Localization and Rigidity

Presenter: Tina Nolte
Readings:
Aspnes et al., A Theory of Network Localization
Connelly, R., Cornell University, rigidity theory manuscript, "Basic concepts"
Next time: Time synchronization papers

1 Network localization

Consider a connected network N in d-space, with n nodes, labelled 1 through n, located at fixed positions $p_1, \cdots, p_n$ in $\mathbb{R}^d$. Each node also has neighboring nodes; the neighbor relation is symmetric. The first m nodes, labelled 1 through m, $0 < m < n$, are special beacon nodes. (It does not seem that the cases where $m = 0$ or $m = n$ present any problems, however.)

We can represent this network as an undirected graph $G_N = (V, E_N)$, with vertices 1 through n, and edges between vertices that are neighbors in the network.

Then, the *network localization (nl) problem with distance info* for network N is, given the network graph, positions of beacons, and distances between neighbor pairs ($\delta_N(i,j)$ for $i,j \in E_N$), to determine positions of all nodes in the network consistent with the distance constraints:

$$nl_N(G_N, \delta_N, \{p_1, \cdots, p_m\}) \text{ returns positions } \{p'_{m+1}, \cdots, p'_n\}.$$

The *network localization solvability (nls) problem* for network N asks whether there is exactly one set of positions, namely $\{p_{m+1}, \cdots, p_n\}$, for any particular *nl* problem above. $nls_N(G_N, \delta_N, \{p_1, \cdots, p_m\})$ returns true if there is exactly one solution to the *nl* problem for the given inputs, and false otherwise.

The *generic network localization solvability problem* for network N with nodes at positions $\{p_1, \cdots, p_n\}$ asks if the *nl* problem is solvable at each point in an open neighborhood of $\{p_1, \cdots, p_n\}$ in $\mathbb{R}^{nd}$.

Intuitively, this says that the problem is solvable not only for the given network, but also for slightly perturbed versions. There are examples of network localization problems that are solvable, but not generically solvable; these networks are degenerate, in a sense that will be made plain later.

In a little more detail, if coordinates of nodes in the network are algebraically independent, we call the positions *generic*. A generic property of a graph is a property that is true of "most" configurations of a graph, namely ones whose positions are generic. As we'll see later, for many graph properties we are interested in, the configurations for which the properties hold form an open dense set in $\mathbb{R}^{nd}$.

2 Rigidity

We can study the network localization solvability problem using concepts from rigidity theory. Rigidity theory developed in civil engineering, math.

2.1 Introductory concepts

We will be concerned with a modified form of a network graph, called a *grounded network graph*. This graph for a network is the same as the graph described above, except that each beacon node has an edge to each other beacon node, in addition to the other edges. Since the exact coordinates are known for beacons, we can calculate the distance between each pair of beacon nodes. From this point on, we will be dealing with grounded graphs.

We can model a network with a *point formation*.
Say we have point locations $p_1, \cdots, p_n$, and a set of links between neighboring points, L.
Then $F_p = (\{p_1, \cdots, p_n\}, L)$ is a *d-dimensional point formation*, and:

1. Each p_i is a point in $\mathbb{R}^d$,
2. $p = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix}$,
3. L is a set of k links such that $L \subseteq \{(i,j)$ unordered pairs $|i \neq j \wedge i, j \in \{1, \cdots, n\}\}$,
4. $\forall (i,j) \in L : length((i,j)) = |i - j|$, the Euclidean distance between points p_i and p_j.

It's a model for an n-node network in $\mathbb{R}^d$, where points p_i represent the positions of nodes in $\mathbb{R}^d$, and the links in L label pairs of nodes for which distances between them are known. These correspond to edges in the grounded graph.

A formation $F_p = (\{p_1, \cdots, p_n\}, L)$ uniquely determines:

1. Undirected graph $G_{F_p} = (V, L), V = \{1, \cdots, n\}$,
2. Distance function $\delta_{F_p} : L \to \mathbb{R}$, where: $\forall (i,j) \in L : \delta_{F_p}((i,j)) = length((i,j))$.

