A Graph Drawing Approach to Sensor Network Localization

Muhammad S. Nawaz and Sanjay K. Jha
School of Computer Science & Engineering
The University of New South Wales
Sydney 2052, Australia
Email: {mnawaz,sjha}@cse.unsw.edu.au

UNSW-CSE-TR-0611

June 2006

THE UNIVERSITY OF
NEW SOUTH WALES

SYDNEY · AUSTRALIA

Abstract

In this work, we present a centralized localization mechanism for wireless sensor networks. Our method is based on a graph drawing algorithm and utilizes inter node distances to localize sensor nodes in a local coordinate system upto a global translation, rotation and reflection without any absolute reference positions such as GPS or other anchor nodes. We show through simulations that, it is possible to avoid folds and flips in the localized network layout if the entire topology is considered as a whole as opposed to distances to immediate neighbors only. We assess the effect of different parameters like scale, node degree and ranging noise on our algorithm. Finally, we propose a hierarchical approach to make the algorithm scalable for large networks, which we would like to pursue as future work.

Keywords: Localization, Wireless Sensor Networks, CSE, UNSW.

1 Introduction

Recent trends in electronic miniaturization and the advances made in wireless communication has given us the ability to create tiny probes that can not only sense but also arrange themselves in networks and beam this information back over the air. These networks allow us to instrument our world in novel ways providing detailed insight that had not been possible before. Since these sensor networks provide an interface to the physical world, therefore, we must have a mechanism for locating each node in the physical space. The size of these networks prohibits the use of manually defined locations for all of the nodes. Therefore, a localization service must be built in the network allowing it to estimate node positions.

Localization is an important middle-ware service in wireless sensor networks. It provides location information to each individual node in the network over which services like event reporting, routing, data aggregation and many other higher level services can be built. This location awareness allows each node to send location stamped data back to base station. Without the location information, the raw data would not be useful at the base station. Having location information also opens up other interesting possibilities like geographic routing [21], robot navigation [5], counter sniper systems [22] and a wide range of other interesting applications.

In section II, we review some of the related work from both the sensor network localization community and some graph drawing algorithms to see the similarities between the two. In section III, we will look in detail at one of the graph drawing algorithms. In section IV, we will present our localization algorithm based on the graph drawing mechanism. Later we will analyze the effect of different parameters on our localization algorithm. Finally we will present an extension of our current work to make the localization mechanism scalable which we would like to explore in future.

2 Related Work

In this section, we will review some of the related work from the sensor network localization community. We will also briefly look at some of the graph drawing algorithms and will try to develop a connection between the two fields.

Location is considered an important attribute in wireless sensor networks and there is a large body of work targeting this problem. A sensor network localization mechanism can generally be separated in two distinct parts, a method for measuring distances between different sensor nodes and an algorithm that converts these distances into sensor node position estimates. A number of ranging technologies like [18], [19], [16] have been used with sensor nodes for distance measurements. The algorithms that use these distances as inputs to determine position estimates can be classified as, anchor based and anchor free algorithms. Anchor based algorithms [15], [20] assume that some of the nodes referred as anchors or beacons have an a priori knowledge of their locations through manual initialization or through some external infrastructure like GPS. These anchor node locations and inter node distances are then used to localize rest of the nodes in the network. A detailed analysis of these anchor based algorithms can be found in [14].

Anchor free algorithms [17], [4] make no assumptions about a priori location knowledge of some nodes in the network and use only inter node distances to localize the entire network in a local coordinate system. This local coordinate system can be readily used with applications like geographic routing [21] where only the relative location of the destination node is required as opposed to absolute GPS coordinates. For other applications this local coordinate system can be consolidated into any other coordinate system using methods described in [9] or [13]. Anchor free algorithms allow the sensor networks to be

decoupled from any external infrastructure or any manual initialization. This is an extremely important and desirable property for these networks to become truly *place and play* systems. Our algorithm also falls in the same category of anchor free localization algorithms.

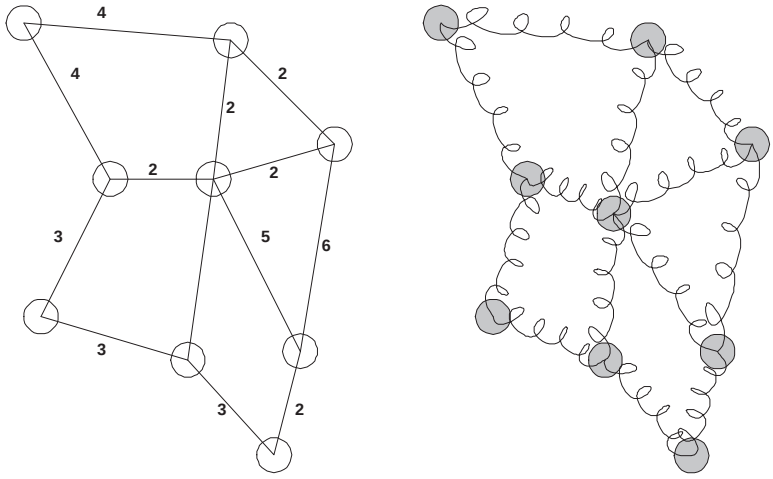
Anchor free algorithms take the distance constraints as input and form a system of non linear equations which is generally solved through some form of non linear optimization. Some approaches model this as a physical analogy [10]. The nodes in the network are replaced with point masses and the edges connecting them are replaced with springs. These point masses are randomly placed in a plane and released. In a real physical system, these springs exhibit tension and try to compress or expand to their normal rest lengths pulling the masses to correct locations. Similarly in these mesh relaxation algorithms, the nodes are randomly placed in a 2D plane and nonlinear optimization is used to move the nodes to locations that satisfy these distance constraints. One advantage of these mesh relaxation methods is that they can be easily implemented in a distributed manner since they involve only neighbor to neighbor communication. However, one major problem with this approach is that the mesh relaxation has a high probability of converging to a false minimum when the initial placement of nodes is random. These false minimums correspond to folded or collapsed layouts of the original network. This problem has been hinted at in [10] and addressed in more detail in [17].

