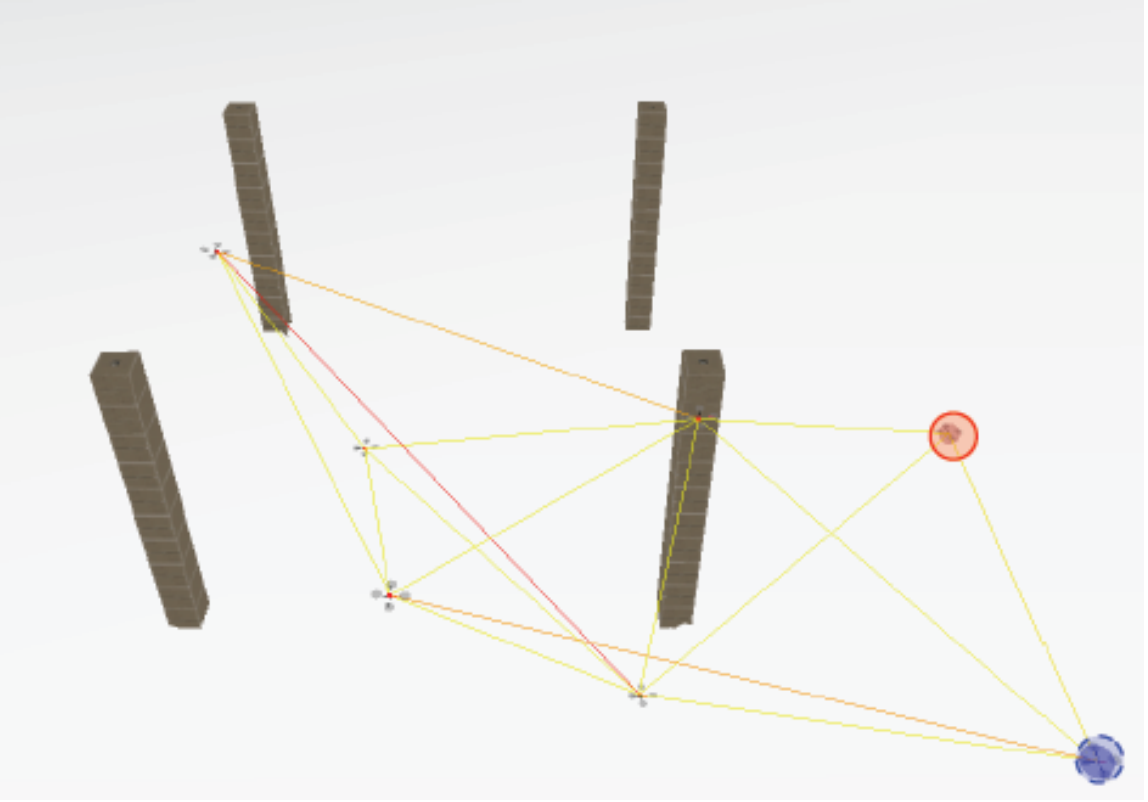
Formation Rigidity: Dynamic Maintenance and Optimality



Australian National University
July 18, 2012

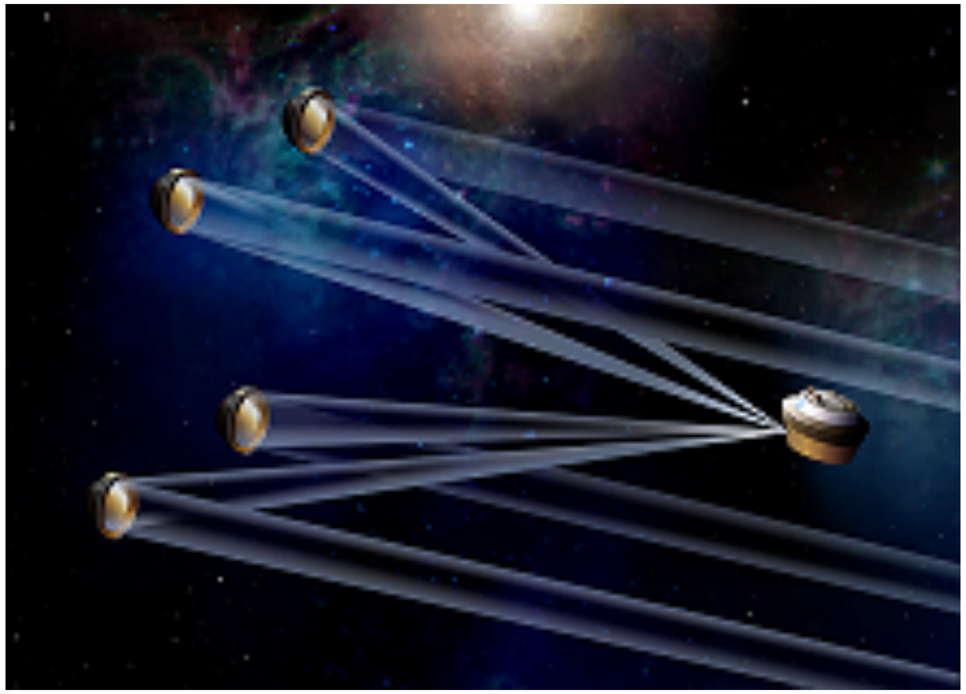
Daniel Zelazo

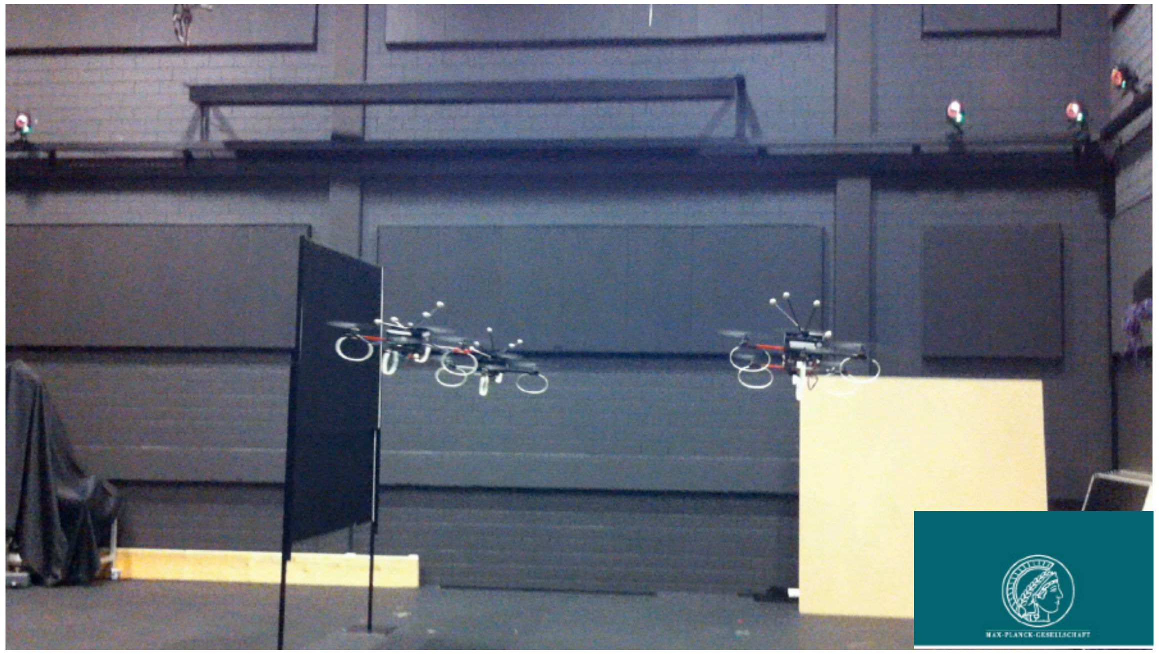
Institute for Systems Theory
and Automatic Control

Universität Stuttgart

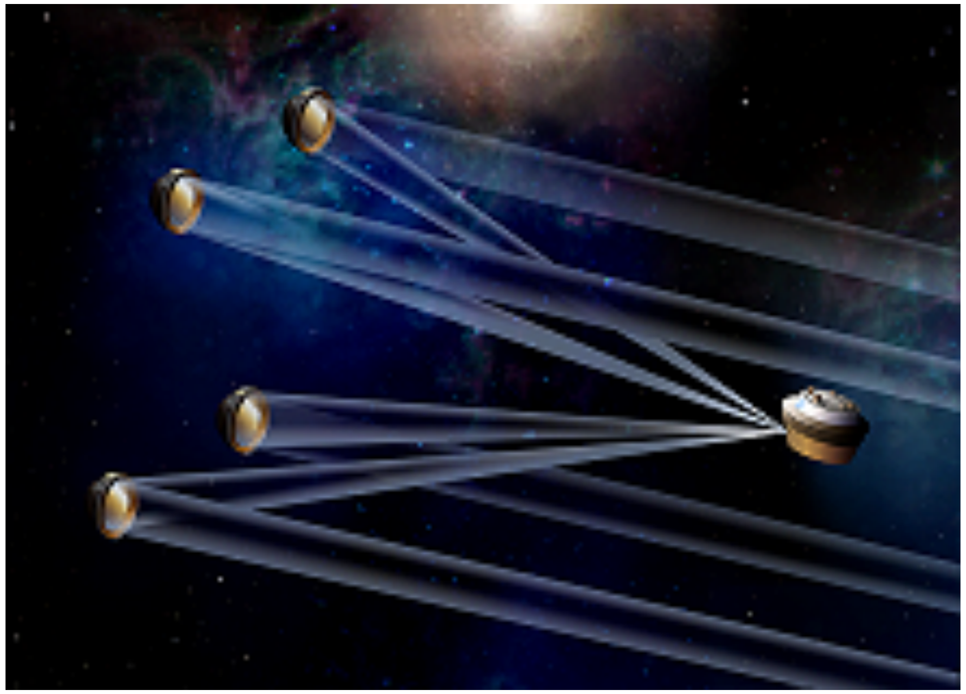
A Real Group Coordination Problem

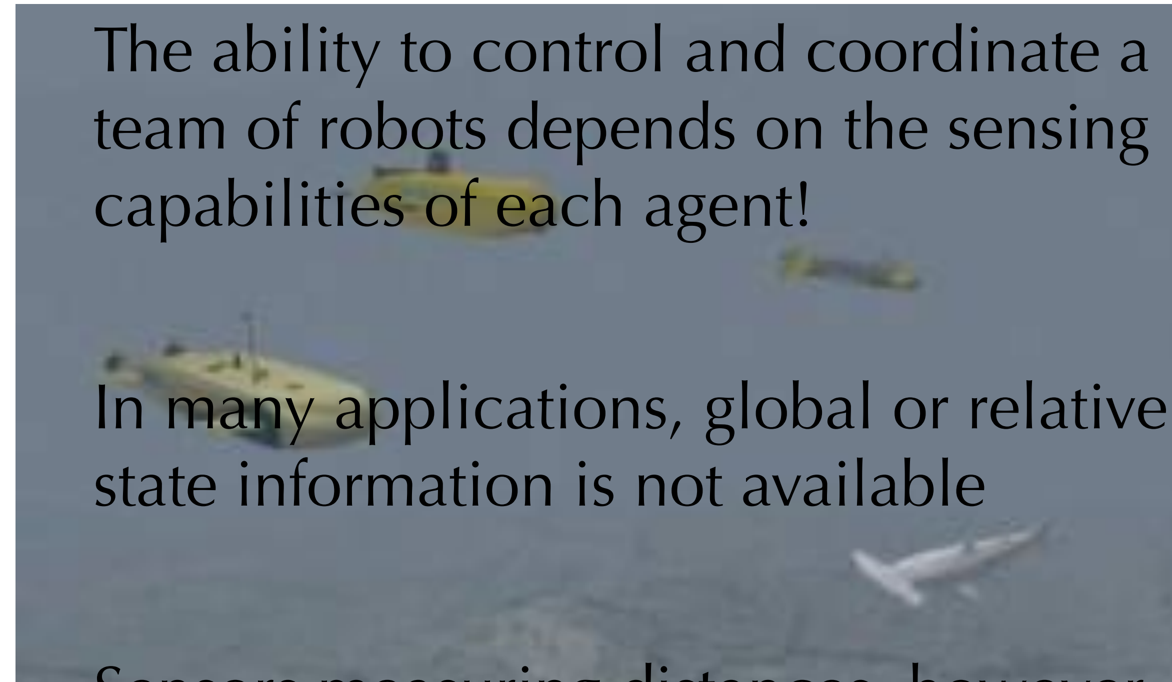
Coordination in harsh environments





Coordination in harsh environments

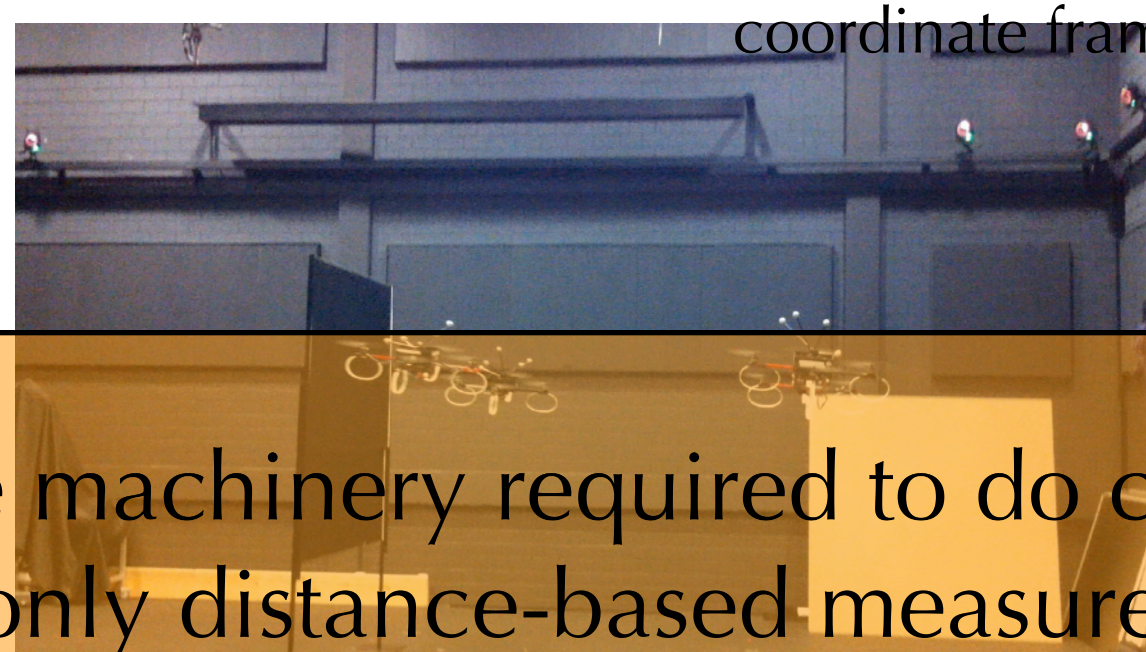




The ability to control and coordinate a team of robots depends on the sensing capabilities of each agent!

