Force-Directed Approaches to Sensor Localization

ALON EFRAT, DAVID FORRESTER, ANAND IYER, and STEPHEN G. KOBOUROV
University of Arizona
CESIM ERTEN
Kadir Has University
OZAN KILIC
Isik University

As the number of application areas where sensor networks are used increases, the sensor network localization problem gains more importance. It is the problem of recovering the correct position of each node in a network of sensors from partial connectivity information such as adjacency, range, or angle between neighboring nodes. In this paper, we consider the anchor-free sensor localization problem in sensor networks that report possibly noisy range information and angular information about the relative order of each sensor's neighbors. Previously proposed techniques seem to successfully reconstruct the original positions of the nodes for relatively small networks with nodes distributed in simple regions. However, these techniques do not scale well with network size and yield poor results with non-convex or non-simple underlying topology. Moreover the distributed nature of the problem makes some of the available techniques useless in a wide range of applications. To address these problems we describe a multi-scale dead-reckoning (MSDR) algorithm that scales well for large networks, can reconstruct complex underlying topologies, and is resilient to noise. The MSDR algorithm takes its roots from classic force-directed graph layout computation techniques. These techniques are appropriately modified with a multi-scale extension to handle the scalability issue and a dead-reckoning extension to overcome the problematic cases arising with non-simple topologies. Furthermore we show that the distributed version of the MSDR algorithm performs as well, if not better than its centralized counterpart, when the qualities of the output layouts measured in terms of appropriate distance metrics are of concern.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Distributed networks*

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Sensor networks, node localization, force-directed

This work is supported in part by TUBITAK grant 106E071 and NSF grant ACR-0222920.
Authors' addresses: A. Efrat, D. Forrester, A. Iyer, and S. G. Kobourov, Department of Computer Science, University of Arizona, 1040 E 4th Street Tucson, AZ 85721-0077, USA. C. Erten, Kadir Has Universitesi, Kadir Has Caddesi, Cibali, Istanbul 34083, Turkey. O. Kilic, Isik Universitesi, Sile Kampusu, Sile, Istanbul 34980, Turkey.
Authors' email addresses: {efrat,forrester,iyer,kobourov}@cs.arizona.edu, cesim@khas.edu.tr, yokilic@isikun.edu.tr.

1. INTRODUCTION

Wireless sensor networks are used in many applications, from natural habitat monitoring to earthquake detection; see [Akyildiz et al. 2002] for a survey. Often, the actual location of the sensors is not known but is necessary for the underlying application, e.g., determining the epicenter of a quake. Further, the location of the sensors can be used to design efficient network routing algorithms [Mauve et al. 2001]. Abstractly, the sensor localization problem can be thought of as a graph layout problem. The true state of the underlying sensor network is captured by a layout D of the source graph G. Given partial information about G (adjacency information, possibly information about edge lengths, or angles between adjacent neighbors), we would like to construct a layout $\hat{D}$ of G that matches D as well as possible. There are many variations of the problem, depending on the quality of the edge length data (e.g., obtained using signal strength), or whether some of the vertices know their exact positions (e.g., GPS-equipped or manually-placed sensors), or whether the vertices can detect the relative order of their neighbors (e.g., obtained by using multiple antennas per sensor). Centralized and distributed algorithms have both been proposed for these problems.

Sensors typically have a *range* that allows them to detect other sensors that fall in that range, thus providing adjacency information for the underlying graph. The strength of the signal, or the time of arrival of the signal are typically used to estimate the actual distance between two sensors. However, sensing neighbors is far from perfect, especially close to the limits. Sensors whose exact positions are known (equipped with GPS or manually-placed) are often called *anchors*. While anchors make the localization problem easier, their use might be problematic as GPS-equipped devices tend to be more bulky, expensive and energy-hungry, and manual placement may not be possible in all situations. Further, GPS-equipped sensors do not work well indoors, under thick tree-cover and underground. In such settings, anchor-free sensor networks are more practical but pose greater challenges in localization. Sensors equipped with multiple antennas can provide *angular information* by reporting the relative order of their neighbors or an estimate on the angle between adjacent neighbors. Multiple antennas add to the cost and size of the sensor, but not nearly as much as in the case of GPS. Once again, the angular information is not perfect, but even allowing for some errors, angular information can be used to find good localizations.

In this paper, we focus on the centralized and distributed sensor localization problem for anchor-free networks. We consider the cases with or without angular information for different types of underlying regions for the sensor network: simple convex polygons, simple non-convex polygons, and non-simple polygons. Classic force-directed methods can be augmented to take into account the edge length information. This approach "works well" for small graphs of up to fifty or so vertices provided that the graphs are well-connected. It successfully reconstructs a layout $\hat{D}$ which under certain appropriate distance metrics closely matches the source layout D. For larger graphs, the simple force-directed algorithms fail to reconstruct the vertex locations. On the other hand we show that *multi-scale* versions of the force-directed algorithms are *scalable* and can extend the utility of these algorithms to graphs with hundreds of vertices, provided that the graphs

are defined inside simple convex polygons. We note that when we refer to scalable algorithms we mean algorithms whose performance do not degrade with larger input sizes as measured by the number of vertices and edges in the input graphs. When we refer to multi-scale algorithms we mean multi-level, multi-stage type algorithms. Next we describe a new centralized multi-scale dead-reckoning (MSDR) algorithm which extends the utility of multi-scale force-directed algorithms to graphs with thousands of vertices, defined inside non-convex and even non-simple polygons. Finally, we describe a distributed version of our new approach.

1.1 Related Work

In the last decade the sensor localization problem has received a great deal of attention in the networks and wireless communities, due to the lowering of the production cost of miniature sensors and due to the numerous practical applications, such as environmental and natural habitat monitoring, smart rooms and robot control [Akyildiz et al. 2002]. Several recent approaches have exploited the natural connections with graph layout algorithms. Priyantha *et al.* [Priyantha et al. 2003] propose a new distributed anchor-free layout technique, based on force-directed methods. Gotsman and Koren [Gotsman and Koren 2004] utilize a stress majorization technique in their distributed method. Neither of these approaches assumes that angular information is available and as a consequence these algorithms need additional assumptions to achieve good results (both approaches assume that sensors are distributed in a simple convex polygon, and Priyantha *et al.* assume that the graph is rigid).

Most algorithms that do utilize angular information, assume that a fraction of the sensors is GPS-equipped. Doherty *et al.* [Doherty et al. 2001] formulate the sensor localization problem as a linear or semidefinite program based on both adjacency and angular information. Savvides *et al.* [Savvides et al. 2001] describe an ad-hoc localization system (AHLoS) which employs anchor-based algorithms for sensor localization using both edge length and angular information. Savarese *et al.* [Savarese et al. 2001] and Niculescu and Nath [Niculescu and Nath 2003] describe anchor-based algorithms for sensor localization utilizing edge lengths information. Fekete *et al.* [Fekete et al. 2004] use a combination of stochastic, topological, and geometric ideas for determining the structure of boundary nodes of the region, and the topology of the region. Basu *et al.* presented a localization algorithm that makes intensive use of angular information, but requires that all nodes are equipped with a compass, so they all "know" the direction to the absolute north [Basu et al. 2006]. Eren *et al.* [Eren et al. 2006] investigate the uniqueness of the localization from a global rigidity perspective when angular information is available.

Approaches based on Monte Carlo localization take roots from robotics localization. Such *range-free* approaches do not assume distance or angular information and utilize the mobility information gathered from the nodes [Hu and Evans 2004; Rudafshani and Datta 2007]. More recently signal processing type of approaches have been suggested for localization. Using the known track of a calibration target and a reference location, compressive sensing ideas have been employed to localize a node [Cevher and Baraniuk 2008]. A passive localization algorithm that constructs location estimates from a set of projected distances which are based on interpreting the time differences of a global event is presented in [Kwon and Agha 2008].

1.2 Our Contributions

We focus on centralized and distributed force-directed sensor localization algorithms for anchor-free networks by considering two variations of the problem: one in which the input contains (noisy) edge lengths information and the other in which the input also contains (noisy) angular information. We perform experiments by varying the sizes of the graphs, in terms of number of vertices and edge density (average vertex degree). We also consider different types of shapes for the region in which the sensors are distributed: simple convex polygons, simple non-convex polygons, and non-simple polygons. Finally, we measure two types of performance metrics: the global quality of the layout and the structure of the boundary of the region.

We describe one new force-directed technique and adapt several standard force-directed techniques to the centralized and distributed sensor localization problem. Two standard force-directed techniques are those of Fruchterman-Reingold [Fruchterman and Reingold 1991] and Kamada-Kawai [Kamada and Kawai 1989]. If we are only given adjacency information about the underlying graph, these algorithms fail to solve the sensor localization problem even for small graphs.

We show that incorporating the (noisy) edge lengths information to these classic force-directed methods works surprisingly well for graphs defined inside simple convex regions. Yet this simple extension is not sufficient to handle the issues of scalability and the non-simple, non-convex underlying network topologies.

