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

LEI ZHANG and LIGANG LIU
Zhejiang University
CRAIG GOTSMAN
Technion
and
STEVEN J. GORTLER
Harvard University

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.

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

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Sensor networks, as-rigid-as-possible, localization, embedding

ACM Reference Format:

Zhang, L., Liu, L., Gotsman, C., and Gortler, S. J. 2010. An as-rigid-as-possible approach to sensor network localization. ACM Trans. Sensor Netw. 6, 4, Article 35 (July 2010), 21 pages.
DOI = 10.1145/1777406.1777414 http://doi.acm.org/10.1145/1777406.1777414

This work is supported by the joint grant of the National Natural Science Foundation of China and Microsoft Research Asia (grant 60776799) and the 973 National Key Basic Research Foundation of China (grant 2009CB320801). C. Gotsman and S. Gortler were partially supported by United States – Israel Binational Science Foundation (grant 2006089).

Authors' addresses: L. Zhang and L. Liu (corresponding author), Department of Mathematics, Zhejiang University, Hangzhou 310027, China; email: cgzhanglei@gmail.com, ligangliu@zju.edu.cn; C. Gotsman, Computer Science Department, Techion, Haifa 32000, Israel; email: gotsman@cs.technion.ac.il; S. J. Gortler, Department of Computer Science, Harvard University, 33 Oxford St., Cambridge MA 02138; email: sjg@cs.harvard.edu.

DOI 10.1145/1777406.1777414 http://doi.acm.org/10.1145/1777406.1777414

1. INTRODUCTION

Sensor networks are a distributed collection of small devices, capable of a limited amount of local processing and wireless communication [Tubaishat and Madria 2003]. They are drawing more and more attention for use in a variety of applications, such as environment monitoring, military surveillance, smart places etc. [Mauve et al. 2001]. 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 sensor graph together with 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*, *layout*, or *drawing* of the measurement graph.

More formally, this article 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 [Saxe 1979; Yemini 1979], that is, 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 *globally rigid*, that is, when the coordinates are completely determined, up to a rigid transformation, from the measured distance data. If we restrict our attention to only "generic" embeddings (i.e., those which are not algebraically degenerate in any way; see Connelly [2005] for a precise definition), then the property of global rigidity depends only on the sensor graph itself and not the particular measurements. A sensor graph whose generic embeddings are globally rigid is called *generically globally rigid*. Only if the sensor graph is known to belong to a special subclass of generically globally rigid graphs, such as *trilateration graphs*, may efficient localization be provably possible [Eren et al. 2004].

Despite the inherent difficulty in the general case, the importance of the localization 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 article 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.

2. RELATED WORK

Assume the sensor network is modeled by a graph $G = (V = \{1, \ldots, n\}, E)$, 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 $P = (p_1, \ldots, p_n)$ that *realizes* all known distances. This embedding can be formally characterized as a minimum of the *stress function*.

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

Note that there might not be a unique minimum to this function, meaning that the measured distances may not uniquely characterize the embedding. 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 (measuring discrepancy between the *squared* distances). Strain is a convex function, thus may be minimized using *classical Multi-Dimensional Scaling* (MDS) [Borg and Groenen 1997], which, in practice, is an eigenvector computation. The situation changes dramatically when only a subset of the pairwise distances is available. Stress is a nonconvex function, hence, in general, we cannot find its global minimum efficiently (even if it is unique), 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.

Although sensor network localization is a nonconvex optimization problem, it can be converted to a Semi-Definite Programming (SDP) problem by relaxing the nonconvex constraints in the formulation [Biswas et al. 2006]. This approach transforms the quadratic constraints on the embedding to a matrix inequality, which can be finally rewritten as a standard SDP problem. With some assumptions on the distances between sensors, it is proven that the SDP relaxation generates a solution to the original embedding problem. Furthermore, if the embedding has a unique solution (e.g., the system is completely noise-free), the algorithm is guaranteed to discover the embedding in polynomial time [So and Ye 2005].

In the most general case, stress may be minimized using iterative *stress majorization*, also called the SMACOF algorithm [Borg and Groenen 1997]. 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 [Gotsman and Koren 2005], 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. [2004] 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 these neighborhoods generically globally rigid. Thus Moore et al. [2004] 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 this 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. [2004] 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. [2005] describe a method to find generically globally rigid subgraphs (on which they run an MDS approach), and thus avoid contaminating their output with underdetermined portions of the network.

A similar "patching" algorithm, called MDS-MAP(P,R), was described by Shang and Ruml [2004]. 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 of the patch, which is subsequently improved by stress minimization. Similarly to Moore et al. [2004], the patches are "stitched" together incrementally in a greedy order by finding the best rigid transformation between a new patch and the global embedding. A postprocessing refinement 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. [2005] solves the error accumulation problem which may affect the method of Moore et al. [2004] and MDS-MAP(P,R). 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. [2005] 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 [Brand 2002; Roweis and Saul 2000], used in machine learning.

A more recent approach of Singer [2008], 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) nonlinear system of equations. Fortunately, this nonlinear 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 performance in return for a very small increase in computational complexity.

