Sensor Network Localization Using Sensor Perturbation

YUANCHEN ZHU and STEVEN J. GORTLER
Harvard University
and
DYLAN THURSTON
Columbia University

Sensor network localization is an instance of the NP-HARD graph realization problem. Thus, methods used in practice are not guaranteed to find the correct localization, even if it is uniquely determined by the input distances.

In this paper, we show the following: if the sensors are allowed to wiggle, giving us perturbed distance data, we can apply a novel algorithm to realize arbitrary generically globally rigid graphs (GGR), or certain vertex subsets in non-GGR graphs whose relative positions are fixed (which include vertex sets of GGR subgraphs). And this strategy works in any dimension.

In the language of structural rigidity theory, our approach corresponds to calculating the approximate kernel of a generic stress matrix for the given graph and distance data. To make our algorithm suitable for real-world application, we also present (i) various techniques for improving the robustness of the algorithm in the presence of measurement noise (ii) an algorithm for detecting certain subsets of graph vertices whose relative positions are fixed in any generic realization of the graph and robustly localizing these subsets of vertices (iii) a strategy for reducing the number of measurements needed by the algorithm.

We provide simulation results of our algorithm.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

General Terms: Algorithm, Performance, Theory

Additional Key Words and Phrases: Generic global rigidity, globally linked, localization, rigidity theory, sensor networks, sensor perturbation

1. INTRODUCTION

In this paper, we revisit the problem of determining the 2D geometric coordinates of a set of sensors in a network, where we have the ability to measure the distances between only certain pairs of sensors. This can be modeled as a graph realization problem.

Due to the importance of the problem, many effective heuristic strategies and

A preliminary version of this article appeared in the *Proceedings of the 28th Conference on Computer Communications (IEEE INFOCOM 2009)*, April 2009, Rio de Janeiro, Brazil, pp. 2796–2800.

numerical algorithms have been proposed [Shang et al. 2003; Doherty et al. 2001; Biswas et al. 2006; Niculescu and Nath 2003; Eren et al. 2004; Hendrickson 1995; Moore et al. 2004; Singer 2008]. Note that GRAPH-EMBEDDABILITY [Saxe 1979] is NP-hard, then (unless P=NP) there is no efficient algorithm for solving the corresponding search problem, graph realization, either. Thus, such methods are not guaranteed to find the correct (or even approximate) localization, even if it is uniquely determined by the input distances. We do note that some methods are guaranteed to work on special families of input. For example, trilateration will work on "trilateration graphs" [Eren et al. 2004], and as explored in [So and Ye 2007; Zhu et al. 2010], and methods based on semi-definite programming such as [Biswas et al. 2006] will work on input that is "universally rigid". Both types of input are proper subsets of "globally rigid" input.

In this paper we investigate a variant of the problem, where we are able to gather more data, in the form of perturbation data. In particular, we assume that the sensors are repeatedly able to move by small amounts, and the distances along the graph edges are repeatedly measured. We do not need to know anything about the directions or specific distances of the movement, as long as the motions are not too large or too small.

Our main theoretical result is that with enough perturbation data, of high enough accuracy, we are guaranteed to be able to localize any "generically globally rigid" (GGR) graph up to an unknown Euclidean transform. The main theoretical tool we use is the "stress normal property" from the structural rigidity literature. This property tells us that the space of "equilibrium stresses" of an embedding of a graph is the orthogonal complement to the tangent space of the image of a certain map. From the perturbation data, we can estimate this tangent space, and thus the space of equilibrium stresses. From these stresses, we are then able to correctly localize the graph. In this paper, we also discuss the class of vertex subsets that we will be able to localize for a given non-GGR graph. We say that these vertex subsets are "stress kernel localizable" (SKL). Moreover, SKL subsets are guaranteed to contain vertex sets of any GGR subgraph.

Although it is important to know that the algorithm will work for "generic" embeddings and under no-noise conditions, in practice, there will of course be noise, and we may be sufficiently close to some singular embedding. In this paper we discuss practical methods to estimate the relevant quantities in order to deal with measurement noise. We also show experiments documenting the behavior of our approach under noise.

Our guaranteed algorithm needs essentially quadratic number of pair-wise distance measurements (a linear number of measurements along the linear number of edges in the network graph). However we show how in practice, a linear number of measurements can often suffice.

Even though our approach requires many measurements, with faster measurement technology, this novel approach may become more practical in the future. Moreover, our approach can also be used to augment current methods, by applying our method only on the difficult sub-sections of a network. Finally, our method immediately extends to 3 and higher dimensions.

2. RELATED WORK

Saxe [1979] showed that Graph-Embeddability and Global-Rigidity in any dimensions is NP-HARD. The Graph-Embeddability problem asks: Given a dimension d, a graph, and a set of desired distances attached to the edges, whether these distances can be realized (even approximately) by a straight line drawing of the graph in $\mathbb{R}^d$? Clearly if one had an algorithm that could always find the embedding whenever it existed, then one could answer the embeddability question, and thus finding embeddings must be at least as difficult as the decision problem. The Global-Rigidity problem asks: Given a graph embedded in $\mathbb{R}^d$, is there a second embedding in $\mathbb{R}^d$ with the same edge lengths? The difficulty of this problem implies that even determining whether a graph realization problem is well-posed is an intractable problem.

Fortunately, the proofs of these results depend on clever arrangements of the graph vertices. Later researchers have shown that if the coordinates of the vertices are *generic*, i.e., there is no algebraic dependencies among the coordinates of vertices of the graph, then the global rigidity problem becomes tractable. Connelly gave a set of sufficient conditions for global rigidity of a generic framework [2005], and Hendrickson described a set of necessary conditions for a generic framework to be globally rigid [1992]. Jackson and Jordan showed that in 2D, these conditions imply each other, and so completely characterize generic global rigidity in 2D [2005]. Gortler et al. [2008] proved that for all dimensions, Connelly's sufficient conditions are also necessary, and thus gave a complete characterization of generic global rigidity in any a fixed dimension. These results also imply that for a given graph and a fixed dimension, either all of its generic embeddings are globally rigid or none of them are. Thus generic global rigidity is a property of the graph, and not any particular embedding. Finally, these results lead to an efficient randomized algorithm for deciding if a graph is GGR [Gortler et al. 2008]. Note, though, that knowing that a graph is GGR does not make the Graph-Embeddability or the graph realization problem any more tractable (for example, the cycle graph used for the reduction in [Saxe 1979] is GGR).

Even though graph realization is an NP-HARD problem, due to its practical importance, various heuristic and numerical methods have been proposed over the years. Here we describe a few notable methods.

Optimization: Shang et al. [2003] apply classical multi-dimensional scaling (MDS) to the problem of sensor network localization. Because classical MDS requires all-pair distances information, they use a shortest path algorithm to fill in the missing distances, which introduces error. Doherty et al. [2001] model range and angular constraints in sensor network localization as convex constraints. The resulting minimization problem can be solved efficiently using semi-definite programming (SDP). However, their range constraint only forces edge lengths to be shorter than certain amounts. Thus the method requires anchor nodes to be either placed at the boundary or distributed densely to achieve good results. Also using SDP, Biswas et al. [2006] incorporate a variance maximizing energy to spread out the vertices. However, this SDP method cannot, in general, enforce that the result is 2-dimensional, so it must essentially be followed by a 2D PCA step, and numerical optimization (e.g. gradient descent). An embedding is called *universally* *rigid* if it is the only embedding (up to a Euclidean transformation) with the same edge lengths in *any* dimensional space. For such embeddings, the dimensionality constraint comes for free from the data, and SDP methods are guaranteed to work [So and Ye 2007; Zhu et al. 2010]. However, there exist GGR graphs with generic embeddings that are not universally rigid. For example, one can embed the 4-cycle generically in one dimension such that it can be "unfolded" when given two dimensions (see also [Gortler and Thurston 2010]).

Multilateration: In $\mathbb{R}^d$, given the positions of $d+1$ or more anchor vertices, a vertex can be localized through multilateration if its distances to the anchor vertices are known. Niculescu and Nath [2003] propose a distance-vector style algorithm that propagates distance-to-anchor information and then uses multilateration to recover vertex positions. Eren et al. [2004] formalize the concept of trilateration graphs, which are efficiently localizable via trilateration. However, trilateration and multilateration graphs are also a strict subset of GGR graphs. The fact that multilateration graphs are actually universally rigid for all generic embeddings follows from the universal rigidity of the simplex and is discussed in [Zhu et al. 2010]. Multilateration often cannot propagate through poorly connected parts of graphs, and requires higher average vertex degree or higher anchor vertex density to localize the entire graph.

Divide, conquer, and stitch: In the context of molecular conformation, Hendrickson [1995] divides a graph into GGR subgraphs, localizes each subgraph separately, and then puts together the pieces. More recently, Moore et al. [2004] also first compute subgraph localization, and then piece them together by computing matching transformation between pairs of localized subgraphs. See also [Biswas and Ye 2006; Carter et al. 2007] for alternative divide-conquer-stitch schemes. The geometric constraint community (see [Gao et al. 2002] for example) has also worked on algorithms that solve for geometric configuration satisfying various constraints, including constraints on distances. Often they solve systems of algebraic equations to find the solution to subproblems once the original problem has been decomposed into small enough ones. Very recently, Singer [2008] suggests a method for consolidating subgraph realization into a global graph realization by computing eigenvectors of a special matrix derived from the subgraph realization. He also notes an alternative construction, which essentially corresponds to building a single stress matrix and computing its kernel. Divide and conquer methods can fail on a GGR graph, when the subgraphs themselves are not GGR, and thus cannot be correctly localized. Our method is also based on computing the kernel of stress matrices, although we do not require any subgraph realization to compute the stress matrix.

A recent paper [Goldenberg et al. 2005] discusses the notion of localizable nodes in a graph. This idea essentially boils down to looking for GGR subgraphs. In this paper, we explore the more general notion of "globally linked subsets of vertices" which captures the most general notion of which vertices of a graph have their positions uniquely determined (up to a Euclidean transform) from the edge lengths of a graph. It turns out that the class of vertex subsets localizable by our methods falls between this most general class and GGR subgraph vertex sets.

3. METHOD OVERVIEW

Here we give a brief overview of our approach. Details follow below.

$$\begin{tikzcd}
\textbf{wiggle} \arrow[r] & \textbf{tangent} \arrow[r] & \textbf{normal} \arrow[dll] \\
\textbf{stress matrix} \arrow[r] & \textbf{affine embedding} \arrow[r] & \textbf{Euclidean embedding}
\end{tikzcd}$$

By taking (squared) length measurements along each of the e edges in the network graph, we obtain a point in $\mathbb{R}^e$. As we wiggle the sensors, we obtain a cloud of points in $\mathbb{R}^e$. Each of these is a point on the "measurement set". The measurement set is the set of points in $\mathbb{R}^e$ representing all possible consistent measurements from graph embeddings in d dimensions (in our case 2 dimensions). The measurement set will be of dimension lower than e (as not all e tuples represent valid measurements). We can then fit a plane (of appropriate dimension) through this cloud (using PCA) to obtain an approximation to the tangent of the measurement set. Once we have a representation of the tangent space, we can take its orthogonal complement to obtain a representation of its normal space.