We propose the multi-scale dead-reckoning (MSDR) algorithm to overcome these problems. The multi-scale approach has been shown to resolve the scalability problem in graph drawing, a slightly different context where layouts of large graphs are computed with no constraints imposed on the edge lengths [Gajer et al. 2004]. Our next extension therefore incorporates the multi-scale technique to the proposed force-directed localization method. However even this modification fails to reconstruct node positions in networks defined in non-simple or non-convex regions. With the aid of (noisy) angular information, we can extend the utility of multi-scale localization algorithm to large networks with complicated underlying regions. The angular information is incorporated to the suggested multi-scale extension via the *dead-reckoning* technique, a position estimation method for mobile objects known for centuries. We show that our new multi-scale dead-reckoning (MSDR) algorithm provides output layouts matching closely the source layouts under the defined similarity metrics, and is tolerant to non-trivial noise for large networks defined in non-simple and non-convex regions. Furthermore we provide a distributed version of the MSDR algorithm and show that the distributed version performs as good as, or in certain cases even better than its centralized counterpart.

2. ALGORITHMS, METRICS, AND EXPERIMENTS

In this section we briefly describe the algorithms we implemented, the metrics used to evaluate performance, and our experimental setup.

2.1 Algorithms

In order to see and compare the performance of force-directed localization algo-

rithms, we implemented and tested seven of them: Fruchterman-Reingold Algorithm (FR), Kamada-Kawai Algorithm (KK), Fruchterman-Reingold Range Algorithm (FRR), Kamada-Kawai Range Algorithm (KKR), Multi-Scale Kamada-Kawai Range Algorithm (MSKKR), Multi-Scale Dead-Reckoning Algorithm (MSDR) and Distributed Multi-Scale Dead-Reckoning Algorithm (D-MSDR). The first two utilize only the graph adjacency information. The next three utilize the graph adjacency information and the edge lengths (range) information. The last two algorithms utilize the graph adjacency information, the edge lengths (range) information and the angular information. Details about these algorithms are provided in the next section.

2.2 Metrics

We compare the performance of various algorithms on different underlying graphs, varying the number of vertices, edge density, as well as the types of regions in which the graphs are defined. We also vary the amount of error in both the edge length and angular information. We implemented six different metrics to capture the performance of the algorithms, some intended to measure the global quality of the layout and the others measuring the quality of the boundary. In this paper, we report the results using the *Frobenius* metrics for comparing the layouts globally and the *BAR* metric for comparing the quality of the boundary reconstruction.

The global quality metrics attempt to measure how the layout $\hat{D}$ created by a given algorithm matches the source layout D. In particular, the Frobenius metric [Golub and Van Loan 1996] is equivalent to the Frobenius norm of a matrix M whose entries are:

$$M_{ij} = \frac{\hat{d}_{ij} - d_{ij}}{n},$$

where n is the number of sensors, d_{ij} is the actual distance between sensors i and j in D, and $\hat{d}_{ij}$ is the distance between those sensors in the layout $\hat{D}$. Thus, we can measure the global quality of the layout[1] in terms of the Frobenius error:

$$FROB1 = \sqrt{\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\hat{d}_{ij} - d_{ij})^2}. \quad (1)$$

The *boundary alignment ratio* (BAR) is the sum-of-squares normalized error value of a boundary matching. Given the true layout D, we compute its boundary and then compute an approximation by taking a sample of the boundary points B.

[1]The *global energy ratio* (GER) defined by Priyantha *et al.* [Priyantha et al. 2003] is similar to the Frobenius metric:

$$GER = \frac{1}{n(n-1)/2}\sqrt{\sum_{i=1}^{n}\sum_{j=i+1}^{n}\left(\frac{\hat{d}_{ij} - d_{ij}}{d_{ij}}\right)^2}.$$

While appropriate for comparing the layouts obtained by different algorithms for graphs of the same size, the GER metric is not well-suited to compare the quality of the layout across different graph sizes.

We compute the same size sample $\hat{B}$ of the boundary of the layout $\hat{D}$ produced by our algorithm. We then apply the iterative closest point algorithm (ICP) [Besl and McKay 1992] to align the two boundaries using rotation and translation. The boundary alignment ratio is defined as:

$$BAR = \frac{\sum_{\hat{p}\in\hat{B}}(\hat{p}-p)^2}{|B|}. \quad (2)$$

The ICP algorithm first computes a match $\hat{p} \to p$ for each point $\hat{p} \in \hat{B}$, based on nearest neighbors. Next, the ICP algorithm aligns the two layouts D and $\hat{D}$ as well as possible using the BAR metric. This process of nearest-neighbor computation and alignment is repeated until the improvement in the BAR score becomes negligible.

2.3 Experiments

We have implemented all the algorithms and created a simulation environment to test them. The implementations and the data regarding all the experiments can be found in [Efrat et al.]. Since we did not have actual sensors to work with, we wrote a plugin for our graph visualization system, Graphael [Forrester et al. 2004], that simulates the placement of the sensors and the reported information from each. The sensor data generator takes the following parameters as input: number of sensors, average connectivity (density), type of the region to place the sensors in (square-shape, star-shape, etc.), range error, and angle error. All of the regions have the same area so that the size of the region does not affect the performance metric results.

The data generator fills the region with the given number of sensors randomly placed inside it. Then the distances between all pairs of sensors are computed so that the sensor range that will give the desired average connectivity can be determined. Finally, the sensors that are within the determined sensor range are connected and the distances between them are reported after the range error is incorporated into the actual distances. The range error specifies standard deviation (in percentage) about 100% of the true edge length using a Gaussian distribution.

Next we compute the angular information. Each sensor chooses a random direction to be called "north". Then, the sensor detects the clockwise angle from north that each of its neighbors are located at, and angle error is factored in. We then sort these edges by reported angle and generate a mapping from each edge to its next clockwise edge about the node, and store it with the angle to that edge. This procedure guarantees that although error may be present in the reported data, the sum of the reported angles between edges is equal to 360^o. Angle error specifies standard deviation (in degrees) about the actual angle from a sensor's declared "north" to an edge using a Gaussian distribution.

3. FORCE-DIRECTED ALGORITHMS FOR LOCALIZATION

Force-directed layout algorithms are some of the most flexible algorithms for calculating layouts of simple undirected graphs. Also known as spring embedders, such algorithms calculate the layout of a graph using only information contained within

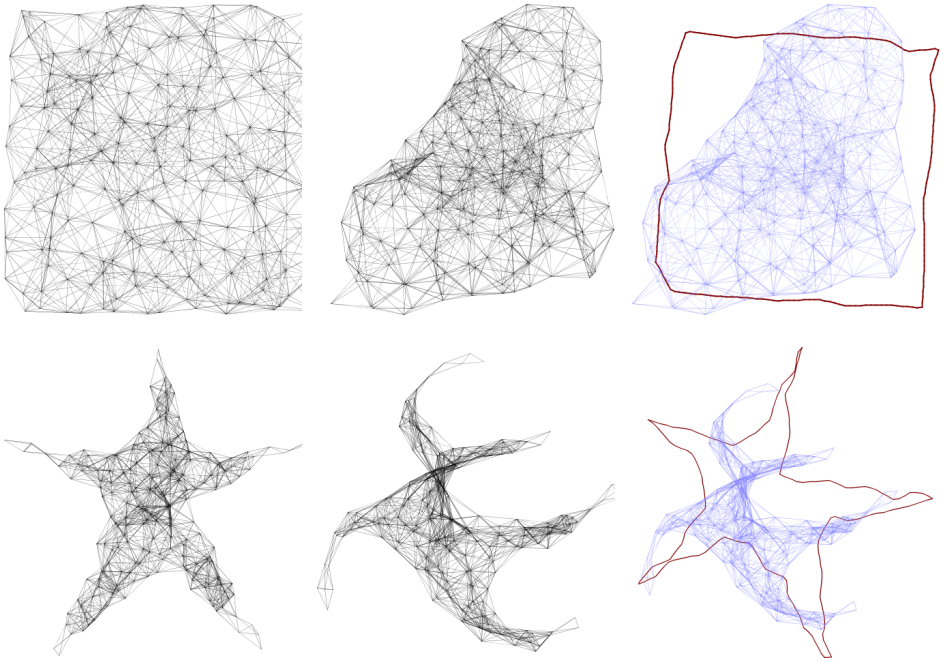

Fig. 1. Typical results illustrating input/output/boundary-alignment for KK (top) and FR (bottom) for graphs with 200 vertices inside square and star-shape regions, respectively.

the structure of the graph itself. In general, force-directed methods define an objective function which maps each graph layout into a number in $\mathcal{R}^+$ representing the energy of the layout. This function is defined in such a way that low energies correspond to layouts in which adjacent nodes are near some pre-specified distance from each other, and in which non-adjacent nodes are well-spaced. A layout for a graph is then calculated by finding a (often local) minimum of this objective function.

The Fruchterman-Reingold (FR) algorithm [Fruchterman and Reingold 1991] defines an attractive force function for adjacent vertices and a repulsive force function for non-adjacent vertices. The vertices in the layout are repeatedly moved according to this function until a low energy state is reached. FR, relies on *edgeLength*: the unweighted "ideal" distance between two adjacent vertices. The displacement of a vertex v of G is calculated by $F_{FR}(v) = F_{a,FR} + F_{r,FR}$, where:

$$F_{a,FR} = \sum_{u \in Adj(v)} \frac{\text{dist}_{R^n}(u,v)^2}{\text{edgeLength}^2}(\text{pos}[u] - \text{pos}[v]),$$

$$F_{r,FR} = \sum_{u \in Adj(v)} s \cdot \frac{\text{edgeLength}^2}{\text{dist}_{R^n}(u,v)^2} \cdot (\text{pos}[u] - \text{pos}[v]).$$

Alternatively, forces between the nodes can be computed based on their graph theoretic distances, determined by the lengths of shortest paths between them.