In many applications, global or relative state information is not available

Sensors measuring distances, however, are very accurate and independent of any coordinate frame



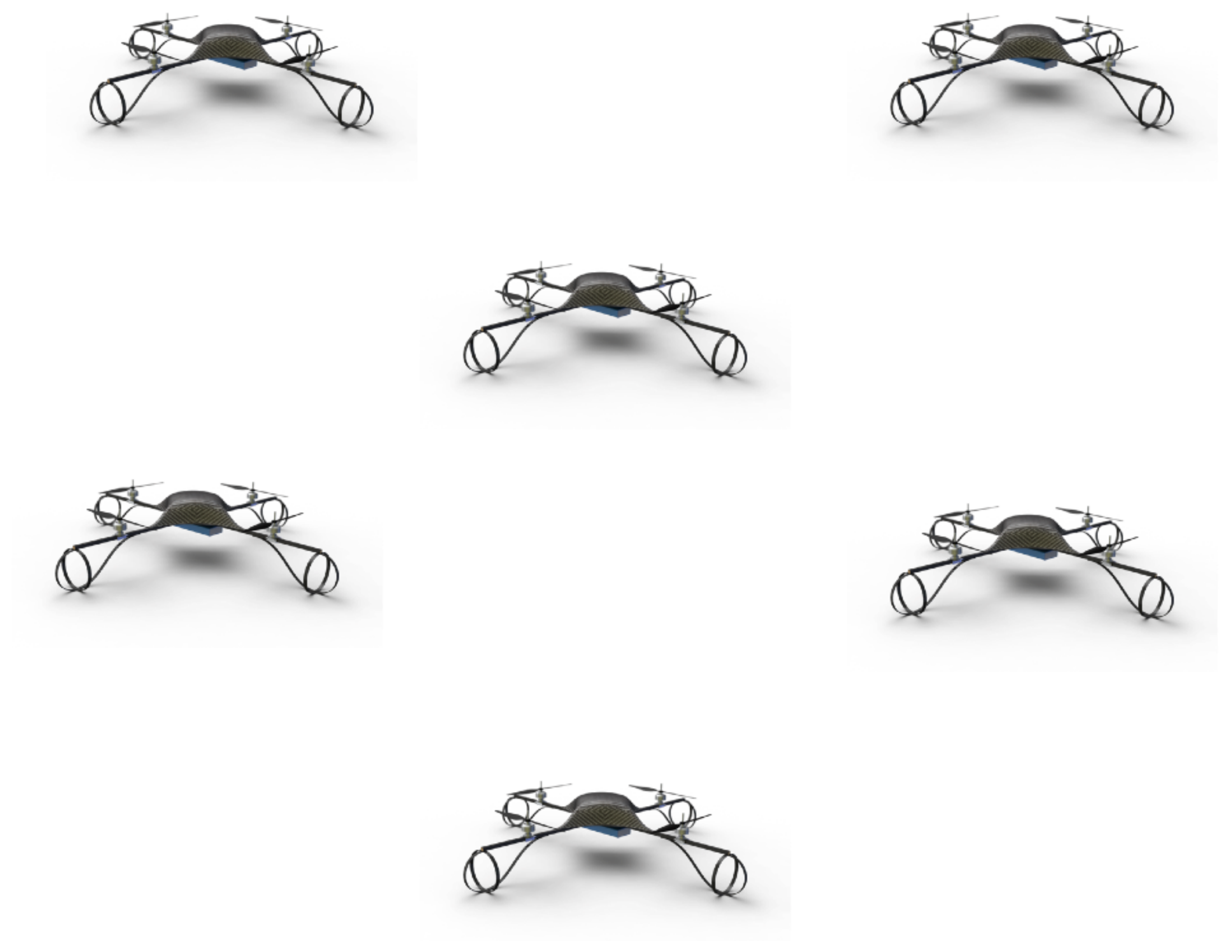
What is the machinery required to do coordination using only distance-based measurements?

Formation Rigidity

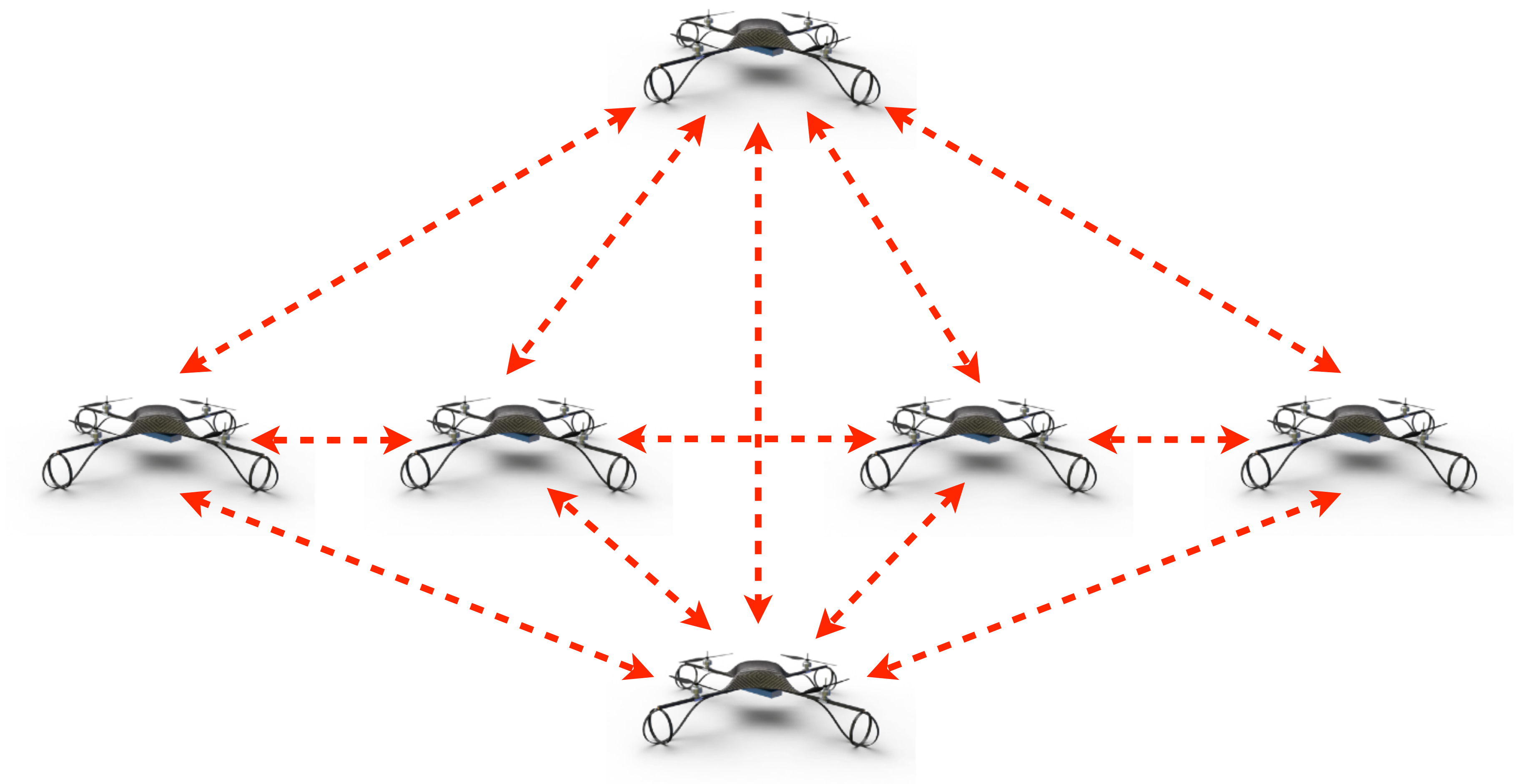
Outline

- ✧ Motivation
- ✧ Graph Rigidity and the Rigidity Eigenvalue
- ✧ Dynamic Rigidity Maintenance
- ✧ Optimally Rigid Formations
- ✧ Outlook

What is rigidity?



What is rigidity?

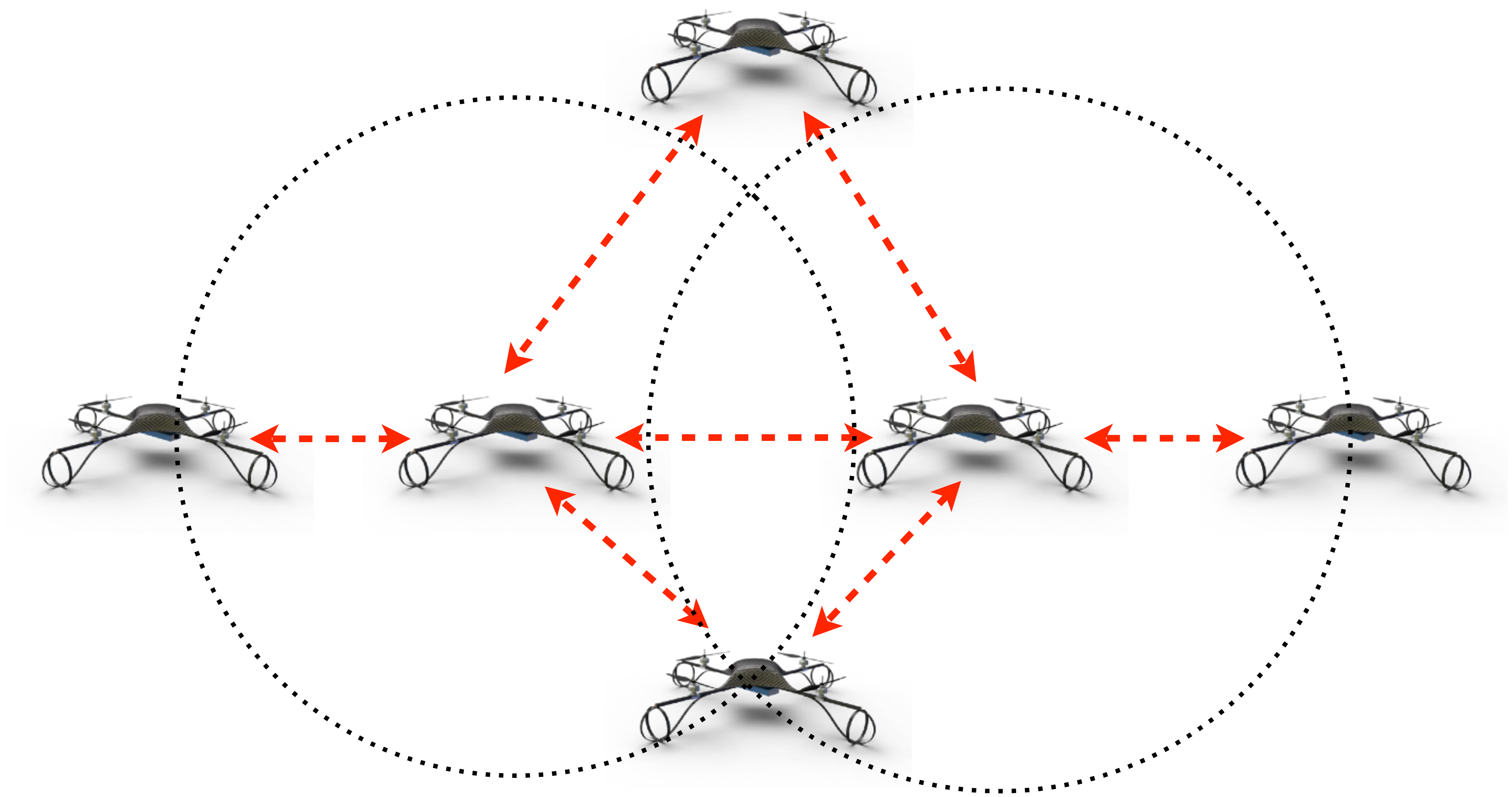


formation specified by a set of inter-agent distances

agents can measure distance to neighbors

sensor limitations only allow a subset of available measurements

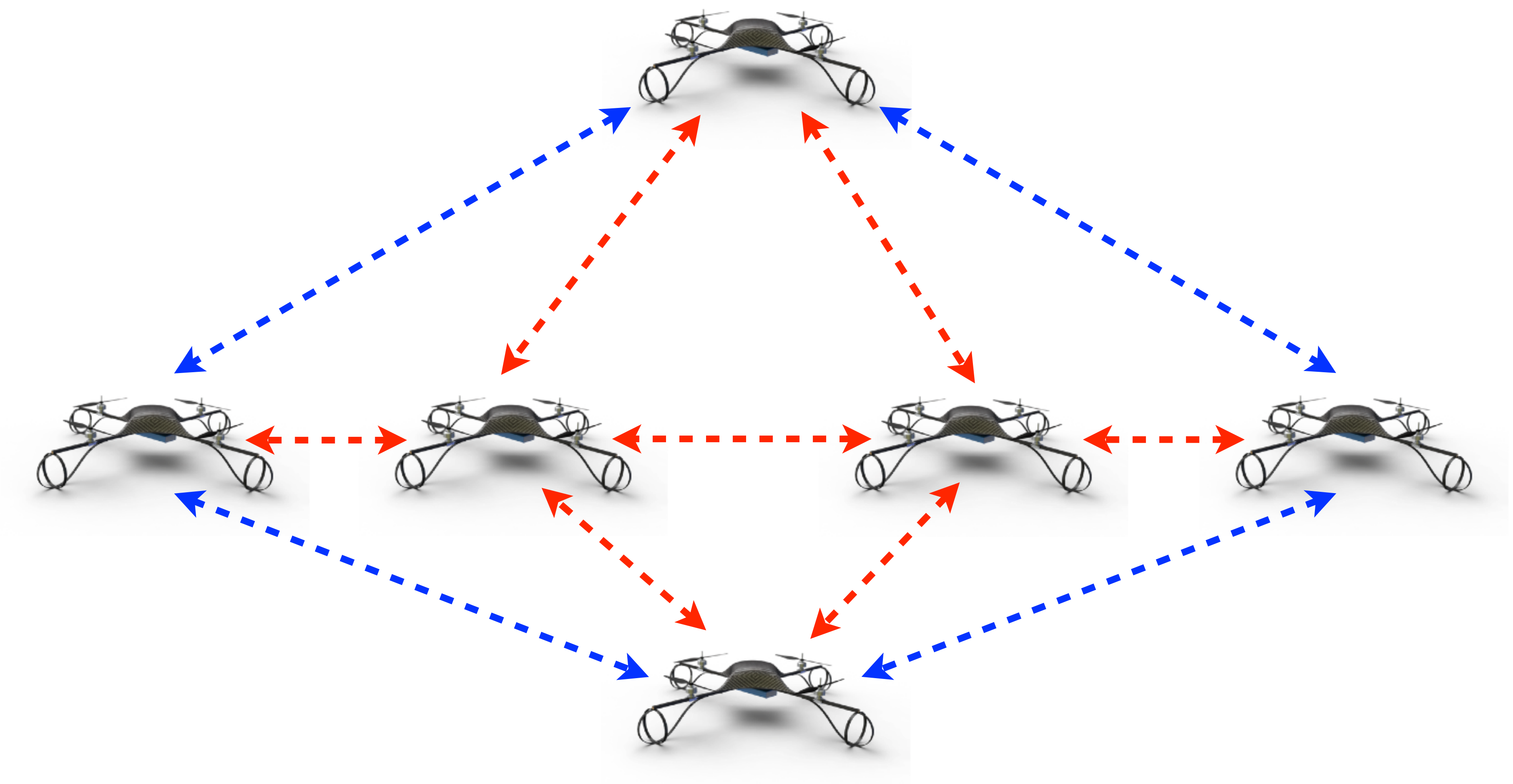
What is rigidity?



Can the desired formation be maintained using only the available distance measurements?

No!

What is rigidity?



A *minimum* number of distance measurements are required to *uniquely* determine the desired formation!