We now use some facts from the rigidity literature to show that this data can be used to obtain the embedding. First, we use a classical fact from the rigidity literature that each vector in the normal space represents a so-called "stress-matrix" of the graph embedding. We put this together with a recent result in the rigidity literature that tells us that when the graph is GGR, then almost every one of its stress matrices will have an appropriately small kernel [Gortler et al. 2008]. This gives us the embedding up to an unknown affine transform. Finally we use another relatively recent result that tells us that when the graph is GGR, we can use the known edge lengths to remove the affine ambiguity and recover the embedding up to an unknown Euclidean transform [Connelly 2005].

Finally, we can extend our algorithm to non-GGR graphs by examining subspaces of the normal space induced by vertex subsets of the graph. This prompts us to define "stress kernel localizable" sets which precisely categorizes vertex subsets localizable by our method.

4. THEORY

In this section we first fix some notations that will be used through out the paper. Most can be found in [Connelly 2005; Gortler et al. 2008]. Then we describe the theory behind our approach.

4.1 Definitions

A *graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a pair consisting of a set of v *vertices*, $\mathcal{V} = \{1, \ldots, v\}$, and a set of e undirected *edges*, $\mathcal{E}$. Elements in $\mathcal{E}$ are two-element (unordered) subsets of $\mathcal{V}$.

A *framework* is a graph $\mathcal{G}$ together with a mapping from $\mathcal{V}$ to $\mathbb{R}^d$, assigning a position in $\mathbb{R}^d$ to each vertex. We use $C^d(\mathcal{G})$ to denote the space of frameworks for a given graph $\mathcal{G}$ and a given dimension d. Given a framework $\rho \in C^d(\mathcal{G})$, $\rho(i)$ is then the position of vertex i in $\mathbb{R}^d$. Two frameworks $\rho, \rho' \in C^d(\mathcal{G})$ are *congruent*, denoted by $\rho \cong \rho'$, if there exists $R \in \text{Euclid}(d)$ such that $\rho = R \circ \rho'$, where $\text{Euclid}(d)$ is the space of Euclidean transforms in $\mathbb{R}^d$. We also give $C^d(\mathcal{G})$ a metric by trivially identifying $C^d(\mathcal{G})$ with $\mathbb{R}^{vd}$.

The *length-squared function*, $l : C^d(\mathcal{G}) \to \mathbb{R}^e$, is a function that takes in a framework ρ and outputs a e-dimensional vector, assigning to each edge in $\mathcal{G}$ its squared length, i.e., $l(\rho)^{\{i,j\}} = \|\rho(i) - \rho(j)\|^2$ for $\{i, j\} \in \mathcal{E}$. Note that we index coordinate components of vectors in $\mathbb{R}^e$ by edges in $\mathcal{E}$.

A framework ρ is *rigid* if there exists $\delta > 0$ such that $\|\rho' - \rho\| < \delta$ and $l(\rho) = l(\rho')$ together implies $\rho \cong \rho'$. This captures the idea that the graph cannot continuously flex. A framework ρ is *globally rigid* if $l(\rho) = l(\rho')$ always implies $\rho \cong \rho'$. In graph realization, we want to compute ρ up to a Euclidean transform given $l(\rho)$. Thus the graph realization problem is well-defined if and only if ρ is globally rigid.

A subset of vertices $\mathcal{S}$ is called *globally linked* in a framework ρ if, in all frameworks of $\mathcal{G}$ with the same edge lengths as ρ, the embeddings of $\mathcal{S}$ are congruent to each other [Jackson et al. 2006]. Note, a vertex subset can be globally linked in a framework, even if they do not induce a globally rigid sub-framework of G, because *all* of the edges of $\mathcal{E}$ can indirectly constrain the embedding of $\mathcal{S}$ (see Figure 1).

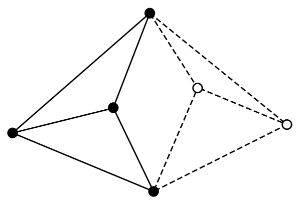


Fig. 1. The filled vertices are globally linked in the framework, but the filled vertices and solid edges alone do not form a globally rigid framework.

A framework $\rho \in C^d(\mathcal{G})$ is *generic* if the coordinates of its vertex positions, i.e., elements of the matrix $(\rho(1), \dots, \rho(v))$, do not satisfy any algebraic equation with rational coefficients. Focusing on generic frameworks allows one to avoid various rare singularities and coincidences.

4.2 Measurement Tangent Space

We denote by $l(C^d(\mathcal{G})) \subseteq \mathbb{R}^e$ the set of all length squared measurements of all frameworks of the graph in R^d and call it the *measurement set* of $\mathcal{G}$ in d dimension. This set is semi-algebraic and thus, is a manifold at $l(\rho)$ for almost every (and for all generic) ρ. The exceptions form a measure zero set, which satisfies some low order polynomial equation, so such singular frameworks are exceedingly rare. For a non-singular ρ, there is a well defined *measurement tangent space* of $l(C^d(\mathcal{G}))$ at $l(\rho)$, which we will denote by $T(\rho)$.

In our sensor network localization approach, we will assume that we can measure not only $l(\rho)$ but also $l(\rho + \Delta_i)$ for some sufficient number of Δ_i's. With such measurements we can then use PCA to fit an approximate $T(\rho)$ to these points. In the limit, with smaller and smaller perturbations, and no noise, this will converge to the correct linear space for all generic ρ.

4.3 The Stress Normal Property

The linearization (Jacobian) of l at point ρ is $l^*_\rho : \mathbb{R}^{vd} \to \mathbb{R}^e$, and its matrix representation with respect to the standard bases in $\mathbb{R}^{vd}$ and $\mathbb{R}^e$ is called the *rigidity matrix.* For non-singular ρ, we have $T(\rho) = l^*_\rho(\mathbb{R}^{vd})$. In other words, the measurement tangent space is equal to the image of the Jacobian, and is represented by the column space of the rigidity matrix.

A *stress vector* of a framework ρ is a vector $\omega \in \mathbb{R}^e$ such that for each $i \in \mathcal{V}$,

$$\sum_{\{i,j\}\in\mathcal{E}} \omega^{\{i,j\}}(\rho(j) - \rho(i)) = 0. \tag{1}$$

In other words, the position of each vertex is described as the weighted average of the positions of its neighbors, with weights given by ω. (And recall that our edges are undirected, so $\omega^{\{i,j\}} = \omega^{\{j,i\}}$.) We denote the vector space of all stress vectors of the framework ρ by $S(\rho)$ and call it the *stress space* at ρ.

The connection between the measurement tangent at $l(\rho)$ and the stresses $S(\rho)$ is described by what we call the "stress normal property":

Stress Normal Property. *For all ρ, $S(\rho)$ is the orthogonal complement to $l^*_\rho(\mathbb{R}^{vd})$ in $\mathbb{R}^e$.*

This fact can be proved with straightforward calculation. It has been used explicitly (see for example Lemma 2.5 of [Connelly 2005]) and implicitly (see for example page 186 of [Asimow and Roth 1979]) in the structural rigidity literature for many years. And thus, for generic ρ, $S(\rho)$ is the orthogonal complement to $T(\rho)$ in $\mathbb{R}^e$.

We can now conclude, that from the measurement tangent, we can compute the stress space for a generic framework.

4.4 Affine Structure from Stresses

A *stress matrix* of a framework ρ is a rearrangement of a stress vector into a matrix form, with suitably chosen diagonal entries. It is a $v \times v$ symmetric real matrix Ω satisfying: (i) For all $i, j \in \mathcal{V}$: $\Omega_{i,j} = \omega^{\{i,j\}}$ if $i \neq j$ and $\{i, j\} \in \mathcal{E}$; (ii) For all $i, j \in \mathcal{V}$: $\Omega_{i,j} = 0$ if $i \neq j$ and $\{i, j\} \notin \mathcal{E}$; (iii) $\Omega\,(1, \ldots, 1)^T = 0$. Because of the bijective correspondence between stress vectors and stress matrices, we will write $\Omega \in S(\rho)$. For Ω a stress matrix, the *stress kernel* $K(\Omega)$ is the kernel of Ω.

From the definition, for any $\Omega \in S(\rho)$, we can state that $K(\Omega)$ contains $\mathbf{1} = (1, \ldots, 1)^T$ and the column space of $(\rho(1), \ldots, \rho(v))^T$. This corresponds to the fact that every stress matrix in $S(\rho)$ is also a stress matrix for any ρ' related to ρ by some d-dimensional affine transform. Suppose now that there exists an $\Omega \in S(\rho)$ with $\dim K(\Omega) = d + 1$, i.e., it contains only the vectors spanned by the vectors mentioned above, then we say that Ω has a *minimal stress kernel.*

Suppose we have computed a basis for such a minimal stress kernel of Ω, denoted by $\{\mathbf{1}, \mathbf{x}_1, \ldots, \mathbf{x}_d\}$. Define $\rho' \in C^d(\mathcal{G})$ such that $\rho'(i)$ is the ith column of $(\mathbf{x}_1, \ldots, \mathbf{x}_d)^T$. Then ρ must be related to ρ' by a d-dimensional affine transformation. We have thus successfully localized our framework up to an affine transform.

In fact [Gortler et al. 2008, Lemma 5.8], for a fixed graph $\mathcal{G}$ and dimension d, $\dim K(\Omega)$ will be the same for almost any randomly selected stress matrix Ω of any randomly selected framework $\rho \in C^d(\mathcal{G})$. So we can use almost any $\Omega \in S(\rho)$.

Moreover, we can say that framework or even the graph itself has a minimal stress kernel.