3. THE ARAP ALGORITHM OVERVIEW

A sensor network can be modeled as a measurement graph $G = (V = \{1, \ldots, N\}, E)$, 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 $N(i) = \{j : (i, j) \in E\} \bigcup \{i\}$ 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, that is, $P = \{p_1, \ldots, p_n\}$, where

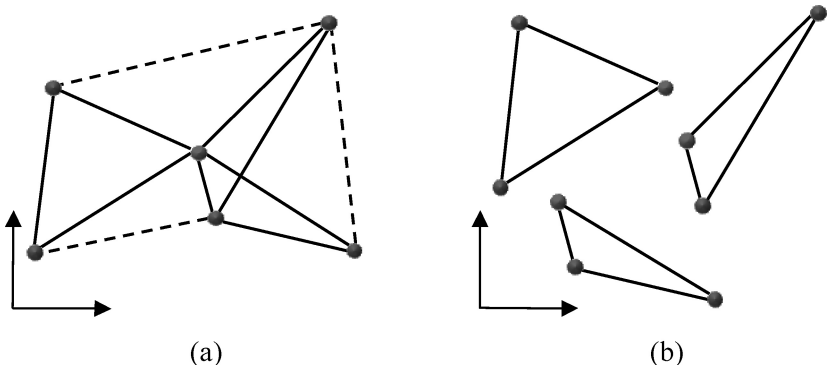

Fig. 1. Possible reference patch and triangles associated with a given vertex (it. grey dot) in a measurement graph. Solid lines are edges in the measurement graph (i.e., the length of the edge is known). Dashed lines are edges not in the measurement graph (i.e., the length of the edge is unknown).

$p_i \in \mathbb{R}^2$, 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 $t = 1, \ldots, T$, and denote their localizations in the local coordinate systems by $Q_t = \{q_i^t, q_j^t, q_k^t\}$.

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 $Q_i = \{q_i, q_{i_1}, \ldots, q_{i_{N_i-1}}\}$. 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.

4. REFERENCE TRIANGLE AND PATCH LOCALIZATION

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 [Koren et al. 2005] to produce a localization Q, which proceeds as follows.

(1) *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}.

$$B_{jk} = \min_{h:(j,h)\in E,(h,k)\in E} \{d_{jh} + d_{hk}\} \tag{2}$$

$$b_{jk} = \max\left\{\max_h\{d_{jh}\}, \max_h\{d_{hk}\}\right\} \tag{3}$$

Then, the estimate for d_{jk} is

$$d_{jk} = (B_{jk} + b_{jk})/2. \tag{4}$$

(2) *Classical MDS*. After estimating all the missing distances between sensors in $N(i)$, the reference patch can be obtained by classical MDS [Borg and Groenen 1997].

(3) *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 $i \in N(i)$. We have

$$q_i \leftarrow \frac{1}{N_i} \sum_{j\in N(i)} \left[q_j + d_{ij}(q_i - q_j)\mathrm{inv}(\|q_i - q_j\|)\right] \tag{5}$$

where

$$\mathrm{inv}(x) = \begin{cases} 1/x, & x \neq 0, \\ 0, & x = 0. \end{cases} \tag{6}$$

Remark. Note that the lower bound estimate (3) on the missing distances is derived from the disk graph assumption. Unfortunately, this lower bound is not reliable for a graph in which two nodes cannot measure distance due to an obstacle (even though they may be very close to each other). Therefore, for general graphs, we may use only the upper bound as an estimate of the missing distance in our ARAP algorithm. We show the results for this scenario, using only upper bounds, in Section 6.1. Our results show that even in this scenario, our ARAP algorithm is still superior to the competition.

5. 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 correctly 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 $Q_i = (q_i, q_{i_1}, \ldots, q_{i_{N_i}})$, where $q_{i_j} = (q_{i_j}^x, q_{i_j}^y)^T$ are the sensor locations in the reference patch $N(i)$ and by $P = (p_1, \ldots, p_N)$, a $2 \times N$ matrix, where $p_i = (p_i^x, p_i^y)^T$ is the global localization of the ith 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, that is, find also $\mathbf{R}_i$ and $\mathbf{T}_i$ for Q_i, such that

$$P_i = \mathbf{R}_i Q_i + \mathbf{T}_i. \tag{7}$$

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 ith 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). We have

$$(P, \mathbf{R}_1, \ldots, \mathbf{R}_N) = \arg \min_{P, \mathbf{R}_1, \ldots, \mathbf{R}_N} \left\{ \sum_{i=1}^{N} \|P_i - \mathbf{R}_i Q_i\|_F^2 : \mathbf{R}_i^T \mathbf{R}_i = I \right\}, \tag{8}$$

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

We now make two key observations.

(1) Given P_i and Q_i, (8) may be solved separately (and locally) for each $\mathbf{R}_i$.

$$\mathbf{R}_i = \arg \min_{\mathbf{R}} \left\{ \|P_i - \mathbf{R} Q_i\|_F^2 : \mathbf{R}^T \mathbf{R} = I \right\} \tag{9}$$

This is the least-squares version of (7), since it will not always be possible to find a $\mathbf{R}_i$ satisfying (7) exactly, as the system may be overconstrained. Sadly, (8) is a nonlinear least squares problem, because of the orthogonality constraint on $\mathbf{R}_i$. Fortunately, it still has a relatively simple solution. Solving (8) 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 [Gower and Dijksterhuis 2004]. A closed form solution is $R_i = V_i U_i^T$, where U_i and V_i are the orthogonal components of the Singular Value Decomposition (SVD) [Golub and van Loan 1996; Press et al. 2002] of $Q_i P_i^T$.

$$Q_i P_i^T = U_i \Sigma V_i^T \tag{10}$$

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

$$P = \arg \min_{P} \left\{ \sum_{i=1}^{N} \|P_i - \mathbf{R}_i Q_i\|_F^2 : P_i \text{ is induced by } P \right\} \tag{11}$$

By setting the gradients to zero, we obtain the following two $N \times N$ linear system of normal equations in the (x and y) components of P (see details in Appendix A). We have

$$\sum_{j \in N(i)} (p_i - p_j) = \frac{1}{2} \sum_{j \in N(i)} \left[\mathbf{R}_i (q_i - q_{i(j)}) + \mathbf{R}_j (q_{j(i)} - q_j) \right], \quad i = 1, \ldots, N, \quad (12)$$

where $q_{i(j)} = L_i(j), j \in N(i)$ and L_i is the local patch localization of $N(i)$ computed in Section 4.

