Open access • Journal Article • DOI:10.1145/1807048.1807057

Force-directed approaches to sensor localization — Source link

Alon Efrat, David Forrester, Anand Iyer, Stephen G. Kobourov ...+2 more authors

Institutions: University of Arizona, Kadir Has University, Işık University

Published on: 04 Oct 2010 - ACM Transactions on Sensor Networks (ACM)

Topics: Brooks–Iyengar algorithm, Visual sensor network, Key distribution in wireless sensor networks, Wireless sensor network and Network topology

Force-Directed Approaches to Sensor Localization*

Alon Efrat, David Forrester, Anand Iyer, and Stephen G. Kobourov
Department of Computer Science
University of Arizona
{alon,forrestd,anand,kobourov}@cs.arizona.edu

Cesim Erten
Department of Computer Science and Engineering
Isik University, Turkey
cesim@isikun.edu.tr

Abstract

We consider the centralized, anchor-free sensor localization problem. We consider the case where the sensor network reports range information and the case where in addition to the range, we also have angular information about the relative order of each sensor's neighbors. We experimented with classic and new force-directed techniques. The classic techniques work well for small networks with nodes distributed in simple regions. However, these techniques do not scale well with network size and yield poor results with noisy data. We describe a new force-directed technique, based on a multi-scale dead-reckoning, 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; see [1] 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 [13].

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 (obtained using signal strength), or whether some of the vertices know their exact positions (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. 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 equipped with GPS are often called *anchors* and while they make the localization problem easier, they are bulky and expensive. 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 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

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

Dagstuhl Seminar Proceedings 05361
Algorithmic Aspects of Large and Complex Networks
http://drops.dagstuhl.de/opus/volltexte/2006/569

larger graphs, the simple force-directed algorithms fail to reconstruct the vertex locations. We show that multi-scale versions of the force-directed algorithms can extend the utility of these algorithms to graphs with hundreds of vertices, provided that the graphs are defined inside simple convex polygons. Finally, we describe a new multi-scale force-directed approach that incorporates the angular information in a dead-reckoning fashion. This approach can extend the utility of multi-scale force-directed 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 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 [1]. Several recent approaches have exploited the natural connections with graph layout algorithms. Priyantha *et al.* [15] 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 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 of the algorithms that do utilize angular information, 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.* [17] 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.* [16] and Niculescu and Nath [14] describe anchor-based algorithms for sensor localization utilizing edge lengths information. Fekete *et al.* [4] 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 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 technique to the centralized sensor localization problem. Two standard force-directed techniques are those of Fruchterman-Reingold [6] and Kamada-Kawai [11]. 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 [7] perform better. However, even these techniques fail to reconstruct graphs defined in non-simple, or non-convex regions.

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, we show that the new *multi-scale dead-reckoning* algorithm performs well and is tolerant to non-trivial noise for large networks defined in non-simple and non-convex regions.

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

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 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 [8] 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:

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

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. 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) [2] to align the two boundaries using rotation and translation. The boundary alignment ratio is defined as:

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

[1]The *global energy ratio* (GER) defined by Priyantha *et al.* [15] 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.

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

Since we did not have actual sensors to work with, we wrote a plugin for our graph visualization system, Graphael [5], 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, 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 specifies standard deviation (in percentage) about 100% of the true edge length using 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 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 specifies standard deviation (in degrees) about the actual angle from a sensor's declared "north" to an edge using Gaussian distribution.

3 Force-Directed Algorithms for Localization

Some of the most flexible algorithms for calculating layouts of simple undirected graphs belong to a class known as force-directed algorithms. Also known as spring embedders, such algorithms calculate the layout of a graph using only information contained within 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

