cse590 Information Processing in Sensor Networks Lecture 3

Rigidity Theory

Lecturer: Jie Gao *Scribe: Amitabh Basu*

1 Overview of the Lecture

In this lecture, we looked at important results in rigidity theory. Given a set of fixed length bars connected by hinges, rigidity theory studies whether one can deform the shape continuously. Some of the main results in rigidity theory are presented in this lecture. In particular, combinatorial rigidity of graphs and the pebble game for testing rigidity of graphs are discussed. In the following sections, we develop the theory in greater detail.

2 Definitions

A *graph* $G = (V, E)$ consists of a set V of vertices and a set E of edges, with $|V| = n$, $|E| = m$.

Define a *configuration* $P(V)$ to be an assignment of the vertices V to points in $\mathbb{R}^d$. In this class we focus on $\mathbb{R}^2$. A configuration is called *generic* if the coordinates are algebraically independent over the rational. Intuitively speaking, a generic configuration has no degeneracy, i.e. no three points staying on the same line, no three lines go through the same point, etc. The study of generic configurations are important because in practice, we can assume that the sensor locations are generic.

A *framework* $G(P)$ refers to a graph G along with a configuration. A particular graph can have different frameworks, with different edge lengths. A framework is called ***flexible*** if there exists a continuous deformation from the given configuration to another, such that edge lengths are preserved. If no such deformation exists, then it is called *rigid*. See Figure 1.

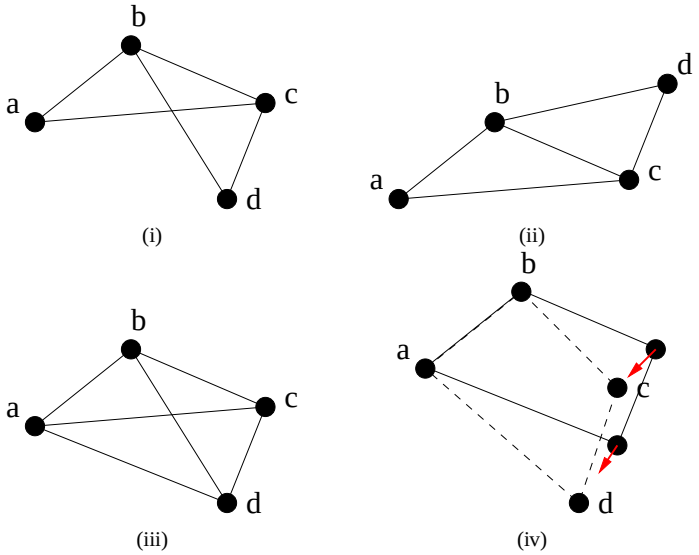


Figure 1: (i) Rigid but not globally rigid; (ii) Rigid but not globally rigid; (iii) Globally rigid; (iv) Flexible.

A configuration $P = \{p_1, p_2, \cdots, p_n\}$ can be represented by a point in $\mathbb{R}^{2n}$, namely, 2 variables for each point p_i. We define the configuration space $C(L)$ with respect to edge length constraints $L = \{\ell_{ij}\}$ as the collections of configurations that satisfy the edge length constraints L. Thus, $C(L)$ is a subspace of $\mathbb{R}^{2n}$.

Many questions regarding a framework can be formulated as questions on the configuration space. For example, whether there is a realization of a graph with distance constraints L is to ask whether $C(L)$ is empty. The degree of freedom of a framework is actually the dimension of the configuration space $C(L)$. If the configuration space has 0-dimension, then the framework is rigid. If $C(L)$ has only one point, then the framework is *globally rigid*.

A rigid framework is *minimally rigid* if it becomes flexible after an edge is removed. A rigid framework is *redundantly rigid*, if it remains rigid upon the removal of any edge.

Generally, questions about the configuration space are hard to answer. They are usually dealt by computational algebraic geometry or topology. However, the tangent space of $C(L)$ is relatively more tractable. The next section illustrates the important results regarding the tangent space.