These two observations mean that we can solve (8) using a two-phase Alternating Least-Squares (ALS) method, sometimes called a *local-global* method: In the first *local* phase, P is assumed to be fixed, and we solve (9) 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 (11) for P.

We call the ALS technique to minimize the ARAP energy (8) the "As-Rigid-As-Possible" (ARAP) localization algorithm. Since the entries of the coefficient matrix of (12) depend only on the sensor graph, this matrix is a sparse matrix which is fixed throughout ARAP. Thus we may prefactor the matrix, and reuse the factorization to efficiently solve (12) with a different right-hand side at each iteration. The observant reader may recognize this matrix as the combinatorial Laplacian matrix of the measurement graph [Bollobas 1998], and (12) as the celebrated *Poisson* equation on a graph [Press et al. 2002].

We emphasize that the ARAP energy is not increased in each of the two phases. Thus the ARAP energy is guaranteed to reach a local minimum when the local-global method is applied.

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.

5.1 AAAP Localization

At this point we make another key observation.

THEOREM. *Let G be a generically globally rigid sensor graph with* 4 *or more vertices and P a generic embedding of G. If all patches of G have been correctly localized (up to a rigid transformation), then it is possible to correctly localize G (up to a global rigid transformation) by solving* (8)*,* while allowing $\boldsymbol{R}_i$ *to be a similarity, or even affine, transformation.*

This theorem is nontrivial and its proof can be found in Appendix B. It relies on the concept of *stress matrices* in rigidity theory, and is implied by the main result of Gortler et al. [2007] that generic global rigidity of a graph implies the existence of a symmetric equilibrium stress matrix of corank 3 (and no more) for the graph. Note that the difficult part of the theorem is showing that *only* the correct embedding (up to an affine transform) will have zero energy. See more detail in Appendix B.

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 $\mathbf{R}_i$ (from rigid to similarity or affine) do not really allow the stitching stage to misbehave. The advantage of relaxing the condition on $\mathbf{R}_i$ from rigid to similarity or affine is that now (8) 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 (8), without the condition on $\mathbf{R}_i$, will be homogeneous, thus admit the trivial solutions $P = 0$ and $\mathbf{R}_i$ 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 post-processing 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 (8) when $\mathbf{R}_i$ are required to only be affine transformations. If we parameterize $\mathbf{R}_i$ as

$$\mathbf{R}_i = \begin{pmatrix} a_i & c_i \\ b_i & d_i \end{pmatrix} \tag{13}$$

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

$$\begin{aligned} p_j^x - p_i^x - \left(q_j^x - q_i^x\right)a_i - \left(q_j^y - q_i^y\right)c_i = 0 \\ p_j^y - p_i^y - \left(q_j^x - q_i^x\right)b_i - \left(q_j^y - q_i^y\right)d_i = 0 \\ i = 1, \ldots, N, j \in N(i) \end{aligned} \tag{14}$$

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 nonlinear equation.

The observant reader might ask why not solve (8) 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. [2005], but a little more general. The interested reader is referred to Section 3.2 of Koren et al. [2005] 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. [2005] compensate for this by applying standard stress majorization in a postprocessing step, but this is quite slow, and has limited effect. Thus AAAP suffices only as an initial embedding for ARAP.

5.2 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 (7). The global linear systems, solved during the AAAP initialization (14) (using a few landmarks) and the global phase (12), are sparse. The associated matrices have nonzero 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 [Press et al. 2002; Saad 2003].

6. 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 nonconvex C-shaped region, and three different values of r, corresponding to sparse, medium, and dense graphs, whose average number of neighbors is 5, 6, and 7, 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. We have

$$Err = \frac{\sum_{i=1}^{N} \| p_i - g_i \|_2}{N \times D}, \tag{15}$$

where p_i is the localization of sensor i, g_i is the ground-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: $d_{ij} = l_{ij} + \varepsilon_{ij}$.