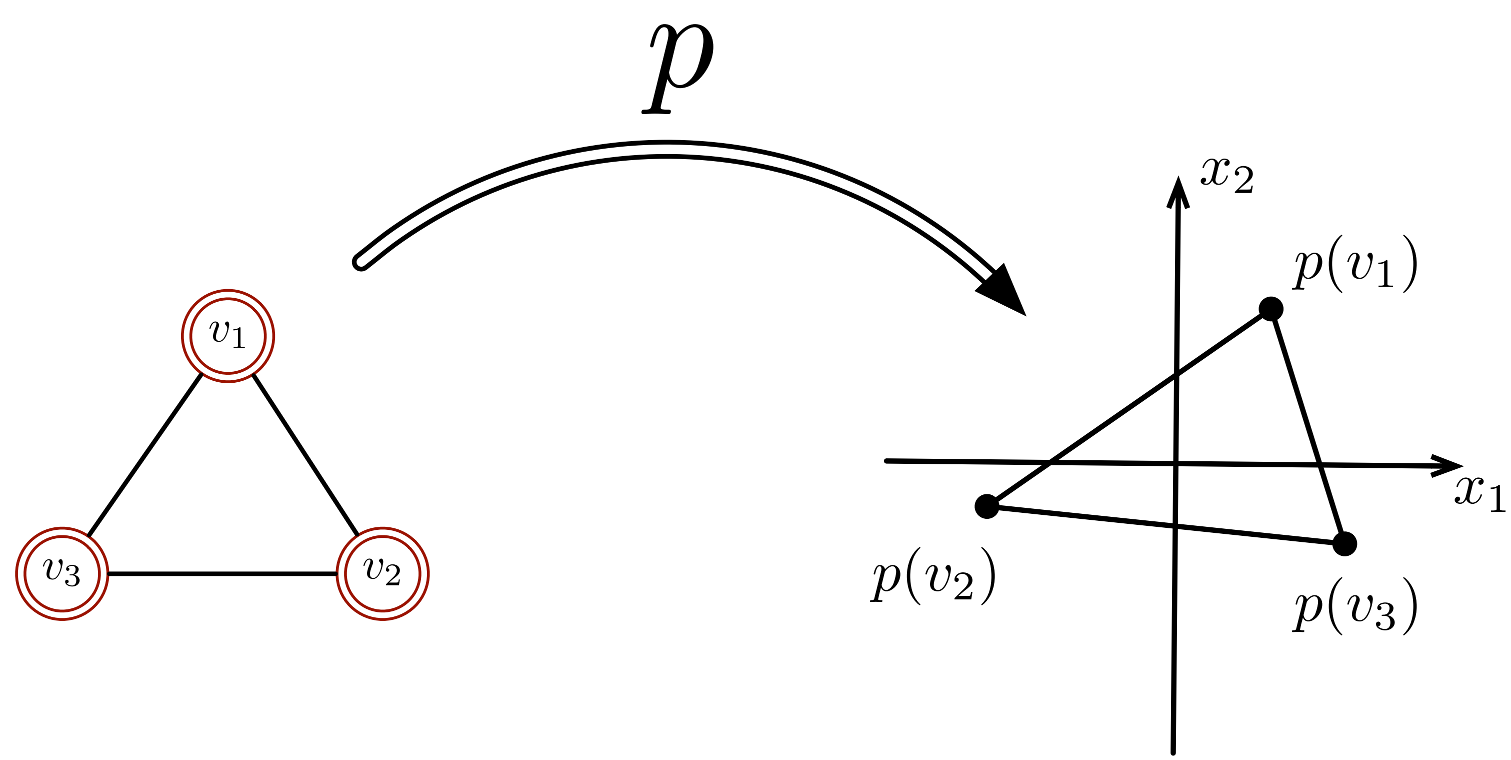
Graph Rigidity

Graph rigidity

bar-and-joint frameworks

$$\begin{cases} \mathcal{G} = (\mathcal{V}, \mathcal{E}) \\ p : \mathcal{V} \to \mathbb{R}^2 \end{cases}$$

maps every vertex to a point in the plane



Two frameworks are *equivalent if* $(\mathcal{G}, p_0)$ $(\mathcal{G}, p_1)$

$$\|p_0(v_i) - p_0(v_j)\| = \|p_1(v_i) - p_1(v_j)\|$$
$$\forall \{v_i, v_j\} \in \mathcal{E}$$

Two frameworks are *congruent if* $(\mathcal{G}, p_0)$ $(\mathcal{G}, p_1)$

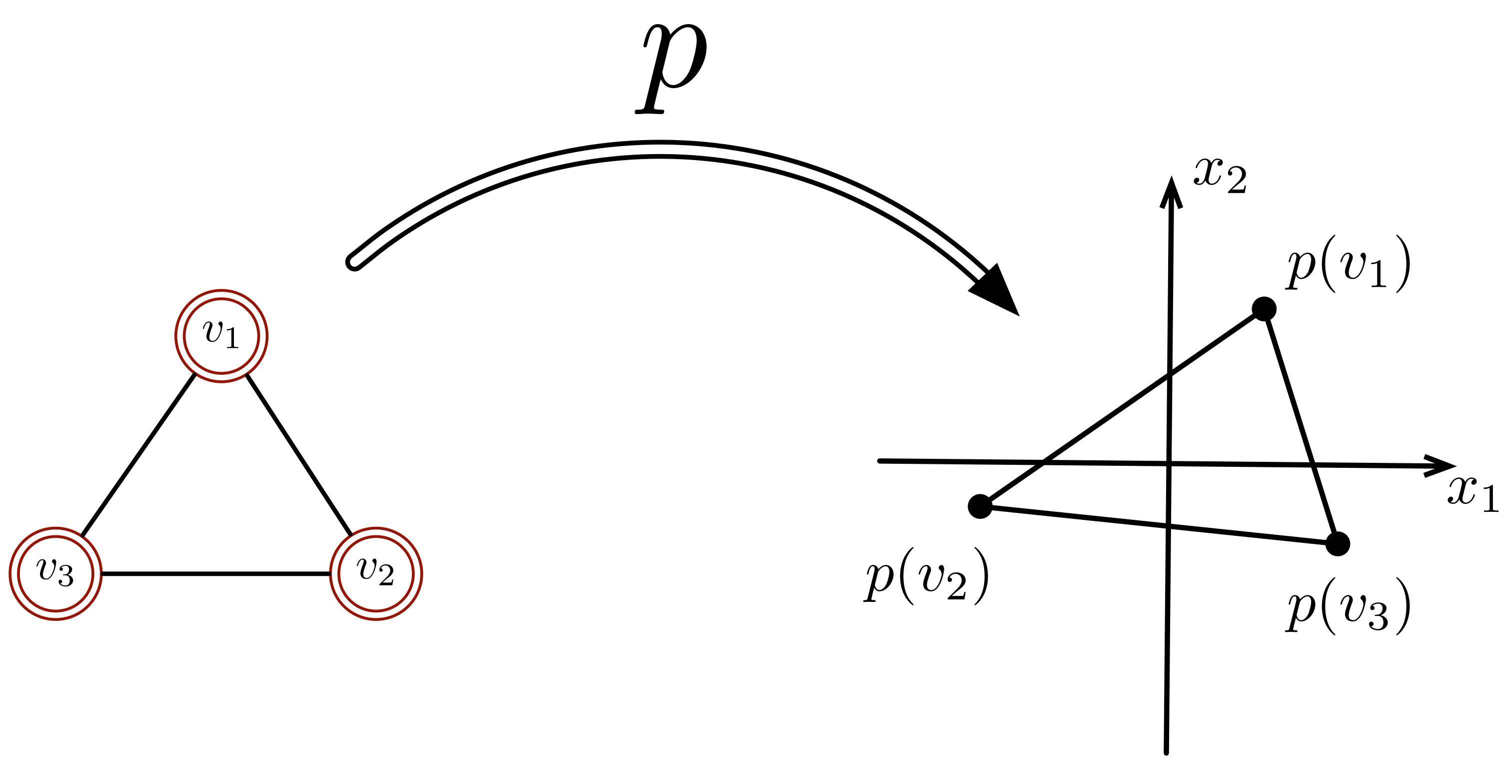
$$\|p_0(v_i) - p_0(v_j)\| = \|p_1(v_i) - p_1(v_j)\|$$
$$\forall v_i, v_j \in \mathcal{V}$$

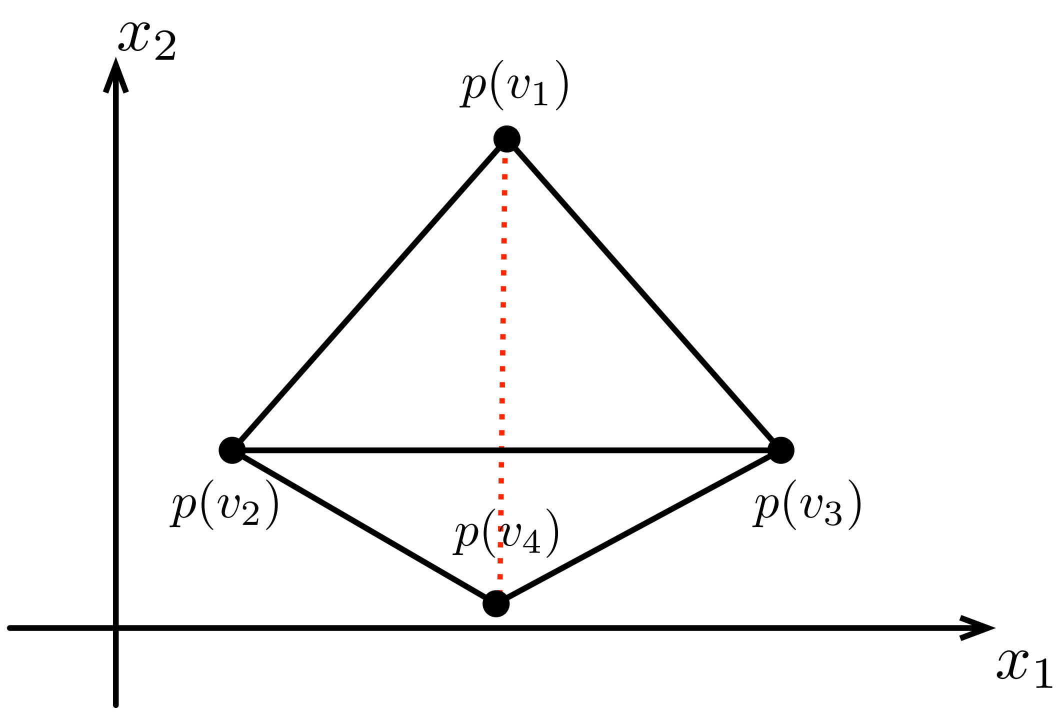
Graph rigidity

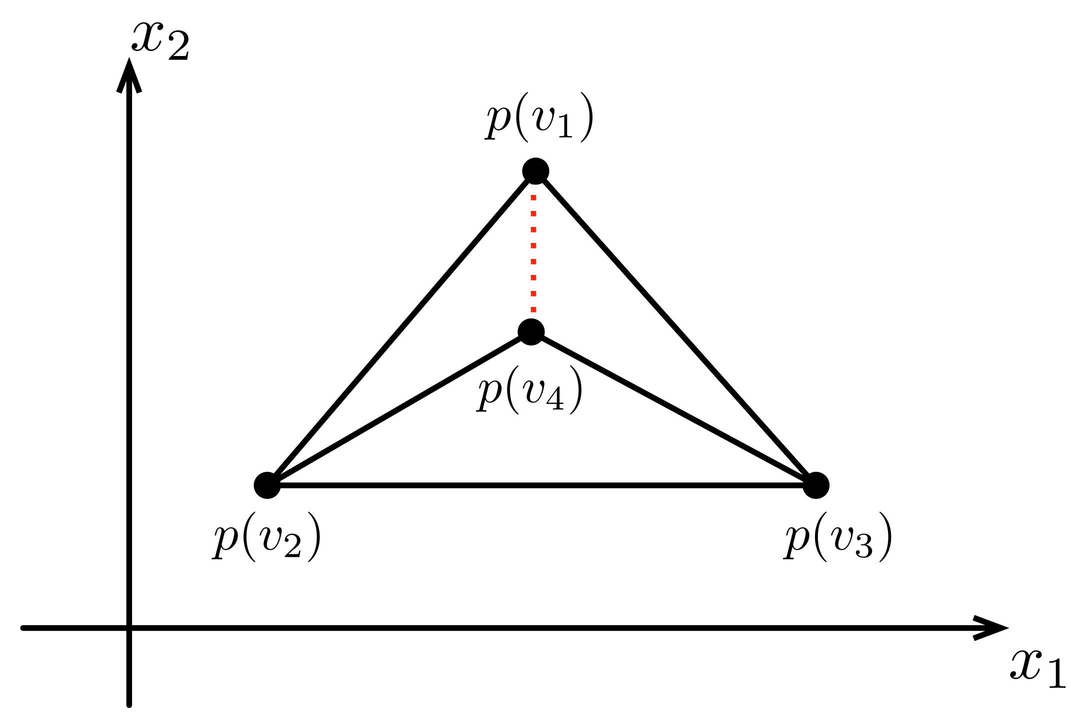
bar-and-joint frameworks

$$\begin{cases} \mathcal{G} = (\mathcal{V}, \mathcal{E}) \\ p : \mathcal{V} \rightarrow \mathbb{R}^2 \end{cases}$$

maps every vertex to a point in the plane





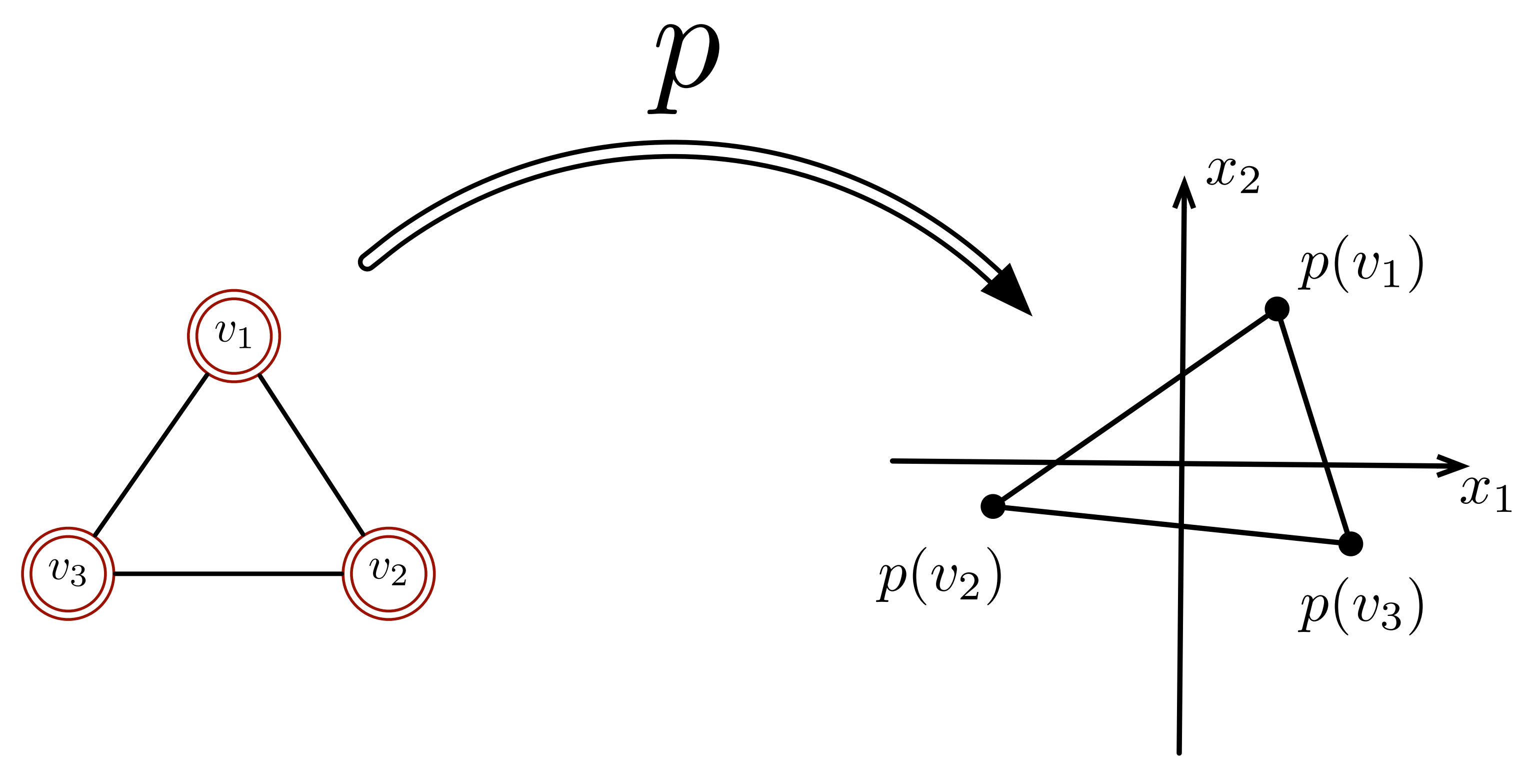


Graph rigidity

bar-and-joint frameworks

$$\begin{cases} \mathcal{G} = (\mathcal{V}, \mathcal{E}) \\ p : \mathcal{V} \to \mathbb{R}^2 \end{cases}$$

