A Force-Directed Approach to Sensor Localization*
(System Demo)

Anand Iyer, Alon Efrat, Cesim Erten, David Forrester, and
Stephen G. Kobourov

Department of Computer Science
University of Arizona
{anand,alon,cesim,forrestd,kobourov}@cs.arizona.edu

Abstract. We consider the centralized, anchor-free sensor localization problem. We propose using classic and new force-directed techniques, depending on the underlying sensor network size and geometry. We consider the case where the sensor network reports range information and also the case where in addition to the range, we also have angular information about the relative order of each sensor's neighbors. In particular, we describe a multi-scale dead-reckoning algorithm that scales well for large networks, is resilient under range errors and can reconstruct complex underlying regions.

1 Introduction

Wireless sensor networks are used in many applications, from natural habitat monitoring to earthquake detection. Often, the actual location of the sensors is not know but is necessary for the underlying application [1], e.g., determining the epicenter of the quake. Moreover, the location of the sensors can be used to design efficient network routing algorithms [12].

The sensor localization problem can be thought of as a graph drawing problem. The true state of the underlying sensor network is captured by a source graph drawing D of the graph G. Given the adjacency information for G, together with possibly some additional noisy information (edge lengths, or angles between adjacent neighbors) we would like to construct a drawing $\hat{D}$ which matches D as best as possible. There are many variations of the problem, depending on the quality of the edge length data (obtained using signal strength), or whether some of the vertices know their exact location (GPS-equipped sensors), or whether the vertices can detect the relative order of their neighbors (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. Often, the strength or the time of arrival of the signal can be used to estimate the actual distance between two sensors. However, sensing neighbors is not perfect, especially close to the limits.

Sensors with GPS are often called *anchors* and while they make the localization problem easier, they are bulky and expensive. Anchor-free sensor networks and more practical but pose greater challenges in localization.

Sensors 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 far from accurate but even allowing for some errors, angular information can be used to find good localizations.

* This work is supported in part by an ACIST grant and an NSF grant ACR-0222920.

In this paper we focus on the centralized sensor localization problem for anchor-free networks. We consider the cases with or without angular information. We also consider 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. For larger graphs, the simple force-directed algorithms fail to reconstruct the vertex locations. Multi-scale versions of the force-directed algorithms extend the utility of these algorithms to graphs with hundreds of vertices, provided that the graphs are defined inside simple convex polygons. Using the angular information, we can extend the utility of these algorithms to graphs with thousands of vertices, defined inside non-convex and even non-simple polygons.

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 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 [1]. However, only a handful of research papers on this topic have exploited the natural connections with graph layout algorithms: Priyantha *et al* [13] survey the existing sensor localization algorithms and propose a new distributed anchor-free layout technique, based on force-directed methods. Gotsman and Koren [9] utilize a stress majorization technique in their distributed method. Neither of these approaches assumes angular information.

Most of the algorithms that do utilize angular information, also assume that a fraction of the sensors is GPS-equipped. Doherty *et al* [3] formulate the sensor localization problem as a linear or semidefinite program based on both adjacency and angular information. Savvides *et al* [15] describe an ad-hoc localization system (AHLoS) which employs an anchor-based algorithms for sensor localization using both edge length and angular information. Savarese *et al* [14] and Niculescu and Nath [4] describe anchor-based algorithms for sensor localization utilizing edge lengths information. Fekete *et al* [5] 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.

1.2 Our Contributions

We focus on centralized force-directed sensor localization algorithms for anchor-free networks. We consider two variations of the problem: one in which the input contains (noisy) edge lengths information and the other in which we 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. We also consider different geometries for the underlying graph: simple convex polygons, simple non-convex polygons. Finally, we measure two types of performance metrics: the global quality of the layout and the structure of the boundary of the region.

The force directed algorithms are based on the Fruchterman-Reingold [7] and the Kamada-Kawai [11] methods. If we are only given adjacency information about the underlying graph, these algorithms fail to solve the sensor localization problem even for small graphs. Incorporating the (noisy) edge lengths information works surprisingly well for graphs defined inside simple convex regions. For larger graphs, the multi-scale graph layout algorithms [8] perform better. However, even these techniques fail to reconstruct graphs defined in non-simple, or non-convex regions.

We augment these approaches to take advantage of angular information. With the aid of (noisy) angular information, we can extend the utility of multi-scale graph layout algorithms to large graphs with complicated underlying regions. In particular, our proposed *multi-scale dead-reckoning* algorithm performs well and is tolerant to non-trivial noise in the edge length and angular information.

