An As-Rigid-As-Possible Approach to Sensor Network Localization

Lei Zhang, Ligang Liu, Craig Gotsman and Steven J. Gortler

TR-01-09

Computer Science Group
Harvard University
Cambridge, Massachusetts

An As-Rigid-As-Possible Approach to Sensor Network Localization

Lei Zhang Ligang Liu
Zhejiang University
Hangzhou, China
{ethan_zhang, ligangliu}@zju.edu.cn

Craig Gotsman
Technion
Haifa, Israel
gotsman@cs.technion.ac.il

Steven J. Gortler
Harvard University
Cambridge, USA
sjg@cs.harvard.edu

Abstract— **We present a novel approach to localization of sensors in a network given a subset of noisy inter-sensor distances. The algorithm is based on "stitching" together local structures by solving an optimization problem requiring the structures to fit together in an "As-Rigid-As-Possible" manner, hence the name ARAP. The local structures consist of reference "patches" and reference triangles, both obtained from inter-sensor distances. We elaborate on the relationship between the ARAP algorithm and other state-of-the-art algorithms, and provide experimental results demonstrating that ARAP is significantly less sensitive to sparse connectivity and measurement noise. We also show how ARAP may be distributed.**

Keywords: *sensor networks, localization, embedding.*

I. INTRODUCTION

Sensor networks are a distributed collection of small devices, capable of a limited amount of local processing and wireless communication [19]. They are drawing more and more attention for use in a variety of applications, such as environment monitoring, military surveillance, smart places etc [12]. In many typical applications, sensors are deployed over a physical area, sometimes at random, without any prior knowledge of their locations. Lacking GPS components, a sensor cannot know its final location in the arena, which may be necessary for it to be useful. Thus autonomous computation of the spatial coordinates of the sensors, based on just the limited information available to the sensors, is an important practical problem, known as the *localization* problem.

One useful source of information for the localization problem is inter-sensor distances. Two sensors with a communication link between them may be able to measure the distance between them and use it for localization. This type of information is the typical input to a localization algorithm. The communication links between the sensors may be modeled as a *sensor graph*, and the distances measured along the edges of this graph are called the *measurement graph*. The coordinates of the sensors computed during localization are sometimes called an *embedding*, *realization* or *layout* of the measurement graph.

More formally, this paper treats the following problem: *Given a set of sensors with unknown spatial distribution, and a mechanism to measure the distances between a sensor and its nearby (neighbor) sensors, determine the coordinates of all the sensors through local sensor-to-sensor communication*.

The localization problem is NP-Hard [16,20], i.e. there is no known efficient algorithm that is guaranteed to find the correct embedding (or even an approximate solution) for all measurement graphs. This is true even when the measurement graph is *generically globally rigid*, i.e., when the coordinates are completely determined, up to a rigid transformation, from the measured distance data. In this case the embedding is called *globally rigid*. Note that generic global rigidity (assuming that the embedding is also "generic") is a property of the *graph structure*, not of the actual distance measurements [8]. Only if the measurement graph is known to belong to a special subclass of globally rigid graphs, such as trilateration graphs [4], efficient localization may be provably possible.

Despite the inherent difficulty in the general case, the importance of this problem has led to many heuristic and numerical algorithms for its approximate solution. The realistic, and most challenging, scenario is when the measurement graph is sparse and noisy. This means that there is only small amount of unreliable information available. Many existing localization algorithms are extremely sensitive to this and degrade very quickly when presented with these types of inputs. This paper presents an algorithm which is relatively easy to implement, can be distributed among the sensors, and, most important, is extremely robust, even when the measurement graph is sparse and noisy.

II. RELATED WORK

Assume the sensor network is modeled by a graph $G = (V=\{1,..,n\}, E)$, such that for every edge $(i,j)\in E$ we know the (noisy) distance d_{ij} between sensors i and j. We assume, without loss of generality, that the distances are always symmetric, so $d_{ij}=d_{ji}$. This can be achieved by requiring neighboring sensors to reach an agreement about their distance (e.g. by averaging d_{ij} and d_{ji}). We would like to generate a two-dimensional embedding $P = (p_1,..,p_n)$ that *realizes* all known distances. This embedding can be formally characterized as the minimum of the *stress function*:

$$Stress(p_1,..,p_n) = \sum_{(i,j)\in E} \left(\|p_i - p_j\| - d_{ij} \right)^2 \quad (1)$$

The stress function and its variants are widely used in problems in distance geometry and in statistics, where distances measure some sort of affinity between subjects. When all pairwise distances are available, it is possible to easily minimize the *strain* function - a function very similar to stress. Strain is a convex function, thus may be minimized using *Classical Multi-Dimensional Scaling* (MDS) [2], which, in practice, is an eigenvector computation. The situation changes dramatically when only a subset of the pairwise distances is available. Stress is a non-convex function, hence, in general, we cannot find its global minimum efficiently, and slow iterative methods must be employed. The main phenomenon observed in the (suboptimal) solutions is that of *foldovers*, where entire pieces of the embedding fold over on top of others, while still realizing well most of the measurement graph.

In the general case, stress may be minimized using iterative *stress majorization*, also called the SMACOF algorithm [2]. Majorization is a modern optimization technique achieving better results than applying traditional optimization methods, such as gradient descent, to the stress function. Although all techniques will eventually get stuck in local stress minima, and require a good foldover-free initial embedding to start from [9], stress majorization is faster and seems to "jump over" many local minima to a generally better solution.