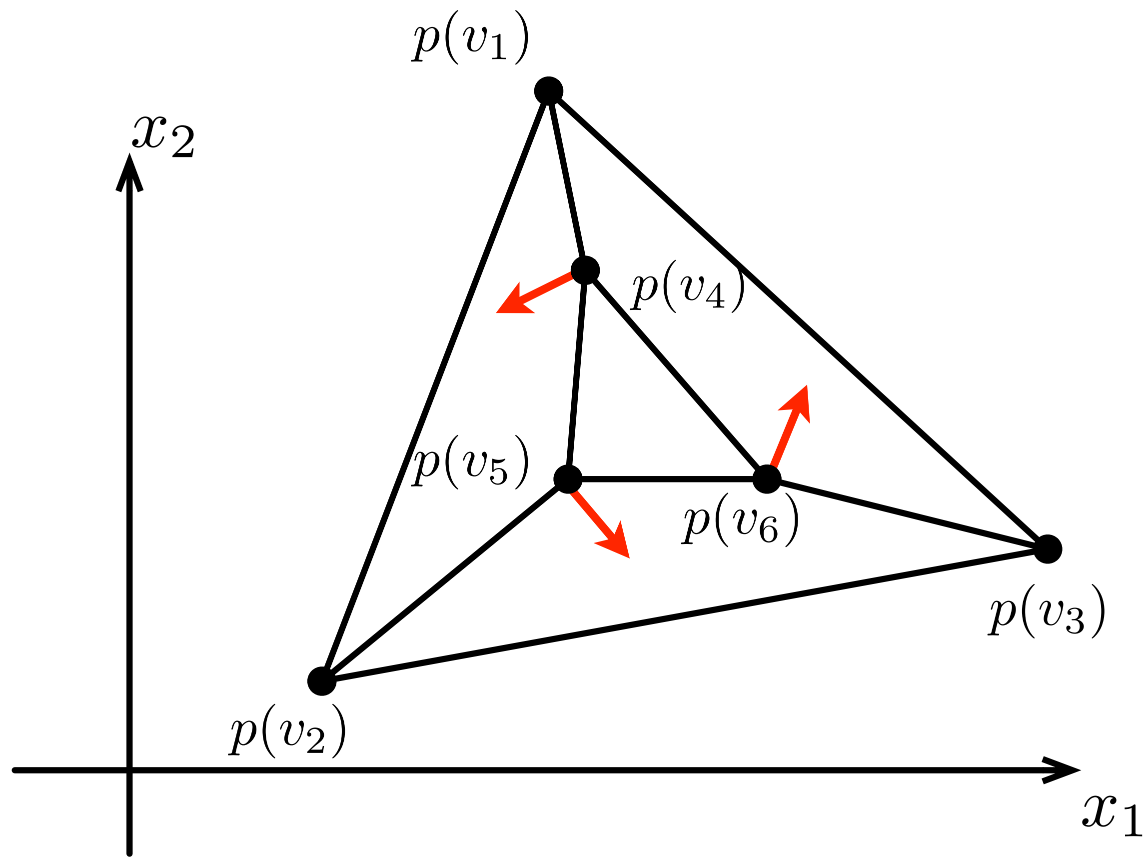
maps every vertex to a point in the plane



A framework $(\mathcal{G}, p_0)$ is *globally rigid* if every framework that is equivalent to $(\mathcal{G}, p_0)$ is congruent to $(\mathcal{G}, p_0)$.

Graph rigidity

$(\mathcal{G}, p)$



An *infinitesimal motion* is the assignment of a velocity vector to each node such that

$$(\xi(v_i) - \xi(v_j))^T (p(v_i) - p(v_j)) = 0$$

$$\forall \{v_i, v_j\} \in \mathcal{E}$$

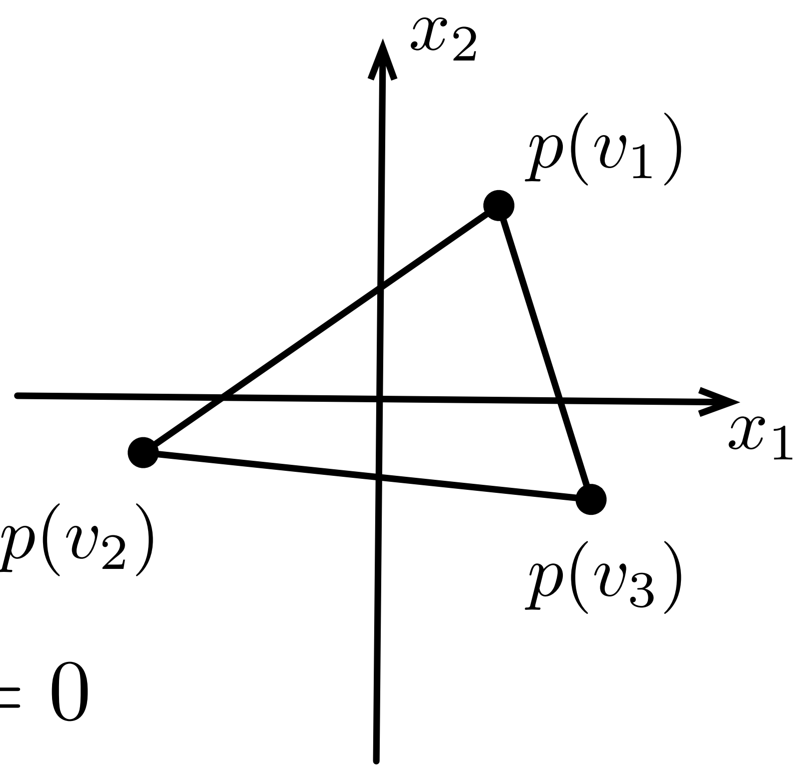
A framework $(\mathcal{G}, p_0)$ is *infinitesimally rigid* if every possible motion is trivial (rotations & translations)

A framework $(\mathcal{G}, p_0)$ is *minimally rigid* if the removal of any edge results in a non-rigid graph

Graph rigidity and the rigidity matrix

The Rigidity Matrix

$$R(p) \in \mathbb{R}^{|\mathcal{E}| \times 2|\mathcal{V}|}$$



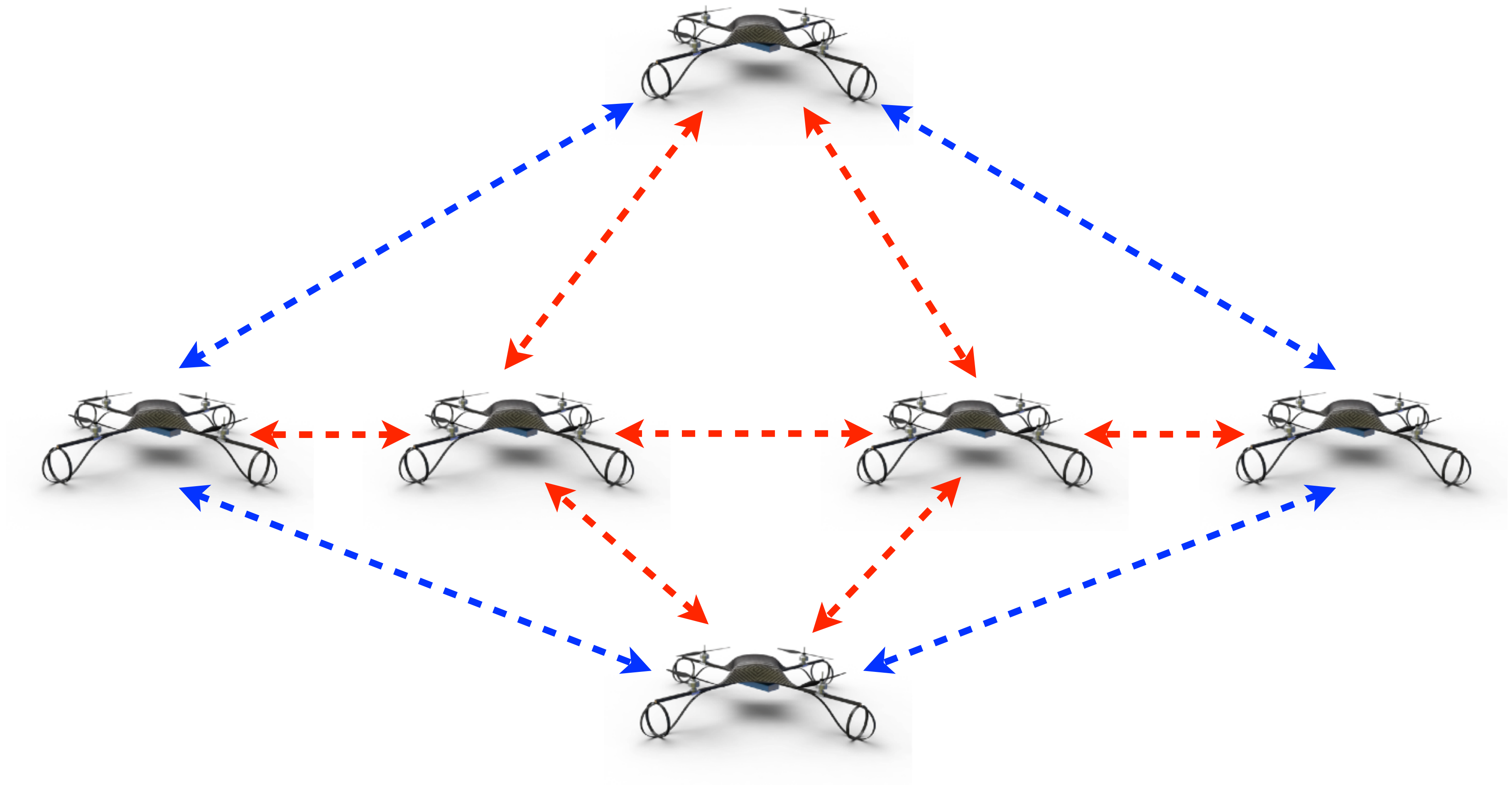
$$p(v_i) = (p_i^x, p_i^y)$$

$$(\xi(v_i) - \xi(v_j))^T (p(v_i) - p(v_j)) = 0$$

$$R(p) = \begin{bmatrix} p_1^x - p_2^x & p_1^y - p_2^y & p_2^x - p_1^x & p_2^y - p_1^y & 0 & 0 \\ p_1^x - p_3^x & p_1^y - p_3^y & 0 & 0 & p_3^x - p_1^x & p_3^y - p_1^y \\ 0 & 0 & p_2^x - p_3^x & p_2^y - p_3^y & p_3^x - p_2^x & p_3^y - p_2^y \end{bmatrix}$$

Lemma 1 (Tay1984) *A framework $(\mathcal{G}, p)$ is infinitesimally rigid if and only if* $\mathbf{rk}[R] = 2|\mathcal{V}| - 3$

Rigidity and Formations

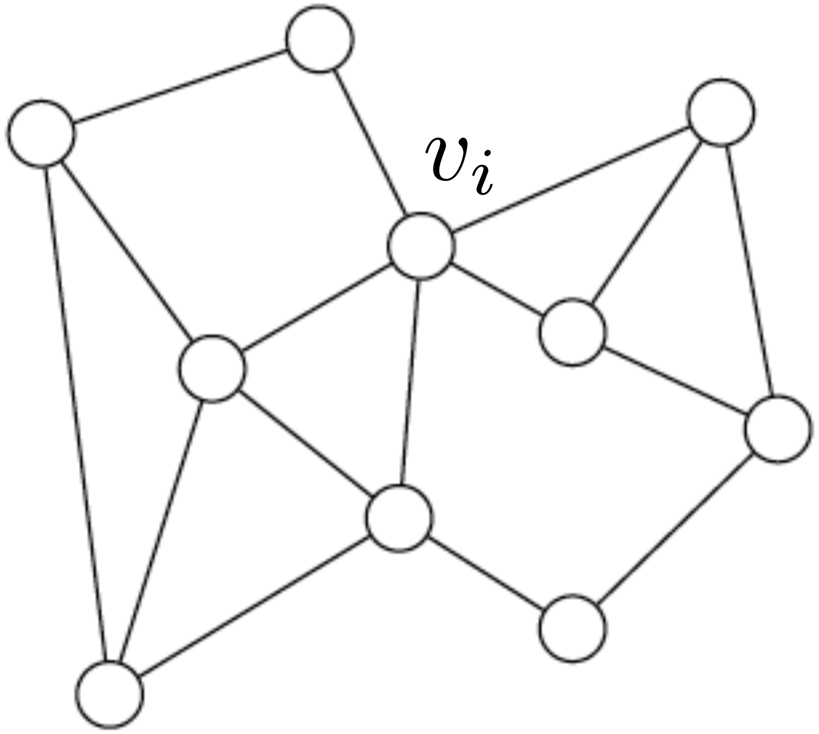


$$R(p) \in \mathbb{R}^{|\mathcal{E}| \times 2|\mathcal{V}|}$$

The Rigidity Matrix

$$R(p) \in \mathbb{R}^{|\mathcal{E}| \times 2|\mathcal{V}|}$$

the "local" graph from the perspective of a single agent



$$E(\mathcal{G}_{v_i})$$

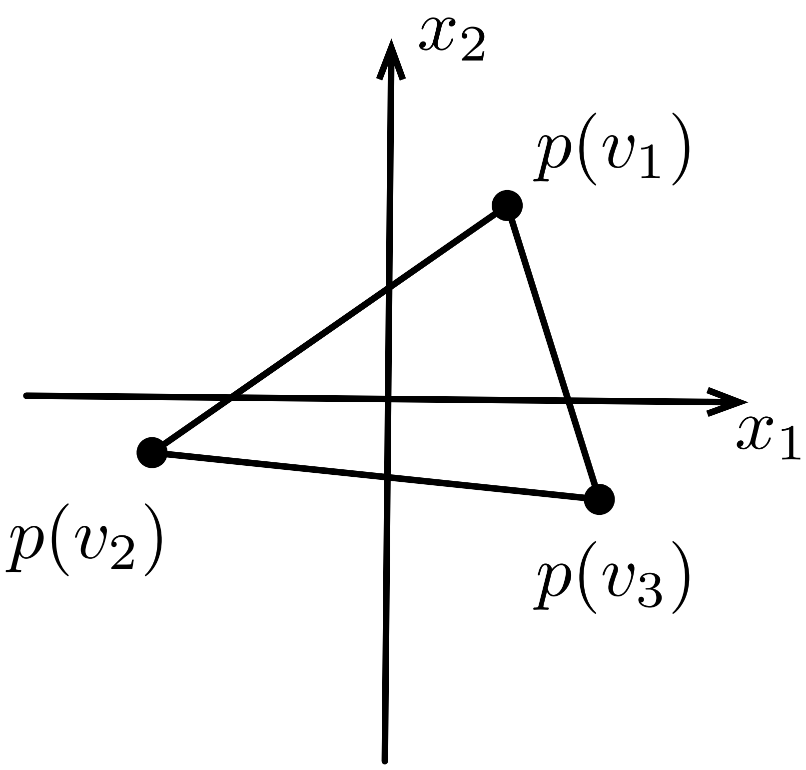
local incidence matrix

$\mathcal{G}$ $\mathcal{G}_{v_i}$

The Rigidity Matrix

The Rigidity Matrix

$$R(p) \in \mathbb{R}^{|\mathcal{E}| \times 2|\mathcal{V}|}$$



$p(v_i) = (p_i^x, p_i^y)$

'local' incidence matrices

$$E(\mathcal{G}_1) = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \quad E(\mathcal{G}_2) = \begin{bmatrix} -1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & -1 \end{bmatrix} \quad E(\mathcal{G}_3) = \begin{bmatrix} 0 & 0 & -1 \\ 0 & -1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

Proposition 1 (Zelazo *et al.* '12) *The rigidity matrix can be defined as*

$$R(p) = \begin{bmatrix} E(\mathcal{G}_1) & \ldots & E(\mathcal{G}_{|\mathcal{V}|}) \end{bmatrix} (I_{|\mathcal{V}|} \otimes p^{(x,y)})$$

The Rigidity Eigenvalue

The Symmetric Rigidity Matrix

$$\mathcal{R} = R(p)^T R(p)$$

λ_4 the *Rigidity Eigenvalue*

a symmetric positive semi-definite matrix with eigenvalues

$$\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{2|\mathcal{V}|}$$

Theorem 1 (Zelazo *et al.* '12) *A framework is infinitesimally rigid if and only if the* rigidity eigenvalue *is strictly positive; i.e.,* $\lambda_4 > 0$.