2 Background

2.1 Algorithms

We implemented and tested six force-directed algorithms: 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) and Multi-Scale Dead-Reckoning Algorithm (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 algorithm utilizes 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 use two metrics to capture the performance of the algorithms, one intended to measure the global quality of the layout and the other measuring the quality of the boundary.

The first metric, the *global energy ratio* is the root-mean-square normalized error value of the point-to-point distances, as defined by Priyantha *et al* [13]:

$$GER = \frac{\sqrt{\sum_{i<j} \hat{e_{ij}^2}}}{n(n-1)/2},$$

where $\hat{e_{ij}} = \frac{\hat{d_{ij}} - d_{ij}}{d_{ij}}$ is the normalized error of the difference between the true distance d_{ij} in D and the distance in the graph layout $\hat{d_{ij}}$ in $\hat{D}$. This metric attempts to measure the global quality of the layout, by considering the distances between all pairs of sensors in D and $\hat{D}$.

The second metric, the *boundary alignment ratio* is the sum-of-squares normalized error value of a boundary matching. Given the true drawing D, we compute its boundary and then compute an approximation by taking a sample of the boundary points B. We compute the same size sample $\hat{B}$ of the boundary of the drawing $\hat{D}$ produced by our algorithm. We then apply the iterative closest point algorithm (ICP) [2] to align the two boundaries using rotation and translation. The boundary alignment ratio is defined as:

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

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 drawings D and $\hat{D}$ as best 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. In order to avoid local minima we run the ICP algorithm from several different initial alignments.

2.3 Experiments

Since we did not have actual sensors to work with, we wrote a plugin for our graph drawing system, Graphael [6], that simulates the placement of the sensors and the reported information from each. Our sensor data generator takes the following parameters as input: number of sensors, average connectivity (density), region to place the sensors in (square-shape, triangle-shape, star-shape, etc.), range error, and angle error. All of our regions have the same area so that the size of the region does not affect the performance metric results.

Our data generator fills the region with the given number of sensors placed at random inside it. Then the distances between all pairs of sensors are computed so that we can determine the sensor range that will give us the desired average connectivity. Finally, we connect the sensors that are within the determined sensor range and report the distance between them after incorporating the range error into the actual distances. The range error is specified as a percentage of the actual range. A 10% range error between sensors that have an actual distance of r implies that the reported range is within $[r - 0.1 \cdot r,\, r + 0.1 \cdot r]$.

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 with it 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 in our experiments is specified as a percentage of 360^o. For example, an angle error of 25% implies that the angle from north reported by a sensor might be up to 45^o on either side of the actual angle.

3 Force-Directed Algorithms for Localization

The FR algorithm defines an attractive force function for adjacent vertices and a repulsive force function for non-adjacent vertices. FR, relies on edgeLenth: the unweighted "ideal" distance between two adjacent vertices. For a vertex v, $F_{FR}(v) = F_{a,FR} + F_{r,FR}$, where the attractive force is defined as

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

and the repulsive force is defined as

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

In the KK algorithm each pair of vertices connected by a path has forces proportional to the length of the path. The displacement of a vertex v of G is calculated by:

$$F_{KK}(v) = \sum_{u \in N_i(v)} \left(\frac{\mathrm{dist}_{R^n}(u,v)^2}{\mathrm{dist}_G(u,v) \cdot \mathrm{edgeLength}^2} - 1 \right) (\mathrm{pos}[u] - \mathrm{pos}[v])$$

Since neither FR, nor KK use the range information, the resulting layouts $\hat{D}$ are not of the same scale as the original graph drawing D. However, these algorithms often manage to reconstruct the underlying structure as well as the boundaries. To address the scale issue, we extend these algorithms to take into account the range information; see Fig. 1.