3 Infinitesimal Motion and Infinitesimal Rigidity

Infinitesimal rigidity is defined over the tangent space. In particular, the configuration space of a framework is represented by

$$C = \{\mathbf{p_i} \in \mathbb{R}^2 | (\mathbf{p_i} - \mathbf{p_j}) \cdot (\mathbf{p_i} - \mathbf{p_j}) = \ell_{ij}^2, \forall ij \in E\}. \tag{1}$$

Now we take the derivative at the configuration point $\{\mathbf{p_1}, \mathbf{p_2}, \cdots, \mathbf{p_n}\}$. We have the following equation:

$$(\partial\mathbf{p_i} - \partial\mathbf{p_j}) \cdot (\mathbf{p_i} - \mathbf{p_j}) = 0, \forall ij \in E. \tag{2}$$

We define an *infinitesimal motion* $\mathbf{v} = \{\mathbf{v_1}, \mathbf{v_2}, \cdots, \mathbf{v_n}\}$ as a vector that satisfies the linear system in 2. We usually factor out the global rigid motion (translation and rotation) by pinning down an edge. So we include the "pin down" equations $\mathbf{v_i} = \mathbf{v_j} = 0$ for an edge ij in the graph.

This linear system (2 with the "pin down" equations) always has a trivial solution $\mathbf{v} = 0$. If it has a non-trivial solution, the infinitesimal motion is called non-trivial. A framework is *infinitesimally flexible* if it has a non-trivial infinitesimal motion. Correspondingly, a framework is *infinitesimally rigid* if it has no non-trivial infinitesimal motion. If a framework is flexible, then there exists a non-trivial infinitesimal motion. Therefore, if a framework is infinitesimally rigid, then it is rigid. However, there exist rigid, but not infinitesimally rigid frameworks. See Figure 2.

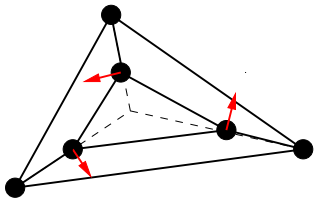


Figure 2: A rigid, but not infinitesimally rigid framework. The arrows show the non-trivial infinitesimal motion.

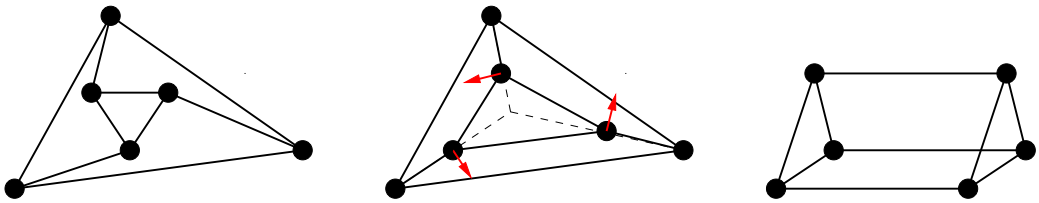
Computing an infinitesimal motion is easy – we simply solve the linear system. Now we represent the linear system in a matrix form:

$$R\mathbf{v} = 0, \tag{3}$$

where $\mathbf{v}$ is a $2n$-dimensional velocity vector, R is a $m \times 2n$ matrix known as the *rigidity matrix*. In particular, each row in the rigidity matrix corresponds to an edge in G. A row takes value $\mathbf{p_i} - \mathbf{p_j}$ corresponding to velocity for i, $\mathbf{p_j} - \mathbf{p_i}$ corresponding to velocity for j, and 0 otherwise. Since we do not include the "pin down" equations, the trivial motions (global translation and rotation) will yield non-trivial solutions which will form a subspace of dimension 3. Therefore, disregarding global translation and rotation, the framework $G(P)$ is infinitesimally rigid if and only if $rank(R) = 2n - 3$.

4 Generic or Combinatorial Rigidity of a Graph

So far, we have explored the rigidity of a *framework*, i.e. a graph *along with* a configuration. A given graph $G = (V, E)$ can have infinitesimally rigid, rigid but infinitesimally flexible, and flexible frameworks. As an example, refer to Figure 3. However, the situation changes if we insist on generic configurations of a graph. Rigidity under this constraint becomes a property of the graph as the following theorem indicates.



(i) (ii) (iii)

Figure 3: (i) Infinitesimally rigid (ii) Rigid but infinitesimally flexible (iii) Flexible

Theorem 1 *If a graph has a single infinitesimally rigid framework, then all its generic frameworks are rigid.*

Thus, a graph can be characterised to be either rigid or flexible, in a generic sense. In other words, if we force a graph's configuration to be generic then rigidity becomes a property of the connectivity and **not** of the geometry of the formation. This suggests that there must be a combinatorial way of deciding the generic rigidity of a graph, without referring to any particular configuration.

Thus we have now two ways of looking at rigidity. A framework can be either infinitesimally rigid or flexible and this property can be tested by linear algebra using the rigidity matrix. A graph can be rigid or flexible, if only generic configurations are considered.