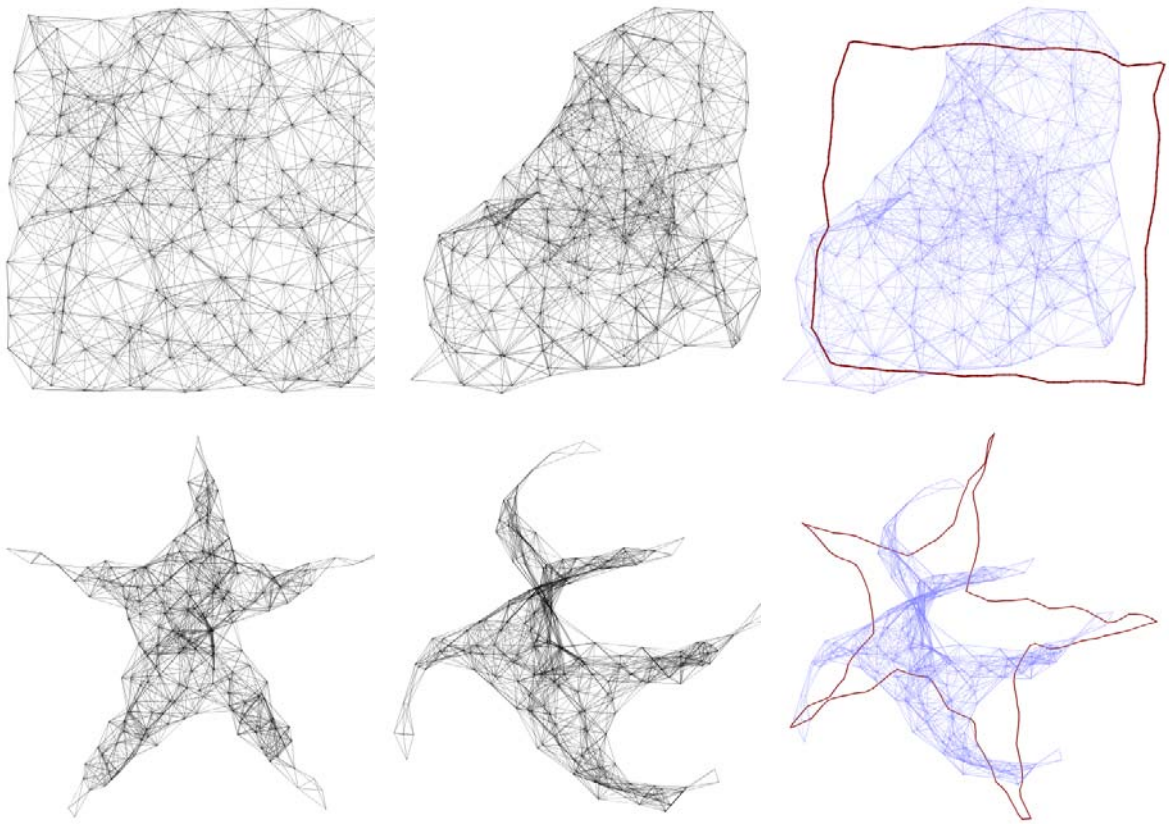

Figure 1: Typical results illustrating input/output/boundary-allignment for KK (top) and FR (bottom) for graphs with 200 vertices inside square and star-shape regions, respectively.

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 [6] 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. The Kamada-Kawai (KK) algorithm [11] 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[2] as the original graph layout D. Still, for small graphs (50-100 vertices) in simple underlying regions these algorithms often manage to reconstruct the underlying

[2]The notions of "scale" and "scalability" can be confusing. In this context, "scale" refers to the edge lengths of the graph. In general, when we refer to "scalable algorithms" we mean algorithms whose performance does not degrade with larger input sizes as measured by the number of vertices and edges in the input graphs. Finally, when we refer to "multi-scale" algorithms we mean multi-level, multi-stage type algorithms.