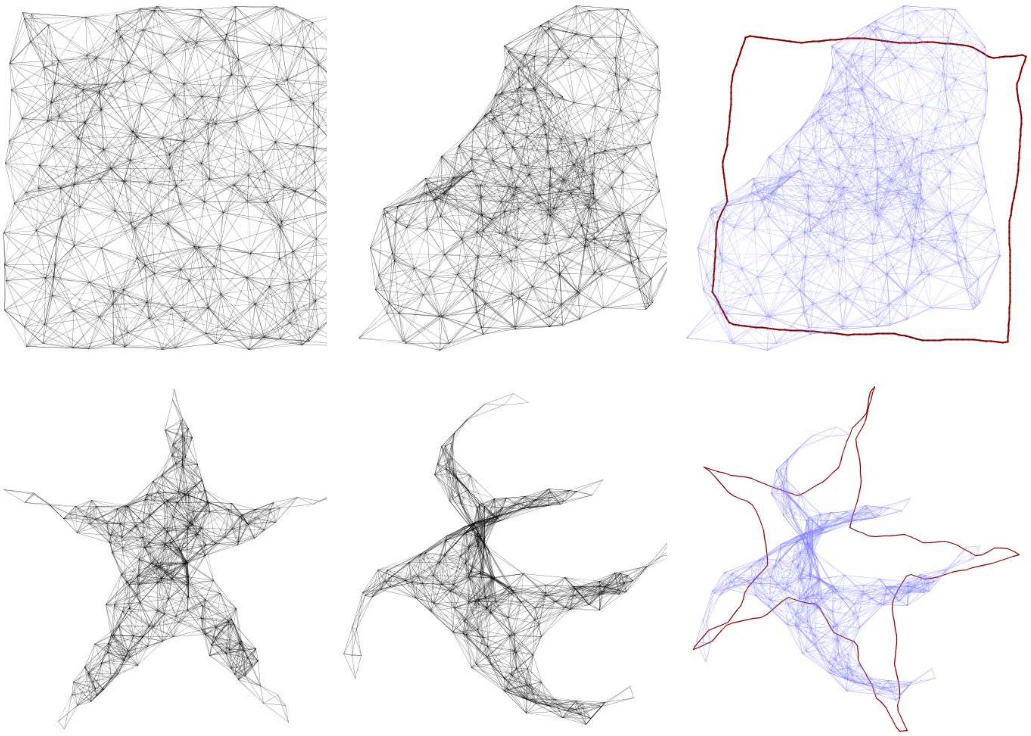


Fig. 1. Typical input/output pairs illustrating input/output/boundary for KK (top) and FR (bottom).

3.1 Range Extensions to Classic Force-Directed Algorithms

In 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. In FRR, edgeLength is different for different edges and is given by the reported distance between the corresponding pair of vertices. A similar result can be achieved by replacing the ideal edgeLength in the original formulation of FR by a factor equal to the average distance between two vertices connected by an edge (which can be easily computed from the range data).

In the range version of Kamada-Kawai, KKR, the forces are defined as follows:

$$F_{KK}(v) = \sum_{u \in N_i(v)} \left(\frac{\mathrm{dist}_{R^n}(u,v)^2}{\mathrm{dist}_P(u,v) \cdot \mathrm{edgeLength}^2} - 1 \right) (\mathrm{pos}[u] - \mathrm{pos}[v]),$$

where $\mathrm{dist}_P(u,v)$ is the weighted distance along the BFS path p from u to v. Again, a similar result can be achieved by using the average edgeLength, computed from the range data.

FRR and KKR work surprisingly well for small graph of fifty or so vertices, defined in simple convex shapes. However, non-convex shapes, larger graph sizes and low density can result in very poor layouts; see Fig. 2.

3.2 Multi-Scale Extensions

One of the problems with the classic force-directed algorithms 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 particular, multi-scale variants of the Kamada-Kawai algorithm have