A map $T : \mathbb{R}^d \to \mathbb{R}^d$ is *distance preserving* if: $\forall p, q \in \mathbb{R}^d : |T(p) - T(q)| = |p - q|$.

Two n-point columns p and q in $\mathbb{R}^d$ are *congruent* if:
$\exists T : \mathbb{R}^d \to \mathbb{R}^d : [T$ is a distance preserving map $\wedge \forall i \in \{1, \cdots, n\} : T(q_i) = p_i]$.

Two point formations F_p and F_q are *congruent* if:

1. $G_{F_p} = G_{F_q}$, and
2. Point columns p and q are congruent.

This is like saying that one formation can be turned into the other by taking the formation and translating, rotating, or reflecting it as a whole. Notice that: F_p congruent to $F_q \Rightarrow \delta_{F_p} = \delta_{F_q}$.

It's clear that G_{F_p} and δ_{F_p} uniquely determines formation F_p *at most* up to congruence.
Global rigidity captures the case where a graph and distance function uniquely determine a formation *exactly* up to congruence.

Point formation F_p in d dimensions is *globally rigid* if:

$$\forall F_q \text{ in } d \text{ dimensions} : [(G_{F_p} = G_{F_q} \wedge \delta_{F_p} = \delta_{F_q}) \Rightarrow F_p \text{ is congruent to } F_q].$$

Here's a useful fact:

Lemma 2.1 Any formation F_p whose graph G_{F_p} is complete is globally rigid.

The way to think about this is that there is no wiggle room.

A set of points $\{p_{i_1}, \cdots, p_{i_{d+1}}\}$ in d dimensions is in *general position* if it does not lie in a proper subspace. For example, in 2 dimensions, a set of 3 points are in general position if the points aren't collinear; in 3 dimensions, a set of 4 points are if they aren't coplanar.

Lemma 2.2 (Lemma 1 in Aspnes) Let F_p be an n-point formation in $\mathbb{R}^d$ that contains $d+1$ points, $\{p_{i_1}, \cdots, p_{i_{d+1}}\}$, in general position. Suppose G_{F_p} contains a complete graph on $\{p_{i_1}, \cdots, p_{i_{d+1}}\}$. If the only n-point formation in $\mathbb{R}^d$ that contains these $d+1$ points and has the same link set as F_p is F_p, then F_p is globally rigid.

Proof sketch: Say there is some formation F_q that has the same graph and edge distances as F_p, but isn't congruent to F_p. As mentioned in Lemma 2.1, a formation whose graph is complete is globally rigid, meaning its positions are unique, up to congruence. Any distance preserving map that maps the complete graph of F_q into the complete graph of F_p also preserves the distances between the other points in F_q as well, meaning that F_q is mapped into a formation with the same graph and link distances as F_p, and contains a complete graph on points $\{p_1, \cdots, p_m\}$, but is different from F_p, a contradiction. ■

Network localization solvability as global rigidity testing

We can now recast network localization solvability in terms of global rigidity.

Network localization solvability for a network N, described with point formation F_p, given G_{F_p}, δ_{F_p}, and beacon positions is written as: $nls_{F_p}(G_{F_p}, \delta_{F_p}, \{p_1, \cdots, p_m\})$, returning either $true$ or $false$. We drop the F_p parameter from nls if it is obvious.

We can restate a network localization solvability problem in terms of its point formation and global rigidity:

Theorem 2.3 (Theorem 1 in Aspnes) In $\mathbb{R}^d, d = 2$ or 3,
(1) $nls_{F_p}(G_{F_p}, \delta_{F_p}, \{p_1, \cdots, p_m\}) \Rightarrow F_p$ globally rigid,
(2) [F_p globally rigid $\wedge \{p_1, \cdots, p_m\}$ contains $d+1$ beacons in general position]
$\Rightarrow nls_{F_p}(G_{F_p}, \delta_{F_p}, \{p_1, \cdots, p_m\})$.