Now we turn our attention briefly to graph drawing. Graph drawing is a burgeoning field dedicated to determine algorithms for drawing *aesthetically pleasing* layouts for different types of graphs. A large variety of different algorithms [2] have been proposed that take into account different aesthetics criteria like edge crossings, vertex spacings etc for graph drawings. For general graphs, there exists a body of algorithms [6], [7], [11] that are referred to as *force directed* drawing algorithms. These graph drawing algorithms, like Mesh Relaxation, also model the drawing process as forces acting on vertices and then try to find vertex locations where these forces become zero. In fact a brief review of these algorithms shows that the Mesh Relaxation is just a slight variation of these approaches. However, these algorithms also introduce repulsive forces between non adjacent vertices which avoid the collapsed layouts and result in uniform population of the drawing area by the vertices. Recently, other graph drawing algorithms [12] have also been used for both centralized [3] and distributed [8] sensor network localization.

3 Kamada Kawai Graph Drawing

In this section, we will review the Kamada Kawai graph drawing algorithm [11]. We will present a slightly modified version of the original algorithm that is not constrained by the drawing area. Let us suppose that there are *n* vertices located in a two dimensional plane. Each vertex is connected to all of the nearby vertices that lie with in a circle of radius *r* centered at that vertex as shown in fig. (1a). The weights on the edges represent the distances between the vertices. Such a graph can be modeled as a physical system of point masses connected together through springs. Traditional spring embedders [6], [7] modeled such graphs by replacing each edge in the graph with a spring that had a rest length equal to the edge weight (distance in our case) to form a mass spring mesh as shown in fig. (1b). However, as we noted earlier such spring embedders always resulted in *collapsed* or *folded* layouts like in [17], [10] in the absence of forces between non adjacent vertices unless initialized with a *good* guess of the original layout.

Kamada and Kawai also present a spring embedder for drawing aesthetically pleasing graphs in [11]. However, it is different from the earlier spring embedders in the sense that it assumes that each point mass is connected to all of the remaining $n-1$ masses with springs having different spring constants and rest lengths irrespective of whether there is an edge between two vertices or not in the original graph. For each pair of vertices, the spring's rest length and the spring constant is derived from the shortest



(a) A simple graph (b) Mass spring mesh

Figure 1: A spring embedder's view of the graph

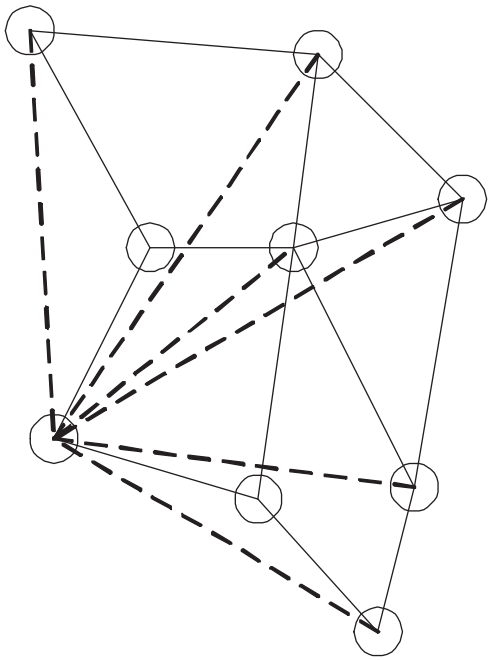


Figure 2: Kamada Kawai spring embedder's view of the same graph. Solid lines represent actual edges and the dashed lines represent created connections. The connections for only one vertex are shown for the purpose of clarity

path distance between the vertices. If the current distance between the point masses is smaller than the rest length of the spring, it pushes them away from each other; if the current distance is larger than the rest length, it pulls the masses closer to each other. The imbalance of such a mass spring system can be modeled as an energy function *E* given as

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{ij} \left(\|p_i - p_j\| - d_{ij} \right)^2 \tag{1}$$

where

p_i and p_j are current coordinates of i and j

d_{ij} is the shortest path distance between i and j

k_{ij} is the spring constant of the spring connecting i and j

The spring constant k_{ij} is given as

$$k_{ij} = \frac{K}{d_{ij}^2}$$

where K is a constant.

Eq. (1) is infact the summation of the differences of the current and desirable distances for all $\frac{n(n-1)}{2}$ pairs of vertices. Therefore, the best graph layout can be determined by minimizing the energy *E*. Since each vertex is connected to all of the remaining $(n-1)$ vertices, so the resulting layout cannot collapse or fold over itself. In a two dimensional plane, the position of each vertex can be expressed by two coordinates, *x* and *y*. Therefore, energy *E* is a function of $2n$ variables $x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n$.

$$E(\mathbf{X}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{ij} \left(\|p_i - p_j\| - d_{ij} \right)^2 \tag{2}$$

where

$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 & \ldots & x_n & y_1 & y_2 & \ldots & y_n \end{bmatrix}^T$$

$$E(\mathbf{X}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{ij} \left\{ (x_i - x_j)^2 + (y_i - y_j)^2 + d_{ij} - 2d_{ij} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \right\} \tag{3}$$

The condition for the minimum of the energy function E is

$$\frac{\partial E(X)}{\partial x_i} = \frac{\partial E(X)}{\partial y_i} = 0 \qquad \text{for } 1 \leq i \leq n \tag{4}$$

where

$$\frac{\partial E(X)}{\partial x_i} = \sum_{j \neq i}^{n} k_{ij} \left\{ (x_i - x_j)^2 - \frac{d_{ij} (x_i - x_j)}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \right\}$$

and

$$\frac{\partial E(X)}{\partial y_i} = \sum_{j \neq i}^{n} k_{ij} \left\{ (y_i - y_j)^2 - \frac{d_{ij} (y_i - y_j)}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \right\}$$