In an attempt to depart from the cumbersome (and slow) global optimization strategies, Moore *et al.* [13] describe a localization algorithm which takes advantage of the special nature of the connectivity of sensor network measurement graphs. Since each sensor typically can communicate only with its local neighborhood, the connectivity tends to be local, with only short edges. These graphs can even be mathematically modeled as "disk graphs", where two sensors are connected if (and only if) the distance between them is less than some parameter. This characteristic of sensor network connectivity implies that the connectivity within small neighborhoods is typically dense, even frequently making those neighborhoods generically globally rigid. Thus Moore *et al.* [13] propose an incremental algorithm which attempts to first localize only small "patches" of the network. Each such patch is a set of four sensors forming a rigid quadrilateral, which is localized using a procedure called "trilateration" and then improved by running stress minimization on that patch. Given one quad which has been localized, another quad is found which has some sensors in common with the first. This is then laid out relative to the first by applying the best possible rigid transformation between the two. In such a manner, a sequence of quads is laid out in breadth-first order until no more sensors can be localized.

While a very promising approach, the algorithm of Moore *et al.* will localize only those sensors contained in a trilateralizable component of the network. The algorithm will not produce anything useful for the other sensors. More unfortunately, there exist generically globally rigid graphs that are not trilateralizable. Goldenberg *et al.* [6] describe a method to find generically globally rigid subgraphs (on which they run an MDS approach), and thus avoid contaminating their answer with underdetermined portions of the network.

A similar "patching" algorithm was described by Shang and Ruml [17]. However, they localize an individual patch by first computing all pairwise shortest paths (in the measurement graph) between sensors in the patch. Classical MDS is then applied to these distances to yield an initial embedding, which is subsequently improved by stress minimization. Similarly to Moore *et al.* [13], the patches are "stitched" together incrementally in a greedy order by finding the best affine transformation between a new patch and the global embedding. A post-processing stage of the algorithm further improves the embedding by minimizing the stress energy of the complete graph. These incremental algorithms seem to be sound, but, unfortunately, like any other incremental algorithm, may accumulate error indefinitely, especially when the measurement graph is very noisy.

The PATCHWORK algorithm of Koren *et al.* [11] solves the error accumulation error problem which may affect the methods of Moore *et al.* [13] and Shang and Ruml [17]. It also first localizes small patches of sensors. However, instead of then gluing them together incrementally, and possibly accumulating error in the process, it applies a global process which attempts to map the patches to a global coordinate system via per-patch affine transformations. Since the patches overlap, this involves solving a linear least-squares problem in order that the transformations agree well on the locations of sensors they have in common. Koren *et al.* [11] observe that using affine transformations between each patch and the global coordinate system, may be interpreted as affine relationships *between patches*. Some algebraic manipulation shows that PATCHWORK reduces to a method very similar to *dimension reduction* methods [3,15], used in machine learning.

A more recent approach of Singer [18], called Locally-Rigid Embedding (LRE), makes an explicit connection between localization and the machine learning technique of Locally-Linear Embedding (LLE). LRE tries to preserve the local affine relationships, present *within* patches, in the global coordinate system. Because of the overlap between patches, this imposes affine relationships *between* patches. A linear system is set up, each sensor contributing one equation relating its location to those of its neighbors. Solving this system results in an embedding of all sensors, from which a global affine transformation must be removed.

The method we propose here, called As-Rigid-As-Possible (ARAP) belongs to the same family as PATCHWORK and LRE. We also start off by localizing small patches in a very similar manner. However, instead of then embedding them in a global coordinate system using *affine* mappings, we embed them using *rigid* mappings. Here too, the overlap between patches constrains the mappings. Using rigid mappings has the advantage of preserving better the local relationships between patches, but has the disadvantage of resulting in a (sparse) non-linear system of equations. Fortunately, this non-linear system may be solved rather simply and efficiently using a two-phase *alternating least-squares* method. Along the way, we show how to improve LRE and PATCHWORK at just a slightly more expensive price of solving a larger (but still sparse) linear system, which we call As-Affine-As-Possible (AAAP). In fact, we use AAAP as the starting point for the iterations of ARAP. Experimental results show that ARAP is more robust to sparsity and noise in the measurement graph than all its predecessors. So, while ARAP might seem to be a straightforward extension of LRE, it provides a very large increase in perform-

ance in return for a very small increase in computational complexity.

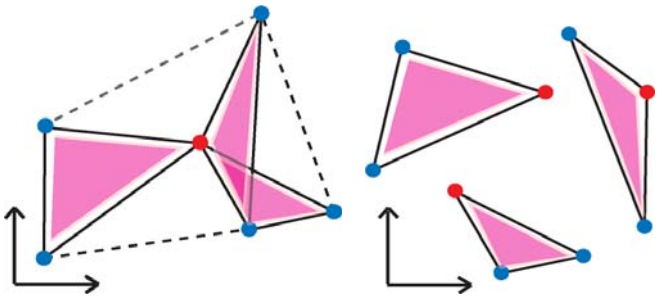
III. THE ARAP ALGORITHM OVERVIEW

A sensor network can be modeled as a *measurement graph* $G = (V=\{1,..,N\}, E)$, where the sensors are treated as the nodes of the graph, and edge $(i,j) \in E$ is associated with the distance d_{ij} measured between sensors i and j. We denote by $N(i) = \{j:(i,j)\in E\} \cup \{i\}$ the set of sensors connected to sensor i, and their number by N_i. Sensor i has no prior knowledge about its location, and knows only the distances to its neighbors in $N(i)$. We would like to find a localization of these sensors in the plane, i.e., $P = \{p_1 ,.., p_n\}$, where $p_i \in \Re^2$, such that $\|p_i - p_j\|$ is as close as possible to the given d_{ij}, essentially a global minimum of the stress energy (1).