Proof sketch: (1) If F_p were not globally rigid, it would be impossible to determine F_p up to congruence, let along to determine it uniquely.
(2) If F_p is globally rigid, solvability reduces to making sure that distance preserving maps $T : \mathbb{R}^d \to \mathbb{R}^d$ for which $T(p_i) = p_i, i \in \{1, \cdots m\}$ also satisfy $T(p_i) = p_i$, for $i \in \{m+1, \cdots, n\}$. We have seen in Lemma 2.2 that if $\{p_1, \cdots, p_m\}$ contains $d+1$ points in general position, then the only T that leaves $\{p_1, \cdots, p_m\}$ the same is the identity map. ■

	Local	*Global*
Formation	rigid first-order rigid	globally rigid generically globally rigid
Graph	generically rigid	redundantly rigid generically globally rigid

Figure 1: Different forms of rigidity we'll discuss.

2.2 Rigidity

As seen in Theorem 2.3, under certain circumstances, solvability of network localization is equivalent to global rigidity.

How do we check for global rigidity efficiently in general? For this, we first define rigidity (sometimes called local rigidity), a weaker form of rigidity.
We'll be introducing a number of different kinds of rigidity, both for point formations and for graphs, local and global, generic, not generic (Figure 1).

What is rigidity? For a point formation, rigidity describes the intuition that there are no continuous motions of the vertices satisfying distance constraints on edges, except for the trivial continuous motions (translations and rotations) coming from congruences of all of $\mathbb{R}^d$.

We say that analytic (or continuous) path $F_{p(t)}$ is an *analytic flex* of F_p if:

1. $F_{p(0)} = F_p$, and
2. $\forall t, 0 \leq t \leq 1 : (G_{F_p}, \delta_{F_p}) = (G_{F_{p(t)}}, \delta_{F_{p(t)}})$.

F_p is *rigid* in $\mathbb{R}^d$ if: $\forall F_{p(t)} : F_{p(t)}$ an analytic flex of $F_p \Rightarrow F_{p(t)}$ congruent to F_p.

It turns out it is hard to characterize whether a formation is rigid.
Want a simple sufficient condition test that is easy to compute. Hence, we study a sort of "linearized" version of rigidity, described with the tangent space of all possible formations with the same distances for edges.
Consider a formation F_p with link set L and one of its analytic flexes, $F_{p(t)}$. Since an analytic flex preserves the distance constraints on links for all $t \in [0, 1]$:

$$\forall (i, j) \in L : |p_i(t) - p_j(t)|^2 = (\delta_{F_p}(i, j))^2.$$

We can study the flex's tangent space by taking the derivative of the above expression w.r.t. t, at $t = 0$. Since distances between points sharing an edge aren't changing, this derivative is equal to 0:

$$\forall (i, j) \in L : 2(|p_i - p_j|)(|p'_i - p'_j|) = 0.$$

A vector p' satisfying this set of equations is called a *first-order (or infinitesimal) flex*.

Notice that first-order flexes will always contain translations and rotations; these are the *trivial flexes*.

A formation F_p is called *first-order rigid* if every first-order flex is trivial. Notice that first-order rigid formations don't have to be globally rigid (see Figure 2).

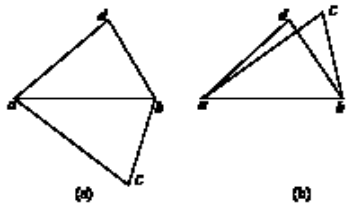

Figure 2: (2D) Two first-order rigid and rigid (see Theorem 2.5), but not globally rigid, formations. They demonstrate flip ambiguity.