However, the $2n$ simultaneous nonlinear equations given by Eq. (4) cannot be solved directly because they are not independent. If, however, only one vertex *k* is selected at a time and energy E is viewed as a function of only two variables x_k and y_k, then it is possible to minimize the energy E with respect to x_k and y_k i.e. choosing a vertex *k* and moving it to a position that minimizes *E* while freezing all of the remaining vertices at their current locations. By iterating this step, a minimum satisfying Eq. (4) can be obtained. In each step, a vertex with largest Δ is selected, where

$$\Delta_k = \sqrt{\left(\frac{\partial E(X)}{\partial x_k}\right)^2 + \left(\frac{\partial E(X)}{\partial y_k}\right)^2}$$

This results in two non linear equations in two unknowns x_k and y_k

$$\begin{aligned} \frac{\partial E}{\partial x_k} &= 0 \\ \frac{\partial E}{\partial y_k} &= 0 \end{aligned} \tag{5}$$

These are solved using the iterative Newton-Raphson method. If $(x_k(0), y_k(0))$ are the initial coordinates of the selected vertex k, then the following iteration is repeated until Δ_k becomes small.

$$\begin{aligned} x_k\left(t+1\right) &= x_k\left(t\right) + \delta x \\ y_k\left(t+1\right) &= y_k\left(t\right) + \delta y \end{aligned} \tag{6}$$

δx and δy are computed from the following pair of linear equations

$$\frac{\partial^2 E}{\partial x_k^2}\left(x_k(t), y_k(t)\right)\delta x + \frac{\partial^2 E}{\partial x_k \partial y_k}\left(x_k(t), y_k(t)\right)\delta y = -\frac{\partial E}{\partial x_k}\left(x_k(t), y_k(t)\right) \tag{7}$$

$$\frac{\partial^2 E}{\partial y_k \partial x_k}\left(x_k(t), y_k(t)\right)\delta x + \frac{\partial^2 E}{\partial y_k^2}\left(x_k(t), y_k(t)\right)\delta y = -\frac{\partial E}{\partial x_k}\left(x_k(t), y_k(t)\right) \tag{8}$$

The process of selecting a vertex with largest Δ and moving it to its stable position continues until the Δ for all of the vertices become small enough and at that point the drawing is finished.

4 Localization

In this section, we will present a localization mechanism for wireless sensor networks in which individual sensor nodes are equipped with devices like [18] and can measure distances to their nearby neighbors. Our localization scheme is based on Kamada Kawai graph drawing algorithm designed to draw *aesthetically pleasing* graphs. However, as we will show later, it is possible to localize a sensor network upto a global translation, rotation and reflection by using a combination of Kamada and Kawai and Mesh Relaxation. If required this localized network layout can then be transformed into a absolute coordinate system with the help of anchor nodes. However, our scheme does not depend on anchors for network localization and it will work in the absence of any anchor nodes.

We assume that after the network is deployed, each node enters into a ranging phase and measures distances to its nearby neighbors within its own ranging distance. After this, all of the successful distance measurements along with the node ID are transferred to a base station using multihop routing. After all of the distance measurements are received at the base station, a graph is constructed in which each edge represents a distance and the end points of each edge represent sensor nodes. Once the graph is

constructed, the localization problem can be viewed as a graph drawing problem. However, we are not interested in aesthetics but a graph drawing that represents the actual sensor network. Later in the section, we will see that the *aesthetically pleasing* drawings come quite close to the original network layout and require only slight refinement.

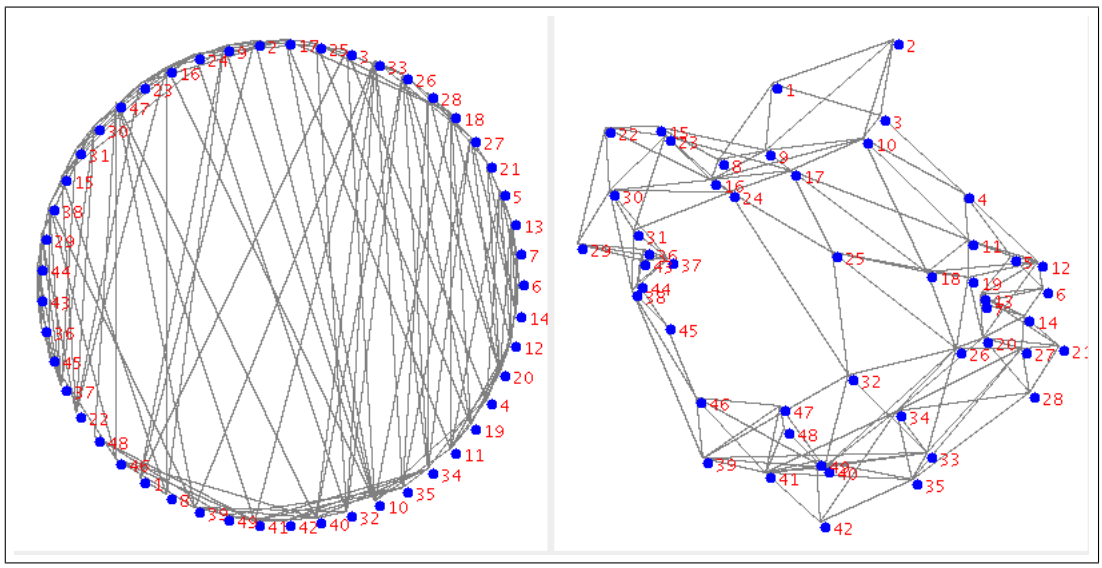
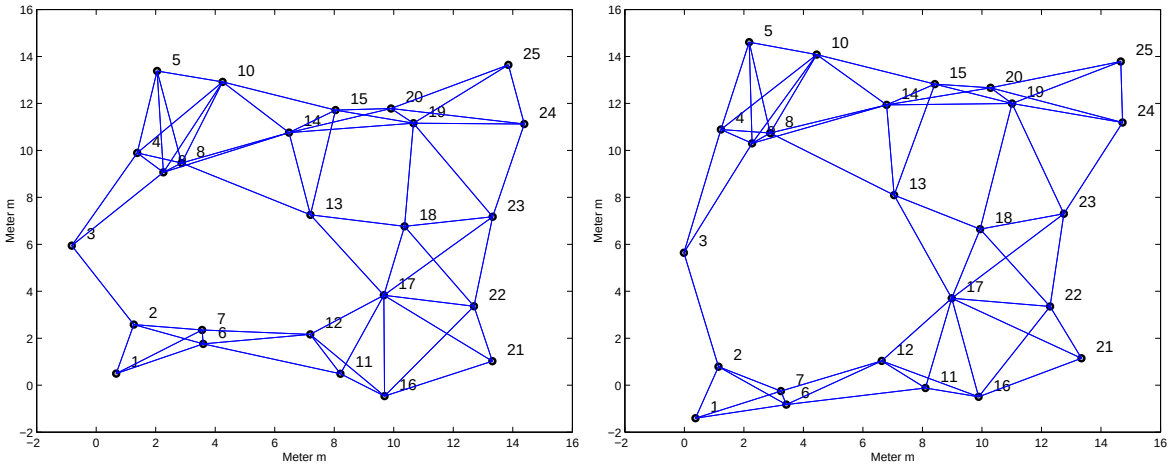