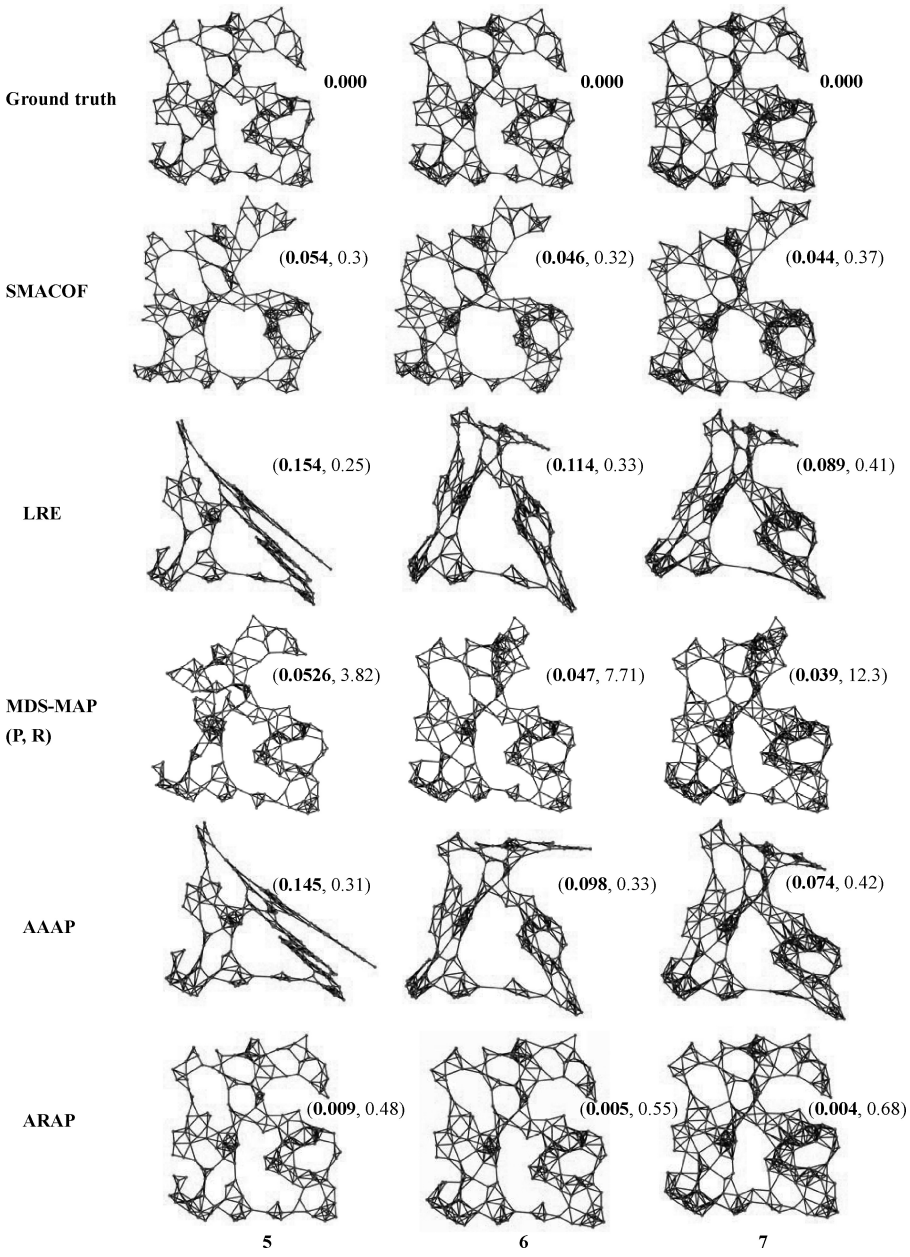

Fig. 2. Localization of 200 sensors in square-shaped region by different algorithms: with $\sigma = 5\%$ noise. Rows: Various algorithms. Columns: Various average number of neighbors. Inlaid numbers are localization error (in boldface) and runtime (in seconds).