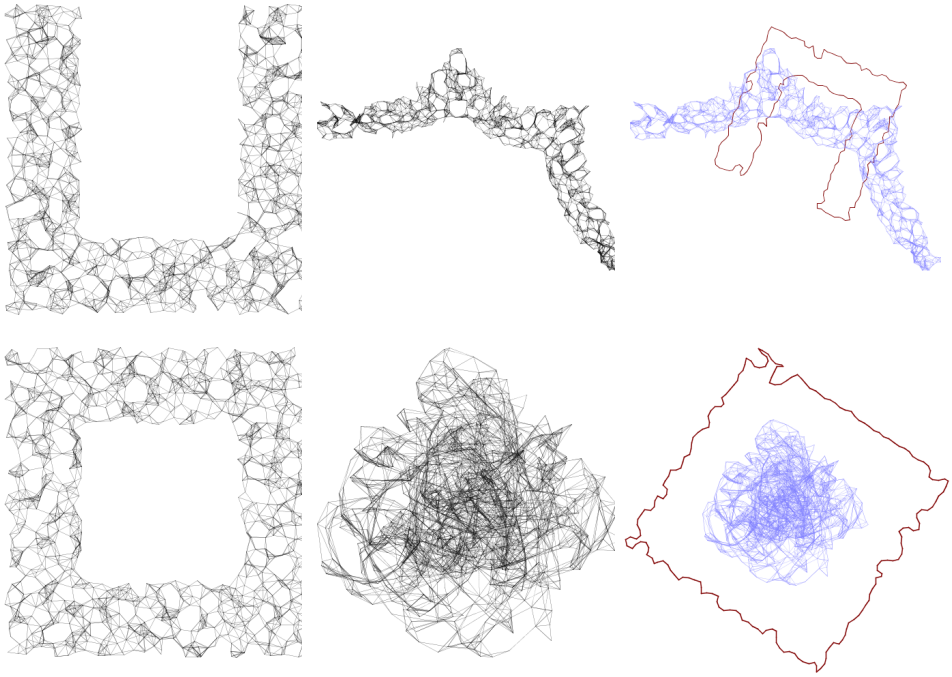

Fig. 2. Typical results illustrating input/output/boundary-alignment for KKR (top) and FRR (bottom) for graphs with 1000 vertices, density 8, range error 10%, angle error 10°, inside U-shape and rectangular donut-shape regions, respectively.

The Kamada-Kawai (KK) algorithm [Kamada and Kawai 1989] uses spring forces proportional to the graph theoretic distances. The displacement of a vertex v of G is calculated by $F_{KK}(v)$:

$$\sum_{\forall u \neq v} \left(\frac{\text{dist}_{R^n}(u,v)^2}{\text{dist}_G(u,v)^2 \cdot \text{edgeLength}^2} - 1 \right) (\text{pos}[u] - \text{pos}[v]).$$

Since neither FR, nor KK use the range information, the resulting layouts $\hat{D}$ are not of the same scale as the original graph layout D. We note that "scale" in this context refers to the edge lengths of the graph. Still, for small graphs (50-100 vertices) in simple underlying regions these algorithms often manage to reconstruct the underlying structure, as well as the boundaries. For larger graphs these algorithms exhibit the typical problems of fold-over and global distortion; see Fig. 1. To address the scale issue, we extend these algorithms to take into account the range information.

3.1 Range Extensions

In the range version of the Fruchterman-Reingold algorithm, FRR, the forces are defined by $F_{FRR}(v) = F_{a,FRR} + F_{r,FRR}$. The difference between the FR and FRR algorithms is in the definition of *edgeLength*. While in FR the ideal *edgeLength* was the same for all edges, in FRR *edgeLength* is different for different edges and

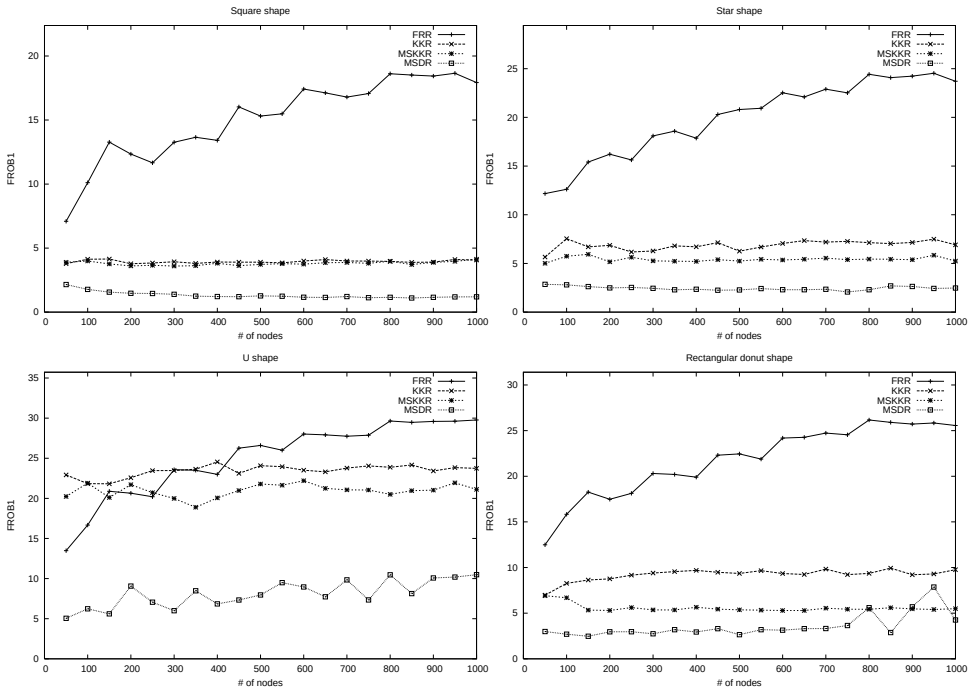

Fig. 3. Comparison between FRR, KKR, MSKKR, and MSDR algorithms measured by the Frobenius metric across square-shape, star-shape, U-shape, and rectangular donut-shape graphs with 50 to 1000 vertices. There were twenty trials per shape, using graphs with density 8, range error of 20% and angle error of 10°.

is defined by the reported distance between the corresponding pair of vertices. In a sensor network setup, this information comes from the range of the sensors and strength-of-signal or time-of-arrival data. In the range version of Kamada-Kawai, KKR, we incorporate the range data and use the weighted graph distance instead of the unweighted graph distance, $dist_G(u, v)$. Similar to KKR, the weight of the edges comes from the range of the sensors and strength-of-signal or time-of-arrival data.

FRR and KKR perform well on some graphs and not so well on others; see Fig. 2. FRR works well for small graphs of fifty to one hundred vertices, defined in simple convex shapes. However, larger graphs pose serious problems as FRR often settles in a local minimum. KKR, performs well on many large graphs, given enough iterations. Yet, KKR performs poorly on graphs defined in non-convex shapes. As we show in Section 4 the poor performance on non-convex shapes of algorithms based on the Kamada-Kawai approach can be addressed with the help of angular information.

3.2 Multi-Scale Extensions

One of the problems with the classic force-directed algorithms, such as Fruchterman-Reingold and Kamada-Kawai, is that they typically do not scale to larger graphs. One way to avoid this problem is to use multi-scale variants of these algorithms. In

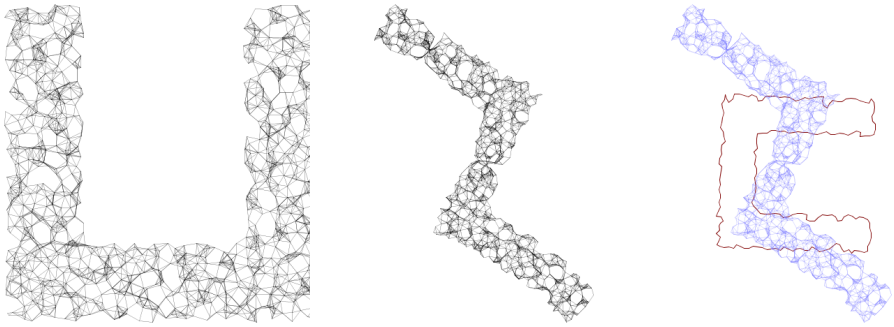

Fig. 4. A typical problem with graphs defined in non-convex shapes. Input/output/boundary-alignment for MSKKR for a graph with 1000 vertices, density 8, range error of 10% and angle error of 10°.

particular, multi-scale variants of the Kamada-Kawai algorithm have already been shown to produce good results in traditional graph drawing setting [Gajer et al. 2004; Harel and Koren 2002]. Our multi-scale algorithm, MSKKR, uses these ideas to extend the utility of KKR to larger graphs.

The MSKKR algorithm relies on *filtration of the vertices, intelligent placement, and multi-scale refinement*. Given $G = (V, E)$, we use a maximal independent set filtration $F: \ V = V_0 \supset V_1 \supset \ldots, \supset V_k \supset \emptyset$, such that each V_i is a maximal subset of V_{i-1} for which the graph distance between any pair of vertices is at least $2^{i-1}+1$. It is easy to see that given this definition $k = O(\log n)$.

The vertices in V_k are placed first, based on an estimate of their graph distances. Then the vertices in each successive set in the filtration are placed based on their graph distances from the vertices which have already been placed, followed by a refinement of the current layout. Details of this approach are discussed in [Gajer et al. 2004].

