marked as circles, and in order to show their errors in z dimension we mark their projected points to the ground by star symbols. The mean and median errors of this three dimensional localization result are 9.79% and 7.82% of the node spacing. The errors are increased compared to the two dimensional result. This is due to the reduced redundancy in anchor nodes and the increased dimension to estimate.

As a final experiment, we perform two dimensional localization when time synchronization errors are present. In order to compare the results with and without clock error, we saved the recordings of all 25 nodes in the previous two dimensional localization. For the recording of each node we removed the first *I* samples, where *I* is an instance of a uniform random variable between 0 and 9 to simulate the initial clock error, and we removed a sample at every *J+1,000* samples, where *J* is an instance of a uniform random variable between 0 and 10,000 to simulate the clock drift error. We added 1,000 to *J* to prevent dropping too many samples. As a result, the initial drops are ranging from 0 to 9 samples and the periodic sample drops are from 2 to 14 samples. That is, the largest clock differences are 14.29% and 17.14% of T_1.

Figure 13 shows the localization result without compensating for the time synchronization error. The mean and median localization errors are 63.00% and 58.45% of node spacing. The figure shows small errors in 45° direction but large errors in 135° direction. This is because the first principal axis closely estimates the node positions, while the second principal axis alone does not and needs more correction with other principal axes.

Figure 14 shows the localization result with the compensation algorithm enabled. With the compensation, the mean and median localization errors are reduced to 6.74% and 5.76% of the node spacing. This error is comparable to the two dimensional localization result without clock error even though this experiment is done with less redundancy in anchor nodes.

5 Conclusion

Our passive localization algorithm interprets recorded time differences of a global event as projected distances and estimates locations from the set of projected distances. The localization algorithm uses a frequency domain event detection algorithm and a phase shift based distance estimation algorithm, along with a measurement quality improvement method. Simulation studies show that the algorithm is efficient and robust in noisy environments. Moreover, because

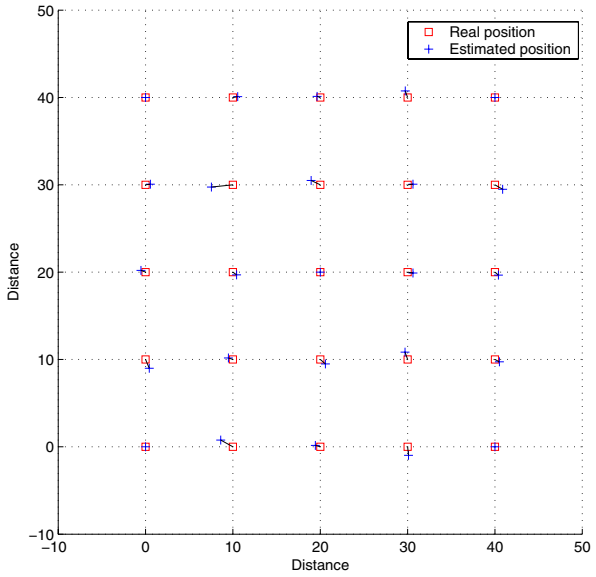

Figure 14. 2D Localization result with clock error and the compensation algorithm is enabled. Mean and median errors are 6.74% and 5.76% of node spacing.

our localization algorithm does not rely on node to node distances, it works in dense as well as sparse systems.

One of the problems we want to address in future research is the use of anchor nodes. Currently anchor nodes are used to compute linear combination coefficients. However, anchor nodes can also be used in estimating distance measurements. For example, the direction of events, or their propagation speed, can be estimated using the anchor nodes, and this information may provide constraints in the steps for estimating distance measurements.

One of the difficulties in passive localization is that the algorithm is inherently centralized: projected distances are obtained by comparing all recordings together, and the localization algorithm requires principal axis analysis on all sets of distance measurements. Thus, each node needs to store its entire recording before it is sent to a central server. Note that other distance measurement methods, such as TDOA, convert the sample to a distance and are able to reuse the sample buffer.

Although there are some difficulties in using global events for localization, using such measurements prevents the accumulation of errors that result from combining small-scale neighborhood localization results. Moreover, we can apply the passive localization algorithm to already existing records, and we can seamlessly combine recordings from different types of sensors. We believe passive localization may provide an energy efficient and accurate solution for large and sparse sensor networks.

References

[1] Y. Baryshnikov and J. Tan. Localiztion for anchoritic sensor networks. In *Distributed Computing in Sensor Systems*, pages 82–95. LNCS 4549, Springer Berlin/Heidelberg, 2007.

[2] Crossbow Technology, Inc. http://www.xbow.com.

[3] X. Ji and H. Zha. Sensor positioning in wireless ad-hoc sensor networks with multidimensional scaling. In *Proceedings of the 23rd Conference of the IEEE Communications Society (IEEE INFOCOM)*, 2004.

[4] Y. Kwon, K. Mechitov, S. Sundresh, W. Kim, and G. Agha. Resilient localization for sensor networks in outdoor environments. In *International Conference on Distributed Computing Systems*, pages 643–652. IEEE Computer Society, 2005.

[5] D. C. Lay. *Linear Algebra and Its Applications*. Addison Wesley, 1994.

[6] R. L.R. and R. Schafer. *Digital Processing of Speech Signals*. Prentice Hall, 1978.

[7] M. Maroti, G. Simon, A. Ledeczi, and J. Sztipanovits. Shooter localization in urban terrain. In *Computer*, pages 60–61. IEEE Computer Society Press, 2004.

[8] K. Mechitov, W. Kim, G. Agha, and T. Nagayama. High-frequency distributed sensing for structure monitoring. In *Trans. of the Society of Instrument and Control Engineers (SICE)*, volume E-S-1, pages 109–114, 2006.

[9] K. Mechitov, S. Sundresh, Y. Kwon, and G. Agha. Cooperative tracking with binary-detection sensor networks. In *International Conference on Embedded Networked Sensor Systems (Sensys)*, pages 332–333. ACM Press, 2003.

[10] D. Niculescu and B. Nath. Ad hoc positioning system (APS). In *GLOBECOM*, November 2001.

[11] A. V. Oppenheim and A. S. Willsky. *Signals and Systems*. Prentice Hall, 1983.

[12] J. Sallai, G. Balogh, M. Maroti, and A. Ledeczi. Acoustic ranging in resource constrained sensor networks. In *Technical Report ISIS-04-504*. Vanderbilt University.

[13] Y. Shang and W. Ruml. Improved MDS-based localization. In *INFOCOM*, volume 7, pages 2460–2651. IEEE, 2004.

[14] Y. Shang, W. Ruml, Y. Zhang, and M. P. Fromherz. Localization from mere connectivity. In *International Symposium on Mobile Ad Hoc Networking & Computing*, pages 201–212, Annapolis, Maryland, USA, 2003. ACM Press.

[15] G. Strang. *Linear Algebra and Its Applications*. Harcourt Brace Jovanovich, 3rd edition, 1988.

[16] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In *Sensys*, 2004.

[17] G. Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, 2nd edition, 2001.