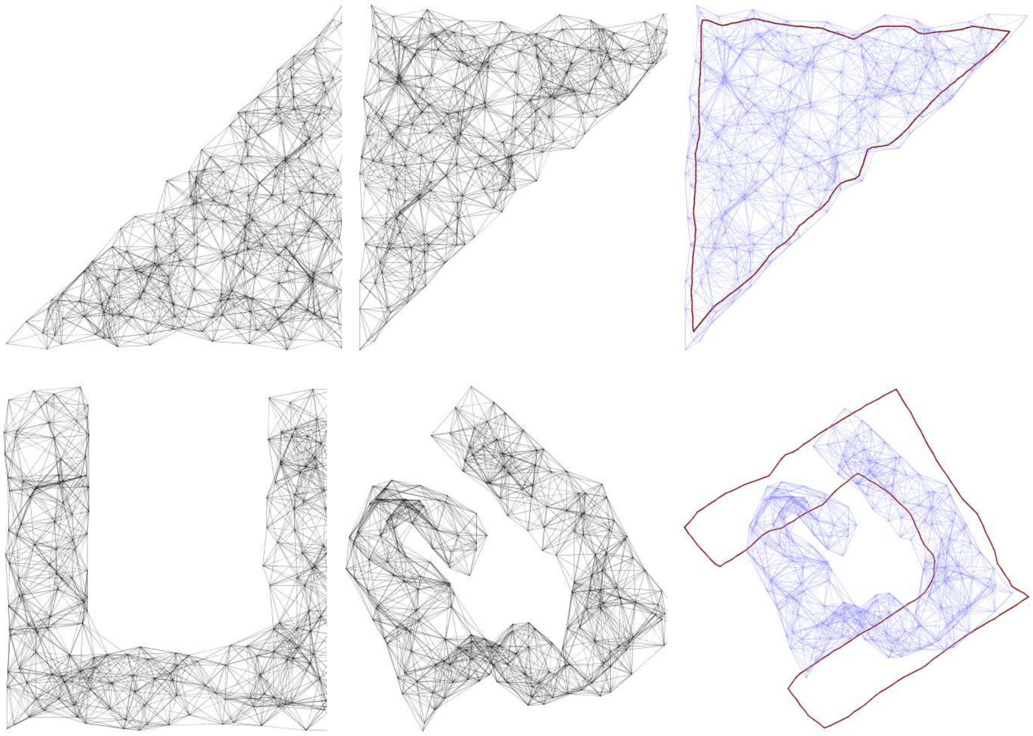


Fig. 2. Typical input/output pairs illustrating input/output/boundary for KKR (top) and FRR (bottom).

already been shown to produce good results in traditional graph drawing setting [8, 10]. The multi-scale algorithm, MSKKR, uses these ideas to extend the utility of KKR to larger graphs.

The MSKKR is an adaptation on the GRIP algorithm [8] to the sensor layout problem. The algorithm computes a filtration of the vertices of the graph into a logarithmic number of progressively smaller sets. The vertices of each set in the filtration are placed by using the placement from the previous set, starting with the smallest. For large graphs, this approach has a better chance of getting right some of the global details of the placement. As the charts in Fig. 3 indicate, MSKKR scales much better than KKR. It is worth noting that the GER metric cannot be used to compare the quality of the layout across different graph sizes. Unlike the BAR metric which clearly indicates the degradation in performance of KKR over larger instances (Fig. 3), the GER metric fails to capture this (Fig. 4). Sample input/output pairs from KKR and MSKKR are show in Fig 5.

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. As the number of vertices increases, the utility of KK and KKR decreases, while MSKKR does quite well on graphs with thousands of vertices. More complicated underlying shapes together with larger graphs, decrease the utility of even MSKKR. Angular information (if available) can be used with

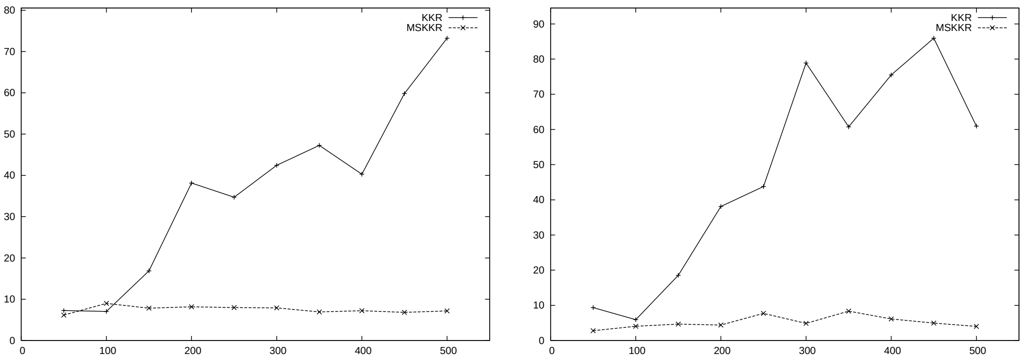


Fig. 3. Comparison between KKR and MSKKR measured by the BAR metrics across square-shape and star-shape graphs with sizes 50 to 500. There were five trials per shape, using graphs with density 8 and 10% range errors.

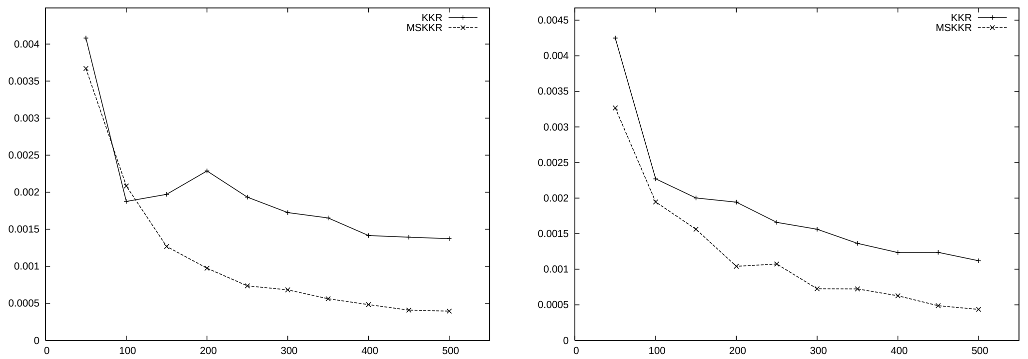


Fig. 4. Comparison between KKR and MSKKR measured by the GER metrics across square-shape and star-shape graphs with sizes 50 to 500. There were five trials per shape, using graphs with density 8 and 10% range errors.