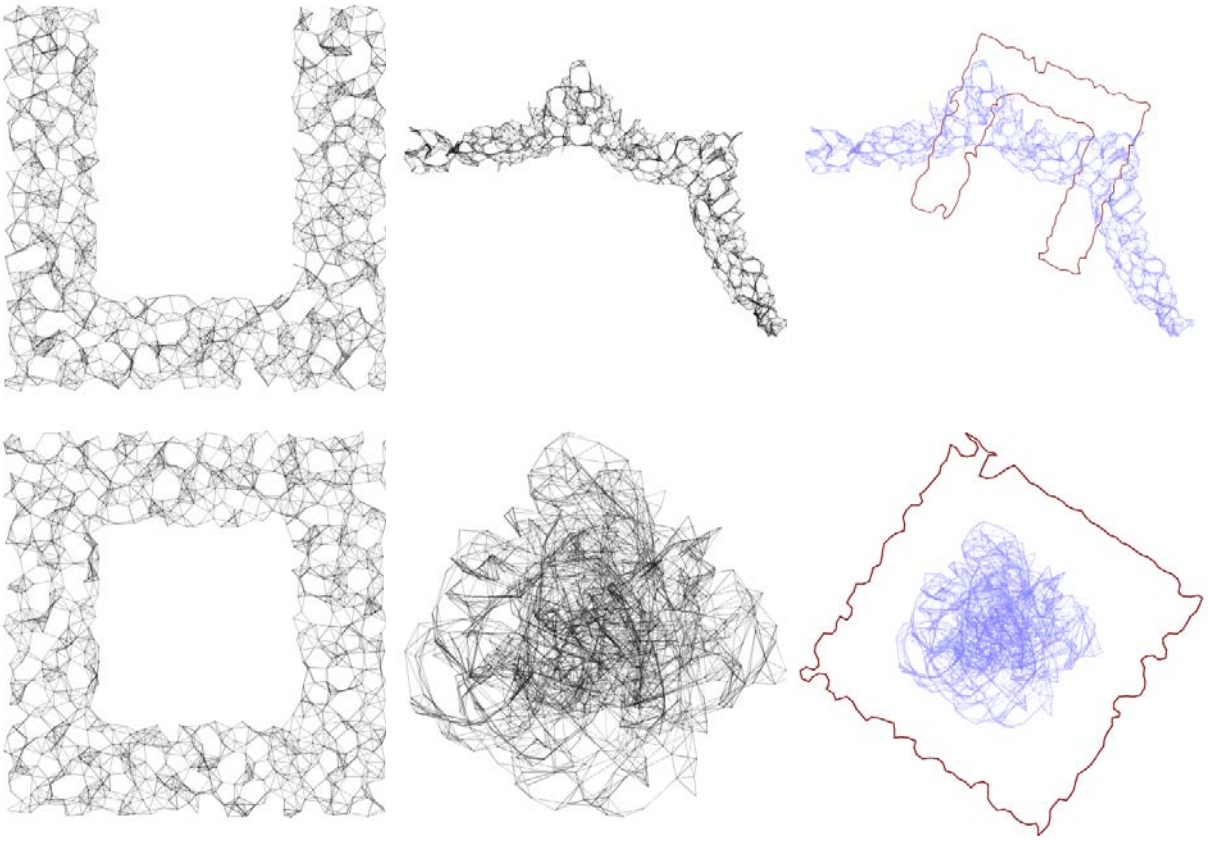

Figure 2: Typical results illustrating input/output/boundary-allignment for KKR (top) and FRR (bottom) for graphs with 1000 vertices, density 8, range error 10%, angle error 10°, inside U-shape and donut-shape regions, respectively.

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 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 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 particular, multi-scale variants of the Kamada-Kawai algo-

