Sensor Network Localization Using Sensor Perturbation

Yuanchen Zhu  
Harvard University  
yzhu@fas.harvard.edu

Steven J. Gortler  
Harvard University  
sjg@cs.harvard.edu

Dylan Thurston  
Columbia University  
dpt@math.columbia.edu

Abstract—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 (GGR) graphs (or maximal vertex subsets in non-GGR graphs whose relative positions are fixed). And this algorithm 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 present techniques for improving the robustness of the algorithm under noisy measurements, and a strategy for reducing the required number of measurements.

I. INTRODUCTION

In this paper, we revisit the problem of determining the 2D geometric coordinates (up to an unknown Euclidean transform) 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 numerical algorithms have been proposed [1], [2], [3], [4], [5], [6], [7], [8]. But graph realization is, in general, NP-HARD, as is its decision problem GRAPH-EMBEDDABILITY [9]. 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” [5], and methods based on semi-definite programming such as [3] will work on input that is “universally globally rigid” [10]. 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 theorem” from the structural rigidity literature. This theorem 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 subgraphs that we will be able to localize with our approach, even when the graph is not GGR.

On the practical side, we discuss numerical methods for estimating the relevant quantities in order to deal with measurement noise, as well as strategies for reducing the required measurements.

Although 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.

We refer interested readers to [11] for a more detailed discussion of our method.

II. THEORY

In this section we first fix some notations. Most can be found in [12], [13]. Then we describe the theory behind our approach.

A. 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 $\mathcal{V}$. Given $i \in \mathcal{V}$, we define its neighboring vertices $\mathcal{N}(i) = \{j : \{i, j\} \in \mathcal{E}\}$.

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 observing that trivially $C^d(\mathcal{G}) \cong \mathbb{R}^{vd}$. 
The length-squared function, \( l : C^d(\mathcal{G}) \to \mathbb{R}^e \), is a function that takes in a framework \( \rho \) 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 elements of vectors in \( \mathbb{R}^e \) by edges in \( \mathcal{E} \).

A framework \( \rho \) is rigid if there exists \( \delta > 0 \) such that \( \|\rho' - \rho\| < \delta \) and \( l(\rho) = l(\rho') \) implies \( \rho \cong \rho' \). This captures the idea that the graph cannot continuously flex. A framework \( \rho \) is globally rigid if \( l(\rho) = l(\rho') \) always implies \( \rho \cong \rho' \). In graph realization, we want to compute measurements of all frameworks of the graph in \( \mathbb{R}^v \) vectors in \( \|\cdot\| \) of \( \mathbb{R}^{G\mathcal{G}} \).

B. Measurement Tangent Space

The connection between the measurement tangent at \( l(\rho) \) and the stresses \( S(\rho) \) is described by what we will call “the stress normal theorem”.

**Theorem 1:** For all \( \rho \), \( S(\rho) \) is the orthogonal complement to \( l_p^r(\mathbb{R}^e) \) in \( \mathbb{R}^e \).

This theorem can be proved with straightforward calculation. It has been used explicitly (see for example Lemma 2.5 of [12]) and implicitly (see for example page 186 of [15]) in the structural rigidity literature for many years. And thus, for generic \( \rho \), \( S(\rho) \) is orthogonal complement to \( T(\rho) \) in \( \mathbb{R}^e \).

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

D. Positions (Almost) from Stresses

A stress vector of a framework \( \rho \in C^d(\mathcal{G}) \) is a vector \( \omega \in \mathbb{R}^e \) such that for each \( i \in \mathcal{V} \),

\[
\sum_{j \in \mathcal{N}(i)} \omega_{i,j}(\rho(j) - \rho(i)) = 0.
\]

In other words, the position of each vertex is described as the weighted average of the positions of its neighbors, with weights given by \( \omega \). (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 \( \rho \) by \( S(\rho) \) and call it the stress space at \( \rho \).

The position of each vertex is described as the weighted average of the positions of its neighbors, with weights given by \( \omega \). (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 \( \rho \) by \( S(\rho) \) and call it the stress space at \( \rho \).

The position of each vertex is described as the weighted average of the positions of its neighbors, with weights given by \( \omega \). (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 \( \rho \) by \( S(\rho) \) and call it the stress space at \( \rho \).

The position of each vertex is described as the weighted average of the positions of its neighbors, with weights given by \( \omega \). (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 \( \rho \) by \( S(\rho) \) and call it the stress space at \( \rho \).

The position of each vertex is described as the weighted average of the positions of its neighbors, with weights given by \( \omega \). (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 \( \rho \) by \( S(\rho) \) and call it the stress space at \( \rho \).
We know there must be some solution $L$ from the corresponding to system of equation, in the unknown coordinates of the vertices in $V$, up to an affine transform.

If the framework does not have a minimal stress kernel, all is not lost. Suppose there is a subset $S \subset V$, such that $K(\rho)$ is in fact of dimension $d+1$ when projected onto the coordinates corresponding to $S$. (The natural inclusion $S \subset V$ induces a projection $\pi_S : \mathbb{R}^{|V|} \to \mathbb{R}^{|S|}$.) Then, using the same reasoning, from the projected stress kernel $\pi_S K(\rho)$, we can compute the coordinates of the vertices in $S$ up to an affine transform.