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 has been used for centuries as a method of estimating the position of a moving object by applying to a previously determined position the course and distance traveled since. Given range and angular information, we can compute the distance between two vertices x and y in the graph 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. Let node C be D's predecessor in the shortest path from A to D, and let B be C's predecessor; see Fig. 6. 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's counter-clockwise.

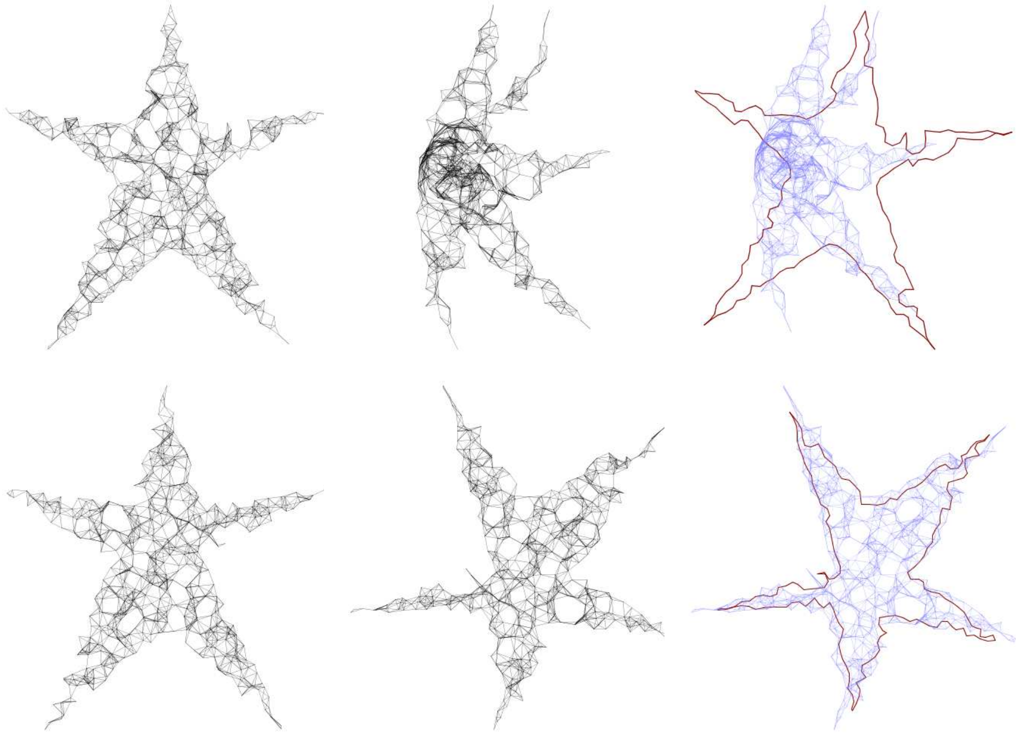


Fig. 5. Some typical input/output pairs illustrating input/output/boundary for KKR (top) and MSKKR (bottom). The underlying graphs have 500 vertices, density 8, range error of 10% and angle error also 10%.

Ultimately, we want to calculate $\angle ACD$ so that we can determine $dr(A, D)$ via the law of cosines. To do this, we must first we compute $\angle BCA$ using the law of cosines:

$$dr(A, B)^2 = edge(B, C)^2 + dr(A, C)^2 - 2 * edge(B, C) * dr(A, C) * cos(\angle BCA)$$

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

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

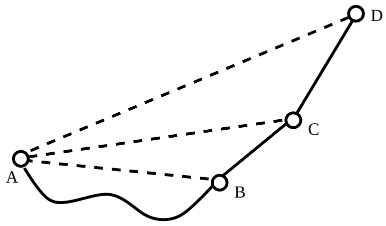
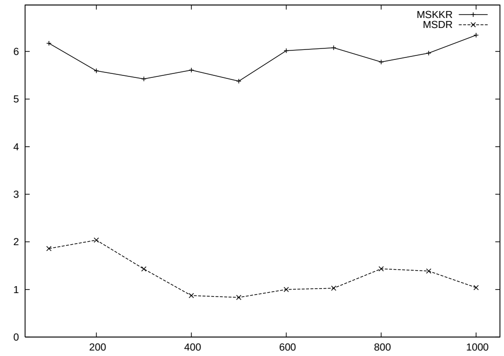


Fig. 6. In the BFS path from vertex A to D, the predecessor of D is C and the predecessor of C is B



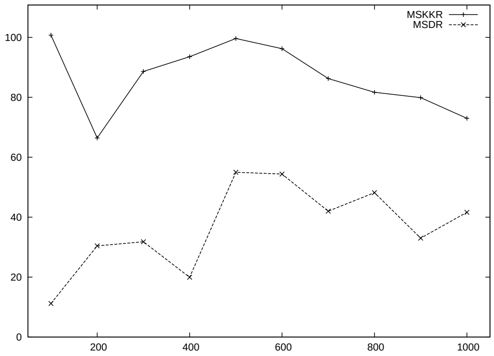


Fig. 7. Comparison between MSKKR and MSDR measured by the BAR metrics across square-shape and U-shape graphs with sizes 50 to 1000. There were five trials per shape, using graphs with density 8 and 10% range errors and 10% angular error.