If the graph does not have a minimal stress kernel, all is not lost. First, suppose $\mathbf{e}_1, \dots, \mathbf{e}_v$ are the standard basis vectors of $\mathbb{R}^v$. Then, given a vertex subset $\mathcal{S}$, the natural inclusion $\mathcal{S} \subseteq \mathcal{V}$ induces a projection $\pi_{\mathcal{S}} : \mathbb{R}^v \to \mathbb{R}^v$ such that $\pi_{\mathcal{S}}(\mathbf{e}_i) = \mathbf{e}_i$ if $i \in \mathcal{S}$ and $\pi_{\mathcal{S}}(\mathbf{e}_i) = 0$ otherwise. Now if for some $\Omega \in S(\rho)$, the stress kernel $K(\Omega)$ is in fact of dimension $d+1$ when projected onto the coordinates corresponding to $\mathcal{S}$, i.e., $\dim \pi_{\mathcal{S}} K(\Omega) = d+1$, then, using the same reasoning as above we can compute the coordinates of the vertices in $\mathcal{S}$ up to an affine transform from this *projected stress kernel* $\pi_{\mathcal{S}} K(\Omega)$. When there is an $\Omega \in S(\rho)$ such that $\dim \pi_{\mathcal{S}} K(\Omega) = d+1$, we say that $\mathcal{S}$ has a *minimal projected stress kernel.*

To find maximum subsets with a minimal projected stress kernel, we start by finding a seed component $\mathcal{S}_0$ with $d+1$ vertices satisfying the condition, and then we sequentially (order does not matter for linear dependence) add any vertex i to the component as long as $\dim \pi_{\mathcal{S}_0 \cup \{i\}} K(\Omega)$ remains $d+1$, until there is none that can be added without increasing the dimension. Conveniently, (again from [Gortler et al. 2008, Lemma 5.8]) all generic stresses of all generic frameworks of a given graph in a given dimension behave identically with respect to this property. So instead of using the noisy input distance data, we find such subsets using exact distances on a random test framework of the graph that we generate ourselves.

Success condition for GGR case: It turns out that the condition of having a minimal stress kernel is in fact equivalent to the concept of generic global rigidity. In particular [Gortler et al. 2008] proves the following two theorems.

THEOREM 1. *For a graph $\mathcal{G}$ with $d+2$ or more vertices, either all generic frameworks $\rho \in C^d(\mathcal{G})$ are globally rigid in $\mathbb{R}^d$, or none of them are in $\mathbb{R}^d$. Thus we can call this condition* generic global rigidity in $\mathbb{R}^d$ *and consider it as a property of the graph.*

With this in mind, a graph that is globally rigid for all generic frameworks is called a *generically globally rigid* (GGR) graph.

THEOREM 2. *For a graph $\mathcal{G}$ with $d+2$ or more vertices, $\mathcal{G}$ is GGR in $\mathbb{R}^d$ if and only if $\mathcal{G}$ has a minimal stress kernel in d-dimensions.*

Thus if a graph is GGR then we can localize it up to an affine transform.

Success condition for non-GGR case: Likewise, if a vertex subset $\mathcal{S} \subset \mathcal{V}$ has minimal projected stress kernel then we can localize these vertices in ρ up to an affine transform.

4.5 Euclidean Structure

We will now use the edge lengths to remove this free affine transform in the coordinates of $\mathcal{V}$ (resp. of $\mathcal{S}$).

Let ρ, the correct answer, be related to ρ', the computed answer by a d-dimensional affine transformation. Let us denote by L the d by d matrix representing the linear part of this affine transform.

Consider the following system of equations, in the unknown $d \times d$ matrix L.

$$\forall \{i,j\} \in \mathcal{E} : \|L(\rho'(i) - \rho'(j))\|^2 = l(\rho)^{\{i,j\}} \tag{2}$$

We know there must be some solution L.

To solve this system let $M = L^T L$ giving us the following set of linear equations (in the $\frac{d(d+1)}{2}$ unknowns of M):

$$\forall\{i,j\} \in \mathcal{E} : (\rho'(i) - \rho'(j))^T M(\rho'(i) - \rho'(j)) = l(\rho)^{\{i,j\}}. \tag{3}$$

(For the case of a subset of vertices, we define $\mathcal{E}$ to be the set of edges with both vertices in the subset $\mathcal{S}$).

Using Cholesky decomposition on M then yields L.

Success condition for GGR case: The only remaining concern is whether this system has more than one solution. Fortunately, it was proved in Prop 4.3 of [Connelly 2005] that if ρ is a generic framework of a graph where each vertex has degree at least d, then the solution to (3) must be unique. Moreover in his proof for his Theorem 1.3, Connelly shows that if a generic framework of graph with $d+2$ or more vertices has a minimal stress kernel, then each vertex must have degree at least $d+1$. Thus we will succeed for any GGR graph.

Success condition for non-GGR case: When dealing with a subset of the vertices and a projected stress kernel, we do not know of any specific guarantees of the vertex degrees, and must enforce this condition manually by repeatedly pruning out vertices with degree less than d from the subgraph. In summary: (1) if $\mathcal{S}$ has minimal projected stress kernel and (2) if each vertex in the induced subgraph is incident to d or more edges in the subgraph, then we can use use the stress kernel to localize up to an affine transform and we can use the length data on these edges to remove this affine transform leaving only a Euclidean transform. We will call such a vertex subset $\mathcal{S}$ a *stress kernel localizable* (SKL) vertex subset of $\mathcal{G}$.

If a vertex subset $\mathcal{S}$ is globally linked in all generic framework of a graph $\mathcal{G}$, we say that it is *generically globally linked* (GGL). It is straightforward to extend the proof of Connelly [2005] to conclude that a SKL subset $\mathcal{S}$ must be globally linked in all generic frameworks in $C^d(\mathcal{G})$, so we can claim that $\mathcal{S}$ is GGL. It is not clear if the converse is true, thus, we are not able to conclude that our approach will be able to localize *all* globally linked subsets of ρ, nor *all* GGL subsets of $\mathcal{G}$.

However, our method is guaranteed to localize every GGR subgraph (as in [Jackson et al. 2006]). To see why this is so, first recall that the vertex set of a GGR subgraph must meet the above two conditions (see remarks at the end of Section 4.5 and at the beginning of this section). Hence if any $d+1$ vertices from a GGR subgraph are used as the seed components in the phase of finding maximum subsets with a minimal projected stress kernel (Section 4.4), then all vertices of the subgraph must be included in the computed component, since they do not increase the dimension of the projected stress kernel. Subsequent pruning of vertices with degree less than d (Section 4.5) will never touch vertices of the GGR subgraph either.

In summary, SKL sets are the precise class of vertex subsets our algorithm can localize for a non-GGR graph. In addition, the following inclusions hold among the three collections of vertex subsets for a graph:

$$\{\text{GGR subgraph vertex sets}\} \subseteq \{\text{SKL sets}\} \subseteq \{\text{GGL sets}\}.$$

For a GGR graph, its vertex set is clearly SKL.

5. NUMERICAL ALGORITHM FOR GGR FRAMEWORKS

In this section, we describe the numerical details of our algorithm. The input to the algorithm is as follows: Suppose there is an unknown, but GGR framework $\rho \in C^d(\mathcal{G})$. (Refer to Section 6 for extension to non-GGR graphs.) We are given the graph connectivity information, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, along with length-squared measurements of edges, $l(\rho)$. In addition, we assume that we can perturb each vertex of the graph to obtain a slightly different framework, and measure its image under l. Formally, let $\delta > 0$ be a predefined parameter which we call *perturbation radius*, and define $B_\delta(\rho) = \{\rho' \in C^d(\mathcal{G}) : \|\rho'(i) - \rho(i)\| < \delta \text{ for all } i \in \mathcal{V}\}$. Then we assume that we can obtain $l(\rho_1), \ldots, l(\rho_M)$ for some fixed number, M, of random samples from $B_\delta(\rho)$. Also the distance measurements can contain noise.

5.1 Estimating the Measurement Tangent and the Stress Space

We first approximate $T(\rho)$ by sampling the manifold $l(C^d(\mathcal{G}))$ near the point $l(\rho)$ and fitting a hyperplane to these points using PCA. The samples around $l(\rho)$ are simply $l(\rho_1), \ldots, l(\rho_M)$ with each $\rho_i \in B_\delta(\rho)$. We next discuss the choice of M and δ, and mention some fine details regarding the PCA process.

Choosing M. In order to properly approximate the measurement tangent space $T(\rho)$ through PCA, we set the number of samples, M, to $\lceil \mu \dim T(\rho) \rceil$, where $\mu \geq 1$ is a *sampling factor* parameter. The exact value of μ depends on how noisy the distance measurements are. In practice, setting $\mu = 4$ often suffices for moderate level of noise. Note that $\dim T(\rho)$ is a generic property, which can be calculated from almost any random framework of $\mathcal{G}$ in $\mathbb{R}^d$. Moreover, when the framework is rigid, $\dim T(\rho)$ takes its upper bound of $(vd - \binom{d+1}{2})$. Thus the number of perturbations we need scales linearly with vd. For each perturbation ρ_i, an entire set of edge length measurements then needs to be made to get $l(\rho_i)$, so the total number of required pair-wise length measurements is $O(vde)$. In Section 7, we present a strategy for reducing the number of required measurement.

Choosing δ. Also crucial to the proper approximation of $T(\rho)$ is the perturbation radius δ. Ideally, δ should be as small as possible. Then in the limit, our approximated measurement tangent space will converge to the true $T(\rho)$. However, in practice, length measurement is noisy, and δ needs to be set large enough to dominate the noise.

PCA details. Because $l(C^d(\mathcal{G}))$ is a cone, $T(\rho)$ must include the origin. Thus we use the origin $\mathbf{0} = (0, \ldots, 0)^T$ rather than the statistical mean of the $l(\rho_i)$'s as the center for PCA. We also verified experimentally that this choice often produces more accurate results in the presence of measurement noise. We apply singular value decomposition to the e by M matrix $(l(\rho_1), \ldots, l(\rho_M))$ to get $U\Lambda V^T$. The matrix $U = (\mathbf{y}_1, \ldots, \mathbf{y}_e)$ gives an orthonormal basis for $\mathbb{R}^e$. The first $t = \dim T(\rho)$ basis vectors $\mathbf{y}_1, \ldots, \mathbf{y}_t$ span our approximation of $T(\rho)$. The rest, $\mathbf{y}_{t+1}, \ldots, \mathbf{y}_e$, forms a basis for the approximate stress space.