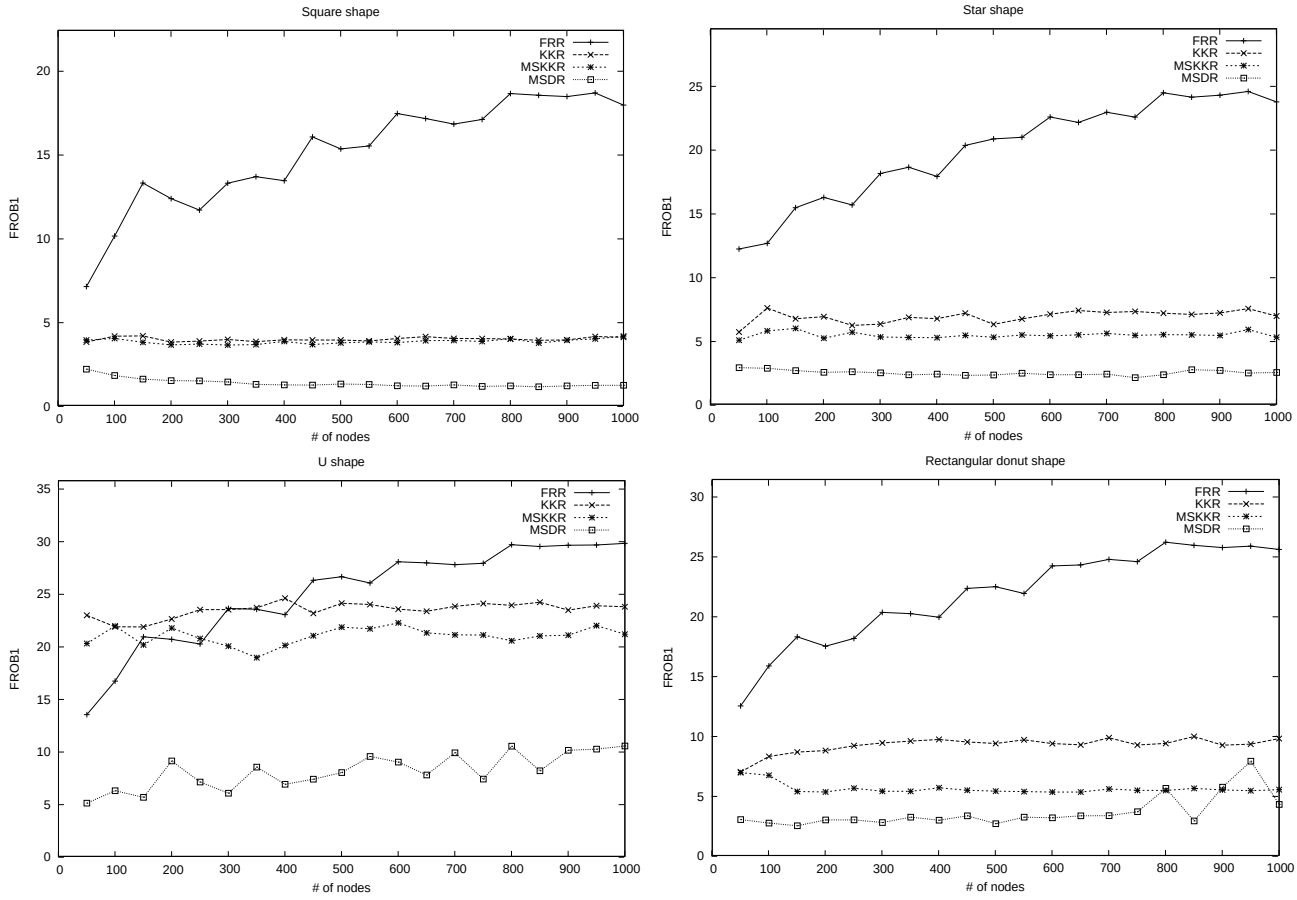

Figure 3: Comparison between FRR, KKR, MSKKR, and MSDR algorithms measured by the Frobenius metric across square-shape, star-shape, U-shape, and donut-shape graphs with 50 to 1000 vertices. There were twenty trials per shape, using graphs with density 8, range error of 20% and angle error of 10°.

rithm have already been shown to produce good results in traditional graph drawing setting [7, 10]. Our multi-scale algorithm, MSKKR, uses these ideas to extend the utility of KKR to larger graphs.

The MSKKR algorithm relies on a *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 that have already been placed, followed by a refinement of the current layout. Details of this approach are discussed in [7].