To find maximum subsets with a minimal projected stress kernel, we start by finding a seed component $S_0$ 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 $\dim \pi_{S_0 \cup \{i\}} K(\rho)$ remains $d+1$, until there is none that can be added without increasing the dimension. Conveniently, all generic frameworks of a given graph in a given dimension behave identically with respect to stress kernel, we do not know of any specific guarantees of this property. So instead of using the noisy input data, we find such subsets using exact distances on a random test graph in the coordinates of $V$ to localize any vertex subset $S$ with $\dim \pi_S K(\rho) = d+1$ and vertex degree at least $d$ in the induced subgraph. In fact, this condition is sufficient for $S$ to be globally linked in all generic frameworks, although whether it is a necessary condition is still an open question.

### III. Numerical Algorithm

#### A. Estimating the Measurement Tangent, and Stress Space

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 $B_\delta(\rho) = \{ \rho' \in C^d(G) : \| \rho'(i) - \rho(i) \| < \delta \text{ for all } i \in V \}$. We also assume that we can obtain $l(\rho_1), \ldots, l(\rho_M)$ for some fixed number, $M$, of random samples $\rho_i \in B_\delta(\rho)$. Then approximate $T(\rho)$ by fitting a hyperplane to these points using PCA.

**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 $\mu$ 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 $G$ in $\mathbb{R}^d$. Moreover, when the framework is rigid, $\dim T(\rho)$ takes its upper bound of $(vd - \frac{(d+1)}{2})$. Thus the number of perturbations we need scales linearly with $vd$. For each perturbation $\rho_i$, an entire set of edge length measurements then need to be made to get $l(\rho_i)$, so the total number of required pair-wise length measurements is $O(vde)$.

In Section III-D we present a strategy for reducing the number of required measurement.

**Choosing $\delta$**. Ideally, $\delta$ 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 $\delta$ needs to be set large enough to dominate the noise. In our simulation, we find that when the noise is normally distributed with variance $\sigma^2$, setting $\delta = 20\sigma$ produces good results for a wide variety of graphs (see Section IV for details).

**PCA details.** Because $l(C^d(G))$ is a cone, $T(\rho)$ must include the origin. Thus we use the origin $O = (0, \ldots, 0)^T$ 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.

#### B. Stress Kernel from Measurement Tangent

The stress kernel is the same as the kernel of any generic stress matrix in $S(\rho)$, so we could take a random stress matrix.
from the stress space and compute its kernel. With no noise, we know that the kernel of such an \( \Omega \) for a GGR graph will be well-defined and of dimension \( d + 1 \).

However, in the presence of measurement noise, there is no longer a distinct gap in magnitude between the \((d + 1)\)th and the \((d + 2)\)th smallest eigenvalues of \( \Omega \), i.e., the kernel of \( \Omega \) 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.

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. Calculating the eigenvectors of \( \Omega \) corresponding to the smallest eigenvalues is equivalent to computing orthonormal vectors \( y_1, \ldots, y_{d+1} \) such that \(|\Omega y_i|\)^2 is minimized in the order of \( i = 1, \ldots, d + 1 \). We extend this notion to multiple stress matrices. With stress matrices \( \Omega_1, \ldots, \Omega_N \), we thus seek orthonormal vectors \( y_1, \ldots, y_{d+1} \) minimizing \( \sum_i |\Omega_i y_i| \) in the order \( i = 1, \ldots, d + 1 \). Since \( \sum_i |\Omega_i y_i| = y^T (\sum_i \Omega_i^T \Omega_i) y \), the \( y_i \)'s are simply the eigenvectors of \( \Sigma = \sum_i \Omega_i^T \Omega_i \) corresponding to the smallest \( d + 1 \) eigenvalues. We call \( \Sigma \) 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.

C. Positions from Stress Kernel

When there is noise in the measurement, we in general cannot expect \( \Sigma \) to be exactly solvable. Our approach is to solve the linear systems in the least square sense with the constraint \( M \succ 0 \), using semi-definite programming \( \text{[16, 8]} \). In addition, as in \( \text{[8]} \), we optionally use \( D \succ (d+1) \) eigenvectors from \( \Sigma \) to recover the positions. In this case we obtain an embedding in \( \mathbb{R}^{D-1} \) and must use PCA to project back down to \( \mathbb{R}^d \).

D. Reducing Required Measurements

To reduce the requirement measurements, we can calculate the stress space of some overlapping subgraphs of \( G: \mathcal{H}_1, \mathcal{H}_2, \ldots \) from a fixed set of perturbed length measurements. Since these subgraphs are much smaller than \( G \), we can use a much smaller measurement set. In particular, the subsets we use are the 2-rings in \( G \), and if there is an upper limit on vertex degree, then the number of required pairwise measurement is reduced to \( O(deg) \), linear in the graph size.