proof: $$P\mathcal{R}P^T = (I_2 \otimes E(\mathcal{G})) \begin{bmatrix} W_x & W_{xy} \\ W_{xy} & W_y \end{bmatrix} (I_2 \otimes E(\mathcal{G})^T)$$

use properties of incidence matrix to show first three eigenvalues must be at the origin

The Rigidity Eigenvalue

The Symmetric Rigidity Matrix

$$\mathcal{R} = R(p)^T R(p)$$

...as a weighted graph Laplacian Matrix

$$P\mathcal{R}P^T = (I_2 \otimes E(\mathcal{G})) \begin{bmatrix} W_x & W_{xy} \\ W_{xy} & W_y \end{bmatrix} \left(I_2 \otimes E(\mathcal{G})^T\right)$$

Weights are a function of
relative positions

$$W_x = (p_i^x - p_j^x)^2 \quad W_y = \left(p_i^y - p_j^y\right)^2 \quad W_{xy} = \left(p_i^x - p_j^x\right)\left(p_i^y - p_j^y\right)$$

Outline

- ✧ Motivation
- ✧ Graph Rigidity and the Rigidity Eigenvalue
- ✧ Dynamic Rigidity Maintenance
- ✧ Optimally Rigid Formations
- ✧ Outlook

Rigidity and Formation control

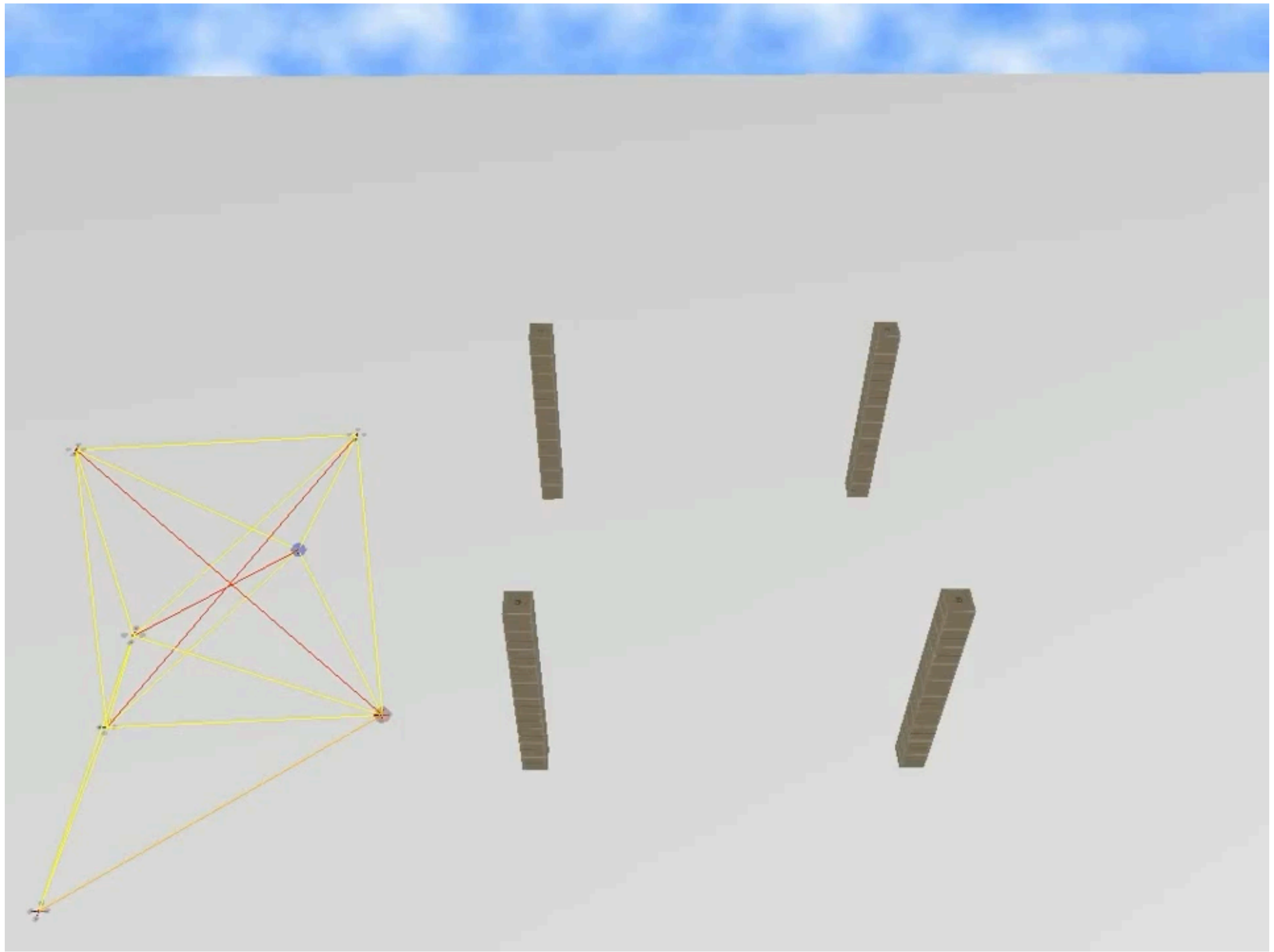
Why is this important or useful?

- formation control
- localization
- exploration

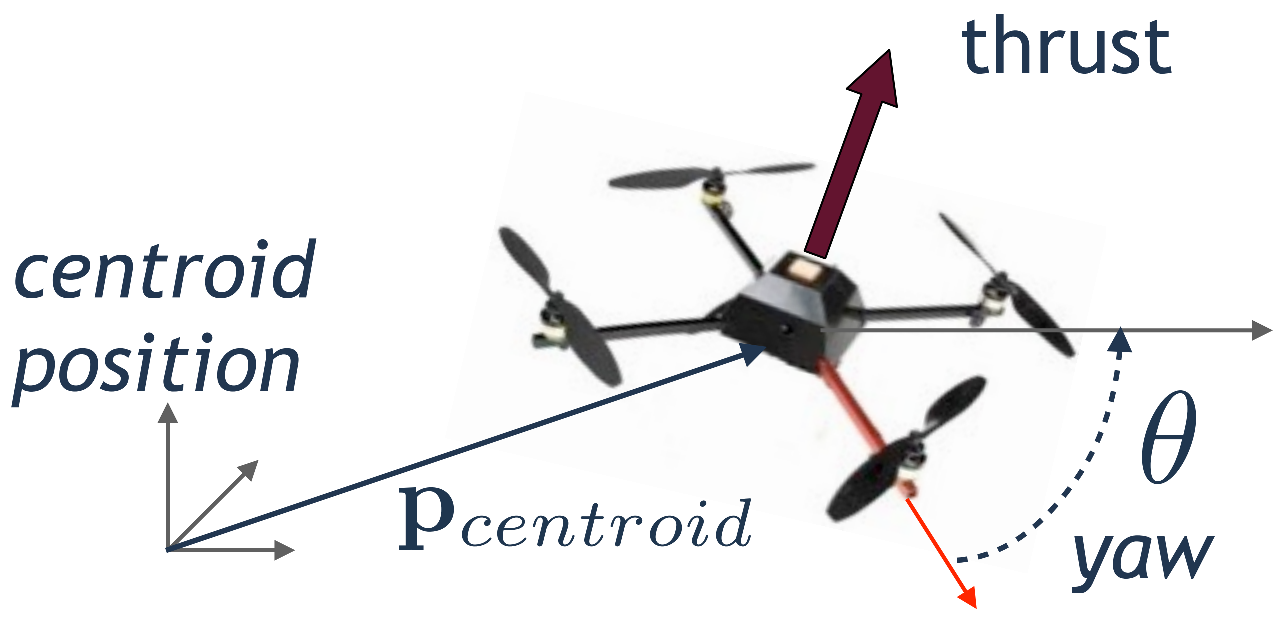
Is it possible to maintain rigidity in a distributed manner?

- the rigidity eigenvalue is the tool

Agents should move to ensure the rigidity eigenvalue is always positive!



Control of a Quadrotor UAV



$$J_i w_i + S(w_i) J_i w_i = \gamma_i + \zeta_i$$

fully-actuated rotational dynamics

$$m_i \ddot{x}_i = -\lambda_i R_i e_3 + m_i g e_3 + \delta_i$$

under-actuated translational dynamics

The position of the **center of mass** and the **yaw** are **flat outputs** [Mistler & al. ISRHIC 2001]

Any **smooth trajectory in the flat outputs** space can be followed by the quadrotor (with a suitable controller)

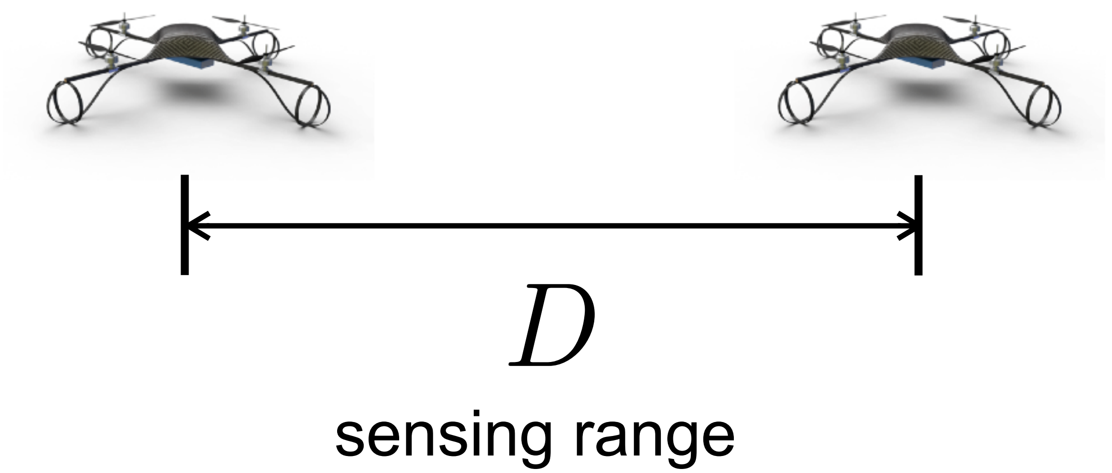
$\Rightarrow$ The UAV is abstracted as a **point oriented in the horizontal plane**

Design a *velocity command* for each quadrotor using only sensed information from neighbors and obstacles

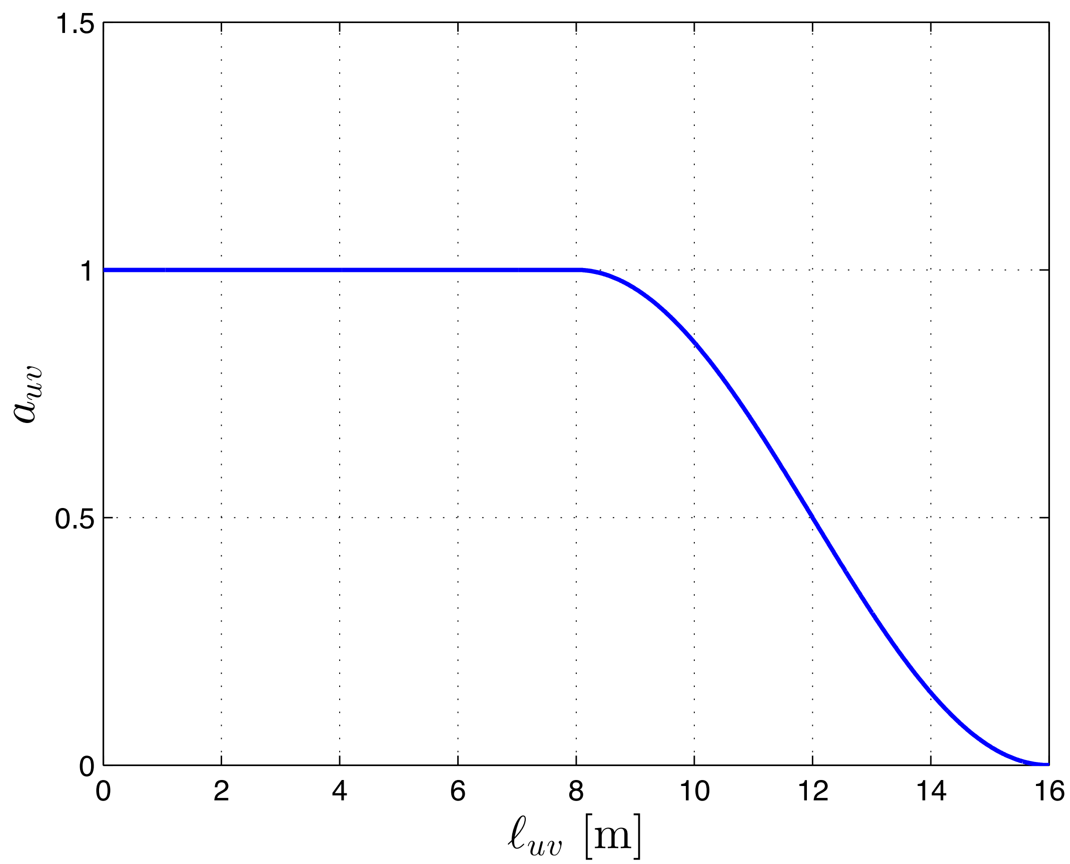
$$\xi_i$$

Quadrotor Sensing Constraints

When is there a sensing link between agents?



"Weights" can be introduced on sensing link between agents to promote or discourage behaviors

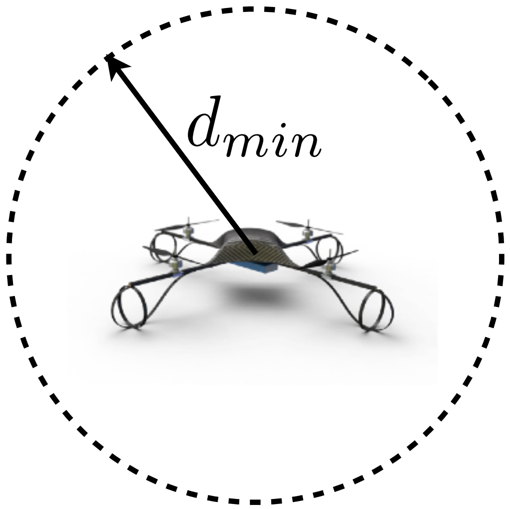


$$a_{ij}(d_{ij}) = \begin{cases} k_a & 0 \le d_{ij} \le d_0 \\ \frac{k_a}{2}(1 + \cos(\alpha_a d_{ij} + \beta_a)) & d_0 < d_{ij} \le D \\ 0 & d_{ij} > D \end{cases}$$

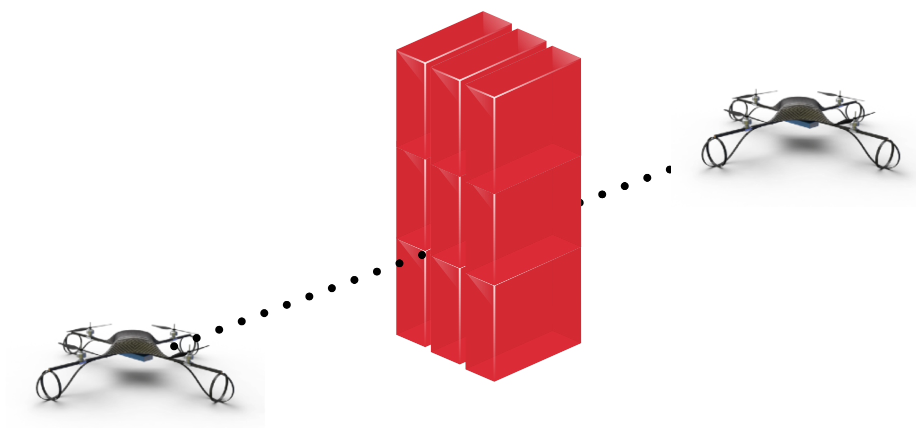
Quadrotor Sensing Constraints

When is there a sensing link between agents?