We can use linear algebra to determine if a formation is first-order rigid. Namely, we recast the equations above as matrices. First, the distance equation:

$$\forall (i,j) \in L : (p_i(t) - p_j(t))^T (p_i(t) - p_j(t)) = (\delta_{F_p}(i,j))^2.$$

The derivative constraints can be rewritten as:

$$\forall (i,j) \in L : (p_i - p_j)^T (p_i' - p_j') = 0.$$

The rigidity matrix for F_p with n vertices in $\mathbb{R}^d$, $R(F_p)$, is an $|L| \times dn$ matrix, with rows indexed by edges in L, and columns indexed by vertices, such that the entry in a row corresponding to link $(i,j) \in L$ is $p_i - p_j$ in column i, $p_j - p_i$ in column j, and 0 otherwise.

The entire system can be written as:

$$R(F_p)p' = 0.$$

(If you were to do the expansion, this is obvious, but running through the expansion is tedious.)

Now, whether a formation only has trivial flexes can be expressed as a question about rank of the rigidity matrix:

Theorem 2.4 (Theorem 2 in Aspnes) Consider a formation F_p with at least d nodes in d-space. The following are equivalent:

1. F_p is first-order rigid.
2. rank $R(F_p) = dn - \begin{pmatrix} d+1 \\ 2 \end{pmatrix}$.

An algebraic geometry proof of this property can be found in the Connelly manuscript.

An n-point formation in d-dimensions has rank at most $dn - \begin{pmatrix} d+1 \\ 2 \end{pmatrix}$ (number of coordinates minus the dimension of the trivial flexes). If this inequality is an equality, then the trivial flexes are the only first-order flexes.

Part (a) of the following theorem is not easy to prove (look in the Connelly paper for details), but is a useful observation:

Theorem 2.5 Consider a formation F_p. Then:

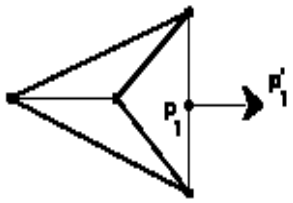

Figure 3: (2D.) Rigid, but not first-order rigid, formation (arrow indicates the non-trivial infinitesimal flex). Also example of a generically rigid graph (see Theorem 2.8). (Also example of a rigid formation in $\mathbb{R}^2$ that is not rigid in $\mathbb{R}^3$.)

1. If F_p is first-order rigid, then F_p is also rigid.
2. If F_p is rigid, it is not necessarily the case that F_p is first-order rigid (see Figure 3).

In the prior statements we have been concerned with rigidity of point formations. However, it turns out that it is convenient (and sometimes easier) to talk about rigidity properties of unweighted graphs, capturing just connectivity information. The idea here is that the unweighted graph of a formation can be generically rigid, in the sense that any nondegenerate formation with that graph is first-order rigid.

Graph G is *generically rigid* if the set $\{F_p | G = G_{F_p} \wedge F_p \text{ first-order rigid in } \mathbb{R}^d\}$ is dense.

With some topology and algebraic geometry, it is possible to prove the following (see Connelly manuscript for details):

Theorem 2.6 Let G be any graph with $n \geq d$ vertices in d-space. Then the set $\{F_p | G = G_{F_p} \wedge F_p \text{ first-order rigid in } \mathbb{R}^d\}$ is either an open dense subset of $\mathbb{R}^{nd}$ or is empty.

Using the above theorem, having one first-order rigid formation for a graph is enough to show that the graph is generically rigid:

Theorem 2.7 (Theorem 3 in Aspnes, simplified) Let G be any graph in d-space. The following are equivalent:

1. $\exists F_p | G = G_{F_p} \wedge F_p$ is first-order rigid in d-space.
2. G is generically rigid.

Another way of saying this is that if a graph has a single first-order rigid formation, then all its generic formations are rigid; if we force a formation to be generic, then rigidity is a property of the connectivity (not the geometry) of the formation. This suggests to us that there might be a combinatorial way to decide the generic rigidity of a graph.