As described in \( \text{[11]} \), we can add together these stress spaces from the subgraphs to obtain a stress subspace of the whole graph, \( S^*(p) \subseteq S(p) \) and an enlarged stress kernel \( K^*(p) \supseteq K(p) \). Although we have no guarantee that \( \dim(K^*(p)) = \dim(K^*(p)) \), in practice it often is, and in any case we can still proceed with our algorithm and localize any vertex subset where \( K^*(p) \) projects down to the appropriate dimension.

IV. RESULTS AND DISCUSSIONS

In this section we briefly evaluate our algorithm using simulation. We refer interested readers to \( \text{[11]} \) for more detailed experimental data and discussions.

Testing graphs: We test our algorithm on two types of random graphs: The first, which we call isotropic graphs, are constructed by placing vertices uniform-randomly in \([0, 1]^2\), and putting an edge between vertices that are within a distance threshold, \( R \). The second, which we call anisotropic graphs, are constructed by first doing the same, but then overlaying an actual floor plan and disconnecting edges that cross walls, thus making connectivity highly non-uniform. In addition, we enforce maximal vertex degree constraints: 7 for isotropic graphs and 9 for anisotropic graphs. The actual mean vertex degree for the resulting graphs are typically around 7 and 7.5 respectively. We optionally prune the graphs to make them GGR. Note that this level of connectivity usually does not allow trilateration based method to localize the entire graph. For brevity, we report simulation results on two fixed instances from these two categories, shown in Figure 2.

Noise model: We use an additive noise model. The noise term follows a zero-mean Gaussian distribution with a standard deviation of \( \epsilon R \), where \( \epsilon \) is a noise level parameter. The commercially available UWB devices described in \( \text{[17]} \) has a maximum range of about 30 meters and its 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: The output of our algorithm by definition leaves free a Euclidean transform. In order to compare the computed localization, \( \rho' \), to the ground truth, \( \rho \), we first align up the two using the least-square optimal \( Q \in \text{Euclid}(d) \) minimizing \( \sum_i \|\rho(i) - Q(\rho'(i))\|^2 \), computed using the Procrustes method \( \text{[13]} \). Note that this alignment step does not improve the intrinsic quality of the output. We then calculate two simple error metrics (which are also used in \( \text{[17]} \)): the positional error, \( \sigma_p \), defined as

\[
\sigma_p^2 = \frac{1}{n} \sum_{i=1}^{n} \|\rho(i) - Q(\rho'(i))\|^2,
\]

and the distance error, \( \sigma_d \), defined as

\[
\sigma_d^2 = e^{-1} \|l^2(\rho) - l^2(\rho')\|^2,
\]
where $l^+(\rho)$ is simply $l(\rho)$ with every element square-rooted, i.e., it is a vector of edge lengths as opposed to squared edge lengths.

From simulation, we found that without noise ($\epsilon = 0$), and with a small perturbation radius ($\delta = 10^{-4}$), our method essentially performs as predicted by the theory, and always recovers entire GGR frameworks and any detected generically globally linked component of non-GGR frameworks with negligible error. Thus we are mostly interested in how our method performs in the presence of noise, which depends heavily on the choice of parameters described in Section III.

- $\mu$ controls how many perturbations are used.
- $\delta$ controls how much we perturb each vertex. To be relevant, we always report the ratio $\frac{\delta}{\mu}$.
- $N$ stress matrices are used to form the aggregated squared stress matrix.
- $D$ coordinate vectors are used to reconstruct the framework.

The numerical accuracy of our algorithm for the two testing graphs and two noise levels, under suitably chosen parameters are summarized in Table I. To provide an intuitive feel for these numbers, we present corresponding plots of the error vectors ($\rho - \rho'$) in Figure 3. Note that our algorithm handles the isotropic graph very well, both at low and high noise levels. The anisotropic testing case is heavily affected by the noise level though. For more experimental data on the effects of the parameters, see [11].

![Fig. 3. Realization results for isotropic (top row) and anisotropic (bottom row) testing cases under high (left) and low (right) noise condition.](image)

Experimentally we find our method produces results of similar quality on detected generically globally linked subgraphs of non-GGR graphs, though the optimal $D$ depends on subgraph size. (Setting $D$ proportional to the logarithm of the number of subgraph vertices produces good results for us.) Our measurement reduction strategy works very well with our isotropic testing graphs. For anisotropic testing graphs, the method can also localize a large portion of the graph. (For details, please see [11].)

### Table I

**Accuracy of Our Algorithm Under Suitable Parameters. In All Cases, $\mu = 4.0$ and $N = 200$.**

<table>
<thead>
<tr>
<th></th>
<th>isotropic (v=200)</th>
<th>anisotropic (v=137)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>high noise</td>
<td>low noise</td>
</tr>
<tr>
<td>$\epsilon R$ in $10^{-4}$</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>$\delta$ in $10^{-4}$</td>
<td>58</td>
<td>23</td>
</tr>
<tr>
<td>$\frac{\delta}{\mu}$</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>$\sigma_\rho$ in $10^{-4}$</td>
<td>48</td>
<td>6.3</td>
</tr>
<tr>
<td>$\sigma_\delta$ in $10^{-4}$</td>
<td>21</td>
<td>2.4</td>
</tr>
</tbody>
</table>

**REFERENCES**