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:

$$dr(A, D)^2 = dr(A, C)^2 + edge(C, D)^2 - 2 * dr(A, C) * edge(C, D) * cos(\angle ACD)$$

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 Kamada-Kawai algorithm results in our MSDR algorithm. It outperforms all of the algorithms discussed earlier in the paper. We study MSDR more carefully below. Comparing MSKKR to MSDR shows that MSDR consistently performs better; see Fig 7. For simple convex shapes both algorithms recover the global structure and the boundary well. In such cases MSDR offers local improvements over MSKKR. More notably, for non-simple and even non-convex shapes, MSDR can perform significantly better by truly capturing the underlying shape, which is often not the case with MSKKR; see Fig. 8.

Moreover, the MSDR algorithm is very tolerant to range errors and somewhat tolerant to angle errors. We compare the quality of the layouts under varying range and angular errors. Under the BAR metric, the algorithm can handle 25% range errors without noticable effects. The effect of angular errors is more pronounced, especially for non-convex shapes; see Fig. 9. The GER metric can be safely applied to compare the performance of MSDR for

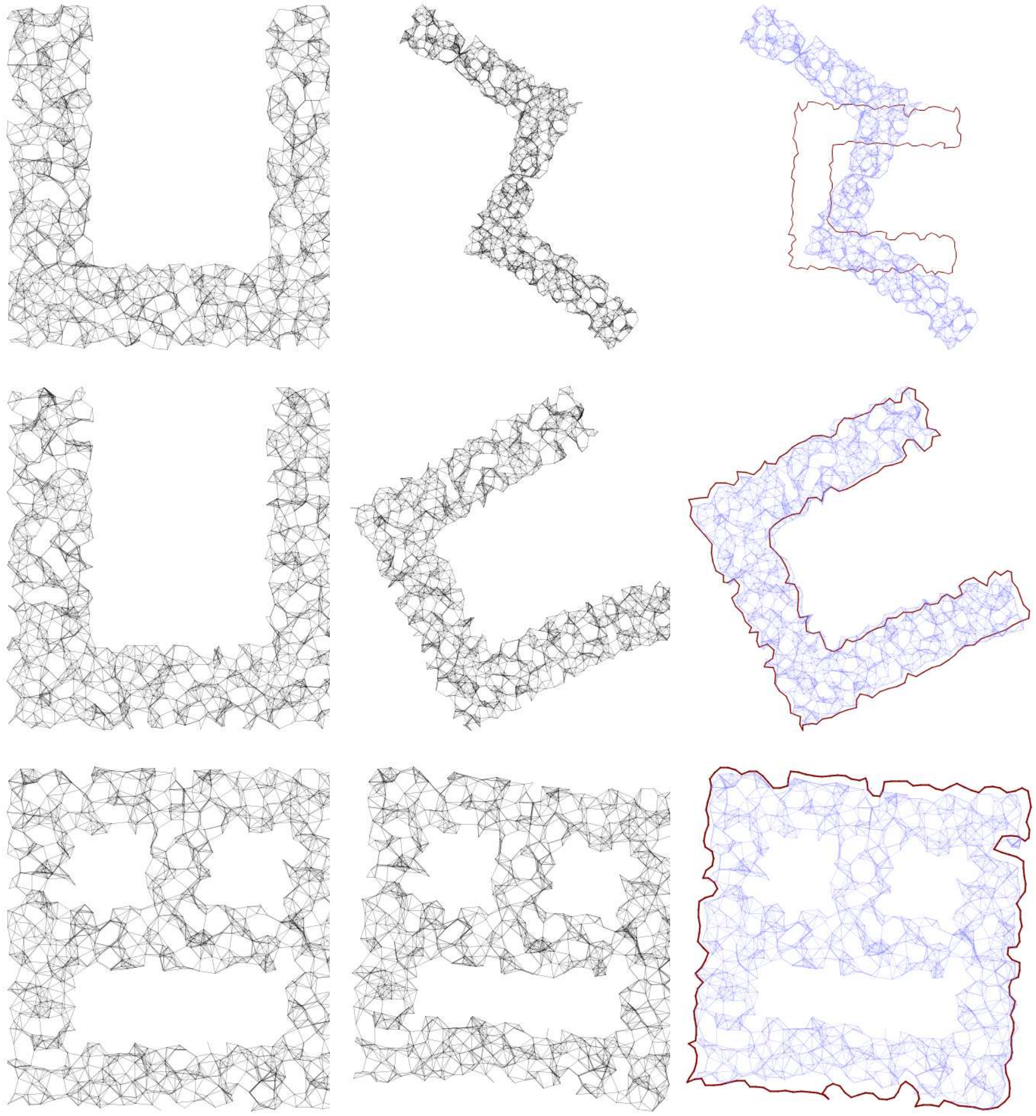


Fig. 8. Some typical input/output pairs illustrating input/output/boundary for MSKKR (top) and MSDR (middle and bottom). The underlying graphs have 1000 vertices, density 8, range error of 10% and angle error also 10%.