Since a triangle is generically globally rigid, any sensor i, along with its two neighbors j and k, whose accurate inter-sensor distances d_{ij}, d_{jk}, d_{ki} are known, may be uniquely positioned (or *localized*) in an arbitrary coordinate system, up to a rigid transformation. Call a triplet of sensors for which this data is given an *available* triangle, and its localization a *reference triangle*. Number the available triangles by $t=1,..,T$, and denote their localizations in the local coordinate systems by $Q^t = \{q^t_i, q^t_j, q^t_k\}$.

More complicated graphs, such as the subgraph of G induced by $N(i)$, are not necessarily generically globally rigid, even if G is. And even if they are generically globally rigid, they still may not be localizable using known methods. But they may be, if the subgraph contains sufficient edges. Thus we may attempt to localize the sensors in $N(i)$ in an arbitrary coordinate system, again up to a rigid transformation. The localization of $N(i)$ is called a *reference patch*, and denoted by $Q_i = \{q_i, q_{i1}, q_{iNi}\}$. See Figure 1 for an illustration of reference triangles and patches.

The spirit of our algorithm is to construct as many of these reference triangles and reference patches as possible, and "stitch" them together in a global coordinate system to localize the entire measurement graph. The latter is done by mapping each patch to the global coordinate system using a rigid transformation. Both phases of the algorithm try to respect the measurement graph as much as possible, thus minimizing the stress energy.



(a) (b)

Figure 1. Possible reference patch and triangles associated with a given vertex (red dot) in a measurement graph. Solid lines are edges in the measurement graph (i.e. the length of the edge is known). Dashed lines are edges not in the measurement graph (i.e. the length of the edge is unknown).

IV. REFERENCE TRIANGLE AND PATCH LOCALIZATOIN

Reference triangles can be obtained easily. If the distances between the three nodes i, j and k are given, they have a stable localization in the plane up to a rigid transformation: a translation, rotation and reflection (sometimes called "flip"). This follows from elementary geometry.

Localizing a patch $N(i)$ is more complicated. The distances between i and the other sensors in $N(i)$ are all known. Only some of the distances between the neighboring sensors are known. We use the three-stage method of PATCHWORK [11], which proceeds as follows.

Estimating missing distances. We estimate the distance d_{jk} for all $(j,k)\notin E$, $j,k\in N(i)$. From the triangle inequality, we obtain an upper bound B_{jk} on d_{jk}, and from the disk graph assumption a lower bound b_{jk} on d_{jk}:

$$B_{jk} = \min_{h:(j,h)\in E,(h,k)\in E} \{d_{jh} + d_{hk}\}$$

$$b_{jk} = \max\left\{\max_h\{d_{jh}\}, \max_h\{d_{hk}\}\right\}.$$

Then, the estimate for d_{jk} is

$$d_{jk} = (b_{jk} + B_{jk})/2.$$

Classical MDS. After estimating all the missing distances between sensors in $N(i)$, the reference patch can be obtained by Classical MDS [2].

Stress majorization. This step is to improve the embedding of the reference patch by using only the known distances between the sensors in $N(i)$. It involves iteratively applying the following update rule, for each $i\in N(i)$:

$$q_i \leftarrow \frac{1}{N_i}\sum_{j\in N(i)}\left[q_j + d_{ij}(q_i - q_j)\mathrm{inv}\left(\|q_i - q_j\|\right)\right]$$

where

$$\mathrm{inv}(x) = \begin{cases} 1/x & x \neq 0 \\ 0 & x = 0. \end{cases}$$

V. RIGID ALIGNMENT

Once the reference triangles and patches are obtained, it remains to align ("stitch") all the triangles and patches together into one coherent layout. If we assume that we have properly localized the triangles and patches up to a rigid transformation, ideally the alignment should allow only rigid transformations between the patches and the global coordinate system. Reference triangles and patches undergo the same "stitch" operation, so we give a detailed derivation of the rigid alignment for reference patches, which is applicable also to the reference triangles.

Denote by the $2\times N_i$ matrix $Q_i = (q_i, q_{i1},.., q_{iNi})$, where $q_i = (q_i^x, q_i^y)^T$ are the sensor locations in the reference patch $N(i)$ and by $P = (p_1 ,.., p_N)$, a $2\times N$ matrix, where $p_i = (p_i^x, p_i^y)^T$ is the global localization of the i'th sensor. Note that we assume that the location vectors are 2D column vectors.We would like to find a P such that all the reference patch localizations relate to it via rigid transformations, i.e. find also $\mathbf{R}_i$ and $\mathbf{T}_i$ for Q_i, such that:

$$P_i = \mathbf{R}_i Q_i + \mathbf{T}_i \quad (2)$$

P_i is the localization of $N(i)$ induced by P, $\mathbf{R}_i$ is a 2×2 rigid transformation matrix, and $\mathbf{T}_i$ is a translation vector.

To eliminate the translational component $\mathbf{T}_i$ immediately, we work only with vector *differences* between the coordinates of the sensors in N_i and the coordinates of the i'th sensor. Thus, to perform localization of the entire sensor network, we have to solve the following optimization problem *simultaneously* for P (from which all P_i are induced), and *all* the $\mathbf{R}_i$ (one for each patch):

$$(P, \mathbf{R}_1, .., \mathbf{R}_N) = \arg\min_{P, \mathbf{R}_i} \left\{ \sum_{i=1}^{N} \left\| P_i - \mathbf{R}_i Q_i \right\|_F^2 : \mathbf{R}_i^T \mathbf{R}_i = \mathbf{I} \right\} \quad (3)$$

where $\| \cdot \|_F$ is the Frobenius matrix norm, and the P_i and Q_i are the appropriately translated vectors. Once this cost function is minimized, we may discard the $\mathbf{R}_i$.