5.2 Stress Kernel from Measurement Tangent

The kernel size is the same for any generic stress matrix in $S(\rho)$, so we could take a random stress matrix Ω from the stress space and compute its kernel. With no noise, we know that the if $\mathcal{G}$ is GGR, then the kernel of Ω will be of dimension

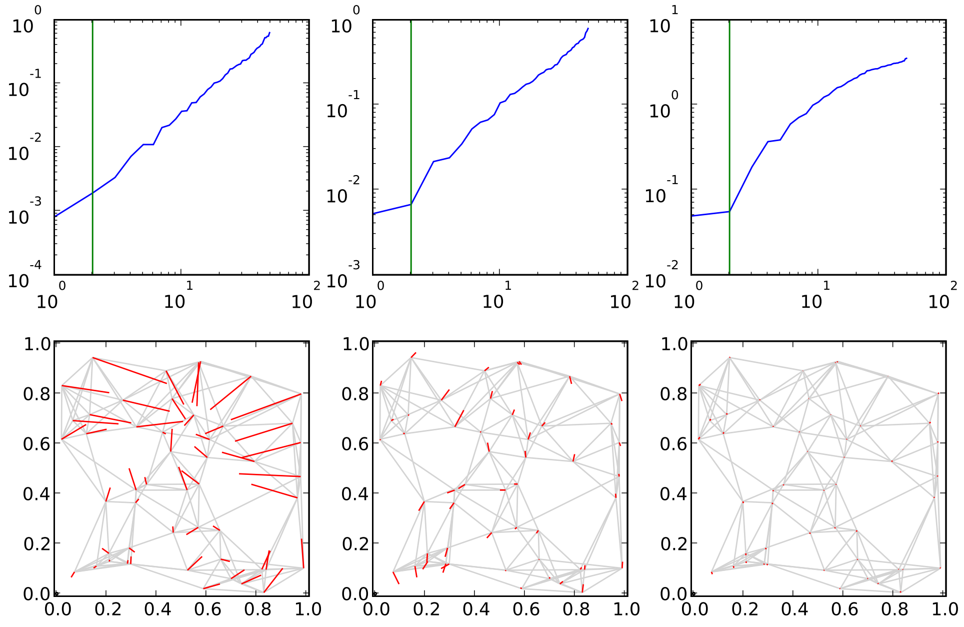


Fig. 2. Plotted in the top row are eigenvalues of aggregated squared stress matrices formed from 1, 2, and 73 stress matrices using noisy distance measurements. The corresponding realization results are shown in the bottom row. Error segments connecting vertex positions in the ground-truth frameworks to the computed frameworks (under optimal alignment) are shown in red. The noise level is 0.1%.

$d + 1$.

However, in the presence of measurement noise, there no longer is a distinct gap in magnitude between the $(d+1)$th and the $(d+2)$th smallest eigenvalues of Ω, i.e., the kernel of Ω is no longer numerically well-defined. If we naively use the eigenvectors corresponding to the smallest $d+1$ eigenvalues to recover the graph embedding, we usually get extremely large error in the resulting realization. For an illustration of the problem, please see Figure 2.

We find it is possible to dramatically increase the accuracy of the computed stress kernel by taking into account more than a single stress matrix. Recall that the kernels of all $\Omega \in S(\rho)$ must include the $d+1$ dimensional space representing affine transforms of ρ, and if $\mathcal{G}$ is GGR, then the kernels of all generic Ω will contain nothing else. Thus we can instead compute the shared kernel of a selected set of stress matrices from $S(\rho)$: $\Upsilon : \{\Omega_1, \ldots, \Omega_N\}$. For Υ, a set of stress matrices, let $K(\Upsilon)$ be the intersection of the kernels of all the stress matrices in the set, i.e., $K(\Upsilon) = \bigcap_{\Omega \in \Upsilon} \ker \Omega$. For a generic framework ρ of a GGR graph for almost all random stress matrices in $S(\rho)$ and almost all random sets of stress matrices $K(\Upsilon) = K(\Omega)$

For a single stress, calculating the eigenvectors of Ω corresponding to the smallest eigenvalues is equivalent to computing orthonormal vectors $\mathbf{y}_1, \ldots, \mathbf{y}_{d+1}$ such that $\|\Omega \mathbf{y}_i\|^2$ is minimized in the order of $i = 1, \ldots, d+1$. For a set of stress matrices Ω_1,

…, Ω_N, we thus seek orthonormal vectors $\mathbf{y}_1$, …, $\mathbf{y}_{d+1}$ minimizing $\sum_j \|\Omega_j \mathbf{y}_i\|^2$ in the order $i = 1, \ldots, d+1$. Since $\sum_j \|\Omega_j \mathbf{y}_i\|^2 = \mathbf{y}_i^T (\sum_j \Omega_j^T \Omega_j)\mathbf{y}_i$, the $\mathbf{y}_i$'s are simply the eigenvectors of $\Sigma = \sum_j \Omega_j^T \Omega_j$ corresponding to the smallest $d+1$ eigenvalues. We call Σ the *aggregated squared stress matrix*. In practice, using even just two stress matrices produces significant better results compared with using a single stress matrix in the presence of noise. For illustration, the eigenvalues of the aggregated squared stress matrix using 1, 2, and 73 stress matrices are plotted in Figure 2, along with the corresponding realization results. From the graph we see that when 2 or more stress matrices are used, the smallest $d+1$ eigenvalues are clearly distinct from the rest.

5.3 Positions from Stress Kernel

Given a basis for a minimal stress kernel of ρ, $\{\mathbf{1}, \mathbf{x}_1, \ldots, \mathbf{x}_d\}$, as computed in Section 5.2, we define $\rho' \in C^d(\mathcal{G})$ such that $\rho'(i)$ is the ith column of $(\mathbf{x}_1, \ldots, \mathbf{x}_d)^T$. Recall from Section 4.5 that ρ' and ρ are related by an affine transformation in $\mathbb{R}^d$, and we can seek a $d \times d$ real matrix L as the solution to (2) in Section 4.5. L can be found by solving the linear system (3) on the $d(d+1)/2$ unknowns in $M = L^T L$. Applying Cholesky decomposition on M will give us L, up to a rigid transformation.

When there is noise in the measurement, we in general cannot expect (3) to be exactly solvable. Our approach is to solve the linear systems in the least square sense, which minimizes the following error term:

$$\sum_{\{i,j\} \in \mathcal{E}} ((\rho'(i) - \rho'(j))^T M (\rho'(i) - \rho'(j)) - l(\rho)^{\{i,j\}})^2 \tag{4}$$

However, due to noisy measurement, the resulting M will not be guaranteed positive semi-definite. A solution to this problem is to use semi-definite programming to minimize the quadratic energy of (4) with the constraint $M \succ 0$ ([Vandenberghe and Boyd 1996; Singer 2008]). This is more computationally expensive than using a least square solver, but often necessary in the presence of noise.

As [Singer 2008] points out, sufficient noise can cause eigenvectors of Σ corresponding to the smallest eigenvalues to be split over multiple eigenvectors. In such a case, using some $D > (d+1)$ eigenvectors of Σ is required to recover the correct coordinate components. Note, however, that now the corresponding matrix M will be of size $D \times D$ and the resulting realization is in $\mathbb{R}^D$ instead of $\mathbb{R}^d$. The simplest way to define the final d dimensional realization is to apply PCA to the embedding in $\mathbb{R}^D$. Due to this projection, increasing D does not automatically guarantees better results and the right choice of D is empirical (see Section 8 for details on how we set D). More advanced dimension reduction algorithms such as [Brand 2003; Zhang et al. 2009] can also be used.

6. HANDLING NON-GGR FRAMEWORKS

If $\mathcal{G}$ is not GGR and hence does not have a minimal stress kernel ($\dim K(\Omega) > d+1$ for all $\Omega \in S(\rho)$ of all generic ρ), we can still look for maximal stress kernel localizable (SKL) vertex subsets of $\mathcal{G}$. Let us first recall from Section 4 some facts about SKL sets: A vertex subset $\mathcal{S}$ is SKL if (1) it has a minimal projected stress kernel, i.e., $\dim \pi_{\mathcal{S}} K(\Omega) = d+1$, where the projection $\pi_{\mathcal{S}}$ is induced by the natural inclusion $\mathcal{S} \subseteq \mathcal{V}$ and $\Omega \in S(\rho)$, and (2) each vertex in the induced subgraph is incident to d or more edges in the subgraph. The first condition allows us to compute the coordinates of the vertices in $\mathcal{S}$ up to an affine transform (Section 4.4). The second condition allows us to remove the free affine transform using squared length measurements (Section 4.5). Also, a SKL set of a graph must be GGL, though the converse may or may not hold. However, the vertex set of any GGR subgraph must be SKL and will be localized by our scheme (Section 4.5).

6.1 Finding Stress Kernel Localizable (SKL) Vertex Subset

Similar to the condition of minimal stress kernel, the condition of minimal projected stress kernel is generic. If the vertex subset $\mathcal{S}$ has a minimal projected stress kernel for one generic framework ρ then it holds for all generic frameworks. Hence instead of using the stress matrix calculated from perturbations which contains noise, we create a random framework for the graph, compute a random stress matrix using exact coordinates, and then detect SKL sets from its kernel.

Suppose ρ' is a framework of the same graph as the input framework ρ but with a known random embedding. Let Ω be a randomly selected stress in $S(\rho')$. Let $\mathbf{1} = (1, \ldots, 1)^T, \mathbf{x}_1, \ldots, \mathbf{x}_{k-1}$ be a basis of $K(\Omega')$ where $k := \dim K(\Omega')$. For notational convenience, pack the basis vectors into a matrix $X = (\mathbf{1}, \mathbf{x}_1, \ldots, \mathbf{x}_{k-1})$. Denote by $X^{\mathcal{S}}$ the sub-matrix of X consisting of rows corresponding to vertices in $\mathcal{S}$. Then $\dim \pi_{\mathcal{S}} K(\Omega') = \operatorname{rank}(X^{\mathcal{S}})$.

We start by finding a seed component, some $\mathcal{S}_0 \subseteq \mathcal{V}$ satisfying (i) $|\mathcal{S}_0| = d+1$, (ii) $\operatorname{rank}(X^{\mathcal{S}_0}) = d+1$, and (iii) $\mathcal{S}_0$ is not contained in any SKL set that we have already found (otherwise that SKL set and $\mathcal{S}_0$ share the same projected stress kernel, so $\mathcal{S}_0$ will be grown into that SKL set). We also weed out numerically degenerate seeds, i.e., almost co-linear vertices. Then a maximal vertex subset with a minimal projected stress kernel is the seed $\mathcal{S}_0$ together with any vertex i such that $\operatorname{rank}(X^{\mathcal{S}_0 \cup \{i\}}) = d+1$. We sequentially (order does not matter for linear dependence) add vertices to the subset until there is none that can be added without increasing the rank. Finally, a maximal SKL set is formed by repeatedly pruning out vertices with degree less than d in the induced subgraph.

While in the worst case we need to enumerate all possible $d+1$ vertex tuples as seeds, in practice we start by enumerating only fully-connected $(d+1)$ subsets of $\mathcal{V}$, i.e., simplices, and only search arbitrary $(d+1)$ subsets of the remaining vertices which do not belong to any detected SKL set. From our experiments, detecting SKL sets for a moderately sized graph of a few hundred vertices takes a tiny fraction of the total running time.

6.2 Computing the Projected Stress Kernel

From the previous subsection, we have computed a collection of, say, m, SKL vertex subsets $U_1, \ldots, U_m \subseteq V$, with $\dim \pi_{U_j} K(\Omega) = d+1$ for each j and some Ω. Fix j and let $\mathcal{S} = U_j$. We now explain how to find a basis for $\pi_{\mathcal{S}} K(\Omega)$.

Again, as in the previous section, for numerical purposes, we will use a collection Υ of stress matrices from $S(\rho)$ instead of a singe one. Since we are looking at a set that is generically globally linked, $\pi_{\mathcal{S}} K(\Upsilon)$ must be the same as $\pi_{\mathcal{S}} K(\Omega)$ for random Ω and Υ.