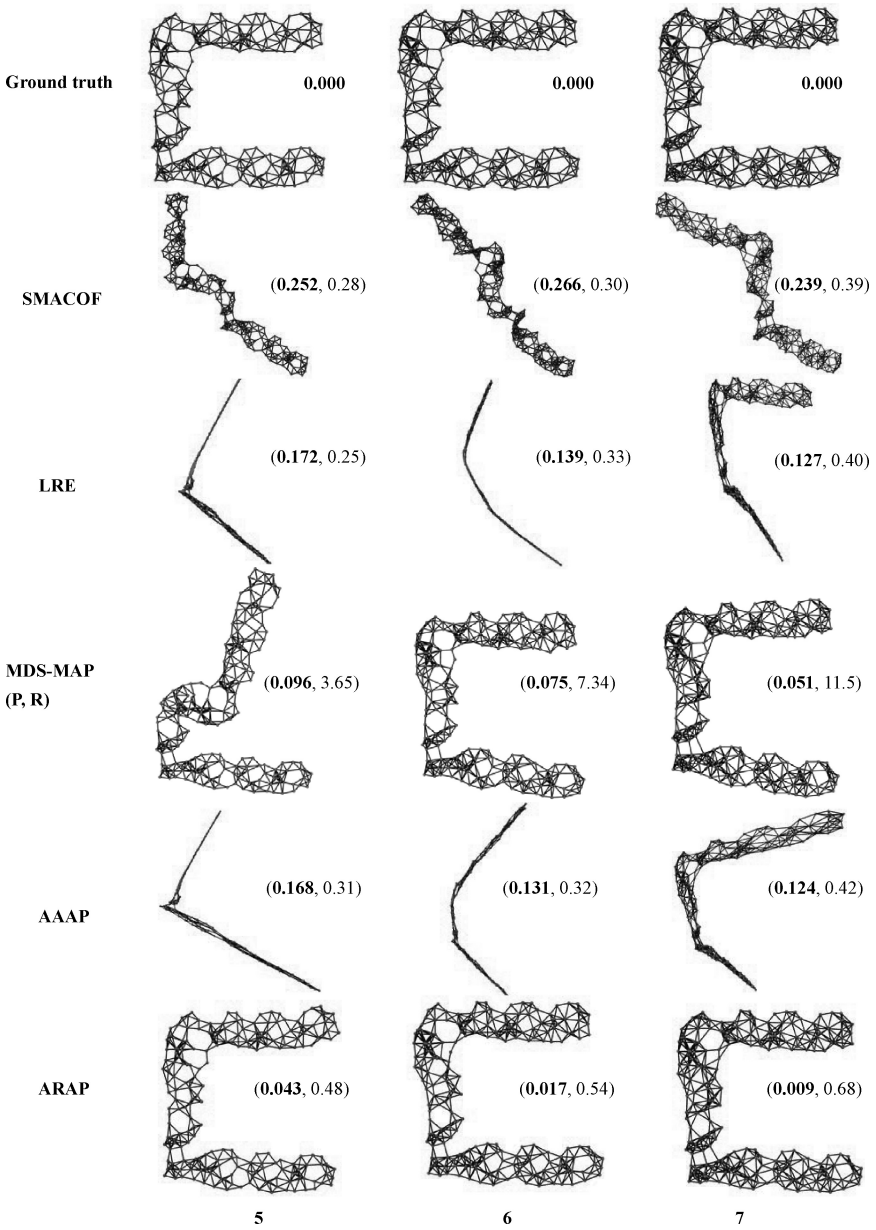

Fig. 3. Localization of 200 sensors in C-shaped region by different algorithms: with $\sigma = 5\%$ noise. Rows: Various algorithms. Columns: Various average number of neighbors. Inlaid numbers are localization errors (in boldface) and runtime (in seconds).

We compared our ARAP approach with the traditional SMACOF approach [Borg and Groenen 1997], when initialized using a Laplacian eigenvector embedding, as in Gotsman and Koren [2005], the LRE approach [Singer 2008], the MDS-MAP(P,R) approach [Shang and Ruml 2004] with one-hop neighbors (using software kindly provided by the authors), 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 4. As in Shang and Ruml [2004], we use only the upper bound B_{jk} of (4) for computing the reference patches in MDS-MAP(P,R). 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 [2008].

We applied the different algorithms to 50 randomly generated networks, and the results are summarized in Figures 2 to 4, which provide a consistent, and clear, picture of the comparative performance of the algorithms. The graphs in Figures 4 show the performance statistics (average and standard deviation) for noise levels between 0% and 10%, at densities equivalent to 5, 6, and 7 neighbors per sensor on the average. The embeddings in Figures 2 and 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 3GHz PC with 3G RAM, along with the resulting localization error, are provided for each embedding. The computation complexity of ARAP is dominated by the local step: the embedding of each patch, which is $O(nk^3)$ for n sensor nodes, where k is the average number of neighbors for the nodes. Each global step (12) requires $O(n)$ time to solve a prefactored sparse linear system.

MDS-MAP(P,R) and ARAP, which are both based on stitching together the local patches, 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 them becomes clear immediately upon departure from this optimistic scenario. As opposed to ARAP, which globally stitches all the patches in a least-squares sense, MDS-MAP(P,R) stitches the patches incrementally in a greedy order by finding the best rigid transformation between a new patch and the global embedding. The incremental algorithm may accumulate error indefinitely, especially when the measurement graph is very noisy.

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

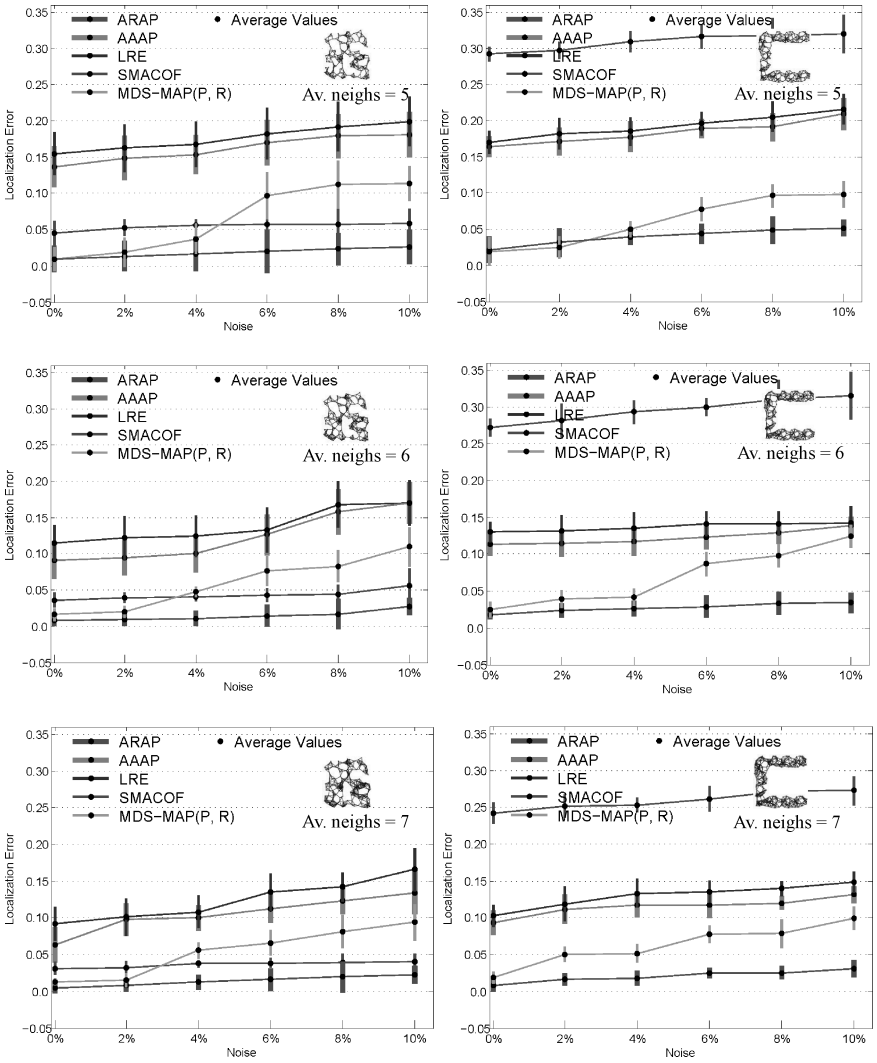

Fig. 4. Performance comparison between localization algorithms. In all experiments 200 sensors were uniformly distributed in a planar region, and a measurement graph constructed by connecting any two sensors i,j such that $d_{ij} < r$. Left column: Square-shaped region. Right column: C-shaped region. Top row: Average number of neighbors = 5. Center row: Average number of neighbors = 6. Bottom row: Average number of neighbors = 7. 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 (15). Each data point represents the average of 50 random inputs, and the short vertical lines represent the standard deviation.