We now make two key observations:

1. Given P_i and Q_i, (3) may be solved separately (and locally) for each $\mathbf{R}_i$.

$$\mathbf{R}_i = \arg\min_{\mathbf{R}} \left\{ \left\| P_i - \mathbf{R} Q_i \right\|_F^2 : \mathbf{R}^T \mathbf{R} = \mathbf{I} \right\} \quad (4)$$

This is the least-squares version of (2), since it will not always be possible to find a $\mathbf{R}_i$ satisfying (2) exactly, as the system may be overconstrained. Sadly, (4) is a *non-linear* least squares problem, because of the orthogonality constraint on $\mathbf{R}_i$. Fortunately, it still has a relatively simple solution. Solving (4) for the best rigid matrix $\mathbf{R}_i$ for each $N(i)$ is an instance of the Rotation Orthogonal Procrustes Problem, which can be solved by Procrustes analysis [10]. A closed form solution is $\mathbf{R}_i = V_i U_i^T$, where U_i and V_i are the orthogonal components of the Singular Value Decomposition (SVD) [7,14] of $Q_i P_i^T$:

$$Q_i P_i^T = U_i \Sigma_i V_i^T \quad (5)$$

2. If all $\mathbf{R}_i$ are given, then the remaining unknown P can be easily obtained by solving a quadratic optimization problem:

$$P = \arg\min_{P} \left\{ \sum_{i=1}^{n} \left\| P_i - \mathbf{R}_i Q_i \right\|_F^2 : P_i \text{ is induced by } P \right\} \quad (6)$$

By setting the gradients to zero, we obtain the following $2N \times 2N$ linear system of *normal* equations in the (x and y) components of P:

$$\sum_{j \in N(i)} (p_i - p_j) = \frac{1}{2} \sum_{j \in N(i)} \left[\mathbf{R}_i (q_i - q_{ij}) + \mathbf{R}_j (q_{ji} - q_j) \right] \quad (7)$$
$$i = 1, 2, ..., N$$

These two observations mean that we can solve (3) using a two-phase Alternating Least-Squares (ALS) method: In the first local phase, P is assumed to be fixed, and we solve (4) for each of the $\mathbf{R}_i$. In the second global phase, all the $\mathbf{R}_i$ are assumed to be fixed, and we solve (6) for P.

We call the use of the ALS technique to minimize (3) the "As-Rigid-As-Possible" (ARAP) localization algorithm. Since the entries of the coefficient matrix of (7) depend only on the reference patch, this matrix is a sparse matrix which is fixed throughout ARAP. Thus we may pre-factor the matrix, and reuse the factorization to efficiently solve (7) with a different left-hand side at each iteration. The observant reader may recognize this matrix as the combinatorial Laplacian matrix of the measurement graph [1], and (7) as the celebrated *Poisson* equation on a graph [14].

To run ARAP, we need an initial localization which the ARAP will refine as it alternates through the two phases. We describe how to do this in the next section.

A. AAAP Localization

At this point we make another key observation.

> If the measurement graph is connected and generically globally rigid and all patches have been correctly localized (up to a rigid transformation), then it is possible to correctly localize the entire graph (up to a global rigid transformation) by solving (3), *while allowing* $\mathbf{R}_i$ *to be a similarity, or even affine, transformation.*

This observation is non-trivial. It relies on the concept of *stress matrices* in rigidity theory, and is implied by the main result of Gortler *et al.* [8] that generic global rigidity of a graph implies the existence of a symmetric equilibrium stress matrix of co-rank 3 (and no more) for the graph.

This observation, surprising at first glance, becomes less surprising when we realize that most of the localization work has already been done in the reference patch localization, and the fact that the graph is generically globally rigid implies that these localizations essentially determine the global localization. Thus relaxing the conditions on $\mathbf{R}_i$ do not really allow the stitching stage to misbehave. The advantage of relaxing the condition on $\mathbf{R}_i$ is that now (3) becomes a single global linear system in the parameters of $\mathbf{R}_i$ and P, so may be solved in "one shot", as opposed to the iterative ARAP. In fact, the resulting per-patch affine transformations will probably be all rigid transformations composed with a common global affine transformation. There is, however, one caveat: The global localization is unique only up to a global transformation taken from the relevant family used: similarities or affine. In particular, this means that (3), without the condition on $\mathbf{R}_i$, will be homogeneous, thus admit the trivial solutions $P = 0$ and $\mathbf{R}_i = \mathbf{0}$ for all i. To avoid this, either a small number of "anchored" sensor positions must be known in advance, and these "pinned down" when solving the system, or the linear system can be solved as an eigenvector problem, which effectively factors out the null-space. In addition, the global transformation can be "factored out" (up to a rigid transformation) in a postprocessing stage. This is done by comparing the inter-sensor distances of the result with the input measurement graph and finding the best transformation that minimizes the distortion of these values.