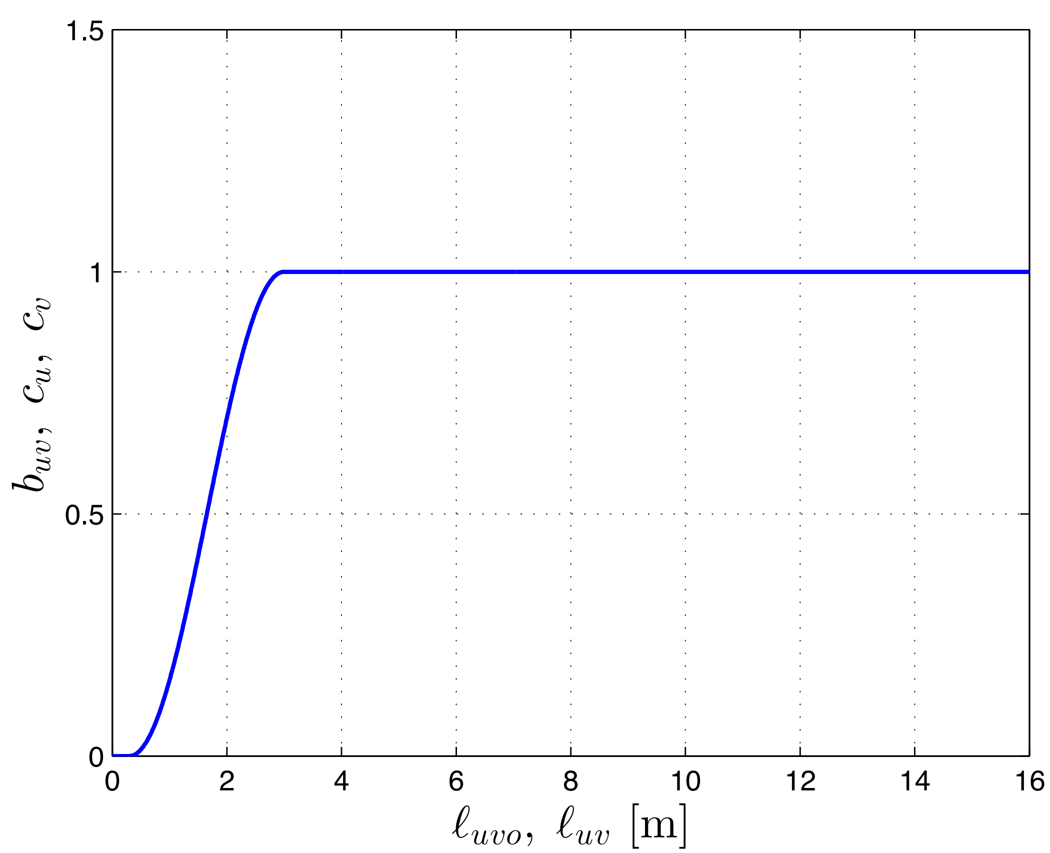
“Weights” can be introduced on sensing link between agents to promote or discourage behaviors



safety zone



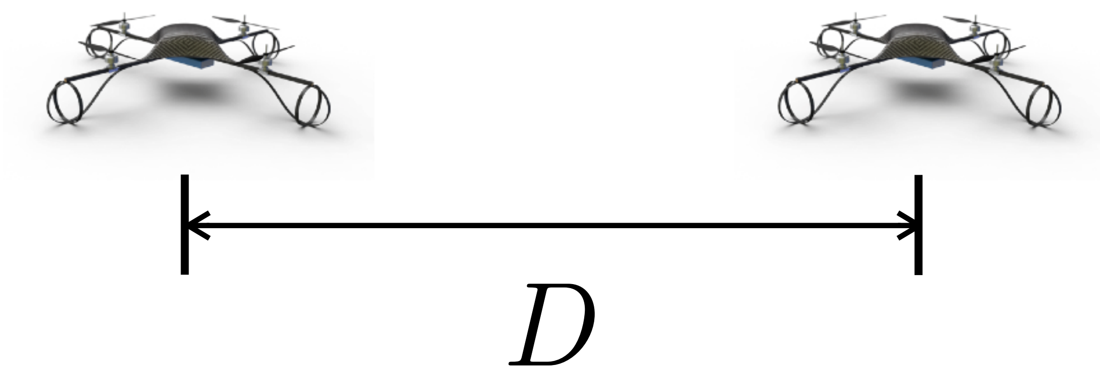
no line-of-sight occlusion



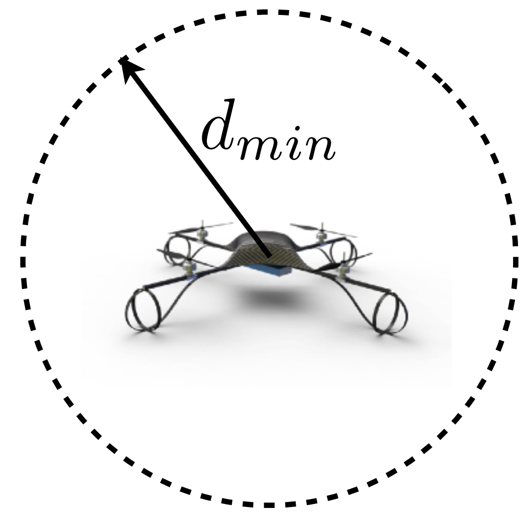
$$b_{ij}(d_{ijo}) = \begin{cases} 0 & d_{oij} \leq d^o_{\min} \\ \dfrac{k_b}{2}(1 - \cos(\alpha_b d_{ijo} + \beta_b)) & d^o_{\min} < d_{ijo} \leq d^o_{\max} \\ k_b & d_{ijo} > d^o_{\max} \end{cases}$$

Quadrotor Sensing Constraints

When is there a sensing link between agents?

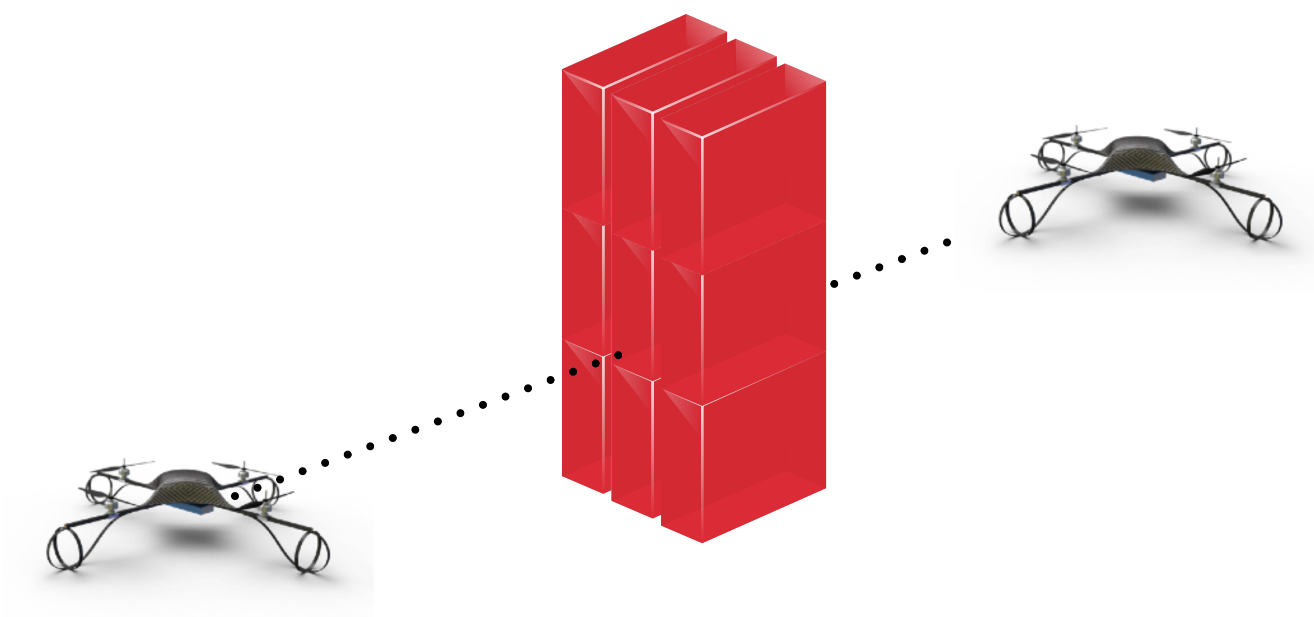


sensing range

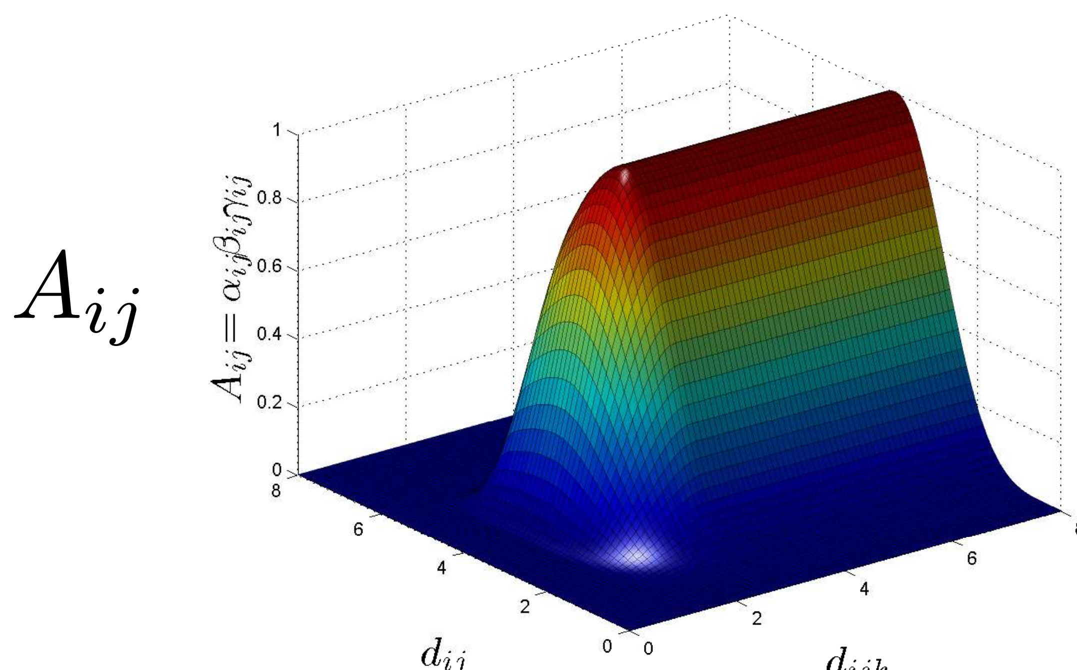


safety zone

composite weight between neighboring agents

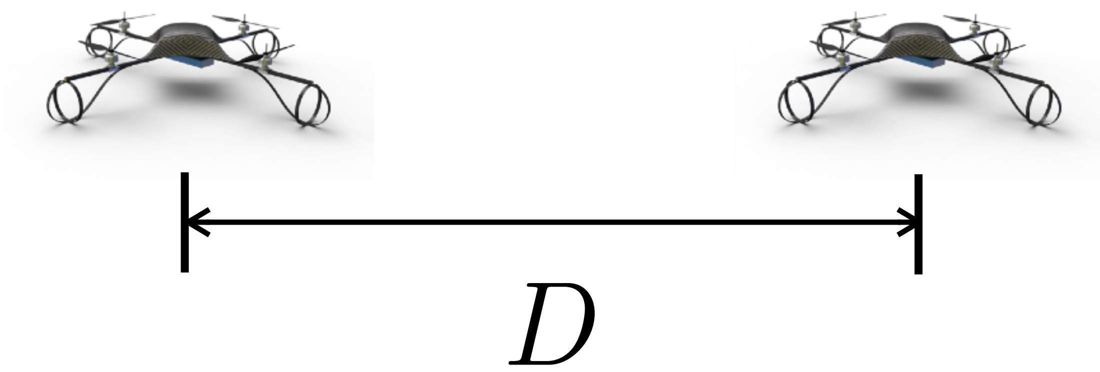


no line-of-sight occlusion

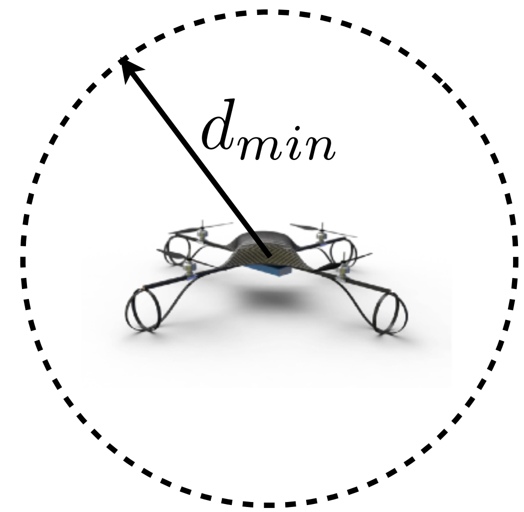


Quadrotor Sensing Constraints

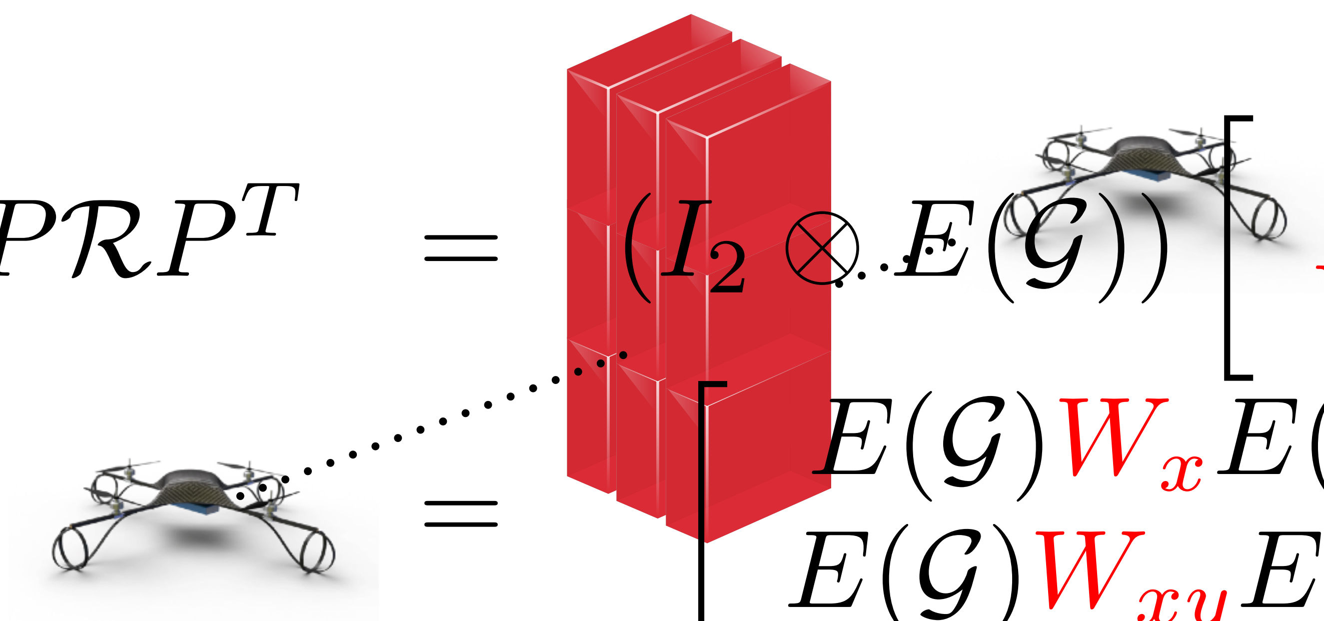
When is there a sensing link between agents?



sensing range



safety zone



no line-of-sight occlusion

$$P\mathcal{R}P^T = (I_2 \otimes E(\mathcal{G})) \begin{bmatrix} W_x & W_{xy} \\ W_{xy} & W_y \end{bmatrix} (I_2 \otimes E(\mathcal{G})^T)$$

$$= \begin{bmatrix} E(\mathcal{G})W_x E(\mathcal{G})^T & E(\mathcal{G})W_{xy} E(\mathcal{G})^T \\ E(\mathcal{G})W_{xy} E(\mathcal{G})^T & E(\mathcal{G})W_y E(\mathcal{G})^T \end{bmatrix}$$

The Rigidity Potential

How can rigidity be maintained with only local information?