While the quality of the layouts obtained by KKR are comparable to those obtained by MSKKR, the multi-scale approach is much faster and offers a better chance of getting right some of the global details of the placement. As the charts in Fig. 3 indicate, MSKKR performs especially well for star-shapes and donut-shapes. The same figure indicates that just as KKR, MSKKR has problems with U-shape graphs that the next algorithm can address.

4. MULTI-SCALE DEAD-RECKONING ALGORITHM

The KK, KKR, and MSKKR algorithms use either the graph theoretical distance or a weighted version of this distance when the range data is taken into account. This approach provides layouts that typically match the underlying graphs. Non-convex underlying shapes, however, yield poor results even for MSKKR. This is a problem exhibited by all of the algorithms considered so far.

Consider the sensor network obtained by distributing sensors in a U-shape region. Both the Kamada-Kawai and Fruchterman-Reingold style algorithm would typically produce layouts in which the bends have been straightened; see Fig. 4. This is not a flaw of the algorithms but a byproduct of the way they compute the layouts as

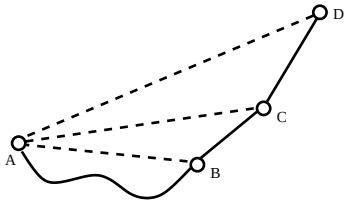

Fig. 5. In the BFS path from vertex A to D, the predecessor of D is C and the predecessor of C is B.

both of these algorithms attempt to place vertices whose graph distances are large, as far away from each other as possible. Pairs of vertices at the tips of the U-shape are at maximum graph distance from each other, but their Euclidean distance is small. Thus, to be able to reconstruct layouts of graphs defined in non-convex or non-simple regions, we need additional information. Most previous approaches rely on anchors (sensor nodes with GPS or manually-placed nodes) but they can be too costly, or may not work indoors, underground, or under thick tree-cover. Instead, angular information can be used with great effect to improve the quality of the layouts. With this in mind, we propose the multi-scale dead-reckoning (MSDR) algorithm.

4.1 Dead-Reckoning

Dead-reckoning, or deduced-reckoning, has been used for centuries as a method of estimating the current position of a moving object by applying the direction and distance traveled to a previously determined position [Krotkov et al. 1995]. It is a common method for calculating the position of a mobile robot, using the robot's measurements of traveled distance and turns made. Although the problem we are considering is a static problem, we can use this technique to obtain better estimates for the relative positions of two distant sensor nodes. Given range and angular information, we can compute the distance between vertices x and y using this idea. We call that distance $dr(x, y)$.

Suppose we want to calculate the dead-reckoning distance from vertex A to a vertex D; see Fig. 5. Let node C be D's predecessor in the shortest path from A to D, and let B be C's predecessor; see Fig. 5. Assume that $dr(A, B)$ and $dr(A, C)$ have already been calculated and that we also know the orientation of $\triangle BCA$. The $\angle BCD$ is also known since the angle between edges on node C is part of the source data, and the lengths of the edges from B to C and from C to D are known as well. To reduce the number of special cases, we convert this angle to a clockwise angle by negating it if it is counter-clockwise.

Ultimately, we want to calculate $\angle ACD$ so that we can determine $dr(A, D)$ via the law of cosines. To do this, we first compute $\angle BCA$ using the law of cosines: $dr(A, B)^2 = edge(B, C)^2 + dr(A, C)^2 - 2 \times edge(B, C) \times dr(A, C) \times cos(\angle BCA)$:

$$\angle BCA = cos^{-1}\left(\frac{edge(B, C)^2 + dr(A, C)^2 - dr(A, B)^2}{2 \times edge(B, C) \times dr(A, C)}\right)$$

To determine the clockwise angle $\angle ACD$, we must either add or subtract $\angle BCA$ to/from $\angle BCD$, depending on the orientation of $\triangle BCA$. If $\triangle BCA$ is clockwise, we simply add the two. If $\triangle BCA$ is counter-clockwise, then the angles overlap and we must therefore take their difference. Put another way, we can just convert $\angle BCA$ to a clockwise angle and add it to $\angle BCD$, then wrap it so that it is in the range $[0^o, 360^o)$.

Now we know the following useful information: $dr(A, C)$, $\angle ACD$, and $edge(C, D)$. Using the law of cosines again, we can compute the distance from A to D: $dr(A, D)^2 = dr(A, C)^2 + edge(C, D)^2 - 2 \times dr(A, C) \times edge(C, D) \times cos(\angle ACD)$. Although $\angle ACD$ may be over 180^o, the law of cosines still yields the proper DR distance (the law of cosines yields the same result for the clockwise angle which is greater than 180^o and the counter-clockwise angle which is less than 180^o). After the DR distance has been computed, we save the orientation of $\triangle ACD$ (determined by whether or not $\angle ACD$ is greater than 180^o) so that we can reference it when calculating the DR distance to further nodes.

There are two base cases that must be considered separately. For nodes adjacent to the starting node, the edge length is the DR distance and no further computation is necessary. For nodes that are 2 edges away from the starting node, $\angle ACD$ is already known and does not need to be calculated. Therefore, only the final law of cosines used in our algorithm needs to be applied to find $dr(A, D)$.

4.2 MSDR Performance

Putting together the dead-reckoning idea with the multi-scale range-based Kamada-Kawai algorithm, by using the angle information in dead-reckoning calculations and the range information in Kamada-Kawai layout calculations, results in our multi-scale dead-reckoning localization algorithm, MSDR.

With regards to a performance comparison in terms of layout qualities measured according to the defined similarity metrics, we note that a direct comparison of previously suggested angle-based approaches with the MSDR algorithm (or even with each other) is quite difficult as they all make different assumptions regarding the problem settings. Some of the algorithms assume an underlying network protocol or require that a certain fraction of nodes are anchors [Savvides et al. 2001]. Some report results only on small graphs [Doherty et al. 2001] and yet some others impose extra constraints on the available angular information such as a knowledge of the direction to the absolute north [Basu et al. 2006]. Therefore a common strategy in most of the related work has been to compare the performance of the proposed angle-based technique to an approach most similar in terms of the problem setting assumptions, but one that makes no use of angle information [Savvides et al. 2001; Basu et al. 2006]. The goal of such a comparison is to decide whether the extra costs of the angle measurement devices are justified by the layout quality gain achieved with the proposed methods employing the angular information. We adopt a similar strategy and show that the MSDR algorithm outperforms all of the algorithms discussed earlier in the paper, even when considerably large angle errors are assumed; see Fig. 3.

Layouts obtained with the MSDR algorithm using small angle and range errors often match near-perfectly the given source graphs; see Fig. 6. Comparing MSKKR to MSDR shows that MSDR with angle errors of less than 10° consistently performs

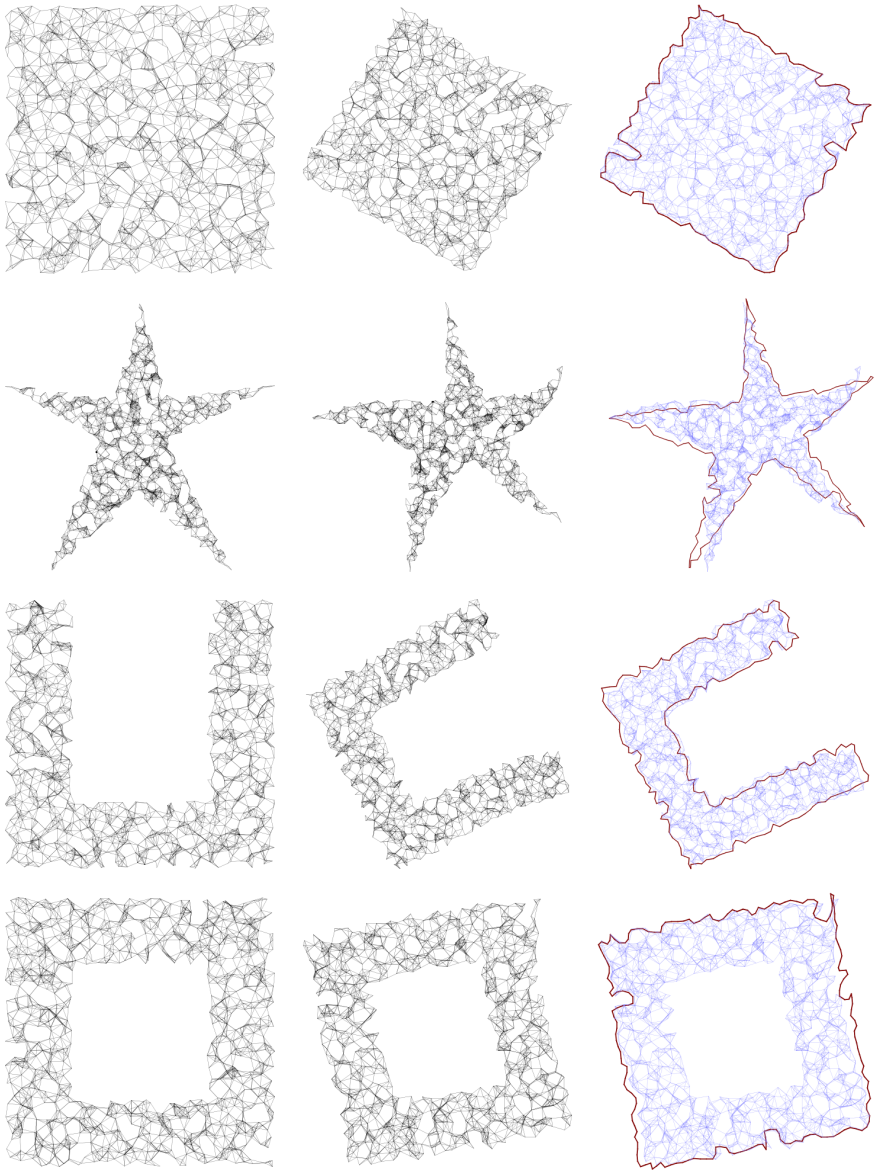

Fig. 6. Typical results illustrating input/output/boundary-alignment for MSDR on square-shape, star-shape, U-shape, and donut-shape graphs. The underlying graphs have 1000 vertices, density 8, range error of 10% and angle error of 10°.

better; see Fig 7. Since MSKKR does not depend on angle errors and is resilient to range-errors it produces stable results in most of the experiments, with the exception of the U-shape. MSDR's performance depends heavily on the angle errors and less on the range errors. For non-convex shapes such as the U-shape, MSDR offers significant advantages even with 50% range error and 25° angle error.