Figure 3: Screen shot of the graph drawing process

Once we have the graph, we must initialize the graph drawing process with initial node locations. After the initialization step, the drawing process continually moves the nodes closer to their original locations in the network layout. We experimented with different starting layouts including a completely random one. However, the best results were obtained with what we call as the *Distance Circle* layout. It is constructed by determining the shortest path distance between every pair of nodes and choosing the end point nodes of the largest one. These nodes are placed at the opposite ends of the diameter of a circle centered at the origin. The diameter d of the circle is chosen such that $d \gg d_{max}$ where d_{max} is the longest shortest path. The rest of the nodes are distributed uniformly along the circumference of the circle. The reason for choosing the *Distance Circle* as the starting layout was that it always resulted in a final layout that was closer to the original network layout as opposed to the random initialization which *at times* resulted in different final layouts. Fig. (3) shows a snap shot of our simulator. On the left, it shows the starting *Distance Circle* layout and on the right, it shows the current iteration of the drawing process.

After the initialization, we use Kamada and Kawai's method of determining the minimum energy state for the graph. In each step, a vertex with largest Δ value is selected. This vertex is then iteratively moved to its lowest energy position using Eq. (6) and Eq. (7). This process is repeated until the Δ values of all of the vertices become very small. We observed that the graph layout generated by Kamada Kawai's method was quite close to the original network layout. However, the generated vertex coordinates were not equal to *exact* node coordinates. The reason for this is that like other graph drawing algorithms, Kamada Kawai's method is meant to produce *pleasing* graph drawings, uniformly distributing the vertices with respect to the edge weights within a bounded area, a problem that is similar but fundamentally different from localization. However, the coordinates generated by this method can be refined using mesh relaxation [10]. Fig. (4) shows a comparison of the original node locations in the network and the graph drawn by Kamada Kawai's method using the inter node distances after it has been initialized with a *Distance Circle* layout. It shows that the drawn graph is a slightly distorted version of the original network.



(a) Original generated network layout

(b) Graph drawn by Kamad Kawai's algorithm

Figure 4: Comparison of original network layout and the output of Kamada Kawai drawing process. The drawn graph looks slightly distorted but quite similar to the original network layout

In the later section, we will have a closer look at the distortion experienced by the network when drawn by this method.

Mesh relaxation requires a good guess of vertex coordinates to converge which in our case is provided by Kamada and Kawai's drawing process. Mesh relaxation is initialized with the vertex coordinates p_i generated by Kamada Kawai's drawing process. During each relaxation step, these coordinates are incrementally improved. In each step t, the force on vertex i due to an adjacent vertex j is calculated using

$$\vec{f}_{ij}\,[t] = \frac{k}{m_i}(\|p_i - p_j\| - d_{ij})\hat{r}_{ij} \tag{9}$$

where

$\hat{r}_{ij}$ is a unit vector directed from i to j

m_i is the number of neighbors of i

The total force $\vec{F}_i\,[t]$ on i due to all of its neighboring vertices is the sum of individual forces

$$\vec{F}_i\,[t] = \sum_j \vec{f}_{ij}\,[t] \tag{10}$$

The new coordinate estimate for vertex i is generated as

$$p_i\,[t+1] = p_i\,[t] + F_i\,[t] \tag{11}$$

In our implementation, we used a fixed number of relaxation steps (t=100). All of our generated topologies converged well before the relaxation steps finished. At the end of mesh relaxation, each vertex coordinate is taken as the respective node's position estimate. Algorithm (1) represents the entire localization process as pseudo code.

It must be noted that our mechanism produces location estimates in a relative coordinate system upto a global translation, rotation and reflection. Therefore, a method for bringing the localized layout in

coincidence with the actual generated network for comparison purposes is required. In other words, we need a transformation from the localized relative coordinate system to the absolute coordinate system in which the network topologies are generated. We use a closed form solution for calculating such transformations as proposed by Horn et al. in [9].