graphs of the same size, with varying range and angular errors. This metric also confirms that the MSDR is stable under range errors of up tp 25%; see Fig. 10.

5 Conclusions and Future Work

We described several adaptations of force-directed graph drawing algorithms for the sensor localization problem. We also presented a new approach that takes advantage of angular information, based on dead-reckoning and multi-scale techniques. All of these algorithms as well as the simulation that generates the data have been implemented as a part of the Graphael [6] system.

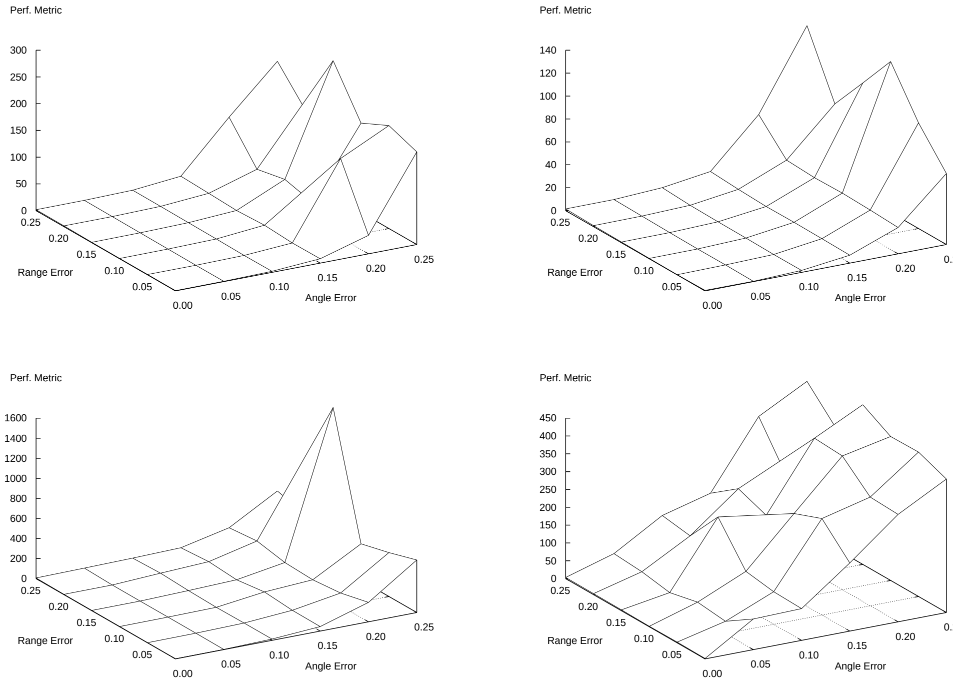


Fig. 9. Using the BAR metric: Error tolerance for MSDR across triangle-shape, square-shape, star-shape and U-shape graphs with sizes 500. There were five trials per for each experiments, using graphs with density 16 and varying the range errors and the angular errors.

We failed to find a performance metric that allows us to compare the quality of the global layout across different graph sizes. While the GER metric can be used to test resilience to different types of errors, it is not suitable for comparing graphs with different number of vertices. The BAR metric that we describe in the paper can indeed be used for that purpose but it only captures the quality of the boundary. Moreover, the ICP method on which the BAR metric is based, is a heuristic that often results in sub-optimal matches (see Fig. 2).