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-

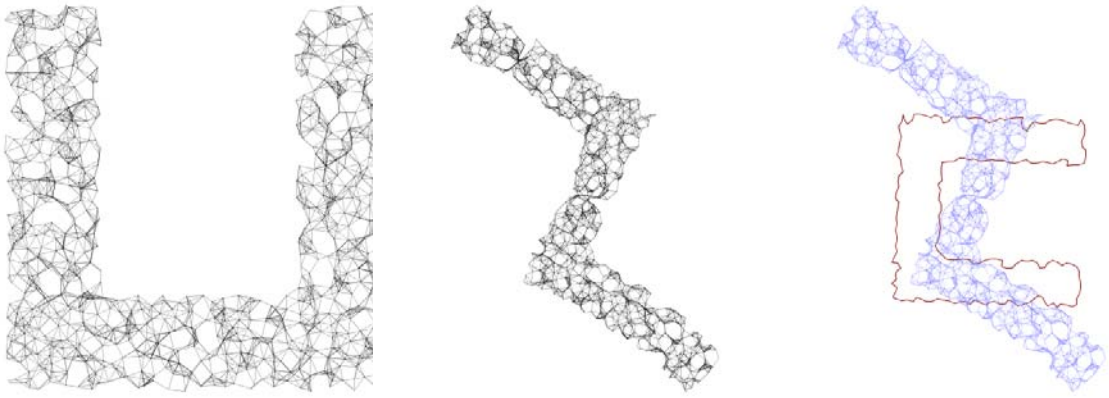

Figure 4: A typical problem with graphs defined in non-convex shapes. Input/output/boundary-allignment for MSKKR for a graph with 1000 vertices, density 8, range error of 10% and angle error of 10°.

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 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 (vertices with GPS) but these are too costly and bulky. Instead, angular information (if available) 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 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 [12]. 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 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. 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's 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: $dr(A, B)^2 = edge(B, C)^2 + dr(A, C)^2 - 2 * edge(B, C) * dr(A, C) * cos(\angle BCA)$:

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

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 difference. Put another way, we can just convert $\angle BCA$

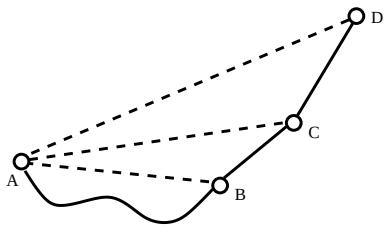

Figure 5: In the BFS path from vertex A to D, the predecessor of D is C and the predecessor of C is B.