Algorithm 1 Sensor Network Localization

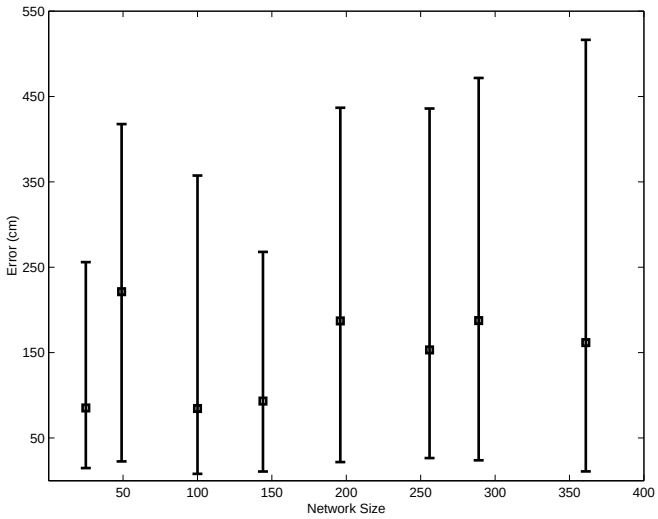
Require: All distance measurements have been collected

Determine the shortest path d_{ij} for all pairs of nodes
Calculate spring constant k_{ij} for all pairs of nodes
Let $d_{max} = max(d_{ij})$
Initialize node positions with a $DistanceCircle$ layout with diameter $100 \times d_{max}$
while $max(\Delta_i) >$ threshold **do**
    Let k be the node with $\Delta_k = max(\Delta_i)$
    while $\Delta_k >$ threshold **do**
        Calculate δx and δy
        $x_k \leftarrow x_k + \delta x$
        $y_k \leftarrow y_k + \delta y$
    end while
end while
$t \leftarrow 0$
while $t <$ Iterations **do**
    for all nodes **do**
        Calculate resultant force and direction
        Update node coordinate estimate
        $t \leftarrow t + 1$
    end for
end while

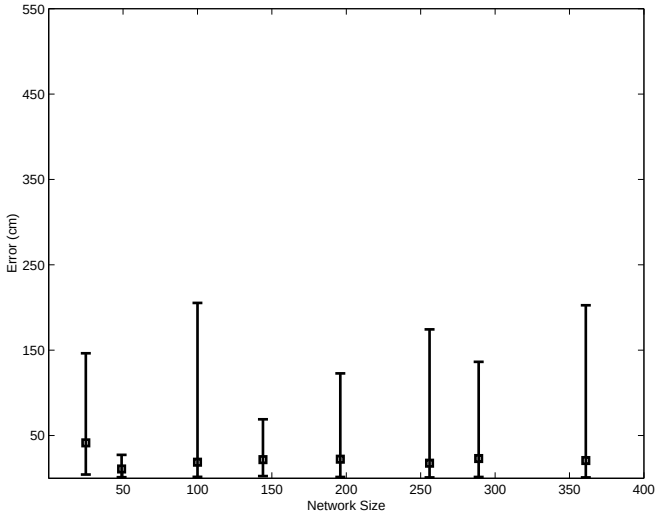
5 Simulation

In this section, we analyze the effect of different parameters on our localization scheme with the help of simulations. We investigate the effect of network size, node degree and ranging error on both phases of our localization method. In order to analyze the effect of different parameters, we decided to generate topologies that not only had some randomness but also some regularity of structure. The reason for choosing such topologies is that some random topologies are more well behaved than others which at times could be very difficult to localize. Therefore, we decided to generate noisy grid topologies like the ones mentioned in [23] for all of our simulations. However, our localization scheme is not limited to these topologies and can also be used to localize random topologies.

In our first set of simulations we investigate the effect of increasing the size of the network on the localization error. We vary the number of nodes in the network while maintaining the average node density constant to around 0.1 nodes/m^2. The smallest topology consists of 25 nodes in an area of $15m \times 15m$ and the largest topology consists of 361 nodes in an area of $57m \times 57m$. Each node in the network has a maximum ranging distance of 5m. The average node degree for all of the topologies was around 6. The results of these set of simulations are shown in fig. (5). Each point in the figure is an average of 10 runs of simulation on 10 different similar sized topologies. Fig. (5a) shows the minimum, average and maximum location error after the Kamada Kawai's graph drawing and fig. (5b) shows the minimum, average and maximum error after applying mesh relaxation on the output of Kamada Kawai.



(a) Location error after applying Kamad Kawai



(b) Location error after applying mesh relaxation on Kamad Kawai's output

Figure 5: Increasing the network size does not affect the average error of our localization scheme

We can see in fig. (5) that the average localization error is around $20cm$ and it is not effected by the size of the network being localized.

Average error is just an indicator and we believe that it makes more sense to look at the entire distribution of errors in the localized network to gain insight into the performance of a localization algorithm. We see from fig. (5b) that the maximum localization error was experienced by one of the topologies of 100 nodes. Therefore, we choose the topology of 100 nodes that experienced the largest maximum error from our previous simulation set to examine the error distribution. This topology is shown in fig. (6). A comparison of actual node coordinates and localized node positions for the selected topology along with the cumulative distribution of errors at both stages of our algorithm is shown in fig. (7). Fig. (7a) is a comparison of actual node positions and the vertex coordinates generated by the Kamada Kawai's drawing algorithm. We observe that the vertex locations seem to be radiating outward from the center. The nodes at the center experience minimum error while the nodes at the edges experience larger errors. We believe that it is due to the fact that the nodes located close to the center experience forces from all of the directions as opposed to the nodes at the edges that experience forces originating only from the inner side of the network. Fig. (7b) shows the plot of empirical cumulative distribution function of vertex coordinate errors. We can see from the CDF plot that 90 percent of the vertices experience less than $150cm$ of location errors and only 10 percent of the remaining vertices experience errors as large as $300cm$. Fig. (7c) compares the location estimates with the actual node locations after applying mesh relaxation. It shows that the location errors are so small that they are not visible at this scale. The CDF plot in fig. (7d) shows that over 95 percent of errors are below $25cm$ and there are only a few nodes in the network that are experiencing large location errors of around $200cm$.