Key observation: Gradient of rigidity eigenvalue has a distributed structure!

$$\lambda_4 = v_4^T P \mathcal{R} P^T v_4$$

$$\frac{\partial \lambda_4}{\partial p_i^x} = 2\left(\sum_{i\sim j}(p_i^x - p_j^x)(v_i^x - v_j^x)^2 + (p_i^y - p_j^y)(v_i^x - v_j^x)(v_i^y - v_j^y)\right)$$

$$\frac{\partial \lambda_4}{\partial p_i^y} = 2\left(\sum_{i\sim j}(p_i^y - p_j^y)(v_i^y - v_j^y)^2 + (p_i^x - p_j^x)(v_i^x - v_j^x)(v_i^y - v_j^y)\right)$$

gradient is only a function of *relative* quantities!

can be computed locally by each agent*

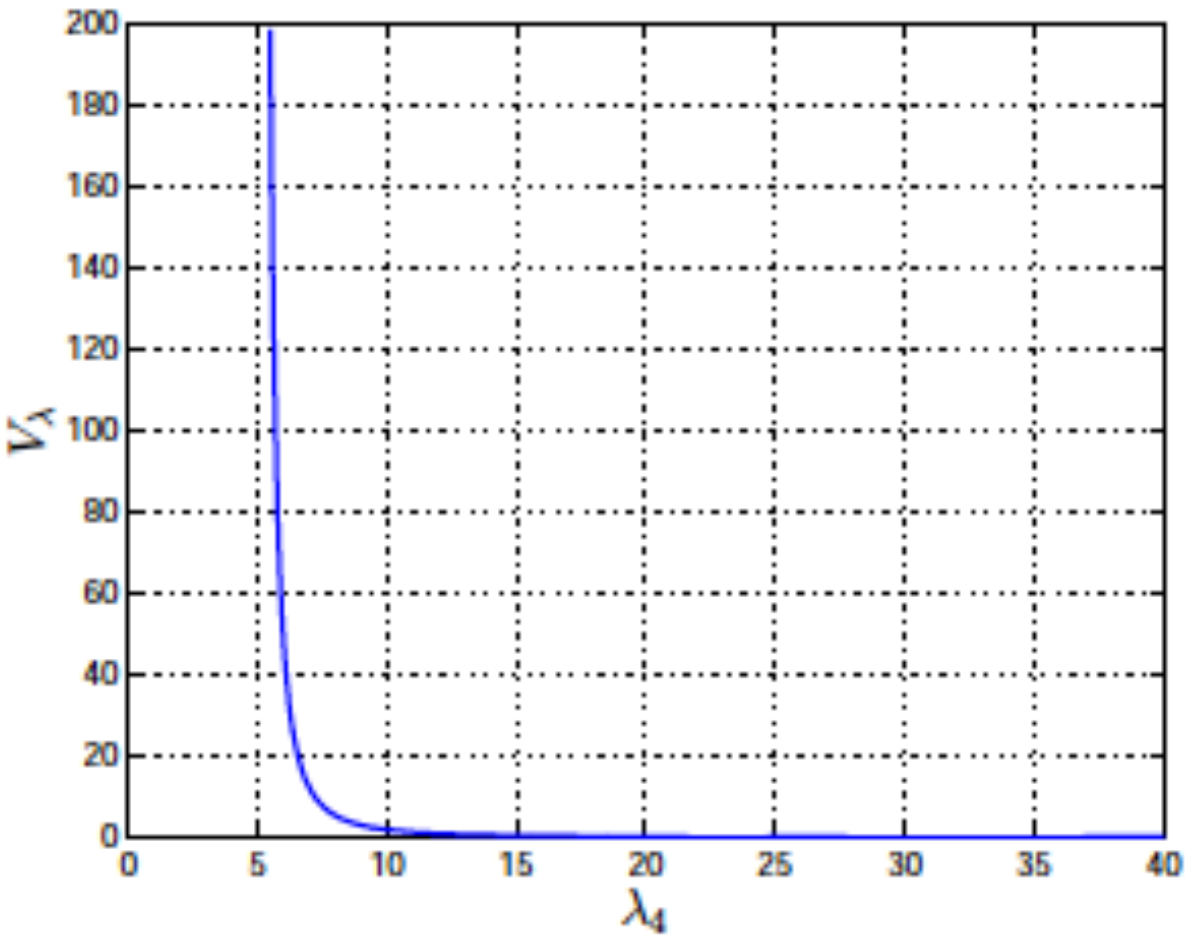
The Rigidity Potential

$$\lambda_4 = v_4^T P \mathcal{R} P^T v_4$$

Define a scalar potential function V_λ

grows unbounded as $\lambda_4 \to 0$

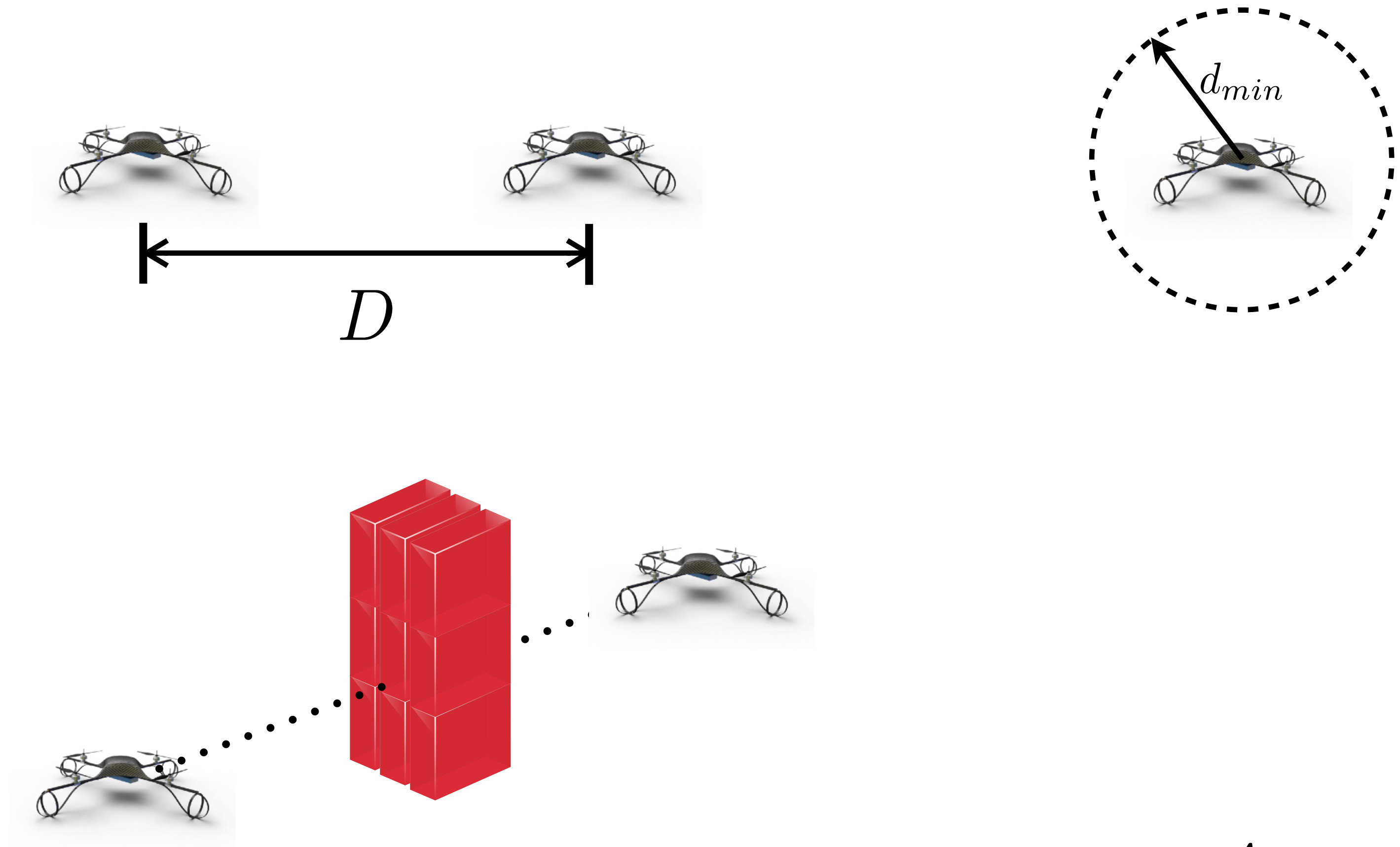
vanishes as $\lambda_4 \to \infty$



velocity command

$$\xi_i = -\frac{\partial V_\lambda}{\partial \lambda_4}\left(\frac{\partial \lambda_4}{\partial p_i}\right)$$

The Rigidity Potential



velocity command

$$\xi_i = -\frac{\partial V_\lambda}{\partial \lambda_4}\left(\frac{\partial \lambda_4}{\partial p_i}\right)$$

A_{ij}

$$\begin{aligned}\lambda_4 \quad = \quad & \Big(\sum_{i\sim j}(p_i^x - p_j^x)^2 A_{ij}(v_i^x - v_j^x)^2\Big) + \\ & \Big(\sum_{i\sim j}(p_i^y - p_j^y)^2 A_{ij}(v_i^y - v_j^y)^2\Big) + \\ & \Big(2\sum_{i\sim j}(p_i^x - p_j^x)(p_i^y - p_j^y)A_{ij}(v_i^x - v_j^x)(v_i^y - v_j^y)\Big)\end{aligned}$$

Weighted Rigidity Eigenvalue

A Note on “how” Distributed

$$\frac{\partial \lambda_4}{\partial p_i^x} = 2\left(\sum_{i\sim j}(p_i^x - p_j^x)(v_i^x - v_j^x)^2 + (p_i^y - p_j^y)(v_i^x - v_j^x)(v_i^y - v_j^y)\right)$$

$$\frac{\partial \lambda_4}{\partial p_i^y} = 2\left(\sum_{i\sim j}(p_i^y - p_j^y)(v_i^y - v_j^y)^2 + (p_i^x - p_j^x)(v_i^x - v_j^x)(v_i^y - v_j^y)\right)$$

Observation: The gradient requires that neighboring agents exchange their component of the *rigidity eigenvector*!

Problem: The rigidity eigenvector is a *global* quantity!

Solution: This control strategy requires a *distributed estimation* of the rigidity eigenvector and eigenvalue for implementation!

Idea: Use consensus filters to implement a distributed version of the *Power Iteration* method for eigenvector estimation
(Yang ’10, Robuffo Giordano ’11)

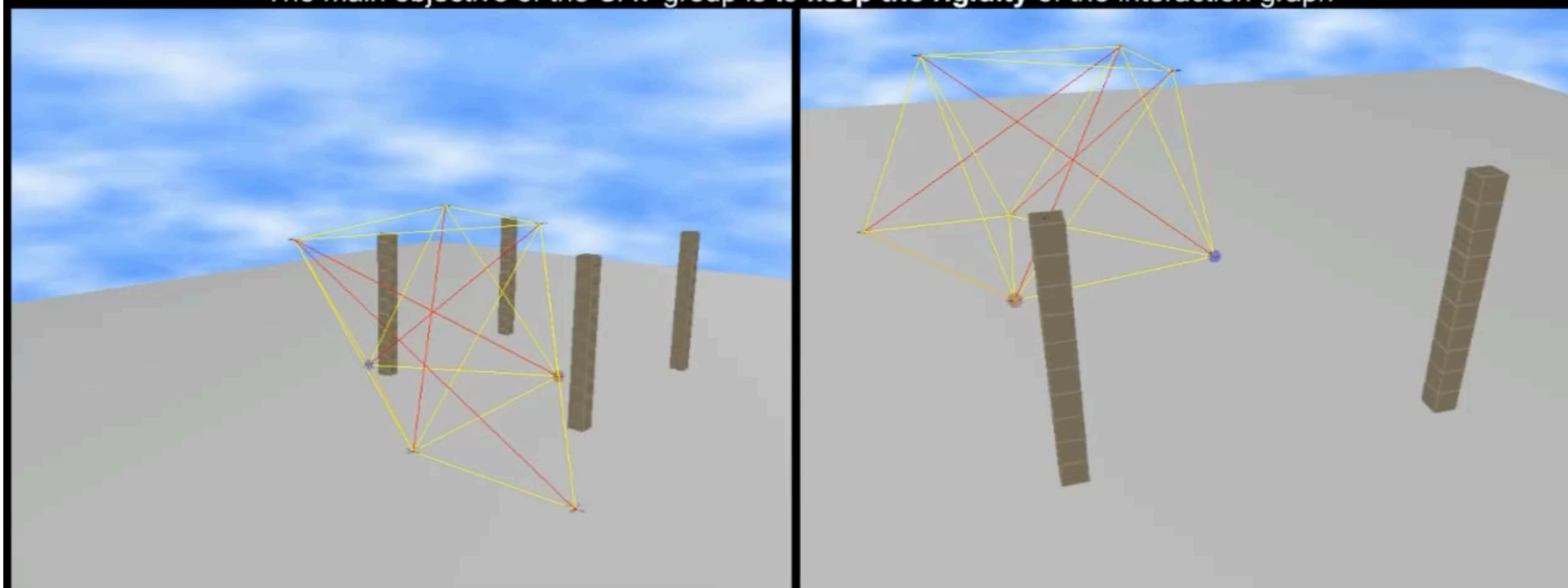
$$\dot{x}(t) = \left(-k_1 TT^T - k_2 \mathcal{R}\right) x(t) - k_3 \left(\frac{x(t)^T x(t)}{n} - 1\right) x(t).$$

Simulation

The 7 UAVs have limited range and line-of-sight communication/perception
resulting in an Interaction Graph
(*red link* = *almost* ***disconnected***)

Rigidity of the graph is a fundamental property in formation control and sensing
(e.g., in order to estimate the relative positions by only measuring distances)

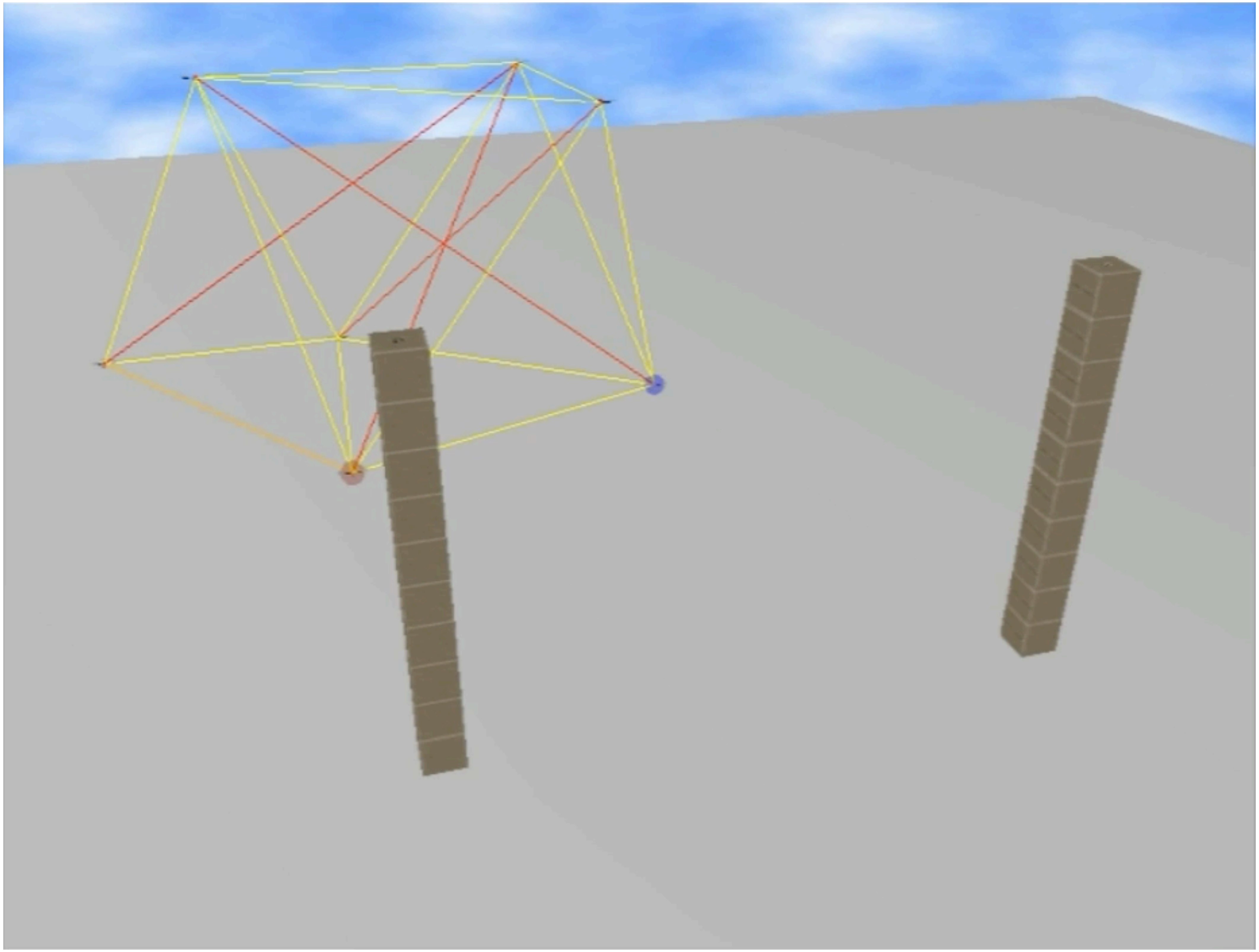
The main objective of the UAV group is to **keep the rigidity** of the interaction graph