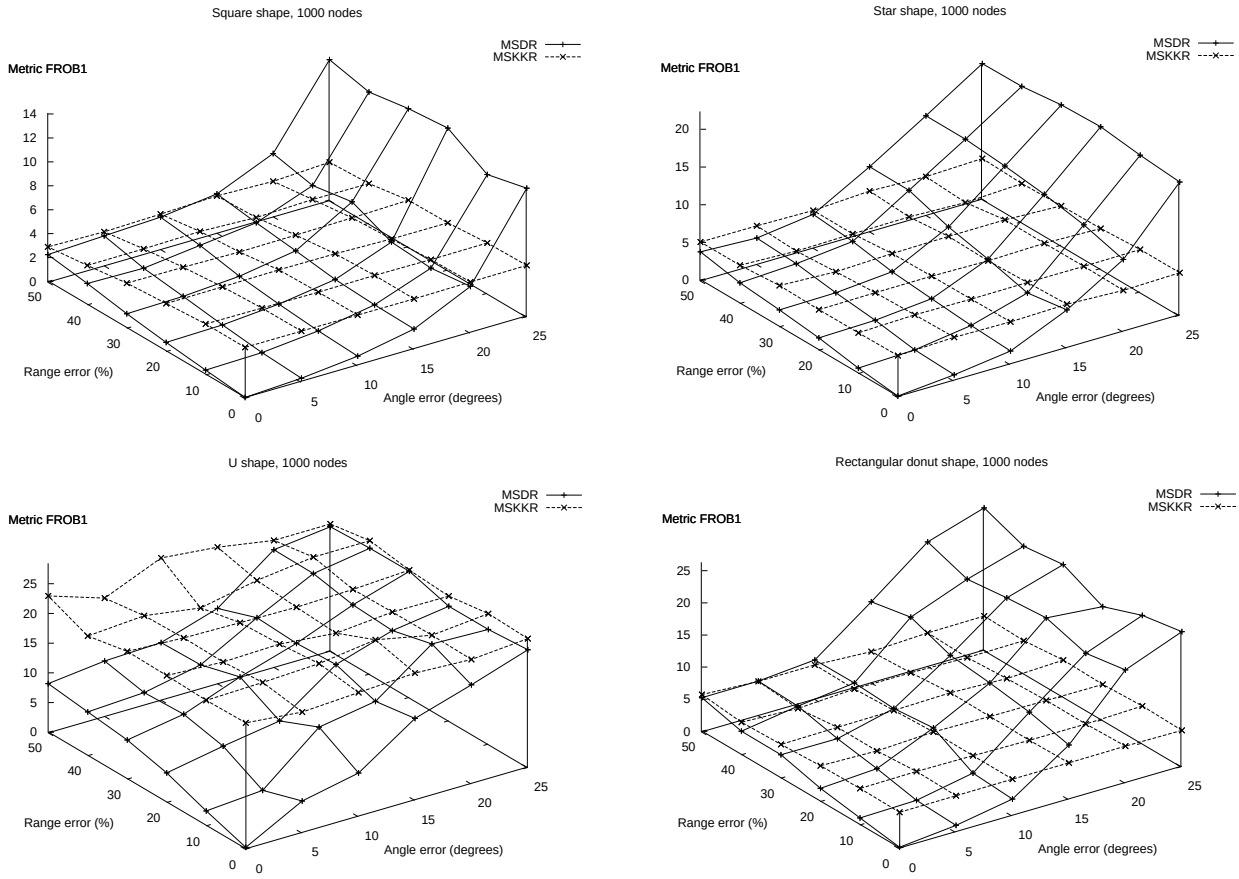

Figure 6: Comparison between MSKKR and MSDR measured by the Frobenius metric across square-shapes, star-shapes, U-shapes, and donut-shapes with 50 to 1000 vertices. There were twenty trials per shape, using graphs with density 8 and range errors 0-50% and angle error $0^\circ - 25^\circ$.

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 range-based Kamada-Kawai algorithm results in our MSDR algorithm. Not surprisingly, it outperforms all of the algorithms discussed earlier in the paper, given small angle errors; see Fig. 3.

Comparing MSKKR to MSDR shows that MSDR with angle errors of less than 10° consistently performs better; see Fig 6. Since MSKKR does not depend on angle errors and is resilient to range-errors it produces stable results for 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.

Layouts obtained with the MSDR algorithm using small angle and range errors often match near-perfectly the given source graphs; see Fig. 7.

The quality of the layouts under varying range and angular errors is captured in Figs. 8-9. Under the Frobenius metric, the algorithm seems 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. 8. 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. 9.

5 Conclusions and Future Work

We presented several adaptations of force-directed graph layout algorithms for the centralized, anchor-free sensor localization problem. 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 approaches. All of these algorithms as well as the simulation that generates the data have been implemented as a part of the Graphael [5] system.

The results presented in this paper are for centralized algorithms, whereas distributed algorithms for the sensor localization problem are more desirable. We plan to explore the possibility of developing practical distributed variants of the two multi-scale algorithms, MSKKR and MSDR.

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] 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.

[5] 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.

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

[7] 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.

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

[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. *Information Processing Letters*, 31:7–15, 1989.

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

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

[14] D. Niculescu and B. Nath. 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.

[15] 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*.

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

[17] 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.