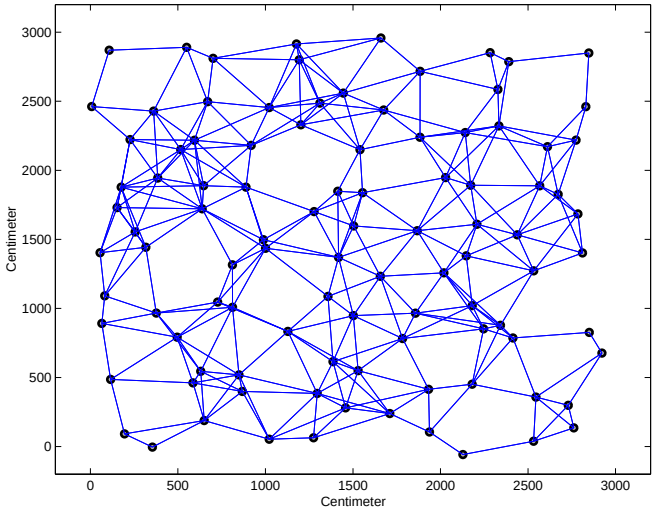
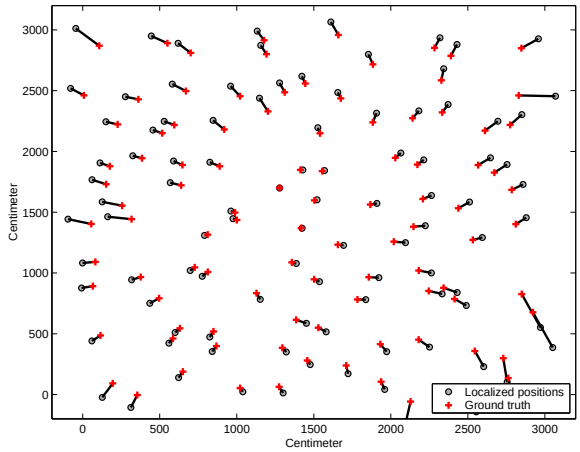
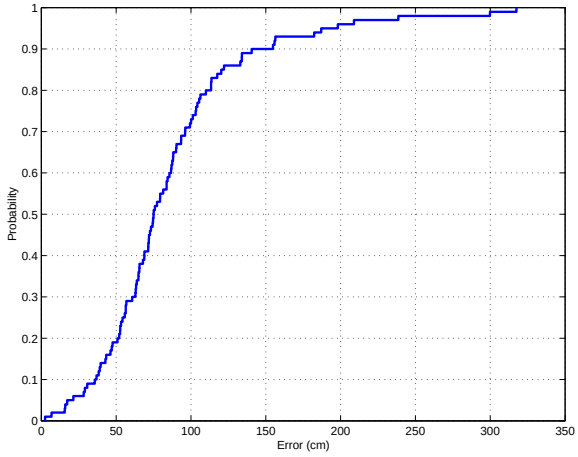


Figure 6: 100 node topology experiencing largest error

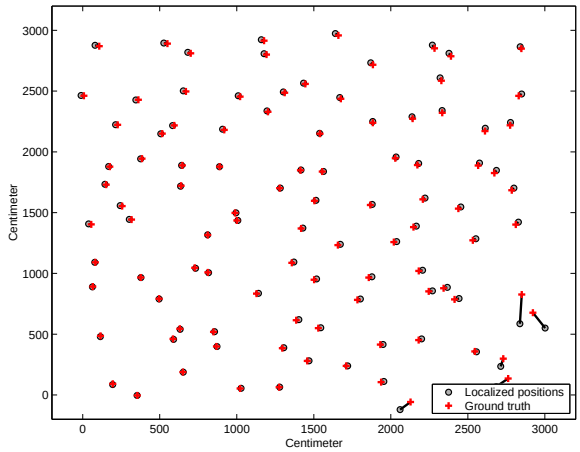
Next we investigate the effect of average node degree on our scheme to see how it affects the localization error. The same topology shown in fig. (6) is used for this simulation. Different average node degrees are achieved by increasing the maximum ranging distance of each node. Fig. (8) shows the result of this simulation where the vertical axis is the localization error and the horizontal axis is average node degree. The two curves show the minimum, average and maximum localization errors of the two stages of our scheme. Increasing the node degree from 7 to 14 rapidly decreases the average error of the coordinates generated by Kamada Kawai's graph drawing scheme. As the node degree is increased beyond 14, the difference between the two curves becomes smaller. Thus adding more edges in the graph allows the Kamda Kawai's algorithm to draw a graph that matches more closely to the real network layout.



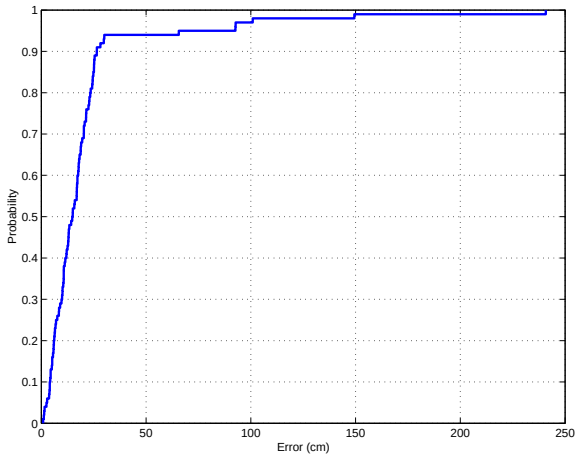
(a) Estimated coordinates after Kamada Kawai Drawing