To compute an initial localization, we solve (3) when $\mathbf{R}_i$ are required to only be affine transformations. If we parameterize $\mathbf{R}_i$ as follows:

$$\mathbf{R}_i = \begin{pmatrix} a_i & c_i \\ b_i & d_i \end{pmatrix} \tag{8}$$

this results in a quadratic optimization problem with unknowns $p_i = (p_i^x, p_i^y)^T$, a_i, b_i, c_i and d_i, whose normal equations are:

$$\begin{aligned} p_j^x - p_i^x - (q_j^x - q_i^x)a_i - (q_j^y - q_i^y)b_i = 0 \\ p_j^y - p_i^y - (q_j^x - q_i^x)c_i - (q_j^y - q_i^y)d_i = 0 \\ i = 1,...,N,\ j \in N(i) \end{aligned} \tag{9}$$

which is linear (and homogeneous) in the variables.

We call this algorithm the As-Affine-As-Possible (AAAP) localization procedure.

Should we want to restrict the transformations $\mathbf{R}_i$ to belong to the class of similarities, this would impose the additional constraints $a_i = d_i$ and $c_i = -b_i$, which are still linear, as opposed to imposing that $\mathbf{R}_i$ are rigid transformations, which would additionally imply that $a_i^2+b_i^2 =1$, a non-linear equation.

The observant reader might ask why not solve (3) for $\mathbf{R}_i$ restricted to the family of similarities, since the system would still be linear, more compact, and closer to the desired class of linear transformations – the rigid ones ? The answer is that, surprisingly, affine transformations are more suitable, since they also allow reflections, which similarities do not allow. These reflections are crucial for the stitching process, since we have no information on the orientation of patches, and their reference localizations must be allowed to reflect ("flip") if needed.

It is relatively simple to see that our initial AAAP localization procedure is quite similar to the PATCHWORK method of Koren *et al.* [11], but a little more general. The interested reader is referred to Section 3.2 of [11] for details. We have already stated that in a noiseless scenario this method would suffice. However, as we shall demonstrate later, in a realistic, noisy, scenario, the method degrades rather quickly. Koren *et al.* [11] compensate for this by applying standard stress majorization in a post-processing step, but this is quite slow, and has limited effect. Thus AAAP suffices only as an initial embedding for ARAP.

B. A Distributed Solution

In order for a localization algorithm to be practical in a real scenario, the algorithm must be distributable, meaning that it can be computed on the sensor network through exchange of information between a sensor and its immediate neighbors only. This is the case for our ARAP algorithm. Obviously, reference patch computation is local, as is the local stage (4). The global linear systems, solved during the AAAP initialization (9) and the global phase (7), are sparse. The associated matrices have non-zero values only between nodes and their immediate neighbors, thus can be solved using a fully distributed version of the well-known iterative Gauss-Seidel method [14,21].

VI. EXPERIMENTAL RESULTS AND COMPARISON

We now present the results of our ARAP approach applied to some sample inputs. As done by previous authors, we generated synthetic inputs on 200 points distributed uniformly within 2D regions of different shapes, endowed with the connectivity of a "disk graph", meaning that all pairs of sensors closer than a parameter r are connected. We used two regions: a simple square, and a non-convex C-shaped region, and three different values of r, corresponding to sparse, medium and dense graphs, whose average number of neighbors is 4, 5 and 6, respectively. The more dense the graph, the more localizable it should be. See Figures 2 and 3. ARAP converged in less than 40 iterations in all the experiments.

To measure the localization performance of the algorithms, we simply compute the average normalized error in localization per sensor:

$$ERR = \frac{\sum_{i=1}^{N} \|p_i - g_i\|_2}{ND} \tag{10}$$

where p_i is the localization of sensor i, g_i is the gound truth location of the sensor (available in our simulation experiments), N is the number of sensors and D is the diameter of the embedding. Before applying this measure, the best rigid transformation between the embedding and the ground truth was computed and factored out. To test the sensitivity of the algorithm to noise, random noise ε_{ij} in the range $[-\sigma l_{ij}, \sigma l_{ij}]$ is added to the true edge lengths l_{ij} when constructing the measurement graph: $d_{ij} = l_{ij}+\varepsilon_{ij}$.

We compared our ARAP approach with the traditional SMACOF approach, when initialized using a Laplacian eigenvector embedding, as in [9], the LRE approach [18], and the AAAP approach, which is essentially our initial embedding for the ARAP procedure. We use the same three-stage reference patch localization procedure for LRE, AAAP and ARAP (typically requiring 20 stress minimization iterations per patch), as described in Section IV. The linear systems involved in LRE and AAAP are solved by an eigenvector calculation, and a global affine transformation is factored out after that, as described by Singer [18].

Our results are summarized in Figs. 2-4, which provide a consistent, and clear, picture of the comparative performance of the algorithms. LRE, AAAP and ARAP, who are all based on the same local reference patch localization, are capable of "reproducing" the correct embedding (up to a rigid transformation) when no noise is present and the measurement graph is dense enough. The difference between the three becomes clear immediately upon departure from this optimistic scenario. The graphs in Fig. 4 show the performance for noise levels between 0% and 10%, at densities equivalent to 4, 5, and 6 neighbors per sensor on the average. The embeddings in Figs. 2-3 were generated for inputs contaminated with 5% noise, at the same densities. These are the most interesting scenarios, as this is more or less where the transition to generic global rigidity typically occurs. The runtime required to generate each embedding

on a 1.86GHz PC with 1G RAM, along with the resulting localization error, are provided for each embedding.