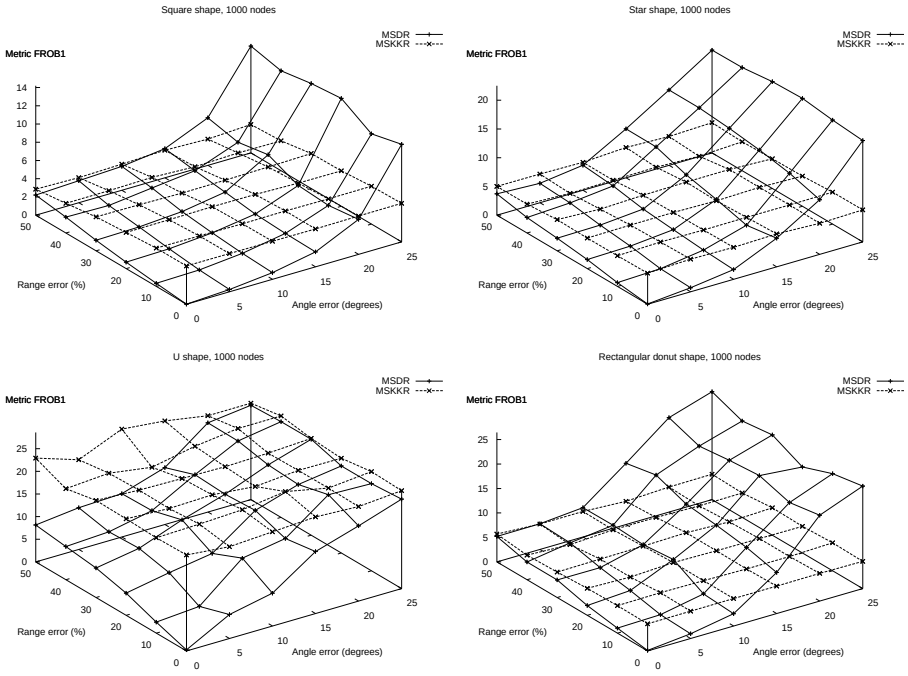

Fig. 7. Frobenius metric error tolerance for MSDR versus MSKKR across square-shape, star-shape, U-shape, and rectangular donut-shape graphs. There were twenty trials for each experiment using graphs with 1000 vertices, density 8 and varying the range and angle errors.

The quality of the layouts under varying range and angular errors is captured in Figs. 7-8. Under the Frobenius metric, the algorithm is stable for range errors of less than 30% and angular errors of less than 10°. As expected, the effect of angular errors is more pronounced; see Fig. 7. MSDR also captures the boundary of the underlying region very well. Experiments with the BAR metric also confirm that the MSDR is stable under range errors of up to 30%; see Fig. 8.

5. DISTRIBUTED MULTI-SCALE DEAD-RECKONING ALGORITHM(D-MSDR)

The presented MSDR algorithm is centralized by nature. However for most sensor network applications it is desirable to have distributed algorithms that execute on each sensor node. Since each node has a limited energy supply, the number of message exchanges should be constrained to an acceptably low amount. Next we extend the hierarchical filtration idea of the MSDR algorithm so that each node can execute it in a distributed manner. This way the number of message exchanges will drop significantly, providing a more efficient method. Our experiments indicate that the distributed algorithm not only provides a more energy-aware way of localization but compared to the centralized MSDR, but it also provides better localizations.

5.1 Filtration in D-MSDR

To achieve distributed filtration, every node applies a selection algorithm simulta-

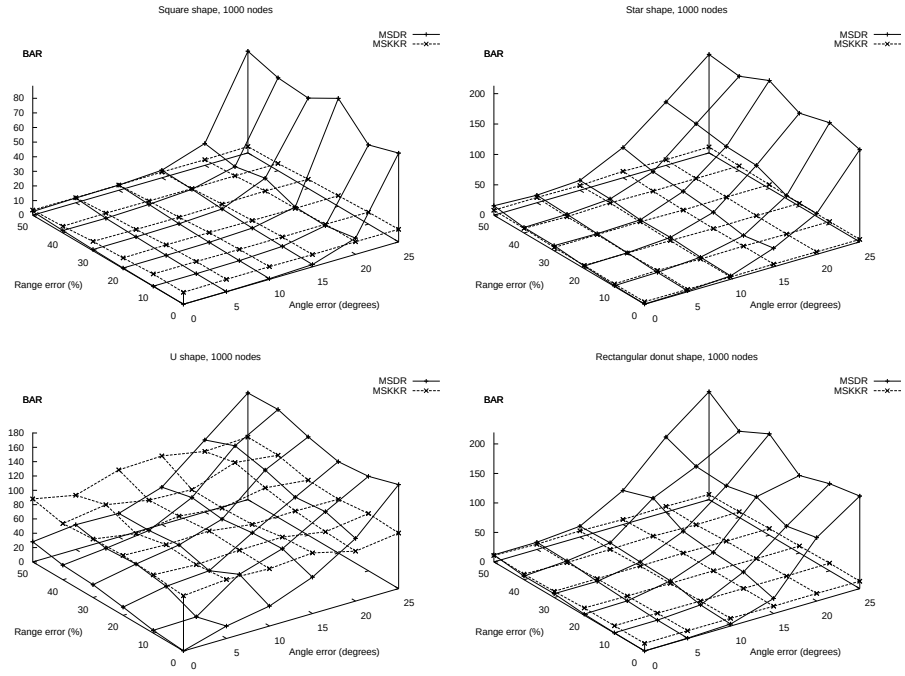

Fig. 8. BAR metric error tolerance for MSDR versus MSKKR across square-shape, star-shape, U-shape, and rectangular donut-shape graphs. There were twenty trials for each experiment using graphs with 1000 vertices, density 8 and varying the range and angle errors.

neously. Similar to the MSDR filtration, for G=(V,E), the distributed filtration results in $F: V = V_0 \supset V_1 \supset \ldots, \supset V_k \supset \emptyset$ such that V_i is a selected subset of V_{i-1}. We note that each node does not keep track of the complete set of selected nodes, but rather it just decides whether it has been selected for the current level.

Every node $u \in V_{i-1}$ checks its current neighborhood $N(u)_{i-1}$. Note that $N(u)_{i-1}$ consists of the nodes of V_{i-1} that are connected to u in the current filtration level. If none of the nodes in $N(u)_{i-1}$ is in V_i, the node selects itself to be in V_i with some probability p.

This selection algorithm is repeated k times for a particular level. Once the iterations are over, we run an additional round of the selection algorithm. However, in this last round, the nodes have the selection probability p=*1*, if they have no selected neighbors in V_i. This ensures that every node in V_{i-1} has at least one neighbor in V_i. A similar selection algorithm was used in [Katz and Wagner]. In our implementation we use $k = 10$. We set the selection probability $p = 0.05$.

5.2 Use of Dead-Reckoning in D-MSDR

If a node is selected to be in V_i, its neighborhood also may change. Each node $u \in V_i$ creates $N(u)_i$ as follows: A subgraph centered around u with radius r is created. Note that the nodes of this subgraph is a subset of V_{i-1} and the edges are determined according to the neighborhoods defined in the filtration level $i-1$.

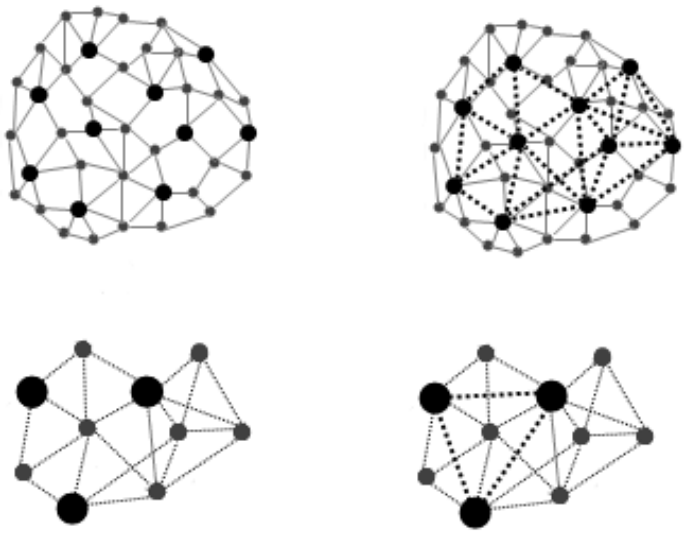

Fig. 9. Illustration of two consecutive filtration steps with $r = 3$. Left: Each dark node selects itself to be in the next level. Right: Every selected node performs dead reckoning in its subgraph and creates a new neighborhood by inserting new edges, shown with dashed segments.