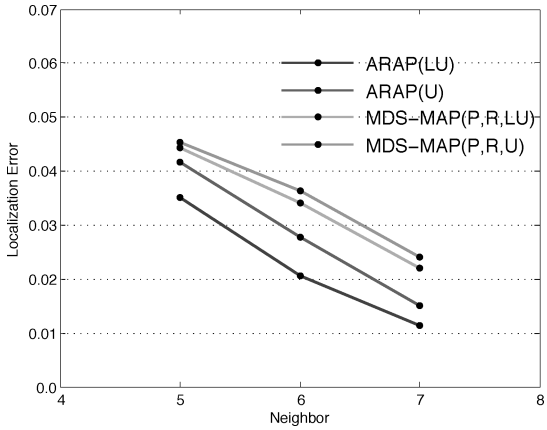

Fig. 5. Comparison between ARAP and MDS-MAP(P,R) using the noise-free sensor data in the square-shaped region. (U): only the upper bound is used to localize patches. (LU): both the upper bound and lower bound are used to localize patches.

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 an 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.

6.1 Patch Localization

Our ARAP approach uses both an upper bound and a lower bound on edge lengths for localization of local patches in the three-stage procedure as described in Section 4. The lower bound is based on the disk graph assumption. For a nondisk graph, only the upper bound is valid, as in Shang and Ruml [2004]. To perform a more fair comparison, we compared the two algorithms when just the upper bound is used, and when both are used. Figure 5 shows the results of this comparison. It can be seen that ARAP achieves better results than MDS-MAP(P,R) in both cases.

6.2 Postprocessing

Like the MDS-MAP(P,R) method, ARAP may also benefit from refinement postprocessing (this is what the "R" in (P,R) stands for) by nonlinear Levenberg-Marquardt (LM) gradient-descent optimization, which reduces the stress (Eq. (1)) of the embedding. Our experiments show that LM optimization decreases the stress slightly, although it might also slightly increase the localization error. As this optimization is very expensive, we do not think it is cost effective for ARAP. In contrast, LM optimization is essential for