In fact, there is, for 2-space:

Theorem 2.8 (Laman, and theorem 4 in Aspnes) An n-node graph G is generically rigid in $\mathbb{R}^2$ iff $\exists E \subseteq L : |E| = 2n - 3 \wedge \forall E' \subset E : |E'| \leq 2n' - 3$, where n' is the number of vertices which are endpoints of edges in E'.

Figure 4: Laman counterexample in $\mathbb{R}^3$ (not generically rigid).

These conditions are checkable in time $O(n^2)$. Result holds only for $d = 2$, and no comparable results are known for $d = 3$ (see Figure 4).

Some interesting things to note:

Theorem 2.9 Consider formations F_p and their graphs G_{F_p}. Then:

1. $\exists F_p$ such that F_p is rigid, but G_{F_p} is not generically rigid (Figure 5).
2. $\exists G, F_p$ such that $G = G_{F_p}$ and G is generically rigid, but F_p is not first-order rigid (see Figure 6). (F_p is then said to be in special position.)

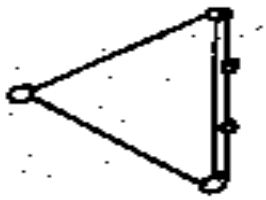

Figure 5: Globally rigid formation, non generically rigid graph.

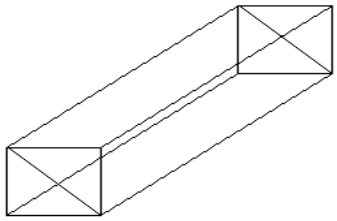

Figure 6: (2D.) Generically rigid graph, flexible formation. Also generically globally rigid graph (see Theorem 2.11).

2.3 Global rigidity

We've looked at (local) rigidity. However, global rigidity is more than just "no continuous deformation". There can be discontinuous flexes, etc. We can extend our study of first-order rigidity and generic rigidity to that of generic *global* rigidity.

Formation F_p is *generically globally rigid* if every sufficiently small perturbation q of p results in a globally rigid F_q.

The following links first-order rigidity and generic global rigidity of formations. Any non-degenerate generically globally rigid formation is first-order rigid (though the converse is not true):

Theorem 2.10 (Averaging, theorem 5 in Aspnes) Consider a non-degenerate formation F_p with nontrivial flex q'. Formations $F_{p+tq'}$ and $F_{p-tq'}$, for all $t > 0$, have the same edge lengths, but are not congruent.

This says that all first-order flexible formations have arbitrarily close formations in their rigidity neighborhoods that are not in the rigidity neighborhoods of each other (see Figure 7).

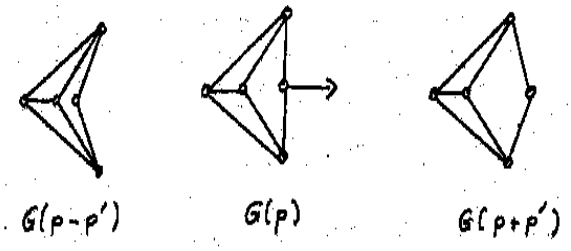

Figure 7: (2D.) First-order flexible formation with arbitrarily close formations that "snap" to each other.

As before, we want to examine properties of graphs, rather than formations.

A graph $G = (V, L)$ is *generically globally rigid* in $\mathbb{R}^d$ if there is an open dense subset of points p at which F_p is a globally rigid formation with link set L.

Again, we can characterize some properties of generically globally rigid graphs in a combinatorial way.

A generically rigid graph is *redundantly rigid* in $\mathbb{R}^d$ if the removal of any one edge results in a graph that is also generically rigid in $\mathbb{R}^d$. The graph has to be generically redundantly rigid to ensure generic global rigidity. A graph that is not redundantly rigid can suffer from what is called a *discontinuous flex ambiguity* (see Figure 8).