Both AAAP and LRE, who allow affine transformations between patches, cannot overcome the inaccurate localizations of the patches, and even amplify the noise. AAAP is typically a little better than LRE. ARAP, on the other hand, compensates for inaccurate patch localization with rigid transformations between the patches, and is able to filter out the noise. It results in localizations which are up to an order of magnitude more accurate than the others, at the price of no more than a factor of two in runtime. Remember, also, that our versions of LRE and AAAP require an eigenvector computation and global affine transformation removal, which are not easily distributable. From our experience, distributable algorithms for achieving a similar effect, such as pinning down a number of anchor vertices and solving a simple linear system (which may be distributed by a iterative Gauss-Seidel procedure) typically produce less accurate results, so the results presented here for LRE and AAAP may be interpreted as an upper bound on the performance of any algorithm within that family. SMACOF, which starts off with a reasonable, but nowhere near accurate, embedding, tries to directly minimize stress relative to the measurement graph, but quickly gets stuck in one of many local minima, typically involving "foldovers" in the embedding.

VII. Discussion

Our new ARAP approach follows the paradigm of "think globally, fit locally". As only information about local distances is available, we are able to localize small reference patches and triangles, and then localize the entire network by fitting the patches together with rigid transformations to a global coordinate system.

It is interesting to explore the relationships between the various methods used in this paper. The iterative SMACOF tries to directly minimize the stress cost function (1). Close inspection of the SMACOF algorithm (see the excellent derivation by Gansner *et al*. [5]) shows that it is actually quite similar in spirit to our iterative ARAP algorithm, and can be considered an *edge-based* version of ARAP. By edge-based, we mean that the algorithm tries to stitch together edges rather than triangles or patches. But apart from its length, an edge does not contain much information on the local rigid structure of the graph, so is a very weak version of ARAP. Since the cost function (1) is known to be plagued with many local minima, SMACOF will quickly get stuck in a suboptimal solution.

AAAP and LRE are extremely similar. Rather than minimize a global stress function, as SMACOF does, AAAP tries to minimize the cost function (3), without constraints on $\mathbf{R}_i$, which is therefore a convex quadratic function, having a unique global minimum, obtained by solving a linear system of equations. The main difference between AAAP and LRE are whether the optimal affine transformation between a patch and the global coordinate system is solved for explicitly along with the global embedding, or solved for separately and hard-wired into the linear system for the global embedding. Koren *et al.* [11] also distinguish between versions of the PATCHWORK algorithm generating N_i linear equations per patch, which is the spirit of AAAP, and just one linear equation per patch, which is the spirit of LRE.

ARAP also minimizes cost function (3), with the non-linear constraint on $\mathbf{R}_i$. This cost function may have local minima, which ARAP will get stuck in, but these seem to be much fewer than those of the stress cost function (1), and provide more accurate localizations than the global minimum corresponding to the cost function of AAAP or LRE.

References

[1] B. Bollobas. Modern graph theory. Graduate Texts in Math. #184, Springer-Verlag, 1998.

[2] I. Borg and P.J.F. Groenen, Modern multidimensional scaling: Theory and applications, Springer-Verlag, 1997.

[3] M. Brand. Charting a manifold. Proc. Neural Information Processing Systems 15 (NIPS), 2002.

[4] T. Eren, D.K. Goldenberg, W. Whiteley, Y.R. Yang, A.S. Morse, B.D.O. Anderson and P.N. Belhumeur. Rigidity, computation, and randomization in network localization. Proc. IEEE Infocom, 2004.

[5] E. Gansner, Y. Koren and S. North. Graph drawing by stress majorization. Proc. Symp. Graph Drawing (GD'04), LNCS Vol. 3383, Springer Verlag, pp. 239--250, 2004.

[6] D. K. Goldenberg, A. Krishnamurthy, W.C. Maness,Y.R. Yang, A. Young, S. Morse, A. Savvides, B.D.O. Anderson. Network localization in partially localizable networks. Proc. IEEE Infocom, 2005, pp. 313- 326.

[7] G. H. Golub and C. van Loan. Matrix computations. Johns Hopkins Unversity Press, 1996.

[8] S.J. Gortler, A. Healy, and D. Thurston. Characterizing generic global rigidity. Unpublished, 2007. Preprint available at arXiv:0710.0926.

[9] C. Gotsman and Y. Koren, Distributed graph layout for sensor networks. Journal of Graph Algorithms and Applications, 9(3):327-346, 2005.

[10] J. C. Gower and G. B. Dijksterhuis. Procrustes problems. Oxford University Press, 2004.

[11] Y. Koren, C. Gotsman, and M. Ben-Chen. PATCHWORK: Efficient localization for sensor networks by distributed global optimization. Technical Report, 2005. Available at http://www.cs.technion.ac.il/~gotsman/patchwork-tr.pdf

[12] M. Mauve, J.Widmer and H. Hartenstein. A survey on position-based routing in mobile ad-hoc networks. IEEE Network, 15, 30-39, 2001.

[13] D. Moore, J. Leonard, D. Rus and S. Teller. Robust distributed network localization with noisy range measurements, Proc. ACM Conf. on Embedded Networked Sensor Systems (SenSys), 2004.

[14] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery. Numerical Recipes in C++: The art of scientific computing (2nd Ed). Cambridge University Press, 2002.

[15] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. Science, 290(5500):2323-2326, 2000.

[16] J.B. Saxe. Embeddability of weighted graphs in *k*-space is strongly NP-hard. Proc. 17th Allerton Conf. Commun. Control Comput. pp. 480-489, 1979.

[17] Y. Shang and W. Ruml. Improved MDS-based localization. Proc. IEEE Infocom, 2004.

[18] A. Singer. A remark on global positioning from local distances. Proceedings of the National Academy of Sciences, 105(28):9507-9511, 2008.

[19] M. Tubaishat and S. Madria. Sensor networks: An Overview. IEEE Potentials, 22: 20-23, 2003.