Naively, once we have a set of basis vectors of the stress kernel of the entire graph $\{\mathbf{1}, \mathbf{x}_1, \ldots, \mathbf{x}_{k-1}\}$ (note that $k > d+1$ since the graph is not GGR and hence does not have a minimal stress kernel) as computed in Section 5.2, we can simply run PCA on their image under $\pi_{\mathcal{S}}$. However this additional PCA step introduces an extra level of numerical error. More importantly, as described in the end of 5.3, when the measurement is sufficiently noisy, we need to take a much larger number of eigenvectors than $\dim K(\Upsilon)$ in order for the actual stress kernel $K(\Upsilon)$ to be contained in their span. The projection under $\pi_{\mathcal{S}}$ of these eigenvectors then spans a space of much higher dimension than $d+1$. It is unclear how we can find a small number of vectors with a span that contains $\pi_{\mathcal{S}} K(\Upsilon)$ from this space.

A more elegant and robust method for finding an orthonormal basis for $\pi_{\mathcal{S}} K(\Upsilon)$ numerically is to formulate these vectors directly in terms of the aggregated squared stress matrix Σ (see Section 5.2). The basis vectors of $\pi_{\mathcal{S}} K(\Upsilon)$ that we will compute are of the form $\pi_{\mathcal{S}}(\mathbf{y}_i)$ where $\mathbf{y}_i \in K(\Upsilon)$. First we set $\mathbf{y}_1 = \mathbf{1}/|\pi_{\mathcal{S}}(\mathbf{1})|$ since the vector of all ones is always in $\pi_{\mathcal{S}} K(\Upsilon)$ and we want $|\pi_{\mathcal{S}}(\mathbf{y}_1)| = 1$. Next we find each $\mathbf{y}_i$ sequentially by solving the following constrained minimization problem for each $i > 1$. Encode $\pi_{\mathcal{S}}$ by the v by v matrix J, so that $J_{p,p} = 1$ if $p \in \mathcal{S}$ and $J_{p,q} = 0$ otherwise. Then

$$\text{minimize: } f(\mathbf{y}_i) = \mathbf{y}_i^T \Sigma \mathbf{y}_i \tag{5}$$

$$\text{subject to: } \mathbf{y}_i^T J \mathbf{y}_i = 1, \tag{6}$$

$$\mathbf{y}_j^T J \mathbf{y}_i = 0 \text{ for } j < i. \tag{7}$$

Minimizing $f(\mathbf{y}_i)$ ensures $\mathbf{y}_i \in K(\Upsilon)$. Constraints (6) and (7) imply that the $\pi_{\mathcal{S}}(\mathbf{y}_i)$'s are orthonormal.

For reasons that will become clear later, we add an extra constraint:

$$\mathbf{y}_i \perp (\ker \Sigma \cap \ker J). \tag{8}$$

This constraint will not change the minimal value of f. Suppose $\mathbf{w}^*$ is a minimizer of f. Then we can write $\mathbf{w}^* = \mathbf{w}_0 + \mathbf{w}_1$ with $\mathbf{w}_0 \perp (\ker \Sigma \cap \ker J)$ and $\mathbf{w}_1 \in (\ker \Sigma \cap \ker J)$. Then $f(\mathbf{w}_0) = f(\mathbf{w}^*)$, so $\mathbf{w}_0$ is also a minimizer of f. Moreover, $\mathbf{w}_0$ satisfies constraints (6), (7), and (8).

Because constraints (7) and (8) are linear, we can eliminate them via a change of variable. Let $\mathbf{y}_i = P\mathbf{x}$ such that the columns of matrix P form an orthonormal basis of the orthogonal complement to $(\ker \Sigma \cap \ker J) \cup \operatorname{span}\{J\mathbf{y}_1, \ldots, J\mathbf{y}_{i-1}\}$. Let $\tilde{\Sigma} = P^T \Sigma P$ and $\tilde{J} = P^T J P$. Then the original constrained minimization problem becomes

$$\text{minimize: } \tilde{f}(\mathbf{x}) = \mathbf{x}^T \tilde{\Sigma} \mathbf{x}$$

$$\text{subject to: } \mathbf{x}^T \tilde{J} \mathbf{x} = 1.$$

Using the Lagrangian multiplier method, we let $g(\mathbf{x}) = \tilde{f}(\mathbf{x}) + \lambda(\mathbf{x}^T \tilde{J} \mathbf{x} - 1)$. Setting $\partial g / \partial \mathbf{x} = 0$ gives

$$\tilde{\Sigma}\mathbf{x} = -\lambda \tilde{J} \mathbf{x}, \tag{9}$$

which is a *generalized eigenvalue problem*. Note that we included the additional constraint (8), so that our instance of generalized eigenvalue problem is guaranteed to be *regular*, i.e., $\det(\tilde{\Sigma} + \lambda \tilde{J})$ as a polynomial in λ does not vanish for all λ.

Hence, solving (9) gives a well defined set of generalized eigenvectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_r\}$. We simply pick q such that $\mathbf{x}_q^T \tilde{J} \mathbf{x}_q \neq 0$ and $\tilde{f}(\mathbf{x}_q/(\mathbf{x}_q^T \tilde{J} \mathbf{x}_q)^{\frac{1}{2}})$ is minimized. $\mathbf{y}_i = P\mathbf{x}_q/(\mathbf{x}_q^T \tilde{J} \mathbf{x}_q)^{\frac{1}{2}}$ is then a solution to the original constrained minimization problem (5).

Finally, note that when the original graph is GGR, J equals the identity matrix, and the algorithm described above reduces to finding the eigenvectors of Σ corresponding to the smallest eigenvalues.

7. REDUCING REQUIRED MEASUREMENTS

Recall from Section 5.1 that the number of perturbations required by our method scales linearly with the size of the graph, and the number of required pair-wise length measurements is $O(vde)$, as opposed to $O(e)$ in, say, multilateration based localization schemes. Since the number of required perturbations scales with graph size, it is natural to ask if it is possible to process much smaller subsets of the graph, and then piece together the stress space of each of these subgraph to get the whole stress space. Each subgraph can be handled using the same measurements of l, so the total number of measurements required is on the order of the size of the largest subgraphs.

Indeed we follow such a strategy. We calculate the stress space of some overlapping subgraphs of $\mathcal{G}$. Each such stress space is a subspace of $S(\rho)$. Hence we add up the computed sub stress spaces to form a subspace $S^*(\rho) \subseteq S(\rho)$. Numerically this is accomplished by running sparse SVD on the basis vectors of all the sub stress spaces. Since $S^*(\rho) \subseteq S(\rho)$, we have $\min_{\Omega^* \in S^*(\rho)} \dim K(\Omega^*) \geq \min_{\Omega \in S(\rho)} \dim K(\Omega)$ And if we are lucky, $\min_{\Omega^* \in S^*(\rho)} \dim K(\Omega^*) = \min_{\Omega \in S(\rho)} \dim K(\Omega)$ If these two quantities are equal, then we will have lost nothing by using the reduced measurement set.

Although we have no guarantee that these quantities are equal, in practice it often is. Even if not, we can still proceed with our algorithm described in Section 6, using any random framework of the graph and exact coordinates to find any vertex subset $\mathcal{S}$, such that the projection $\pi_{\mathcal{S}} K(\Omega^*)$ is of dimension $d+1$. We then know that we will be able to correctly localize this vertex subset from our smaller measurement set.

Because sub stress space calculation can use the same set of length measurements of perturbed frameworks, the total number of whole graph measurements we need is on the order of the size of the largest subgraph used for calculating the sub stress spaces. In other words we only need to set $M = \lceil \mu d v_{\max} \rceil$, where $v_{\max}$ is the number of vertices in the largest subgraph used for calculating the sub stress spaces.

In practice, we use all 2-rings in $\mathcal{G}$ for calculating $S^*(\rho)$ (recall that a k-ring of a vertex v is the set of vertices that is connected to v via a path consisting of at most k edges), and then proceed as usual with the algorithm replacing reference to $S(\rho)$ with that to $S^*(\rho)$. If we assume there is an upper bound c on the vertex degree of $\mathcal{G}$, then the number of pairwise measurements is reduced from $O(vde)$ to $O(c^2de)$, which scales linearly with the size of the graph.

8. RESULTS AND DISCUSSIONS

In this section we evaluate the various parts of our algorithm using simulation. We did find that without noise and with a very small perturbation radius, our method essentially performs as predicted by the theory, and always recovers entire GGR frameworks[1] or any detected SKL vertex subsets with negligible error. Hence we are mostly interested in how well the method performs under noisy measurement.

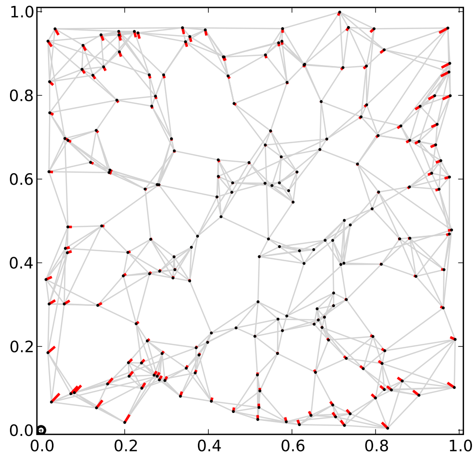
8.1 Simulation Setup

Testing graphs: We test our algorithm on two types of random graphs: The first type is constructed by placing vertices uniform-randomly in $[0, 1]^2$, then putting an edge between vertices that are within a *distance threshold*, R, and finally pruning incident edges of each vertex to enforce a *vertex degree cap.* Given sufficiently many vertices, this kind of graphs are very uniform and we call them *isotropic graphs.* The second type of graphs are constructed by doing the same, but in addition constraining edges not to cross walls in an actual floor plan overlaid on top of the graph. Given the floor plan we use, as shown in Figure 3b, these graphs are highly non-uniform and we call them *anisotropic graphs.* In a physical world, capping of vertex degree means that if a node is in a highly connected area, it only needs to communicate with a fixed number of closest neighbors, as opposed to all nodes within range, and hence can potentially power down its ranging device. Examples of the two types of testing graphs are shown in Figure 3. Finally to make the connectivity of the graph locally invariant with respect to the number of vertices in the graph, we set R to be $v^{-\frac{1}{2}}$ multiplied by a fixed constant.