Every node u runs dead reckoning on its subgraph and creates dead reckoning edges accordingly. Thus every node from this subgraph that belongs to V_i is now in $N(u)_i$; see Fig. 9. In our implementation we set $r = 3$.

5.3 Layout Computation in D-MSDR

Once the network reaches the last level of filtration V_k, each node localizes itself in a manner similar to the layout computation in MSKKR, going back to the first filtration level. However, there are two main differences. When a node localizes itself at some level i, it never computes localization again in lower levels $i-1$,$i-2$,...,0. This way the costs of communication and processing are reduced by preventing extra runs of the localization algorithm throughout many levels. Another difference is that each node runs the previously described force-directed localization algorithm, KK, only on its r-radius subgraph created for that level, instead of the whole graph of that level. This is also an important factor that limits node-to-node communication during localization. The communication is limited to only r-radius subgraphs. Here again, we set $r = 3$.

One problem with distributed localization is that, as the number of filtration levels increases, so do the distances between pairs of nodes in higher levels. Here the distances are defined in terms of the number of hops along the shortest path between the nodes in the original graph. This necessitates a solution to the routing problem for efficient message exchanges between pairs of nodes connected at

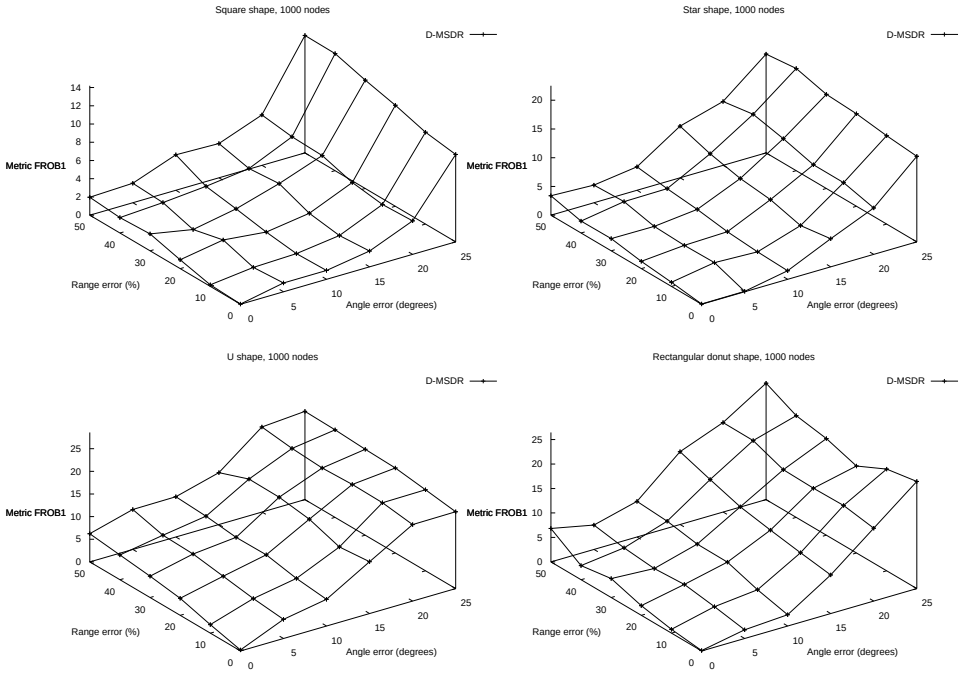

Fig. 10. Frobenius metric error tolerance for D-MSDR across square-shape, star-shape, U-shape, and donut-shape graphs. There were ten trials for each experiment using graphs with 1000 vertices, density 8 and varying the range and angle errors.

some filtration level. Since the distant nodes have dead-reckoning edges between them, the information contained in a dead-reckoning edge can be used to solve this problem. If a node sends a message to another node through a dead reckoning edge, the message will follow the edges from which we computed the particular dead-reckoning edge. Such a solution would resolve the issue for a low-level implementation. However, since our goal is to compare the effectiveness of alternative force-directed methods applied to the sensor localization problem at a higher level, we omitted such low-level issues in our implementation.

5.4 Performance of D-MSDR

5.4.1 *Layout Quality.* We implemented D-MSDR and conducted several experiments with settings similar to those of MSDR.

The output layouts of D-MSDR are nearly perfect for angle errors under 15° and they are reasonable for angle errors between 15° and 30°. Under 50% range errors the difference between the output layout and the correct layout is quite small, considering the BAR metric. Similarly, considering the Frobenius metric, a reasonable output layout is obtainable for range errors under 30%; see Fig. 10 and Fig. 11.

Compared to MSDR, the distributed version D-MSDR performed quite well. Under small angle and range error values, the output layout quality of D-MSDR is almost the same as that of MSDR. However, when we have angle and range

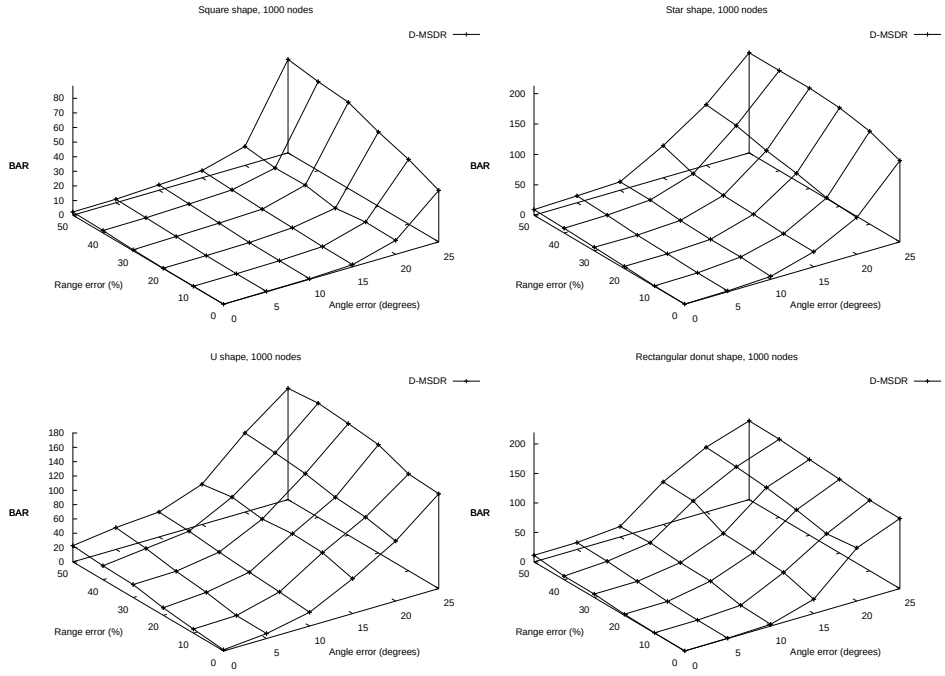

Fig. 11. BAR metric error tolerance for D-MSDR across square-shape, star-shape, U-shape, and donut-shape graphs. There were ten trials for each experiments using graphs with 1000 vertices, density 8 and varying the range and angle errors.

error rates greater than 20° and 20% respectively, D-MSDR outperforms MSDR in terms of layout quality. This is demonstrated in Fig. 12 where the output layouts of MSDR and D-MSDR are shown for networks with two different shapes. Each network has 1000 nodes with density 8. D-MSDR provides a layout almost the same as the original under 20° angle and 15% range errors, whereas the layout quality of MSDR is not as good.

In the centralized version of the algorithm, when there is a considerable amount of angle/range error in some part of the sensor network, the layout of the rest of the network is also affected by this error. This is because the centralized algorithm assumes data regarding the whole network is available and reflects the use of this data in the making of the final global output. Although the use of such global data creates an advantage for cases with small errors, when large errors are introduced, these errors are also not localized and are shared with the rest of the network together with useful data. In the distributed version, however, the local angle/range errors remain local and affect only parts of the network. A detailed comparison between the two methods is shown in Fig. 13 and Fig. 14. With small range/angle error values the output quality of both algorithms almost overlap. However the quality of MSDR degrades sharply starting from 15° angle error as the measurement error increases under both error BAR and FROB error measures, whereas D-MSDR, although providing a degrading quality with increasing error, degrades more gracefully.

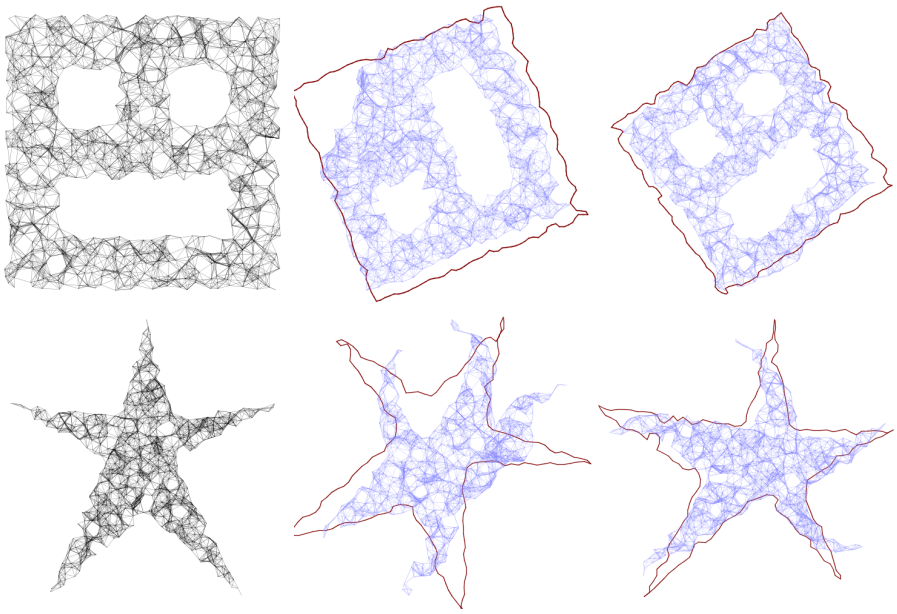

Fig. 12. Comparison of MSDR and D-MSDR for two different shapes. The measurement errors are the same for both MSDR and D-MSDR. **Left:** Original layout of the network **Middle:** MSDR output-boundary alignment **Right:** D-MSDR output-boundary alignment.