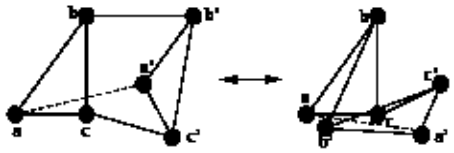

Figure 8: (2D.) Generically rigid 3-connected graph that is not redundantly rigid. Rigid, but not globally rigid, formations demonstrate discontinuous flex ambiguity.

A graph is *k-connected* if it remains connected after removal of any set of $< k$ vertices. The k-connectivity of a complete graph is $n - 1$. For more than $d + 1$ vertices in dimension d, we need at least $d + 1$-connectivity to avoid reflection of one component through a mirror placed on a disconnecting set of size d (referred to as a *flip ambiguity*) (see Figure 2).

For 2-space, we then have the following characterization of generically globally rigid graphs:

Theorem 2.11 (Theorem 6 in Aspnes) A graph G with $n \geq 4$ vertices is generically globally rigid in $\mathbb{R}^2$ iff it is 3-connected and redundantly rigid in $\mathbb{R}^2$.

Tests for both 3-connectivity and redundant rigidity in $\mathbb{R}^2$ are known. Hence, we have tests for generic network solvability in 2D.

Robust quads, which we've seen before, are an example of a generically globally rigid graph (see Figure 9).

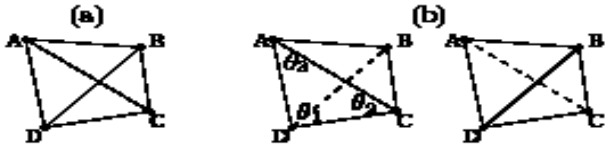

Figure 9: Robust quadrilateral.

Note also that collaborative multilateration's correctness follows from this result. Consider a complete graph on the beacons. The constructed tree is 3-connected, and redundantly rigid, guaranteeing the graph is generically globally rigid.

Also, in 2-space, the existence of one generically globally rigid formation implies the graph is generically globally rigid. In 3-space, however, it is an open question whether this holds.

We don't have a generalization of the above theorem to 3-space, but we can extend it as a necessary, but not sufficient, condition:

Theorem 2.12 (Theorem 7 in Aspnes) If a graph with more than $d+1$ vertices is generically globally rigid in d-space, then G is redundantly rigid and at least $d+1$ connected. In all $d \geq 3$, there are redundantly rigid and at least $d+1$ connected graphs that are not generically globally rigid (in 3-space, the complete bipartite graph $K_{5,5}$ is one such example).

It's good to note that there are generically globally rigid graphs with flexible formations. Such formations will be in special position (Figure 6 again).

2.4 Inductive construction of generically globally rigid graphs

We have sufficient conditions and inductive constructions for generically globally rigid graphs in all dimensional spaces. The construction inserts new nodes of degree $d+1$ into existing generically globally rigid formations to create larger generically globally rigid formations. This is a generalization of the term *trilateration* in the case of 2-space:

Lemma 2.13 (Lemma 2 in Aspnes) Given a generically globally rigid point formation F_p and a new point p_0 linked to $d+1$ nodes $p_1, \cdots, p_{d+1}$ that are non-collinear in 2-space (or non-coplanar in 3-space, etc.), then the extended formation F_{p+p_0} is generically globally rigid.

Proof sketch: We'll consider the proof in 2-dimensions, though the same argument will work in all dimensions, starting with the two points of intersection for all d spheres.

We start with three non-collinear points p_a, p_b, and p_c, and the distance from p_0 to each of these points. The distances from p_a and p_b define two intersections of corresponding circles centered at p_a and p_b. The distance from the third point p_c to these two solutions are different, since the three points were non-collinear. Therefore, there is a unique position for p_0 given the three distances (see Figure 10).

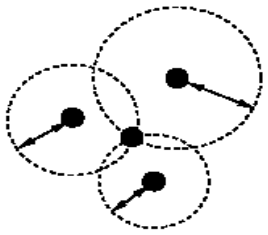

Figure 10: Finding unique position.