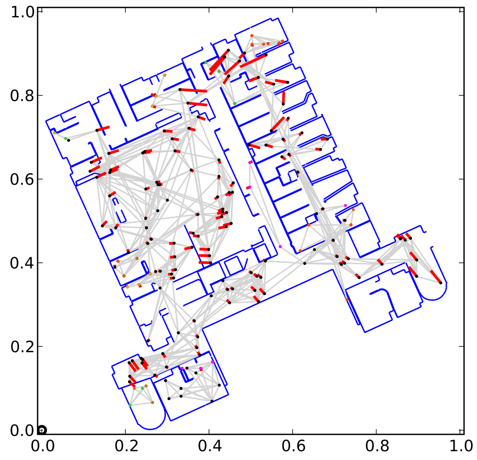
Noise model: We use a simple Gaussian additive noise model for distance measurements in our experiments. The noise term follows a zero-mean Gaussian distribution with a standard deviation of ϵR, where ϵ is a *noise level* parameter. We set these parameters as inspired by the commercially available UWB devices described in [Park et al. 2008]. These devices have a maximum range of about 30 meters and their high-probability error has a standard deviation of 3cm. This corresponds to setting the noise level $\epsilon = 0.1\%$. We also present testing results of setting $\epsilon = 0.01\%$, representing a future generation of ranging devices. In our tests, we refer to these two cases as *high noise* and *low noise* settings.

Error metrics: Our algorithm localizes any maximal SKL vertex subsets in the input graph, up to a Euclidean transform. In order to compare the computed localization ρ' to the ground-truth ρ, for each SKL vertex subset $\mathcal{S}$, we align up its embedding in the two frameworks using the least-square optimal Euclidean transform $Q \in \text{Euclid}(d)$ minimizing $\sum_{i \in \mathcal{S}} \|\rho(i) - Q(\rho'(i))\|^2$, computed using the Procrustes method [Schönemann 1966]. For a vertex i, let Q_i denote such a transform computed for one SKL vertex subset containing i. We then calculate a simple

[1]One simple strategy for generating a random GGR framework is to repeatedly generate a random framework, and accept only if the randomized GGR deciding algorithm in [Gortler et al. 2008] determines the framework to be GGR. We have used this method to verify that our method indeed always works on GGR graphs.



(a) An isotropic testing graph with average vertex degree 5.9. The positional error $\sigma_n = 0.66$m



(b) An anisotropic testing graph with average vertex degree 6.4. The positional error $\sigma_n = 1.30$m. The graph is not GGR so vertices in different detected SKL vertex subsets are shown in different colors.

Fig. 3. Sample realization results for isotropic (left) and anisotropic (right) testing graphs under the high noise setting. The standard deviation of measurement noise $\epsilon R = 0.03$m and the perturbation radius $\delta = 0.6$m. The positional error σ_n is shown under each realization. Error segments connecting vertex positions in the ground-truth frameworks to the computed frameworks (under optimal alignment) are shown in red.

error metric (also used in [Moore et al. 2004]): the *positional error*, σ, defined as

$$\sigma^2 = v^{-1} \sum_{i=1}^{v} \|\rho(i) - Q_i(\rho'(i))\|^2,$$

We report two variants of this metric: σ_n for using our method *without* the measurement reduction strategy, and σ_r for using our method *with* the measurement reduction strategy.

It is important to note that we seek to evaluate the output of our algorithm, which by definition leaves free a Euclidean transform. The positions of all vertices are only used to best align up the output with the ground-truth, and does not improve the *intrinsic* quality of the computed realization.

A separate, but equally important problem is: given the framework computed from our algorithm as a starting point, and a set of more than d anchor vertices with known ground-truth positions, what is the best way to assign positions to the rest of the vertices? One option is to use the moving least square method ([Schaefer et al. 2006]): for each vertex, apply an optimal affine transform computed from all anchor vertices but with weights dependent on the anchor vertices' distance to the vertex in question.

Like [Moore et al. 2004], we also measure the largest localizable subgraph such that any vertices within the subgraph are rigidly localized with respect to each other. This is simply the largest SKL subset detected. Suppose $\mathcal{S}$ is the vertex set of such a subgraph. We define the *localizable ratio* L as

$$L = v^{-1}|\mathcal{S}|.$$

For comparison, we report three variants of this metric: L_n for using our method *without* the measurement reduction strategy, L_r for using our method *with* the measurement reduction strategy, and finally L_t for using trilateration (without regard for noise). Note that L_t, which equals the ratio of vertices in the largest trilateration subgraph over those in the entire graph, is a theoretical maximum for trilateration based methods. In practice, noisy measurements will further shrink the largest trilateralizable subsets, as observed in [Moore et al. 2004].

Real-world correspondence: To provide a more intuitive understanding of our results, we report length measurements (e.g., the noise standard deviation ϵR and the positional error σ) in real-world units. We again use the UWB devices in [Park et al. 2008] as a model. The maximum range of these devices is about 30m, and hence we set our conversion scale so that the edge distance threshold R corresponds to 30m. Everything else then scales accordingly. For example, the high noise setting described previously corresponds to $\epsilon R = 3$cm. Examples of our realization results along with its positional error is shown graphically in Figure 3. Hopefully it will give readers a more concrete feel for the numbers reported later. From the figure we can see that realization results with positional error under 1 meter are actually very good.

We do note that a 3cm measurement error seems small when the maximum edge length can be 30m. However, since we cap the vertex degrees in our testing graphs, a *typical* edge in the graph is much shorter: The median edge length in our testing graphs is usually in the range of 4–6 meters. In other words, for more than half of the edges in the graph, the standard deviation of the measurement noise is more than 0.5%–0.7% of its length.

8.2 Choice of Algorithm Parameters

In the presence of noisy measurements, the performance of our method is greatly influenced by the choice of the various parameters described in Section 5:

—The sampling factor μ controls how many perturbations are used.

—The perturbation radius δ controls how much we perturb each vertex.

—N is the number of stress matrices used to form the aggregated squared stress matrix.

—D is the number of coordinate vectors used to reconstruct the framework.

The optimal parameters definitely vary for different input graphs. Nonetheless, by doing some simple brute force exploration of the parameter space, we are able to find parameters that generally give good and consistent results for our input graphs. We next discuss the effects of these parameters and describe our choice used for later simulations.

Understandably, increase in either μ and N causes an almost monotonic decrease in σ, but with diminishing return. However we want to use as few measurements as possible, so we restrict ourselves by setting $\mu = 4$. A large N stabilizes the kernel of the aggregated squared stress matrix in the presence of noise (Section 5.2).

Moreover the marginal computational cost of increasing N is small. Hence we always set $N = 200$.

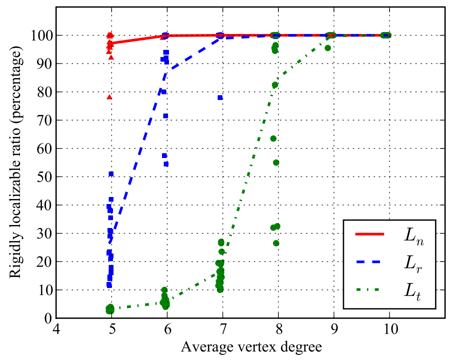
The effect of the perturbation radius is more subtle. The higher δ is relative to the noise level ϵR, the less noise prone the computed approximation of $T(\rho)$ will be. On the other hand, if the local geometry of the manifold $l(C^d(\mathcal{G}))$ around $l(\rho)$ has large second-order derivatives, we need a small δ to get a good approximation of $T(\rho)$. From simulation we found that as $\delta/(\epsilon R)$ increases, σ first goes down (as the approximation becomes less noise prone), and then goes up (as the local geometry becomes more poorly approximated). Also, if the noise level is low, there is more room to tune up $\delta/(\epsilon R)$ before the local geometry around $l(\rho)$ becomes poorly approximated. From simulations, we set $\delta = 20\epsilon R = 60\text{cm}$ for the high noise setting, and $\delta = 80\epsilon R = 24\text{cm}$ for low noise setting.

Finally we found that increasing D in general improves realization accuracy, until some threshold D^* is crossed, at which point it no longer helps and can even be detrimental. This threshold signals that the true stress kernel (or projected stress kernel) is contained in the space spanned by the first D^* basis vectors found by the algorithm (Section 6.2). Also when the SKL vertex subset is really small, setting D too high can allow too much freedom for the embedding and results in large error. From simulations, we settled with using $D = 10$ by default, but $D = 4$ for vertex subsets with less than 20 vertices. Very rarely, due to measurement noise, even $D = 10$ is not enough to recover the true stress kernel. We detect such cases by noticing that there is an abnormally large difference between edge lengths of the resulting realization and the original measurements, and then bump D to 20.

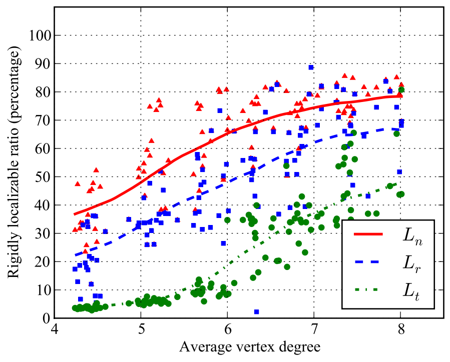
8.3 Simulation Results

We first generate 20 different random graphs, with 200 vertices, for each different (graph type, noise level, vertex degree cap) combination, where the graph type can be isotropic or anisotropic, the noise level can be low or high, and the vertex degree cap ranges from 5 to 10. We then run our algorithm on these random graphs, both with and without employing the measurement reduction strategy. Each run produces a data point with its associated metrics, which we collect and plot. Note that for isotropic graphs, the average vertex degree of the generated graphs cluster around the integral values that we set as the vertex degree cap. However, for anisotropic graphs, due to blockage by walls in the floor plan, the induced average vertex degree is usually smaller than the vertex degree cap and is less clustered around integral values.

Our method can localize a large portion of a graph even if the graph has low average vertex degree. This is a major qualitative advantage of our method over existing methods. Figure 4 shows plots of the localizable ratio versus the average vertex degree, for using our method with and without measurement reduction (L_n and L_r) as well as using trilateration (L_t). From the plots, we can see that when the graph has moderately low connectivity, trilateration can only propagate through a very small part of the graph. L_t is always much smaller than L_n (which can be considered as an upper bound of L). Moreover, when the measurement reduction strategy is used, our method still performs surprisingly well and stays way ahead of trilateration. Also, for isotropic testing graphs, the gap between L_n and L_r is quickly shortened as the vertex degree increases, and mostly disappears when the vertex degree reaches 7.



(a) Isotropic testing graph.



(b) Anisotropic testing graph.

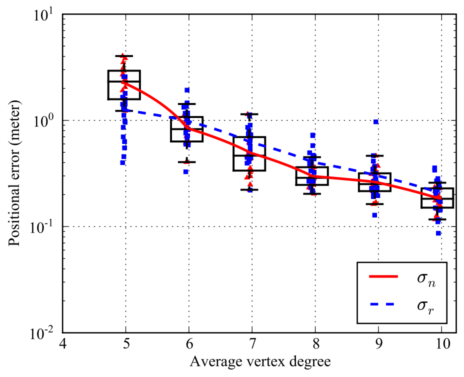
Fig. 4. Plots of the localizable ratio versus average vertex degree, for using our method with and without measurement reduction (L_n and L_r), as well as using trilateration (L_t). A moving average of the data points is overlaid on each plot.