(b) Location Error CDF after Kamada Kawai Drawing



(c) Estimated coordinates after Mesh Relaxation



(d) Location Error CDF after Mesh Relaxation

Figure 7: Analysis of localization errors of our scheme

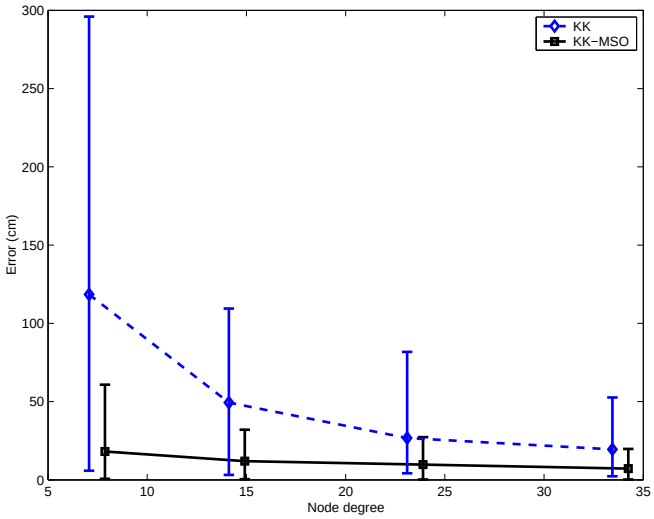


Figure 8: Effect of increasing node degree

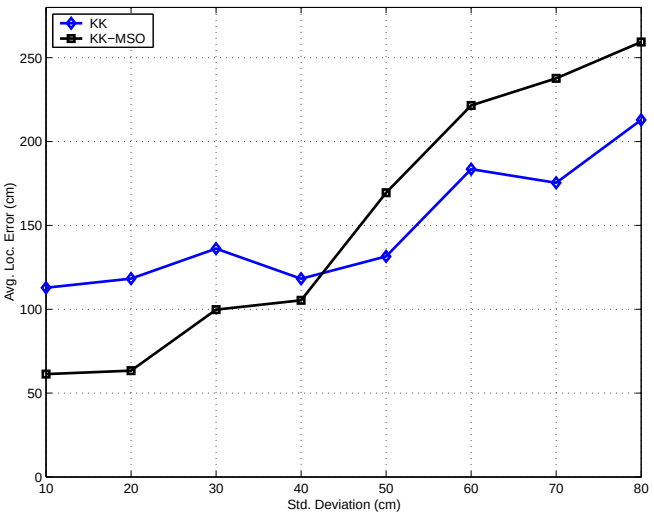
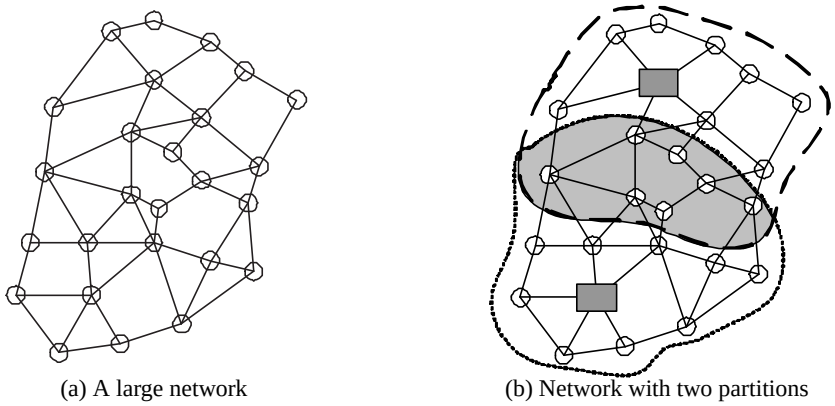


Figure 9: Increasing the ranging noise deteriorates position estimates

Now we look at the effect of ranging noise on the localization error. We selected one of the topologies of 25 nodes in $15m \times 15m$ for this set of simulations. Fig. (9) shows the result of this simulation where the vertical axis is the average localization error and the horizontal axis is the standard deviation of a normal distribution of distance errors with zero mean. The two curves show the average localization error for both the stages of our algorithm. Each point on the curve is an average of 10 runs of simulation with the same standard deviation. The curves show that increasing the distance measurement error deteriorates the performance and increases localization error. However, an interesting cross over occurs at around $50cm$. When the noise in distance measurement is increased beyond $50cm$, the mesh relaxation starts to increase location errors instead of improving location estimates. This is due to the sensitivity of mesh relaxation to its proper initialization. Beyond $50cm$ the graph layout generated by Kamada Kawai becomes so distorted that the mesh relaxation does not converge to actual node positions. These results show that our algorithm can withstand small amount of ranging noise typical of ultrasound devices.

6 Future Work

In the previous few sections, we proposed a localization algorithm for sensor networks and analyzed the effect of different parameters on localization errors with the help of simulations. In this section, we present the planned future extensions of our current work.

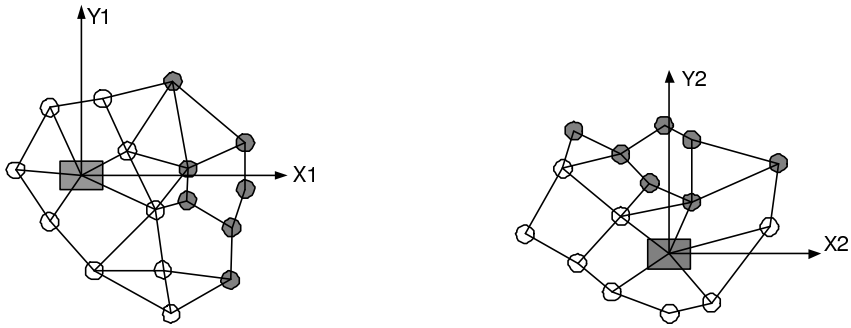


(a) A large network (b) Network with two partitions

Figure 10: Dividing a large network in partitions for localization. Chosen cluster heads are shown with rectangles. Nodes in the shaded region belong to both partitions.

Recent studies suggest that there is always some difference between simulation and real world performance of localization algorithms [23]. Therefore, we are interested in running our localization mechanism on a real medium sized deployment of Mica2 motes to see how it behaves in a real world deployment in the presence of real noise. At this stage, we are investigating different ranging technologies like [19], [1], [18] for Mica2 platform. We are specially interested in [19] because of its ability to use the existing hardware on Mica2 motes for distance measurement with reasonable errors. The audible sound frequency used with this method also makes it more omnidirectional than other ultrasonic alternatives.