5 Testing Generic Rigidity of a Graph

5.1 An Intuitive Approach to Testing Rigidity

Informally, we need to answer the following question : How many distance constraints are necessary to limit a framework to only trivial motion ? This is equivalent to asking how many edges are necessary for a graph of n nodes to be generically rigid. *A priory*, the n nodes have $2n$ degress of freedom (recall we are dealing with $\mathbb{R}^2$). Each edge that we add to the graph can remove at most one degree of freedom (it may not remove any degree of freedom, in which case it will be a redundant edge, as we shall see later). Global rotations and translations are always going to be possible, so at least $2n - 3$ edges are necessary for a graph to be rigid.

However, as we noticed above, $2n - 3$ edges are not always sufficient, because some edges might be redundant. See Figure 4.

Clearly, we need $2n - 3$ well-distributed edges. To put it more formally, if a subgraph has more edges than necessary, some edges are redundant. Non-redundant edges are independent, in the sense that their corresponding rows in the rigidity matrix are independent. Each independent edge always removes a degree of freedom. Therefore, $2n - 3$ independent edges will guarantee rigidity. The following subsection puts these intuitive arguments more formally.

5.2 Laman Condition and Henneberg Constructions

The following theorem is known as the Laman condition.

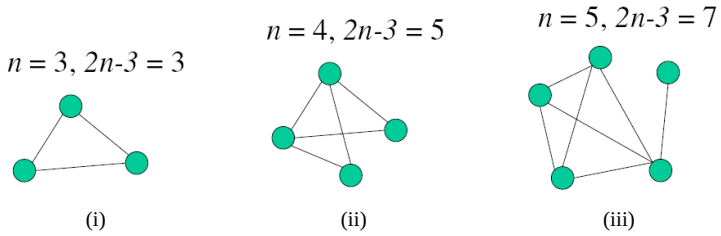


Figure 4: (i) Rigid (ii) Rigid (iii) Flexible

Theorem 2 *A graph is generically, minimally rigid in 2D if and only if it has $2n-3$ edges and no subgraph of k vertices has more than $2k-3$ edges.*

The Laman condition is a purely combinatorial condition, emphasizing that rigidity is a property of the connectivity, and not the geometry (if we insist on generic configurations). Moreover, from a computational point of view, the Laman condition provides a counting algorithm to test the generic rigidity of a graph.

To visualize Laman graphs better, we present a tool called the Henneberg construction.

Theorem 3 (Henneberg constructions) *A Laman graph can be constructed inductively by adding one vertex at a time :*

- *Start with an edge;*
- *At each step add a new vertex;*
- *Type I step : join the vertex to two old vertices via two new edges;*
- *Type II step : join the vertex to three old vertices with at least one edge in between, via three edges. Remove an old edge between these three end-points.*

See Figure 5.

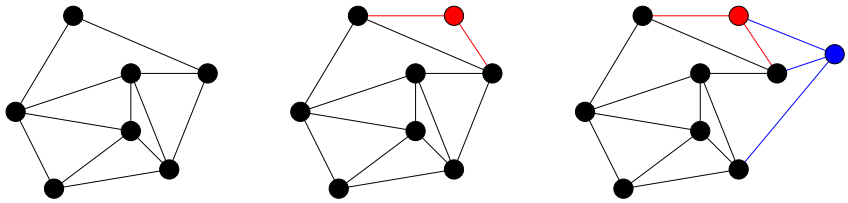


Figure 5: (i) Graph before Henneberg step (ii) Type I step (iii) Type II step

We show the equivalence of Laman graphs and Henneberg constructions in the following two claims.

Claim 4 *A graph constructed using Henneberg constructions is Laman.*

Proof By Induction. Suppose the current graph G is Laman with n vertices, $2n-3$ edges.
Type I step: Add node x. We have $n+1$ vertices, and $2n-3+2 = 2(n+1)-3$ edges. For any subgraph with k nodes, if it does not include x, by the induction hypothesis, there are at most $2k-3$ edges. If the subgraph includes x, for the other $k-1$ nodes, there are at most $2(k-1)-3$ edges between them (induction hypothesis), so in total there are at most $2(k-1)-3+2 = 2k-3$ edges.
Type II step: Add node x and connect it to a, b, c. We have $n+1$ vertices, and $2n-3+3-1 = 2(n+1)-3$ edges. For any subgraph with k nodes, if it does not include x, by the induction hypothesis, there are at most $2k-3$ edges. If the subgraph includes x, for the other $k-1$ nodes, there are at most

1. $2(k-1)-3$ edges, if not all of a, b, c are included.
2. $2(k-1)-4$ edges, if a, b, c are all included.