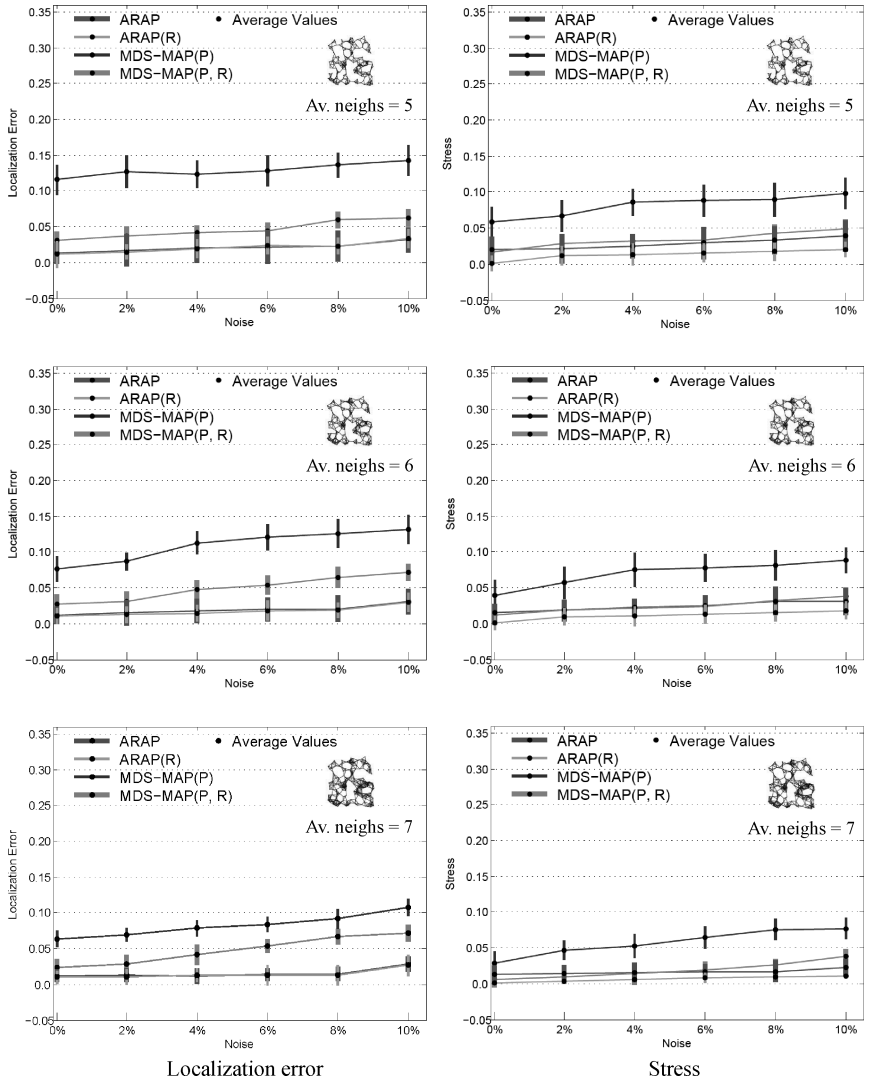

Fig. 6. Effect of LM optimization postprocessing on ARAP and MDS-MAP(P,R) using the sensor data in the square-shaped region with different levels of noise. Left column: localization error. Right column: stress. Rows are different average numbers of neighbors. ARAP: the basic approach of this article; ARAP(R): ARAP followed by LM optimization; MDS-MAP(P): MDS-MAP without LM optimization; MDS-MAP(P,R): MDS-MAP with LM optimization.

MDS-MAP(P,R). The incremental merging results are typically quite bad and may be significantly improved by LM optimization. See Figure 6 for quantification of this.

7. DISCUSSION

Our new ARAP approach follows the paradigm of "think globally, fit locally." As only information about short-range distances is available, we are able to

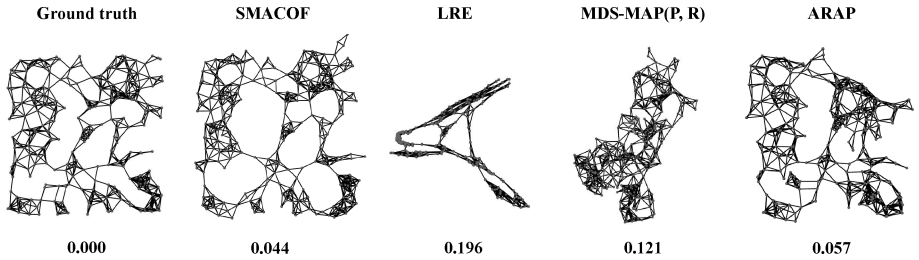

Fig. 7. ARAP (right) may get stuck in a local minimum, which generates a bad embedding. The average number of neighbors is 6, and the boldface numbers denote the localization error.

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 article. 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. [2004]) 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 minimizing a global stress function, as SMACOF does, AAAP tries to minimize the cost function (8), 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. [2005] 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.

Our ARAP algorithm is superficially similar to MDS-MAP(P,R), as they both solve the localization problem in two stages: first building a local patch for each node and then merging these local patches together to generate a global embedding. However, MDS-MAP(P,R) uses an incremental greedy algorithm for merging the local patches one by one while ARAP stitches all the local patches in a global manner, which is formulated as a global optimization problem.

ARAP also minimizes cost function (8), with the nonlinear constraint on $\mathbf{R}_i$. This cost function may have local minima, which ARAP will get stuck in, and Figure 7 shows an example of such a case. However, 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 LRE.

APPENDIXES

A.

Derivation of Eq. (12).

For a specific i, the gradient ∇_{p_i} of (11) is influenced by all the patches that contain p_i, which include $N(i)$ and all $N(j)$ with $i \in N(j)$.

For the patch $N(i)$, the corresponding energy is

$$E_i = \|P_i - \mathbf{R}_i Q_i\|_F^2. \tag{A1}$$

Setting $\nabla_{p_i} E_i = 0$ leads to

$$\sum_{j \in N(i)} (p_i - p_j) = \sum_{j \in N(i)} \mathbf{R}_i \big(q_i - q_{i(j)}\big). \tag{A2}$$

For the patch $N(j)$ that contains i, setting $\nabla_{p_i} E_j = 0$ leads to

$$p_i - p_j = \mathbf{R}_j \big(q_{j(i)} - q_j\big), \quad j \in N(i). \tag{A3}$$

Summing up all the equations in (A2) and (A3) results in the final.

$$\sum_{j \in N(i)} (p_i - p_j) = \frac{1}{2} \sum_{j \in N(i)} \big[\mathbf{R}_i (q_i - q_{i(j)}) + \mathbf{R}_j (q_{j(i)} - q_j)\big] \tag{A4}$$

B.

Proof of the Theorem of Section 5.1.

First we define the AAAP energy of an embedding as

$$(P, \mathbf{R}_1, \ldots, \mathbf{R}_N) = \arg \min_{P, \mathbf{R}_1, \ldots, \mathbf{R}_N} \left\{ \sum_{i=1}^{N} \|P_i - \mathbf{R}_i Q_i\|_F^2 \right\}, \tag{B1}$$

which is analogous to the ARAP energy (8) without the orthogonality constraints on $\mathbf{R}$.

Clearly the correct embedding has zero AAAP energy. We need to prove that the *only* solutions with zero AAAP energy are embeddings that differ from the correct solution by only a global affine transform.

Given an embedding P of a sensor graph G having n vertices, an $n \times n$ matrix Ω is called an *equilibrium stress matrix* for G with P if it satisfies the following.

(1) Ω is symmetric.
(2) Ω_{ij} is 0 if $i \neq j$ and there is no edge of G connecting vertex i and vertex j.
(3) The diagonal entries Ω_{ii} are equal to the negation of the sum of the off-diagonal entries: $\Omega_{ii} = -\sum_{j \neq i} \Omega_{ij}$. Consequently the all-ones vector is in the nullspace of Ω.
(4) The n-vector defined by the x coordinates of the embedding P is in the nullspace of Ω. The same is true for the y coordinates.