5.4.2 *Communication Costs.* It is difficult to evaluate the communication cost of D-MSDR because of its dependence on many attributes. Available bandwidth, synchronization, the techniques employed for the necessary measurements may all affect such an evaluation. More importantly we do not assume a specific routing protocol that manages node to node communication between multihop neighbors as the main focus is to analyze the force-directed approaches to sensor localization in a high-level context.

D-MSDR includes two steps where inter-sensor messaging occurs, namely the filtration and layout computations. Within the filtration phase, when going from level $i-1$ to i two separate processes give rise to inter-node communications. One consists of the selection process of a node $u \in V_{i-1}$ to be in V_i and the other is the construction of $N(u)_i$ for $u \in V_i$. During the selection process the selection probability, denoted with p, and the maximum number of iterations before a committed selection of probability 1, denoted with k, are chosen to provide a high probability (≈ 1) for selecting at least one node from the set $N(u)_i \cup \{u\}$. The number of inter-node communications is $|V_{i-1}| \times d_{i-1}$ where d_{i-1} indicates average degree at level $i-1$. Assuming a constant initial average degree, the average degree remains constant throughout the filtration levels. The number of filtration levels is $O(\log D)$ where D is the diameter of the network, that is the number of hops between the two furthest (in terms of the number of hops) nodes in the network. For the network sizes considered in this paper, the number of filtration levels is a small constant. Therefore the number of inter-node communications throughout the selection processes of the filtration phase is $O(n)$; see Fig. 15 for an experimental

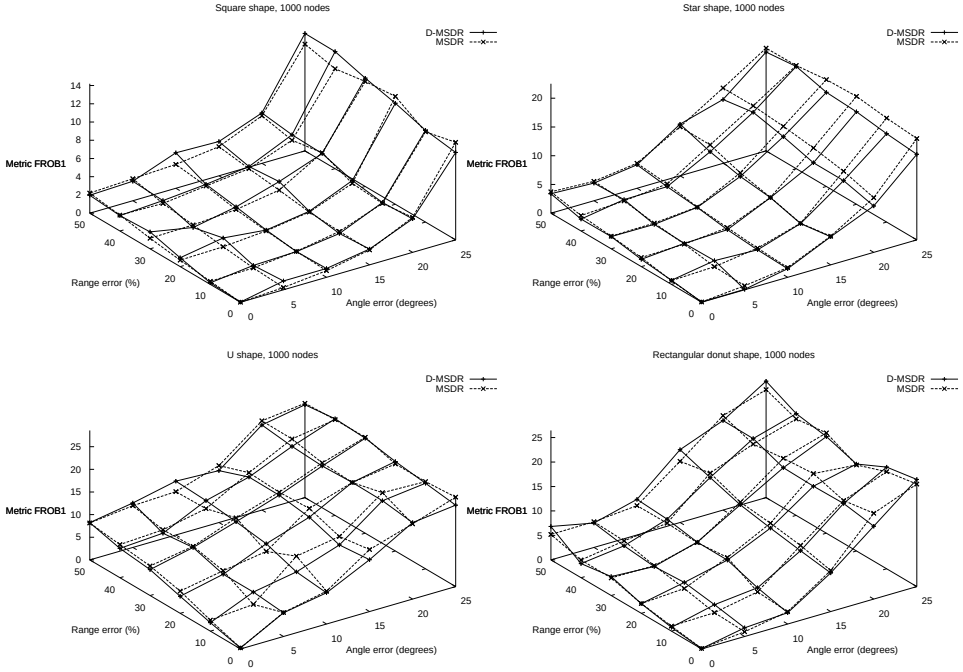

Fig. 13. Comparison of D-MSDR and MSDR measured by the Frobenius metric across four different shapes with 50 to 1000 vertices. There were ten trials per shape, using graphs with density 8 and range errors 0-50% and angle error $0^\circ - 25^\circ$.

plot of the estimated communication cost in terms of the total number of node-to-node inter-sensor messages.

Note that this analysis reflects only the number of node-to-node communication initiations. The specific routing protocol employed for this type of a communication and the actual implementation play an important role that is not reflected within this analysis. Both the neighborhood construction within the filtration and the localization phase incur similar communication costs.

6. ANCHOR EXTENSIONS

D-MSDR can be generalized to run on both anchor-free and anchor-based sensor networks. We consider extensions to D-MSDR that make use of the anchor information if available. For this purpose we modify the filtration and the layout phases of D-MSDR.

In the filtration phase, we change the selection probability of an anchor node to *p=1*. This way, the anchor nodes are always selected for the following filtration levels, finally reaching to the top level. Since the position of an anchor node is already available (either via global positioning devices or manual placement) an anchor node does not need to localize itself. Therefore, in the layout phase, we modify the localization algorithm so that no positions are computed for the anchor nodes.

We conducted additional experiments to measure the impact of anchor nodes on

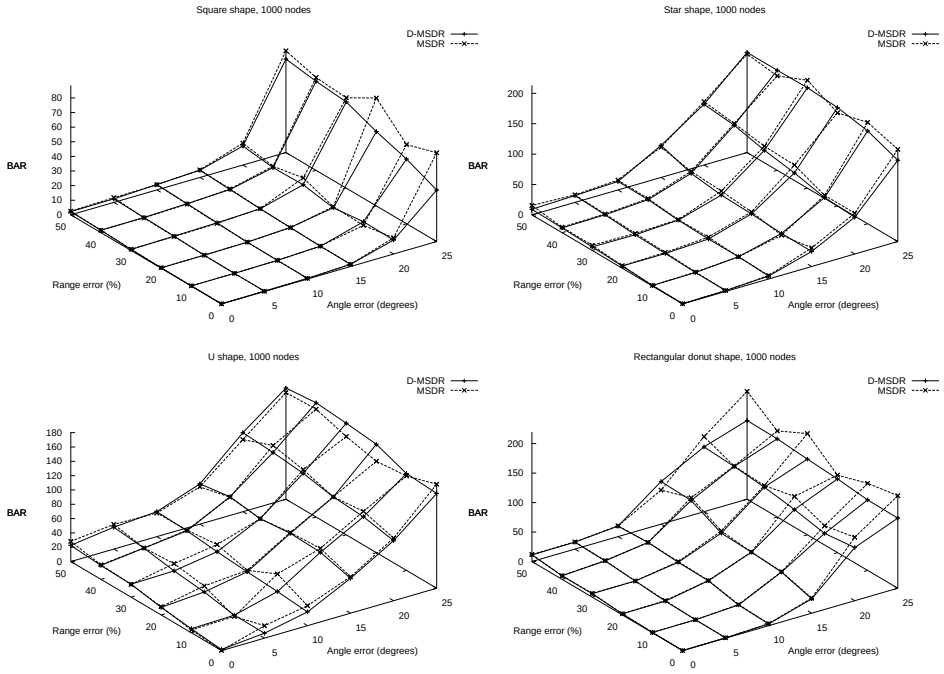

Fig. 14. Comparison between D-MSDR and MSDR measured by the BAR metric across square-shapes, star-shapes, U-shapes, and donut-shapes with 50 to 1000 vertices. There were ten trials per shape, using graphs with density 8 and range errors 0-50% and angle error $0^\circ - 25^\circ$.

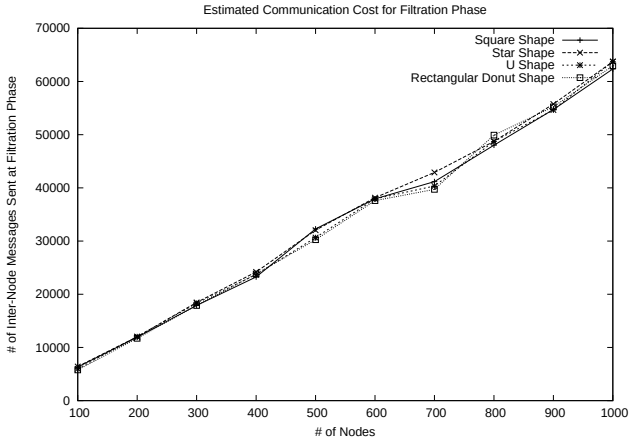

Fig. 15. Estimated communication cost for the selection processes in the filtration phase of D-MSDR in terms of the total number of inter-node messages sent across square-shape, star-shape, U-shape, and rectangular donut-shape graphs.