[20] Y. Yemini. Some theoretical aspects of location-location problems. Proc. IEEE Sympos. Found. Comput. Sci., pp. 1-8, 1979.

[21] Y. Saad. Iterative methods for sparse linear systems. SIAM, Philadelphia, Pennsylvania, 2003.

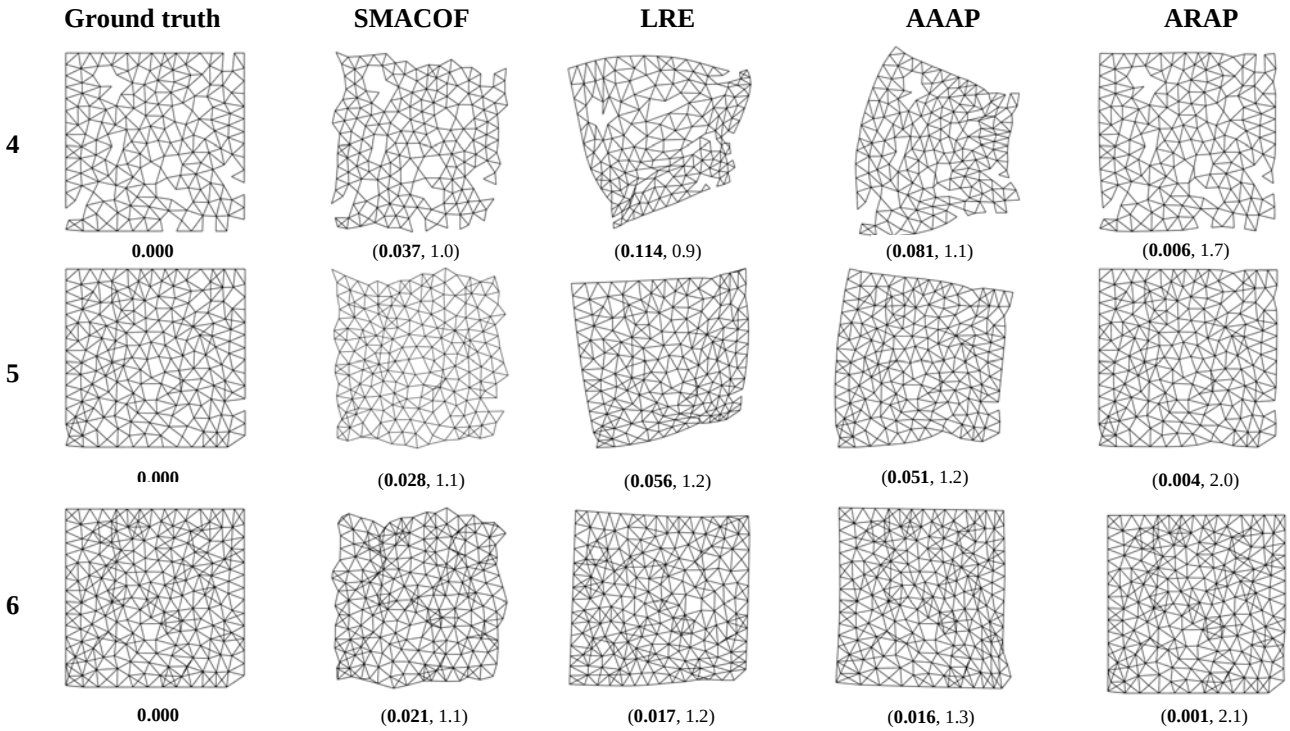


Figure 2: Localization of 200 sensors in square-shaped region by different algorithms: σ = 5%. **Columns:** Various algorithms. **Rows:** Various average number of neighbors. Inlaid numbers are localization error (in boldface) and runtime (in millisec).