From Connelly [2005], Jackson and Jordán [2005], and Gortler et al. [2007], it is known that if G has 4 or more vertices, and is generically globally rigid in the plane, and P is generic, then there must exist an equilibrium stress matrix Ω for G with P of corank 3. (This is the largest possible rank for *any*

equilibrium stress matrix, as at least the x-coordinates, the y-coordinates, and the all-ones vector must be in the nullspace.)

By definition, for any embedding P' that has zero AAAP energy, the coordinates of each one ring in G must differ from its corresponding coordinates in P by only a local affine transform.

Now the embedding of each vertex of G is related to the embedding of its one-ring neighbors by a single row of Ω, and this relationship is invariant to all affine transforms of that one ring. Thus, if each one ring has only undergone a local affine transform, then each of the coordinates of P' must remain in the nullspace of Ω.

However, because Ω has corank 3, and the all-ones vector is contained in its nullspace, any two vectors in its nullspace must be related through a global affine transform. Thus P' is equivalent to P up to a global affine transform.

Finally, from an algorithmic point of view, we note that this affine transform can be computed (up to a Euclidean transform) by solving a least squares system using the given squared edge lengths.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments and Y. Shang for providing an implementation of MDS-MAP(P,R).

REFERENCES

Biswas, P., Lian, T. C., Wang, T. C., and Ye, Y. Y. 2006. Semidefinite programming based algorithms for sensor network localization. *ACM Trans. Sensor Netw. 2*, 2, 188–220.

Bollobas, B. 1998. *Modern Graph Theory*. Graduate Texts in Math. No. 184. Springer.

Borg, I. and Groenen, P. J. F. 1997. *Modern Multidimensional Scaling: Theory and Applications*. Springer-Verlag.

Brand, M. 2002. Charting a manifold. In *Proceedings of the Conference on Neural Information Processing Systems*. ACM, 480–489.

Connelly, R. 2005. Generic global rigidity. *Discr. Comput. Geom. 33*, 4, 549–563.

Eren, T., Goldenberg, D. K., Whiteley, W., Yang, Y. R., Morse, A., Anderson, B. D. O., and Belhumeur, P. N. 2004. Rigidity, computation, and randomization in network localization. In *Proceedings of the IEEE Infocom Conference*. IEEE Computer Society, 2673–2684.

Gansner, E., Koren, Y., and North, S. 2004. Graph drawing by stress majorization. Lecture Notes in Computer Science, vol. 3383, Springer, 239–250.

Goldenberg, D. K., Krishnamurthy, A., Maness, W. C., Yang, Y. R., Young, A., Morse, S., Savvides, A., and Anderson, B. D. O. 2005. Network localization in partially localizable network. In *Procedings of the IEEE Infocom Conference*. IEEE Computer Society, 313–326.

Golub, G. H. and van Loan, C. 1996. *Matrix Computations*. Johns Hopkins Unversity Press.

Gortler, S., Healy, A., and Thurston, D. 2007. Characterizing generic global rigidity. *Amer. J. Math*. To appear.

Gotsman, C. and Koren, Y. 2005. Distributed graph layout for sensor networks. *J. Graph Algor. Appl. 9*, 3, 327–346.

Gower, J. C. and Dijksterhuis, G. B. 2004. *Procrustes Problems*. Oxford University Press.

Jackson, B. and Jordán, T. 2005. Connected rigidity matroids and unique realizations of graphs. *J. Comb. Theory Ser. B 94*, 1, 1–29.

Koren, Y., Gotsman, C., and Ben-Chen, M. 2005. Patchwork: Efficient localization for sensor networks by distributed global optimization. Tech. rep. http://www.cs.technion.ac.il/ gotsman/AmendedPubl/Koren/patchwork–tr.pdf.

Mauve, M., Widmer, J., and Hartenstein, H. 2001. A survey on position-based routing in mobile ad-hoc networks. *IEEE Netw. 15*, 6, 30–39.

MOORE, D., LEONARD, J., RUS, D., AND TELLER, S. 2004. Robust distributed network localization with noisy range measurements. In *Proceedings of the Conference on Embedded Networked Sensor Systems (SenSys)*. ACM, 50–61.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 2002. *Numerical Recipes in C++: The Art of Scientific Computing* 2nd ed. Cambridge University Press.

ROWEIS, S. AND SAUL, L. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Sci. 290*, 5500, 2323–2326.

SAAD, Y. 2003. *Iterative Methods for Sparse Linear Systems* 2nd ed. Society for Industrial and Applied Mathematics.

SAXE, J. B. 1979. Embeddability of weighted graphs in k-space is strongly NP-hard. In *Proceedings of the 17th Allerton Conference on Communication, Control and Computing*. Society for Industrial and Applied Mathematics, 480–489.

SHANG, Y. AND RUML, W. 2004. Improved MDS-based localization. In *Proceedings of the IEEE Infocom Conference*. IEEE Computer Society, 2640–2651.

SINGER, A. 2008. A remark on global positioning from local distances. *Proc. Nation. Acad. Sci. 105*, 28, 9507–9511.

SO, A. M.-C. AND YE, Y. 2005. Theory of semidefinite programming for sensor network localization. In *Proceedings of the Symposium on Discrete Algorithms*. SIAM, 405–414.

TUBAISHAT, M. AND MADRIA, S. 2003. Sensor networks: An overview. *IEEE Potentials 22*, 2, 20–23.

YEMINI, Y. 1979. Some theoretical aspects of position-location problems. In *Proceedings of the IEEE Symposium on Foundations of Computer Science.* IEEE Computer Society, 1–8.

Received February 2009; revised September 2009; accepted November 2009