In case 1, there are at most $2(k-1)-3+2 = 2k-3$ edges. In case 2, there are at most $2(k-1)-4+3 = 2k-3$ edges. □

Claim 5 *Every Laman graph has a Henneberg construction.*

Proof If the number of edges in a graph is $2n-3$, there exists at least one vertex of degree 2 or 3. Otherwise, if all nodes have degree at least 4, we have at least $\frac{4n}{2} = 2n$ edges. If there is a degree 2 vertex in the graph, remove the vertex and its adjacent edges (Type I step in reverse). If there is a degree 3 vertex, remove that vertex and consider its 3 neighbours a, b, c. These cannot form a triangle, since it would violate the Laman condition for $k = 4$ with a, b, c, x. Put the missing edge among these 3 vertices. (Type II step in reverse). It can be shown like the previous proof that the resulting graph is again Laman. A simple inductive argument now completes the proof. □

The Laman condition in 2D does not have a neat extension to 3D. The following condition is necessary but not sufficient for rigidity in 3D : the graph must have $3n-6$ edges and no subgraph of k vertices has more than $3k-6$ edges. The combinatorial condition for rigidity in 3D is a long standing open problem. We will state the theorem of global rigidity before we move to the next section on the computational aspect of testing rigidity.

Theorem 6 *A graph is globally rigid if it is 3-connected and redundantly rigid.*

5.3 The Pebble Game : An Algorithm to Test Rigidity

The Laman condition taken literally leads to a poor algorithm, as it involves checking all subgraphs. A more efficient and intuitive algorithm exists, based on counting degrees of freedom. This is what is known as the Pebble Game.

The basic aim of the algorithm is to find a subset of independent edges. It is incremental in nature : pick an edge, test if it is independent of the current subset. The algorithm's correctness relies on an alternate formulation of the Laman theorem :

Theorem 7 *The edges in G are independent in 2D if and only if for each edge (a, b), the graph formed by quadrupling (a, b) has no induced subgraph of k nodes and $> 2k$ edges.*

The basic framework of the algorithm is as follows.

1. Grow a maximal set of independent edges one at a time.
2. New edge is added if it is independent of existing set. To check this, quadruple the edges and test the Laman condition.
3. If the Laman condition fails, then output "not rigid".

The question is how to test the Laman condition in step 2 quickly. The Pebble game does precisely this.

5.3.1 Pebble Game Algorithm

Each node is assigned 2 pebbles, giving us $2n$ pebbles in total. An edge is covered by having a pebble placed on either of its ends. A pebble covering is an assignment of pebbles such that all edges in the graph are covered. To test if a new edge is independent of the existing set, quadruple the edge and find a pebble covering for the 4 new edges. Formally, the pebble game algorithm works as follows :

1. Assume we have a set of edges covered with pebbles and we want to add a new edge. We have to repeat the following procedure 4 times for every copy of the edge that we are adding.
2. Look at the vertices adjacent to the new edge.
   - If either has a free pebble, use it to cover the edge. Give a direction to this edge such that it points away from the vertex which donated the pebble. Continue with a new edge.
   - Otherwise, their pebbles already cover existing edges. Then search for free pebbles in the directed graph of existing edges. Once a free pebble is found, swap pebbles and reverse edges appropriately, so that the new edge is covered. If no free pebble is found, then the new edge is not independent of the current set. We remove this edge and test the next edge.
3. Once all four copies have been processed, remove 3 of those and retain 1 copy for the edge that was just added to the independent set, with a pebble for it and its appropriate direction.
4. If we end up with $2n-3$ independent edges covered by pebbles, the graph is rigid. If we use up edges and do not find $2n-3$ independent edges, then the graph is not rigid.

5.3.2 Analysis and Applications of the Algorithm

Testing an edge for independence takes $O(n)$ time : we do a depth-first search in the directed graph with $O(n)$ edges (since we have at most $2n-3$ edges in the independent set). At most m edges will be tested. The total running time is therefore $O(nm)$.

Intuitively, each pebble corresponds to a degree of freedom. The final pebble covering always has at least 3 free pebbles. The algorithm is also amenable to a distributed implementation.

Other applications of the algorithm include identifying redundantly rigid sections of the graph. Also, using this in conjunction with 3-connectivity testing algorithms, it is possible to decompose networks into uniquely localizable regions.