Figure 5 shows plots of the positional error versus average vertex degree, for using our method with and without measurement reduction (σ_n and σ_r). The positional error generally decreases as vertex degree increases, since there are more edge lengths to serve as constraints. It is also worth noting that the error we achieve is often not much higher than the perturbation radius. In real world terms, this means that as a sensor node wiggles, the positional error is not much larger than how far it has wiggled.

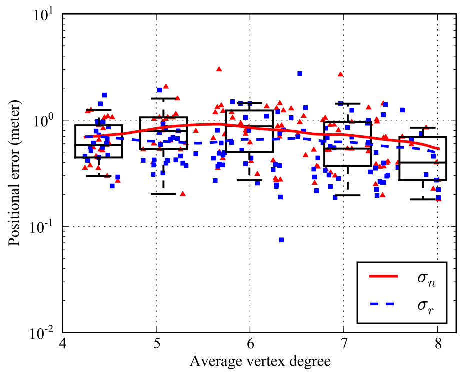
From Figure 5, it can be seen that our method produces very satisfactory results for isotropic graphs, with positional error dropping below 0.8m as soon as the average vertex degree reaches 6, even in the high noise setting. For anisotropic graphs, the highly irregular connectivity clearly poses more challenges, but our method still performs well in general. However, there are a small number of outlying data points with positional error above 2 meters. We manually looked at these runs and found that we can always significant reduce the error (e.g., down to below 1 meter) by slightly tweaking the perturbation radius, δ, or the number of basis vector, D. The reason for this sensitivity is discussed in the previous section. In the future it will be nice to have some automatic scheme for setting δ based on, for example, local average edge length. Similarly, an intelligent method for choosing D will be a welcome addition to our algorithm.

We can also observe from Figure 5 that the measurement reduction strategy does not increase the positional error much, even though the number of measurements needed is drastically reduced. In the case of isotropic graphs in the low noise setting, there is a slightly larger gap between σ_n and σ_r. However both errors are already exceptionally small.

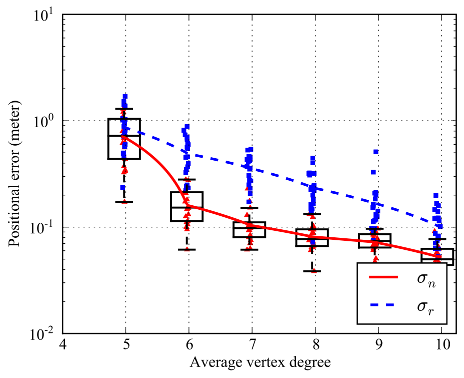
For a somewhat apple-to-orange comparison, we also ran the SDP-based localization codes of Biswas et al. [2006] taken from the authors' website on our testing graphs. To apply their codes, we first prune the input graph to its largest GGR



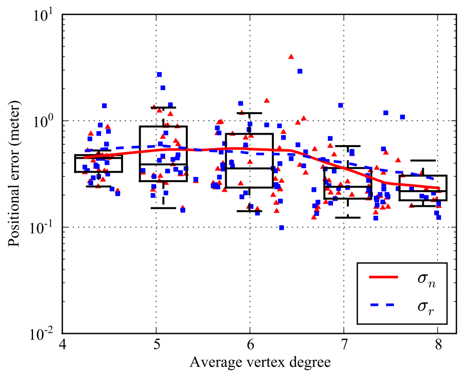
(a) High noise: $\delta = 0.6$m, $\epsilon R = 0.03$m



(b) High noise: $\delta = 0.6$m, $\epsilon R = 0.03$m



(c) Low noise: $\delta = 0.24$m, $\epsilon R = 0.003$m



(d) Low noise: $\delta = 0.24$m, $\epsilon R = 0.003$m

Fig. 5. Plots of the positional error versus average vertex degree, for using our method with and without measurement reduction (σ_n and σ_r). The left column corresponds to isotropic testing graphs and the right anisotropic testing graphs. The top row corresponds to the high noise setting with the noise standard deviation $\epsilon R = 0.03$m and the perturbation radius $\delta = 0.6$m. The bottom row corresponds to the low noise settings with $\epsilon R = 0.003$m and $\delta = 0.24$m. A moving average of the data points is overlaid on each plot. In addition, for the σ_n data points, we cluster them by rounding the x coordinates to the nearest integer, and draw a box-and-whisker diagram for each cluster to better illustrate error distribution.

component by recursively taking the graph's largest SKL induced subgraph till there is only one left. We next choose three anchors by taking the left most vertex, and two subsequent vertices farthest away from previously chosen anchor(s). We then feed noiseless distance measurement to the codes. Finally we optimally align the localization output produced by the codes to the ground-truth (as described in Section 8.1) and measure the corresponding positional error, denoted by σ_s.

Figure 6 shows plots of σ_s versus average vertex degree. Even with noiseless distance measurements, the SDP-based approach produces large error (e.g., greater than 4m) for a significant portion of the graphs, especially in the anisotropic case. This indicates that large portion of these graphs are not universally rigid, even though they are all GGR. To give intuition to how these cases look, two sample localization results with large errors are shown in Figure 7. Of course the SDP-

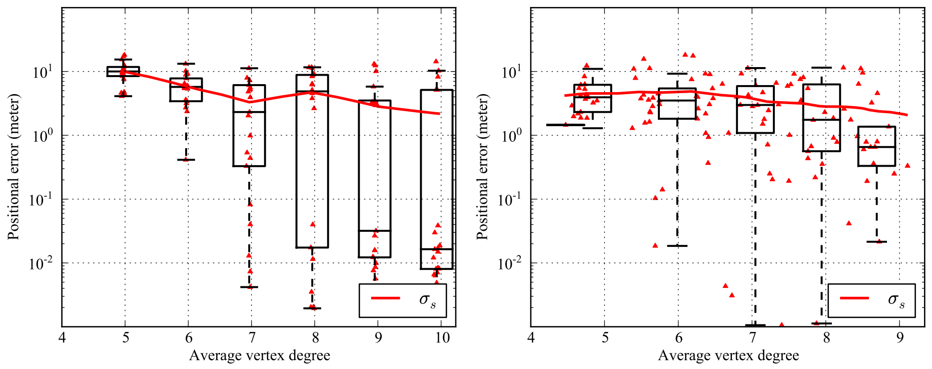
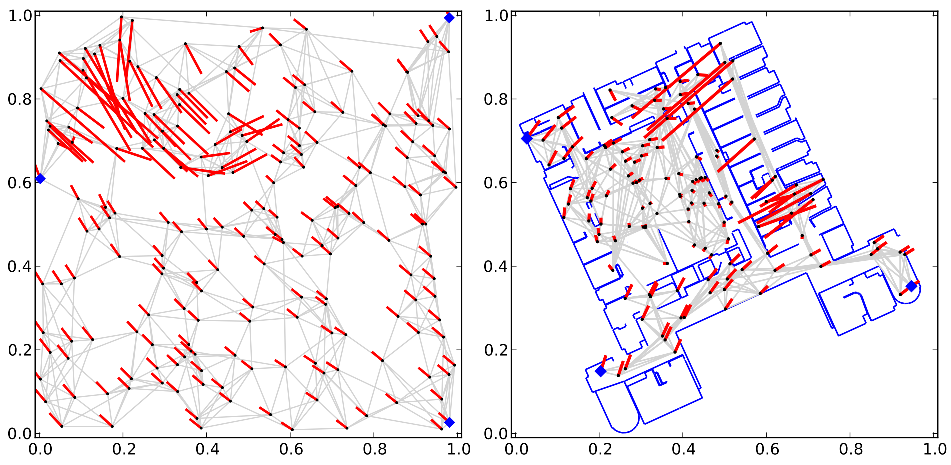


Fig. 6. Plots of positional error versus average vertex degree for the SDP-based approach of [Biswas et al. 2006]. The left and right plots correspond to isotropic and anisotropic testing graphs respectively. Note that the input measurements to the SDP-based approach has zero noise.

based approach uses much fewer measurements and is faster than our method since its solves a smaller semi-definite program. On the other hand, the extra perturbed measurements used by our method is not usable by the SDP-based approach anyway. Thus the two approaches handle different cases using different inputs, and are hard to compare.

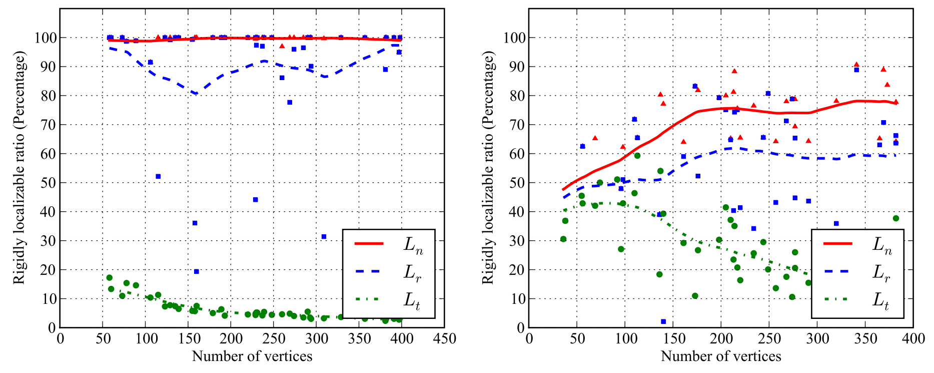


(a) Isotropic case: $v = 200$, average vertex degree $= 6.95$, $\sigma_s = 4.8$m.

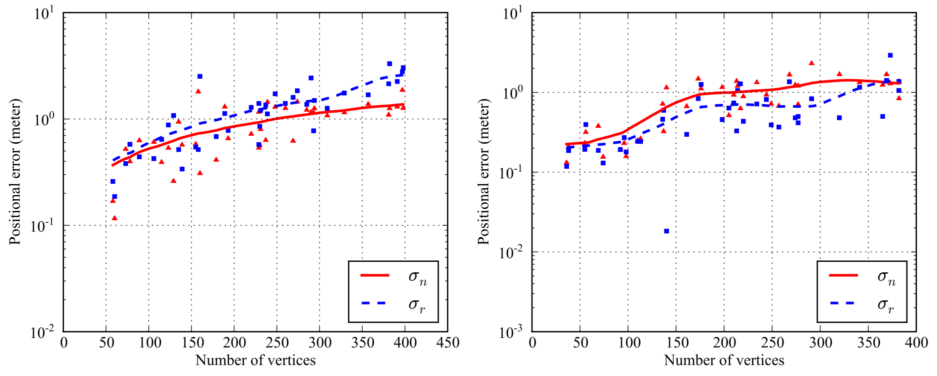
(b) Anisotropic case: $v = 131$, average vertex degree $= 6.96$, $\sigma_s = 4.98$m.

Fig. 7. Examples of when the SDP-based approach of [Biswas et al. 2006] gives large error. The output of the SDP-codes are optimally aligned with the ground-truth. The anchors are shown as blue diamonds.