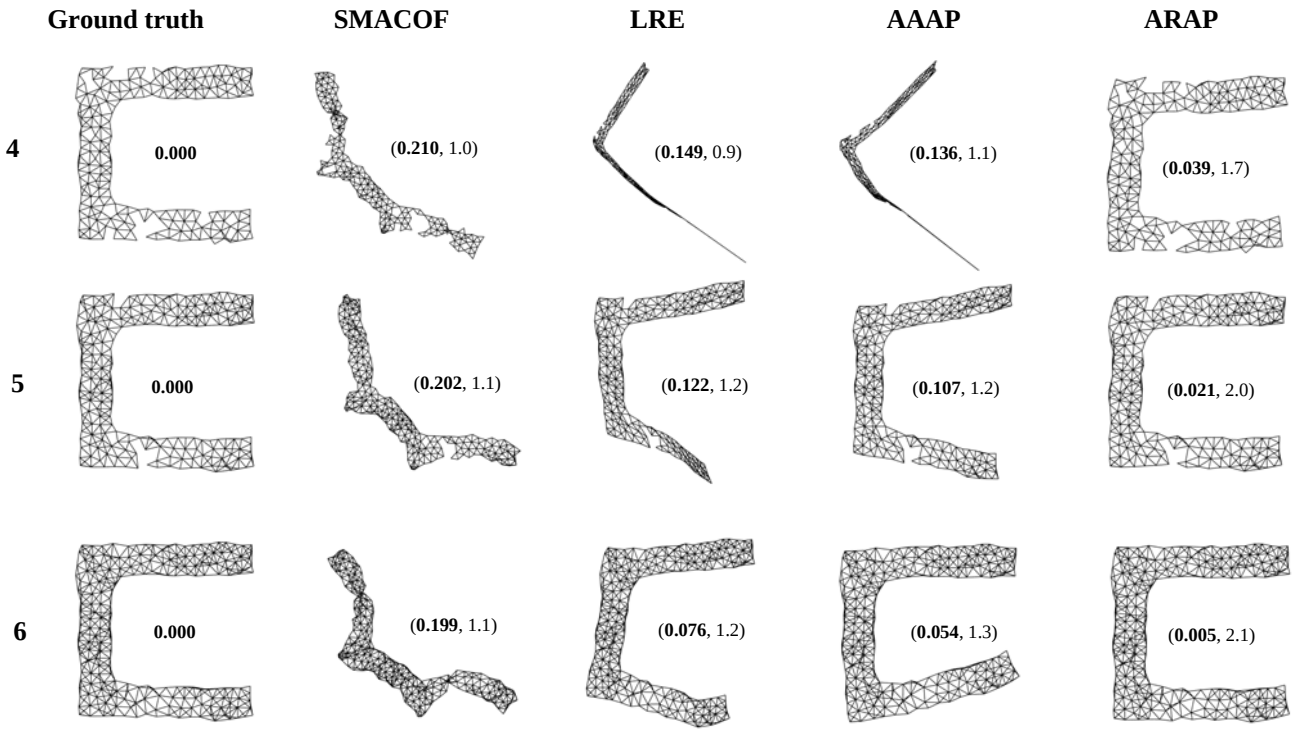


Figure 3: Localization of 200 sensors in C-shaped region by different algorithms: σ = 5%. **Columns:** Various algorithms. **Rows:** Various average number of neighbors. Inlaid numbers are localization errors (in boldface) and runtime (in millisec).

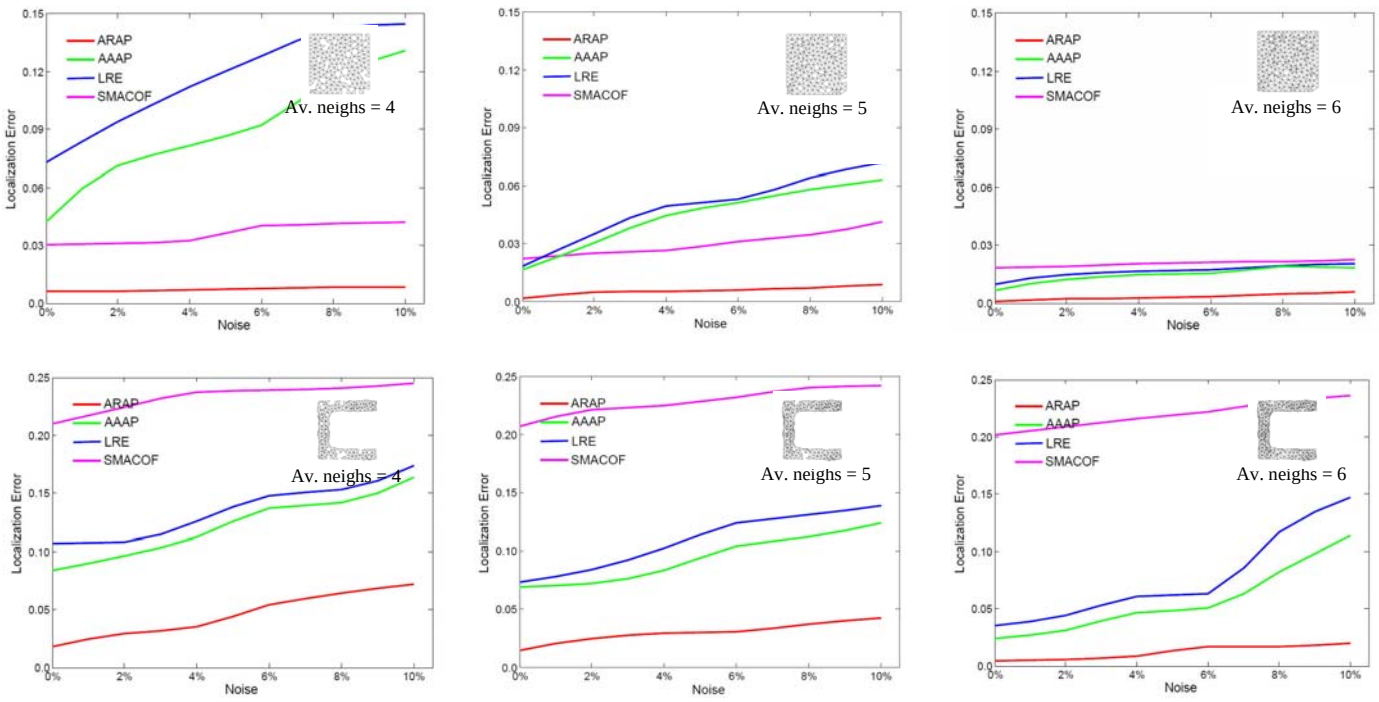


Figure 4: Performance comparison between localization algorithms. In all experiments 200 sensors were uniformly distributed in a planar region, and a measurement graph constructed by connecting any two sensors i,j such that $d_{ij} < r$. **Top row:** Square-shaped region. **Bottom row:** C-shaped region. **Left column:** Average number of neighbors = 4. **Center column:** Average number of neighbors = 5. **Right column:** Average number of neighbors = 6. The noise parameter σ along the ***x*-axis** means that the inter-sensor distances d_{ij} were corrupted by additive noise uniformly distributed in the range $[-\sigma d_{ij}, \sigma d_{ij}]$. The localization error measure along the ***y*-**axis is computed as defined in (10).