Now consider a second formation F_{q+q_0} with the same edge lengths. We know that the generically globally rigid formation F_p is contained in this extended formation, and hence the location of its nodes is unique up to congruence. The unique congruence T defined by the points of attachment induces a position $T(p_0)$. Since the constructed point was unique, we know that $T(p_0) = q_0$ and the two extended formations are congruent, giving that the extended formation is globally rigid.

Global rigidity holds for small perturbations of general position points, so the extended formation is generically globally rigid. ■

Now, we can see a practical connection to the generic solvability problem. For example, in 2-D, this means that we can start with a globally rigid formation on $m \geq 3$ beacons, the complete graph. We can then sequentially add new nodes as $p_{m+1}, \cdots, p_n$, each with 3 edges to distinct nodes in the existing formation to make a new formation. If all sets of points used in extensions are non-collinear, the new formation will be generically globally rigid.

Referring back to robust quads, this lemma explains why their pasting within connected overlap graphs is something that results in a globally rigid formation. Two robust quads can uniquely localize wrt each other if they share three nodes; these three nodes can be seen as forming the initial $d+1$ nodes in some trilateration graph, etc.

More generally, a *trilateration extension* in dimension d of a graph (not formation) $G = (V, E), |V| \geq d+1$, produces a new graph $G' = (V \cup \{v\}, E \cup \{(v, w_1), \cdots, (v, w_{d+1})\})$, where $v \notin V, w_i \in V$. A *trilaterative ordering* for dimension d and graph G is an ordering of the vertices $1, \cdots, n$ such that the complete graph on the initial $\geq d+1$ vertices is in G and for every vertex $j > d+1$, there are at least $d+1$ edges to vertices earlier in the sequence. Graphs that have a trilaterative ordering in dimension d are called *trilateration graphs* in dimension d.

Theorem 2.14 (Theorem 8 in Aspnes) Trilateration graphs in dimension d are generically globally rigid in dimension d.

Proof sketch: Any point formation for the complete graph on $d+1$ vertices in general position is generically globally rigid in dimension d. We repeatedly apply the previous lemma to add each point along the trilaterative ordering to get larger generically globally rigid formations with non-collinear/non-coplanar/etc. nodes. We can add additional edges beyond the $d+1$ needed, without violating generic global rigidity. We repeatedly do this. Since the points were in general position, we get that the graph is generically globally rigid. ■

Trilateration graphs are another way to see that the collaborative multilateration tree Nancy presented earlier is globally rigid; construct the graph from the beacons, up.

However, there are graphs that are generically globally rigid, but are not trilateration graphs (see Figure 11).

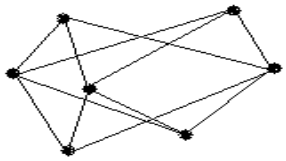

Figure 11: (2D.) Generically globally rigid graph, but not trilateration graph.

3 Complexity of localization

Now we move from the question of solvability of network localization to that of graph realization, the associated search problem. *Graph realization* is defined as the problem of assigning coordinates to vertices of a weighted graph G such that the edge weight of every edge (i, j) is equal to the distance between vertices i and j. We'll start by looking at localization of trilateration graphs, which can be done in poly time, and then we'll see that is a special case.

3.1 Trilateration graphs

Given a trilateration graph with realizable edge weights, it is realizable (we can assign coordinates) in a poly number of trilaterations. If we know an initial complete graph on $d+1$ vertices, we can localize one of the nodes of this graph at the origin, another on the positive x-axis, and the remaining node at a positive y-coordinate. At each trilateration step, we calculate the unique position of the node being localized. Positions for all nodes will be calculated within $O(|V|)$ trilaterations. If we don't know an intial complete graph, we can guess it in $\begin{pmatrix} n \\ 3 \end{pmatrix}$ tries. A guess is correct if it succeeds in localizing all nodes in a linear number of steps. All together, this still leaves us with a polynomial number of steps.