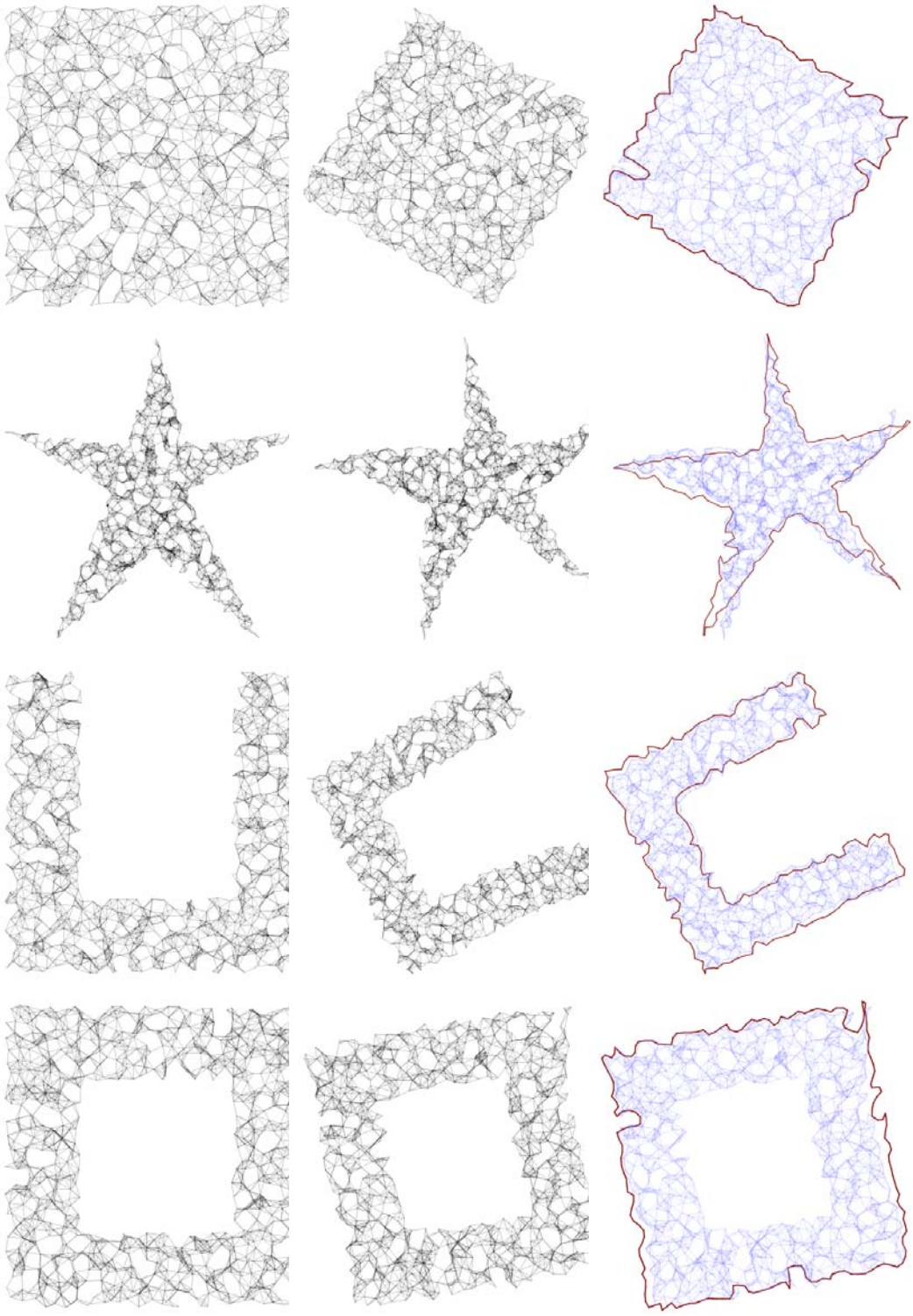

Figure 7: Typical results illustrating input/output/boundary-allignment for MSDR on square-shape, star-shape, U-shape, and donut-shape graphs. The underlying graphs have 1000 vertices, density 8, range error of 10% and angle error of 10°.