References

1. I. F. Akyildiz, S. Weilian, Y. Sankarasubramaniam, and E. E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, 2002.
2. P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and machine Intelligence*, 14(2):239–258, Feb. 1992.
3. L. Doherty, K. Pister, and L. E. Ghaoui. Convex optimization methods for sensor node position estimation. In *Proceedings of the 20th IEEE Computer and Communications Societies (INFOCOM-01)*, pages 1655–1663, 2001.
4. B. N. Dragos Niculescu. Ad hoc positioning system (aps) using aoa. In *Proceedings of the 22 Conference of the IEEE Computer and Communications Societies (INFOCOM-03)*, pages 1734–1743, 2003.
5. S. P. Fekete, A. Kröller, D. Pfisterer, S. Fischer, and C. Buschmann. Neighborhood-based topology recognition in sensor networks. In *ALGOSENSORS*, volume 3121 of *Lecture Notes in Computer Science*, pages 123–136. Springer, 2004.

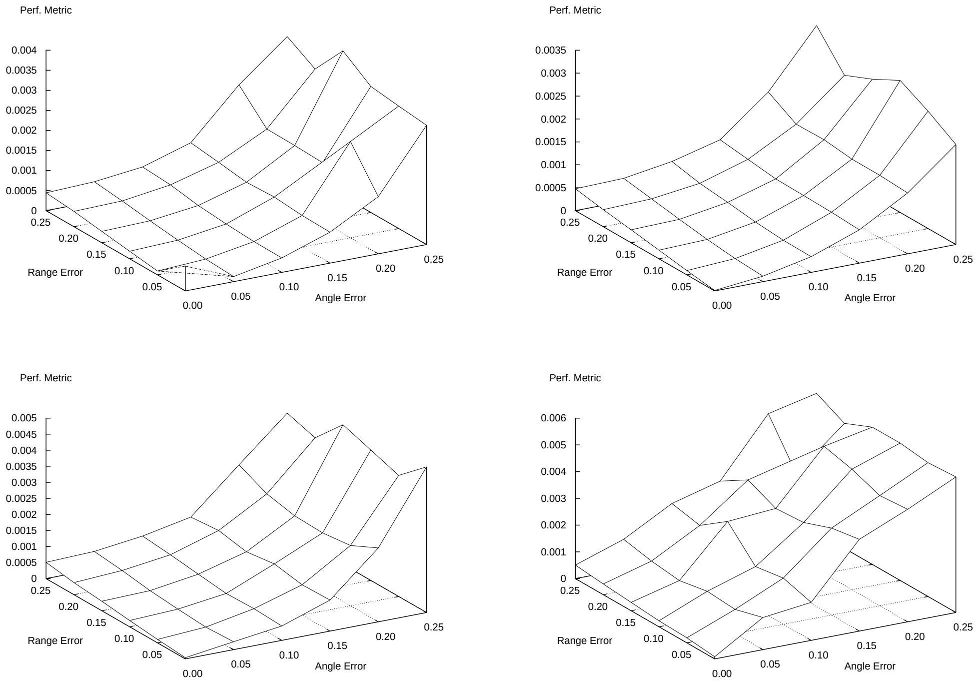


Fig. 10. Using the GER metric: Error tolerance for MSDR across triangle-shape, square-shape, star-shape and U-shape graphs with sizes 500. There were five trials per for each experiments, using graphs with density 16 and varying the range errors and the angular errors.

6. D. Forrester, S. G. Kobourov, A. Navabi, K. Wampler, and G. Yee. graphael: A system for generalized force-directed layouts. In *12th Symposium on Graph Drawing (GD)*, pages 454–466, 2004.
7. T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Softw. – Pract. Exp.*, 21(11):1129–1164, 1991.
8. P. Gajer, M. T. Goodrich, and S. G. Kobourov. A fast multi-dimensional algorithm for drawing large graphs. *Computational Geometry: Theory and Applications*, 29(1):3–18, 2004.
9. C. Gotsman and Y. Koren. Distributed graph layout for sensor networks. In *12th Symposium on Graph Drawing (GD)*, pages 273–284, 2004.
10. D. Harel and Y. Koren. A fast multi-scale method for drawing large graphs. *Journal of graph algorithms and applications*, 6:179–202, 2002.
11. T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inform. Process. Lett.*, 31:7–15, 1989.
12. M. Mauve, J. Widmer, and H. Hartenstein. A Survey on Position-Based Routing in Mobile Ad-Hoc Networks. pages 30–39, November 2001.
13. N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller. Anchor-free distributed localization in sensor networks. In *1st International Conference on Embedded Networked Sensor Systems (SenSys-03)*, pages 340–341, 2003. Also *TR #892, MIT LCS, 2003*.
14. C. Savarese, J. Beutel, and J. Rabaey. Locationing in distributed ad-hoc wireless sensor networks. In *Proc. 2001 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2037–2040, 2001.
15. A. Savvides, C. Han, and M. Srivastava. Dynamic Fine-Grained localization in Ad-Hoc networks of sensors. In *Proceedings of the 7th Conference on Mobile Computing and Networking (MOBICOM-01)*, pages 166–179, 2001.