We next investigate how our method scales with respect to graph size. For the two graph types, we generate 40 different random graphs by letting the noise level to be fixed to high, the vertex degree cap to be 6 and 8 respectively for isotropic and anisotropic graphs, and the number of vertices v to be drawn uniform randomly from $[50, 400]$. The metrics resulting from running our algorithm on these graphs are plotted in Figure 8. We note that the induced average vertex degree is 6 for the isotropic graphs, but ranges from 5.5 to 7 for the anisotropic graphs as the graph size increases. This is because the floor plan used for constructing the anisotropic graphs is static and does not grow in complexity. Hence, as the graph size increases, local blockage of edges by walls decreases and the mean vertex degree tends to increase.



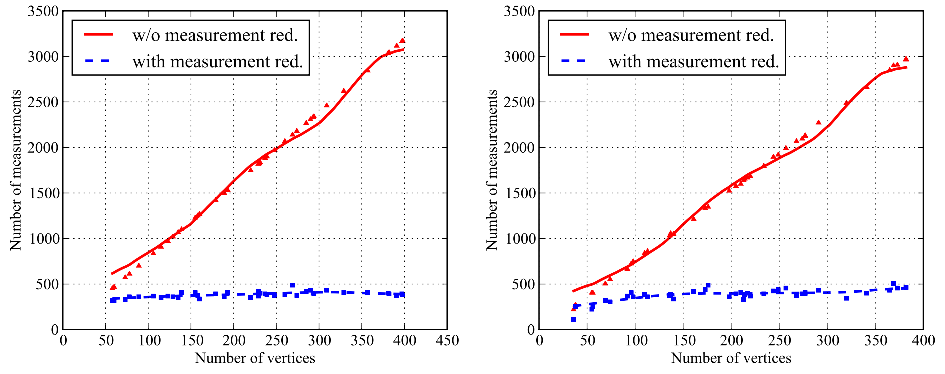
(a) Localizable ratio versus graph size



(b) Positional error versus graph size

Fig. 8. Plots of the various metrics versus graph size for isotropic (left) and anisotropic (right) graphs. The noise level is set to high ($\delta = 0.6$m, $\epsilon R = 0.03$m). The mean vertex degree is 6 for the isotropic graphs, and ranges from 5.5 to 7 for the anisotropic graphs as the graph size increases.

From the plots we can see that in terms of localizable ratio, our method, both with and without measurement reduction strategy, scales well as the the graph size increases, whereas trilateration based methods can only localize an increasingly smaller subset of the vertices. The positional error, however, does exhibit a slowly increasing trend as the graph size increases.



(a) Isotropic graphs (b) Anisotropic graphs

Fig. 9. Plots of the number of (whole graph) distance measurements used by our method, with and without employing the measurement reduction strategy. Various metrics for the corresponding realization results are plotted in Figure 8.

Finally, we want to point out that σ_n and σ_r are generated with drastically different numbers of measurements, as shown in in Figure 9.

8.4 Opportunities for Further Improvements

Note that we choose to use the simplest method, PCA, to flatten our computed realization result in $\mathbb{R}^D$ down to $\mathbb{R}^d$ (see Section 5.3). Using a more sophisticated algorithm such as [Brand 2003; Zhang et al. 2009] will likely further reduce the positional error.

Similarly, running additional numerical optimization on top of our realization results, like [Biswas et al. 2006], will definitely result in smaller positional error. From Figure 3, it is clear that most of the error present can be removed by some gradient descent.

Finally, having and making use of a few well placed anchor nodes will also likely improve our realization results, especially for the anisotropic testing cases.

All these enhancements should be possible and welcomed in a real application, but are beyond the scope of this paper.

8.5 Timing

Our prototype implementation in Python uses the Scipy [Jones et al. 2001] module for numerics and the Cvxopt [Dahl and Vandenberghe 2007] module for SDP optimization. The implementation is single threaded and not optimized for speed. On an Intel Core 2 2.4GHz machine, it usually takes less than a minute to compute embedding for a 200-vertex graph with mean vertex degree 6. We also note that the running time is dominated by numerical computation: PCA in estimating stress space, matrix multiplication and solving eigen problems in estimating stress kernel, and solving SDP in fixing affine transformation from lengths. Of these, solving SDP is the most expensive part.

9. CONCLUSION AND FUTURE WORK

We presented and evaluated a novel algorithm for realizing arbitrary generically globally rigid (GGR) graphs from perturbed distance data. The new algorithm is robust against measurement noise, handles non-GGR graphs by localizing all maximal SKL subsets (which includes vertex sets of GGR subgraphs), and can be improved to require a linear number of pair-wise distance measurements. Our method also extends to higher dimensions without modification. For example, a 3d example is shown in Figure 10. Possible future research directions include:

— Design intelligent methods for choosing algorithm parameters.

— Design hybrid localization strategies that combine our method with methods that use geometry constraints (e.g., multilateration), so that even fewer measurements are required while potentially higher accuracy can be reached.

— Design a distributed version of our method.

— Implement on physical sensor networks. One challenge is that we have assumed in this paper that each set of perturbed distance measurement is consistent. This means that sensor nodes need to synchronize their wiggling and measuring actions, which would be extremely expensive for very large networks. However, with the measurement reduction strategy, synchronization only needs to be done within each subgraph used for calculating sub stress spaces, which is much smaller than the whole graph.

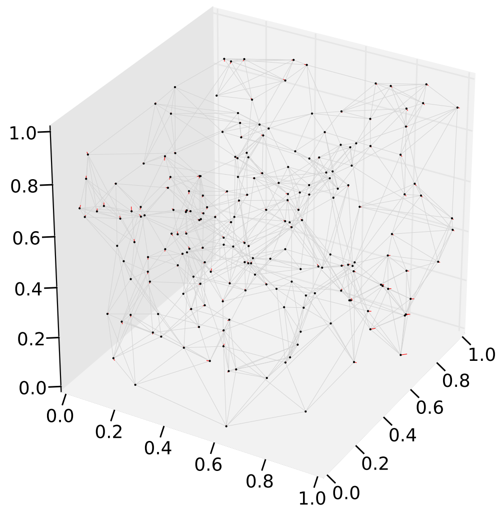


Fig. 10. Example of using our algorithm to recover a framework in 3D. Error vectors are drawn in red.

REFERENCES

Asimow, L. and Roth, B. 1979. The rigidity of graphs. II. *J. Math. Anal. Appl. 68*, 171–190.

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

Biswas, P. and Ye, Y. 2006. A distributed method for solving semidefinite programs arising from ad hoc wireless sensor network localization. In *Multiscale optimization methods and applications, volume 82 of Nonconvex Optim. Appl.* 69–84.

Brand, M. 2003. Charting a manifold. In *Advances in Neural Information Processing Systems 15.* MIT Press, 961–968.

Carter, M., Jin, H., Saunders, M., and Ye, Y. 2007. Spaseloc: An adaptive subproblem algorithm for scalable wireless sensor network localization. *SIAM Journal on Optimization 17,* 4, 1102–1128.

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

Dahl, J. and Vandenberghe, L. 2007. Cvxopt - a python package for convex optimization. In *21st European Conference on Operational Research.*

Doherty, L., pister, K., and El Ghaoui, L. 2001. Convex position estimation in wireless sensor networks. *INFOCOM 2001 Proc.*, 1655–1663.

Eren, T., Goldenberg, O., Whiteley, W., Yang, Y., Morse, A., Anderson, B., and Belhumeur, P. 2004. Rigidity, computation, and randomization in network localization. *INFOCOM 2004 Proc.*, 2673–2684.

Gao, X.-S., Hoffmann, C. M., and Yang, W.-Q. 2002. Solving spatial basic geometric constraint configurations with locus intersection. In *SMA '02: Proceedings of the seventh ACM symposium on Solid modeling and applications.* ACM, New York, NY, USA, 95–104.

Goldenberg, D., Krishnamurthy, A., Maness, W., Yang, Y., Young, A., Morse, A., and Savvides, A. 2005. Network localization in partially localizable networks. *INFOCOM 2005 Proc.*, 313–326 vol. 1.

Gortler, S. J., Healy, A., and Thurston, D. 2008. Characterizing generic global rigidity. arXiv:0710.0926.

Gortler, S. J. and Thurston, D. 2010. Characterizing the universal rigidity of generic frameworks. arXiv:1001.0172.

Hendrickson, B. 1992. Conditions for unique graph realizations. *SIAM J. Comput. 21,* 1, 65–84.

Hendrickson, B. 1995. The molecule problem: Exploiting structure in global optimization. *SIAM J. Optim. 5,* 4, 835–857.

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

Jackson, B., Jordán, T., and Szabadka, Z. 2006. Globally linked pairs of vertices in equivalent realizations of graphs. *Discrete & Computational Geometry 35,* 3, 493–512.

Jones, E., Oliphant, T., Peterson, P., et al. 2001. SciPy: Open source scientific tools for Python.

Moore, D., Leonard, J., Rus, D., and Teller, S. 2004. Robust distributed network localization with noisy range measurements. In *SenSys '04.* 50–61.

Niculescu, D. and Nath, B. 2003. Dv based positioning in ad hoc networks. *Kluwer journal of Telecommunication Systems*, 267–280.

Park, J., Demaine, E. D., and Teller, S. 2008. Moving-baseline localization. In *IPSN '08 Proc.* 15–26.

Saxe, J. B. 1979. Embeddability of weighted graphs in k-space is strongly np-hard. In *Proc. 17th Allerton Conf. in Communications, Control, and Computing.* 480–489.

Schaefer, S., McPhail, T., and Warren, J. 2006. Image deformation using moving least squares. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers.* ACM Press, New York, NY, USA, 533–540.

Schönemann, P. H. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika 31*, 1–10.

SHANG, Y., RUML, W., ZHANG, Y., AND FROMHERZ, M. P. J. 2003. Localization from mere connectivity. In *MobiHoc '03*. 201–212.

SINGER, A. 2008. A remark on global positioning from local distances. *Proc. of the National Academy of Sciences 105,* 28, 9507–9511.

SO, A. M.-C. AND YE, Y. 2007. Theory of semidefinite programming for sensor network localization. *Math. Program. 109,* 2, 367–384.

VANDENBERGHE, L. AND BOYD, S. 1996. Semidefinite programming. *SIAM Rev. 38,* 1, 49–95.

ZHANG, L., LIU, L., GOTSMAN, C., AND GORTLER, S. J. 2009. An as-rigid-as-possible approach to sensor network localization. Tech. Rep. TR-01-09, Harvard University, MA. `ftp://ftp.deas.harvard.edu/techreports/tr-2009.html`.

ZHU, Z., SO, A., AND YE, Y. 2010. Universal Rigidity: Towards Accurate and Efficient Localization of Wireless Networks. In *INFOCOM 2010 Proc.* 1–9.