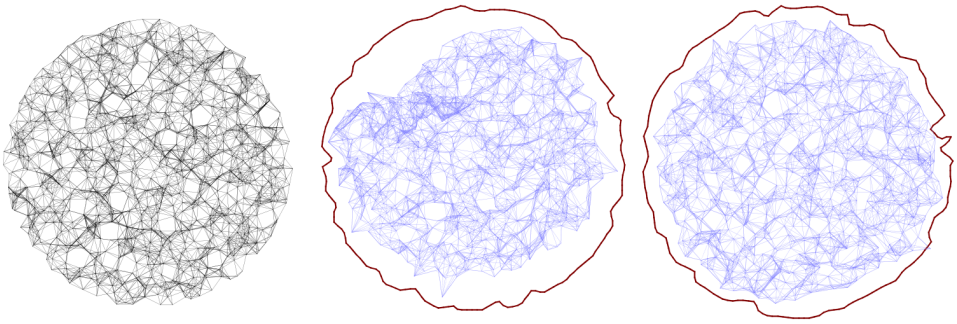

Fig. 16. The comparison of D-MSDR with anchor nodes involved. **Left:** The original layout **Middle:** D-MSDR output with no anchors **Right:** D-MSDR output when 5% of the nodes are anchors.

the final layout quality. Fig. 16 depicts a network with 1000 nodes and density 8. The angle/range errors are assumed to be 30°, 20% respectively. D-MSDR output layout almost matches the original layout even when only 5% of the nodes are assumed to be anchors, whereas D-MSDR performs poorly under the same measurement error rates, when the network is anchor-free. The results of this comparison under the BAR and FROB metrics can be seen in more detail in Fig. 17.

It might be of interest to determine the thresholds where increasing the number of anchors in a given network does not introduce any noticeable gain in the output quality. For example, Fig. 18 shows that for a circle-shaped network under 30° angle error and 20% range error, increasing the number of anchors beyond 30% does not provide a noticeable gain.

7. CONCLUSIONS AND FUTURE WORK

We presented several adaptations of force-directed graph layout algorithms for the sensor network localization problem under centralized/distributed models of computation. We also presented a new approach that takes advantage of angular information, based on dead-reckoning and multi-scale techniques. Our results indicate that incorporating angular information can significantly improve the performance of force-directed sensor localization. We also note that for relatively large measurement errors, the distributed model of computation provides better results than its centralized counterpart. All of these algorithms as well as the simulation that generates the data have been implemented as a part of the Graphael [Forrester et al. 2004] system.

8. ACKNOWLEDGEMENTS

We would like to thank Ahmet Ardal for his help in the implementation of various experiments.

REFERENCES

Akyildiz, I. F., Weilian, S., Sankarasubramaniam, Y., and Cayirci, E. E. 2002. A survey on sensor networks. *IEEE Communications Magazine 40,* 8, 102–114.

Basu, A., Gao, J., Mitchell, J. S. B., and Sabhnani, G. 2006. Distributed localization using

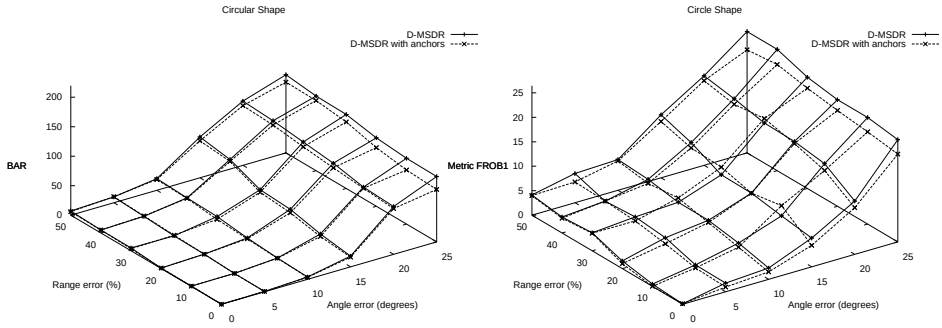

Fig. 17. The plot comparing the BAR and Frobenius values of D-MSDR and D-MSDR with anchors. The network consists of 1000 nodes with density 8 under 30° angle a nd 20% range error. The percentage of the anchors is 5%.

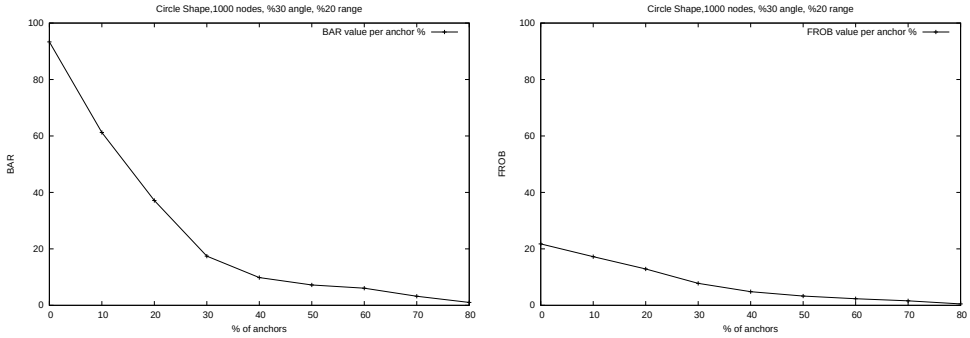

Fig. 18. Figure showing changing BAR and Frobenius error rates with the percentage of the anchor nodes in the sensor network. Experiments were done with a network of 1000 sensor nodes with density 8, 30° angle error and 20% range error.

noisy distance and angle information. In *MobiHoc '06: Proc. 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. 262–273.

Besl, P. J. and McKay, N. D. 1992. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14,* 2 (Feb.), 239–258.

Cevher, V. and Baraniuk, R. 2008. Compressive sensing for sensor calibration. *Sensor Array and Multichannel Signal Processing Workshop, 2008. SAM 2008. 5th IEEE*, 175–178.

Doherty, L., Pister, K., and Ghaoui, L. E. 2001. Convex optimization methods for sensor node position estimation. In *Proceedings of the 20th IEEE Computer and Communications Societies (INFOCOM-01)*. 1655–1663.

Efrat, A., Erten, C., Forrester, D., Iyer, A., Kilic, Y., and Kobourov, S. Force-directed approaches to sensor localization, implementation at `http://hacivat.khas.edu.tr/~cesim/sensorloc.html`.

Eren, T., Whiteley, W., and Belhumeur, P. N. 2006. Using angle of arrival (bearing) information in network localization. In *IEEE CDC '06: Proceedings of the 45th IEEE Conference on Decision and Control*. 4676–4681.

Fekete, S. P., Kröller, A., Pfisterer, D., Fischer, S., and Buschmann, C. 2004. Neighborhood-based topology recognition in sensor networks. In *ALGOSENSORS*. Lecture Notes in Computer Science, vol. 3121. Springer, 123–136.

Forrester, D., Kobourov, S. G., Navabi, A., Wampler, K., and Yee, G. 2004. graphael: A

system for generalized force-directed layouts. In *12th Symposium on Graph Drawing (GD)*. 454–466.

Fruchterman, T. and Reingold, E. 1991. Graph drawing by force-directed placement. *Software – Practice and Experience 21,* 11, 1129–1164.

Gajer, P., Goodrich, M. T., and Kobourov, S. G. 2004. A fast multi-dimensional algorithm for drawing large graphs. *Computational Geometry: Theory and Applications 29,* 1, 3–18.

Golub, G. H. and Van Loan, C. F. 1996. *Matrix Computations*. Johns Hopkins Press, Baltimore, MD.

Gotsman, C. and Koren, Y. 2004. Distributed graph layout for sensor networks. In *12th Symposium on Graph Drawing (GD)*. 273–284.

Harel, D. and Koren, Y. 2002. A fast multi-scale method for drawing large graphs. *Journal of Graph Algorithms and Applications 6*, 179–202.

Hu, L. and Evans, D. 2004. Localization for mobile sensor networks. In *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*. 45–57.

Kamada, T. and Kawai, S. 1989. An algorithm for drawing general undirected graphs. *Information Processing Letters 31*, 7–15.

Katz, B. and Wagner, D. Multi-scale anchor-free distribution positioning in sensor networks.

Krotkov, E., Hebert, M., and Simmons, R. 1995. Stereo perception and dead reckoning for a prototype lunar rover. *Autonomous Robots 2,* 4, 313–331.

Kwon, Y. and Agha, G. 2008. Passive localization: Large size sensor network localization based on environmental events. *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, 3–14.

Mauve, M., Widmer, J., and Hartenstein, H. 2001. A Survey on Position-Based Routing in Mobile Ad-Hoc Networks. 30–39.

Niculescu, D. and Nath, B. 2003. Ad hoc positioning system (APS) using AOA. In *Proceedings of the 22 Conference of the IEEE Computer and Communications Societies (INFOCOM-03)*. 1734–1743.

Priyantha, N. B., Balakrishnan, H., Demaine, E., and Teller, S. 2003. Anchor-free distributed localization in sensor networks. In *1st International Conference on Embedded Networked Sensor Systems (SenSys-03)*. 340–341. Also *TR #892, MIT LCS, 2003*.

Rudafshani, M. and Datta, S. 2007. Localization in wireless sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*. 51–60.

Savarese, C., Beutel, J., and Rabaey, J. 2001. Locationing in distributed ad-hoc wireless sensor networks. In *Proceedings of the 2001 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2037–2040.

Savvides, A., Han, C., and Srivastava, M. 2001. Dynamic Fine-Grained localization in Ad-Hoc networks of sensors. In *Proceedings of the 7th Conference on Mobile Computing and Networking (MOBICOM-01)*. 166–179.