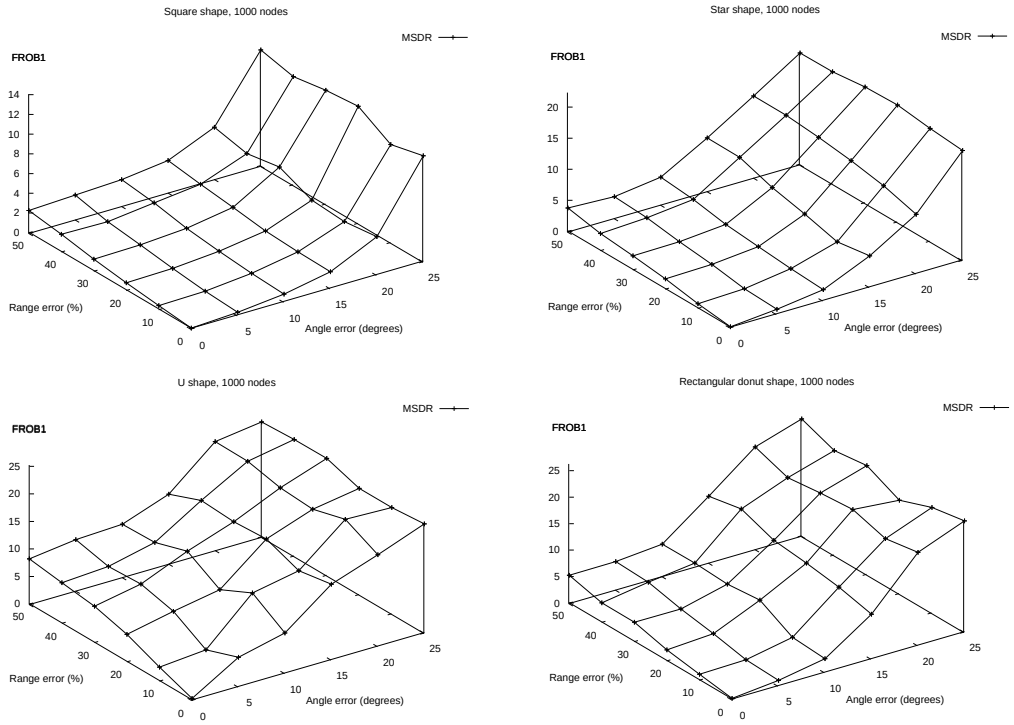

Figure 8: Frobenius metric error tolerance for MSDR across square-shape, star-shape, U-shape, and donut-shape graphs. There were twenty trials for each experiment using graphs with 1000 vertices, density 8 and varying the range and angle errors.

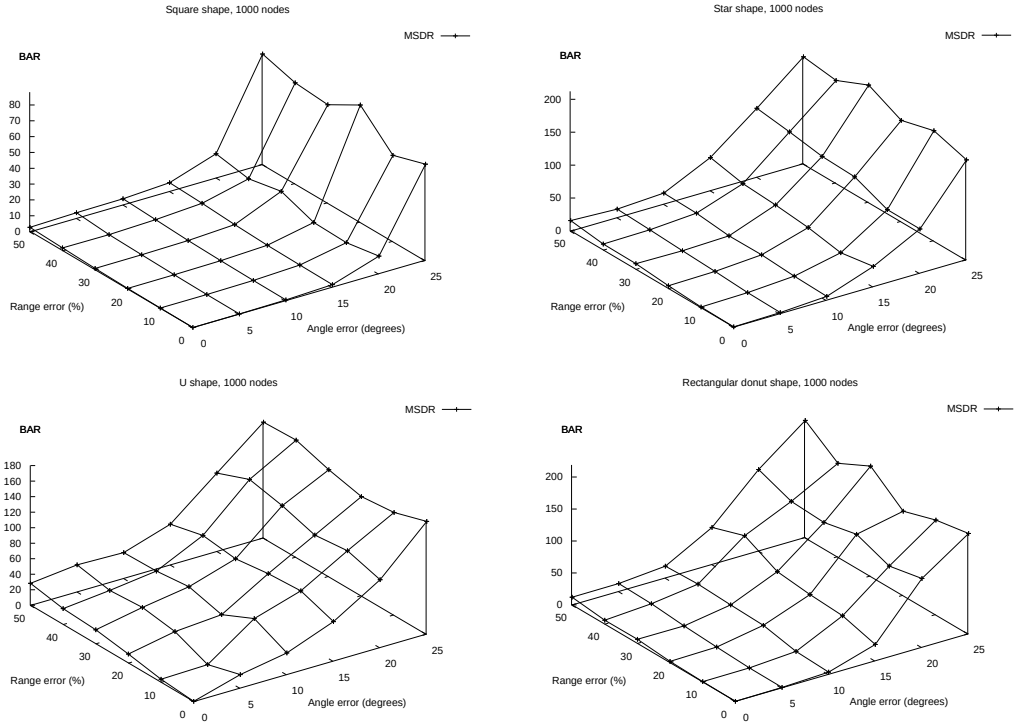

Figure 9: BAR metric error tolerance for MSDR across square-shape, star-shape, U-shape, and donut-shape graphs. There were twenty trials for each experiments using graphs with 1000 vertices, density 8 and varying the range and angle errors.