Since our localization algorithm requires all of the distance measurements from the sensor field to be collected at a central base station for location estimation, this might pose a scalability issue for large networks. We propose a hierarchical scheme to use our current localization algorithm with large networks which we would like to pursue in future. Here we briefly describe the proposed scheme. Once a large network is deployed, it is divided into several overlapping partitions with one cluster head chosen in each partition. Fig. (10) shows a network divided into two such partitions. All of the distance measurements



(a) Localized partition 1 (b) Localized partition 2

Figure 11: Each partition is localized at the cluster head in its own coordinate system defined by X_1Y_1 and X_2Y_2.

from each partition are collected at the respective cluster head. Each cluster head runs our localization algorithm to localize the nodes in its own partition in a local coordinate system. Fig. (11) shows localized layouts of both partitions in their respective coordinate systems defined by axes X_1Y_1 and X_2Y_2. Grey circles represent those nodes that belong to both of the partitions and therefore, each cluster head has their estimated locations in its own coordinate system. Now the cluster heads can exchange the local coordinates of these common nodes with each other and after this exchange each cluster head can calculate an inter-cluster transformation using the method described in [9]. Using this transformation any of the two local coordinate systems can be aligned and stitched to the other one as shown in fig. (12).

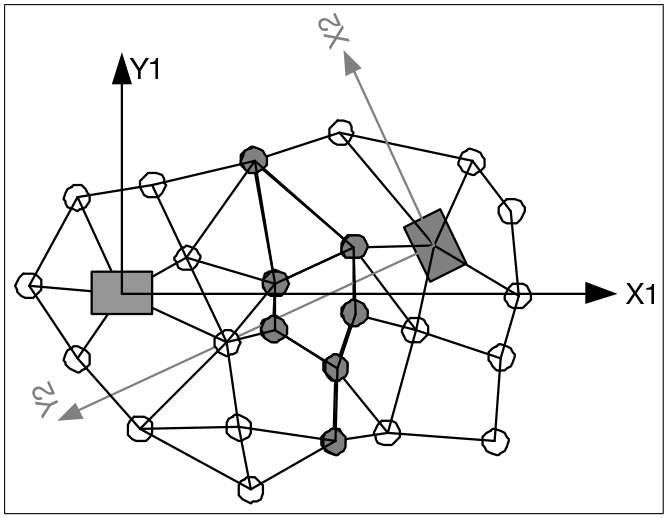


Figure 12: Aligning the coordinate system of cluster head 2 into the local coordinate system of cluster head 1 with the help of common nodes.

7 Conclusion

We presented a centralized localization algorithm for wireless sensor networks based on a graph drawing algorithm and analyzed the effect of different parameters on the resulting localization errors. We showed that by considering the entire topology it is possible to avoid the collapsed or folded layouts. We also showed that the graph drawing process is capable of generating layouts that are quite similar to the actual network layout and thus require only slight refinement to generate node position estimates. Finally we outlined a planned extension of our current work that would allow our localization mechanism to be used with large networks in a distributed manner.

References

[1] Berkley ranging boards: http://www.tinyos.net/scoop/special/hardware#mica2dotsensorboards.

[2] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Comput. Geom. Theory Appl.*, 4(5):235–282, 1994.

[3] M. Broxton. Localization and sensing applications in the pushpin computing network, 2005.

[4] M. Broxton, J. Lifton, and J.A. Paradiso. Localizing a sensor network via collaborative processing of global stimuli. In *Second European Workshop on Wireless Sensor Networks*, pages 321–332, 2005.

[5] P. Corke, R. Peterson, and D. Rus. Coordinating aerial robots and sensor networks for localization and navigation. In *Proc. 7th Int. Symp. on Distributed Autonomous Robotic Systems (DARS'04)*, Toulouse, June 2004.

[6] Peter Eades. A heuristic for graph drawing. *Congressus Numeration*, 42:149–160, 1984.

[7] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Softw. Pract. Exper.*, 21(11):1129–1164, 1991.

[8] Craig Gotsman and Yehuda Koren. Distributed graph layout for sensor networks. In *Graph Drawing*, pages 273–284, 2004.

[9] Horn and B. K. P. Closed form solution of absolute orientation using unit quaternions. *Journal of the Optical Society*, pages 629–642, April 1987.

[10] Andrew Howard, Maja J. Matarić, and Gaurav S. Sukhatme. Relaxation on a mesh: a formalism for generalized localization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1055 – 1060, Wailea, Hawaii, Oct 2001.

[11] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inf. Process. Lett.*, 31(1):7–15, 1989.

[12] Yehuda Koren. On spectral graph drawing. In *COCOON*, pages 496–508, 2003.

[13] Jack B. Kuipers. Quaternions and rotation sequences, 1999.

[14] Koen Langendoen and Niels Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. *Comput. Networks*, 43(4):499–518, 2003.

[15] Dragos Niculescu and Badri Nath. Ad hoc positioning system (APS). In *GLOBECOM (1)*, pages 2926–2931, 2001.

[16] Neal Patwari and III Alfred O. Hero. Using proximity and quantized rss for sensor localization in wireless networks. In *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 20–29, New York, NY, USA, 2003. ACM Press.

[17] Nissanka B. Priyantha, Hari Balakrishnan, Erik Demaine, and Seth Teller. Anchor-free distributed localization in sensor networks. Technical Report 892, MIT, MIT Laboratory for Computer Science, April 2003.

[18] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 32–43, New York, NY, USA, 2000. ACM Press.

[19] János Sallai, György Balogh, Miklós Maróti, Ákos Lédeczi, and Branislav Kusy. Acoustic ranging in resource-constrained sensor networks. In *International Conference on Wireless Networks*, pages 467–, 2004.

[20] Andreas Savvides, Chih-Chieh Han, and Mani B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 166–179, New York, NY, USA, 2001. ACM Press.

[21] Karim Seada, Marco Zuniga, Ahmed Helmy, and Bhaskar Krishnamachari. Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 108–121, New York, NY, USA, 2004. ACM Press.

[22] Gyula Simon, Mikl'os Mar'oti, 'Akos L'edeczi, Gy"orgy Balogh, Branislav Kusy, Andr'as N'adas, G'abor Pap, J'anos Sallai, and Ken Frampton. Sensor network-based countersniper system. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 1–12. ACM Press, 2004.

[23] Kamin Whitehouse, Chris Karlof, Alec Woo, Fred Jiang, and David Culler. The effects of ranging noise on multi-hop localization: An empirical study. In *Fourth International Conference on Information Processing in Sensor Networks (IPSN 2005)*, pages 73–80, April 2005.