3.2 NP-hardness

We saw that we could efficiently realize a trilateration graph. Here we show that was a special case. We're in 2D.

Theorem 3.1 (Section V-A of Aspnes) Graph realization is NP-hard, even if it is known the graph is globally rigid and has a realization, and we restrict ourselves to 2-D.

Proof sketch: This will be done via a reduction with *set-partition-search*. We first show set-partition-search is NP-hard. We then describe wheel graphs and our reduction from set-partition-search to graph realization.

Given a set of numbers S with a set-partition, set-partition search finds the set partition. This is NP-hard. If you had an algo A for the search problem, you could use it to answer the question of whether a set-partition exists at all, which is NP-hard. Just run the algo for the amount of time equal to the running time of the algo on a valid input of the size. If it doesn't terminate, S has no set-partition. If it does, there is.

A *wheel graph* W_n is a graph with n vertices such that one vertex is in the center of a circle (the hub), with the remaining vertices on the circle perimeter. Each vertex on the circle is called a *rim node*, and is connected to the hub via a *spoke*. If we remove two rim vertices, the graph stays connected via the hub. If we remove the hub and a rim vertex, the rest of the graph is connected via the other vertices. Hence, the graph is 3-connected. It's also redundantly rigid, making it generically globally rigid.

Now for the main result. Say we have an algo that takes a realizable globally rigid weighted graph and outputs the unique realization. We describe how to translate a set-partition-search problem into graph realizability. Consider a set of n positive rational numbers $S = \{s_1, \cdots, s_n\}$, for which a set-partition exists, scaled so that $\Sigma_{i=1}^{n} s_i \leq \pi/2$. We label the hub 0, and the rim nodes 1 through n, where there is an edge from i to $i+1$ (with an edge from n to 1).

Each spoke has positive weight r. The weight of the rim edge between spoke i and $i+1$ is $2r\sin(s_i/2)$. This weighted graph has a realization in the plane. Since S has a set-partition, we can form a cycle of chords in 2-D. Note it's not necessarily the case that spokes are sequential on the rim; it's better to think of them arranged as a fan. Given a realization, we can determine the angle and sign (whether it is clockwise or counterclockwise) between spoke i and $i+1$. The positives s_i's are one partition, and the negative ones are the other partition (see Figure 12). ■

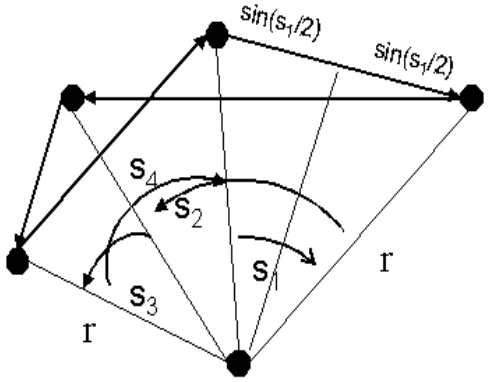

Figure 12: Example of set partition realization.

3.3 Sensor networks

OK, so localization is NP-hard in general. But what if we have some restrictions typical of sensor nets? In sensor networks with a concept of relatively fixed communication radius, we can represent the network using unit disk graphs. A *unit disk graph* is one where there is an edge between a pair of nodes if they are within some radius distance parameter r of each other.

The *unit disk graph reconstruction* decision problem asks if a particular graph with given edge lengths can be realized as a unit disk graph with a given disk radius (2-D). It turns out this problem is also NP-hard. This can be used to show that there is no efficient algo for localization in sparse sensor nets (unless P=NP). Turns out there is also no efficient randomized algo for sparse sensor nets that have unique reconstructions (unless RP-NP). These results still hold in cases where approximations to locations are allowed (within ϵr of the correct, for ϵ a constant).