Outline

- ✧ Motivation
- ✧ Graph Rigidity and the Rigidity Eigenvalue
- ✧ Dynamic Rigidity Maintenance
- ✧ Optimally Rigid Formations
- ✧ Outlook

Rigidity and formation control



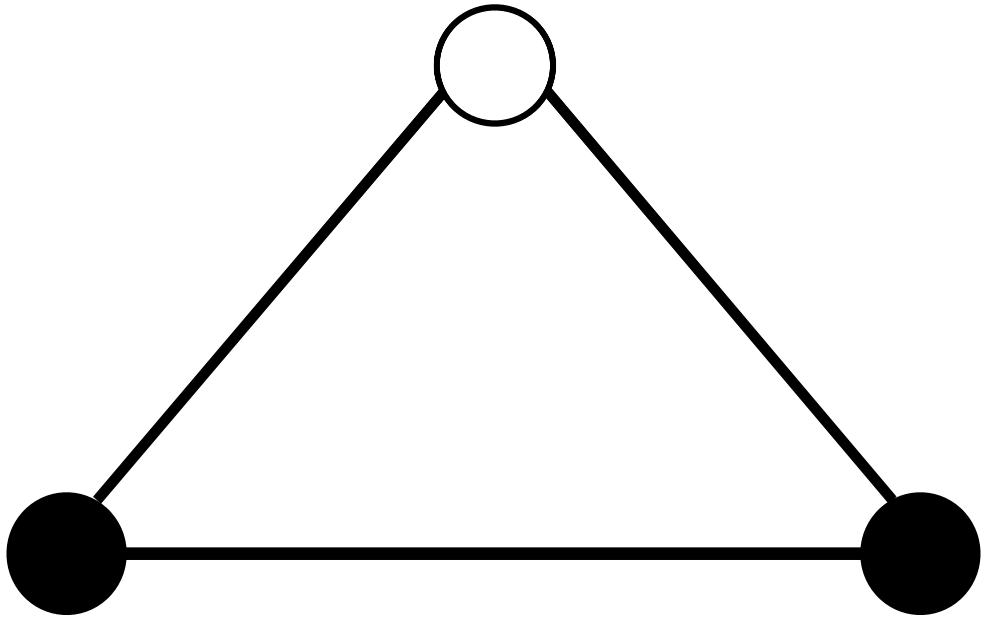
Henneberg Constructions

A constructive method for generating all minimally rigid graphs in the plane
[Henneberg, 1911]

Henneberg Constructions

A constructive method for generating all minimally rigid graphs in the plane
[Henneberg, 1911]

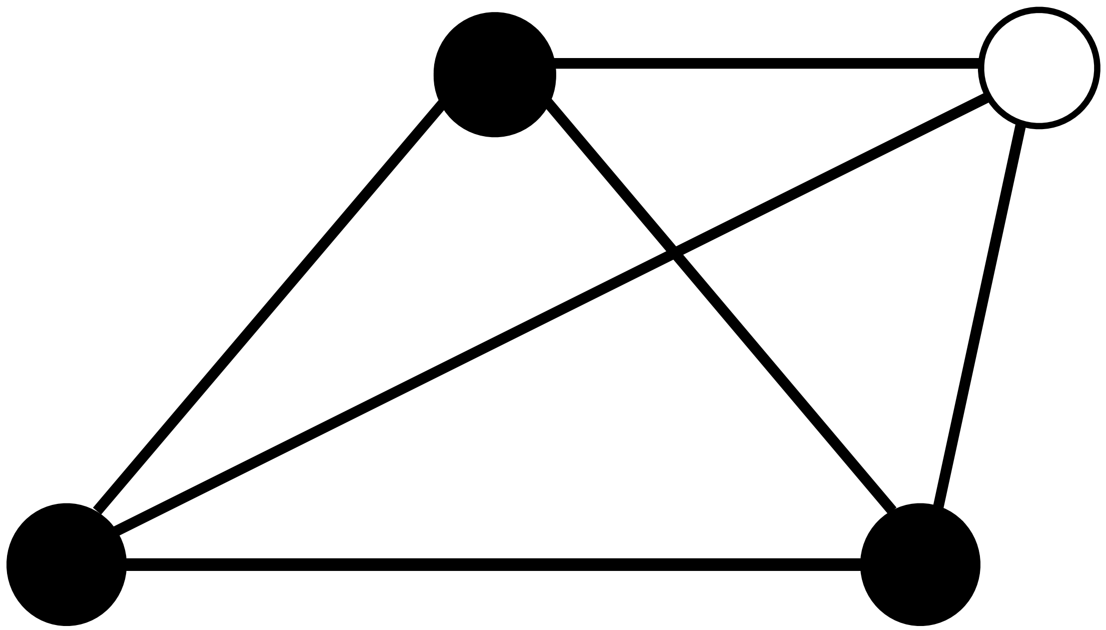
Vertex Addition



Henneberg Constructions

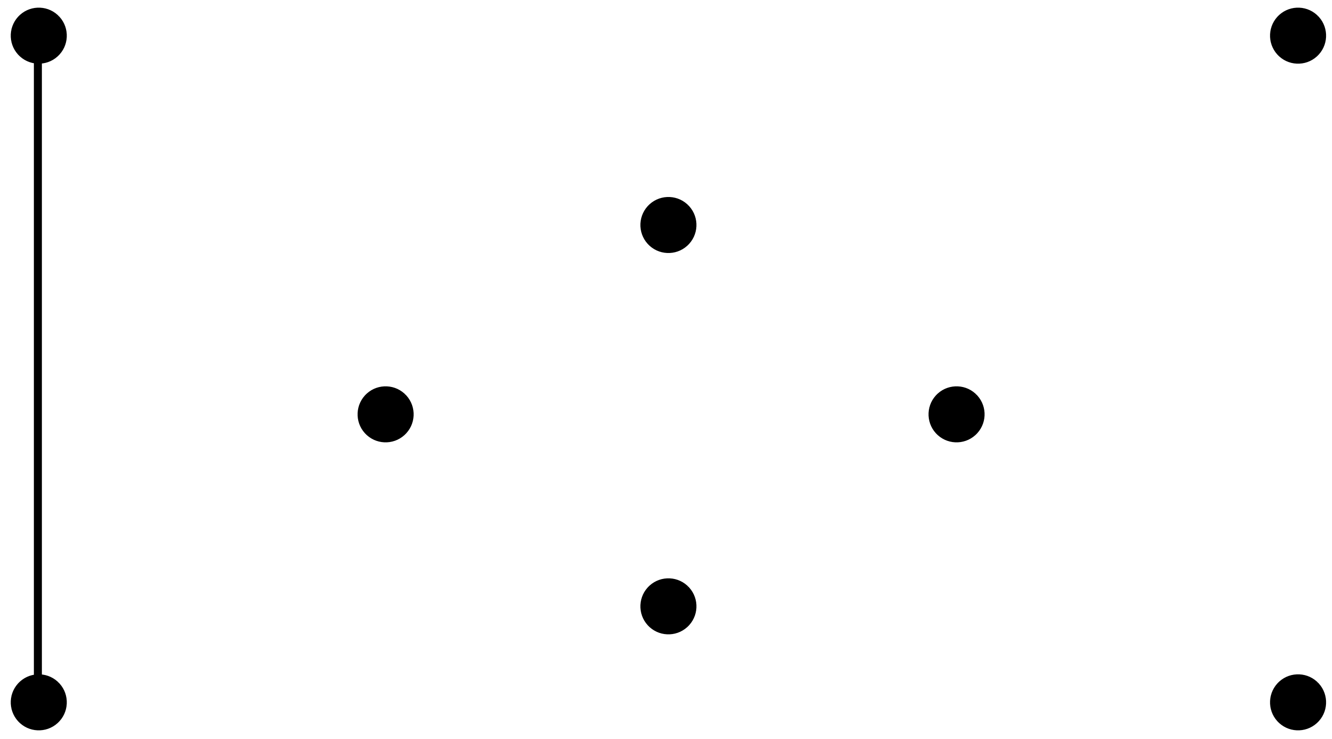
A constructive method for generating all minimally rigid graphs in the plane
[Henneberg, 1911]

Edge Splitting



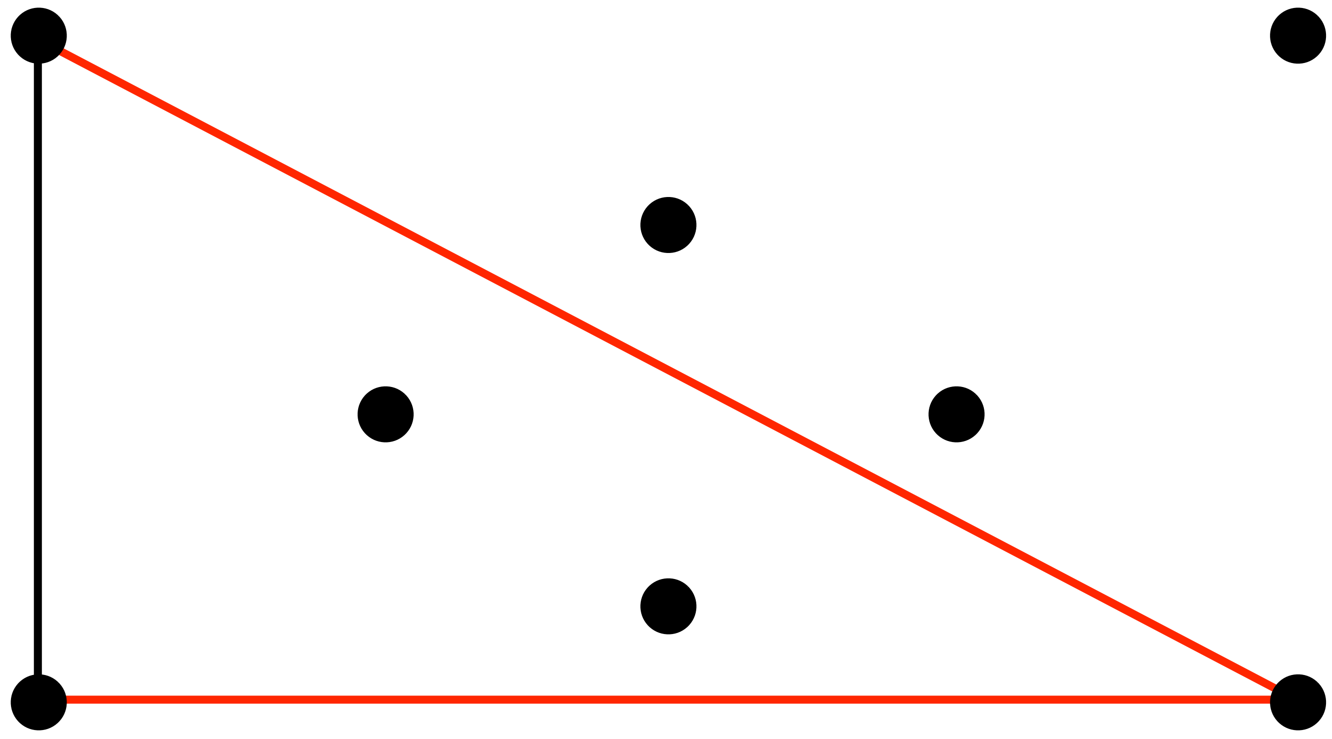
Henneberg Constructions

Example



Henneberg Constructions

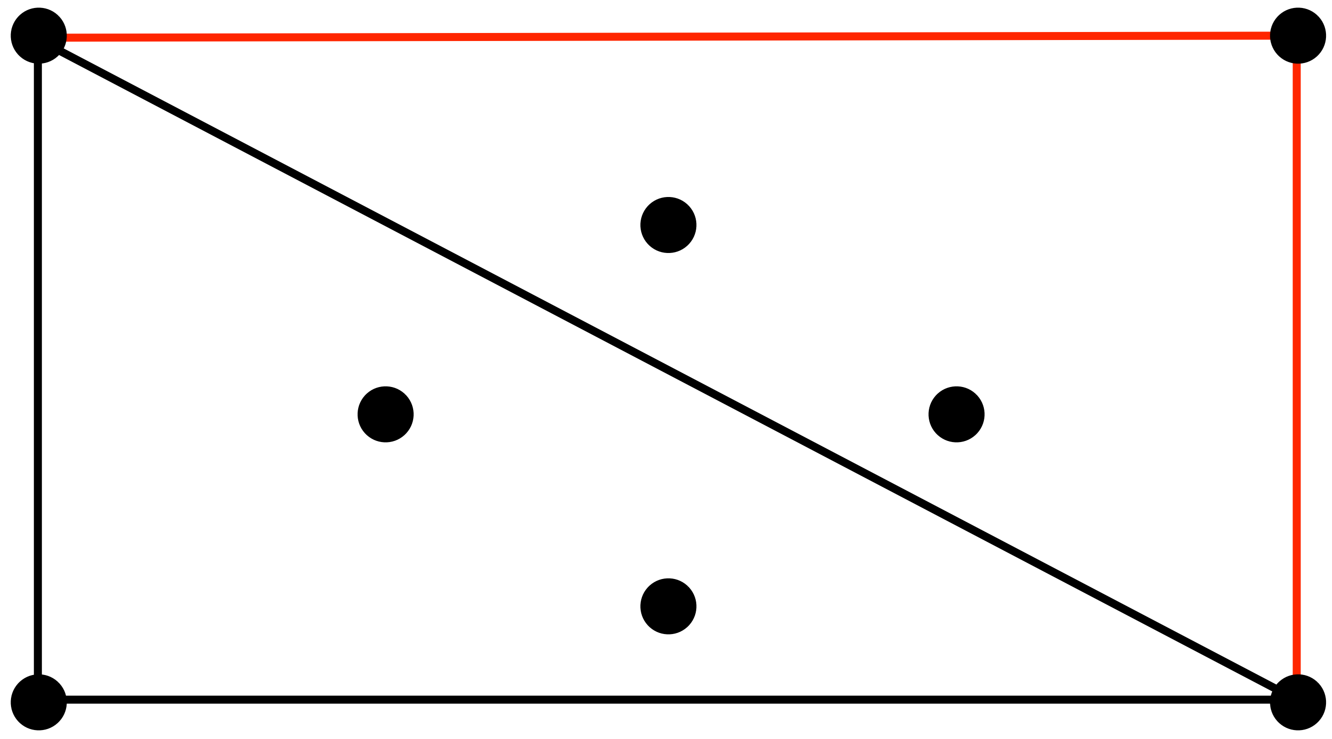
Example



Vertex Addition

Henneberg Constructions

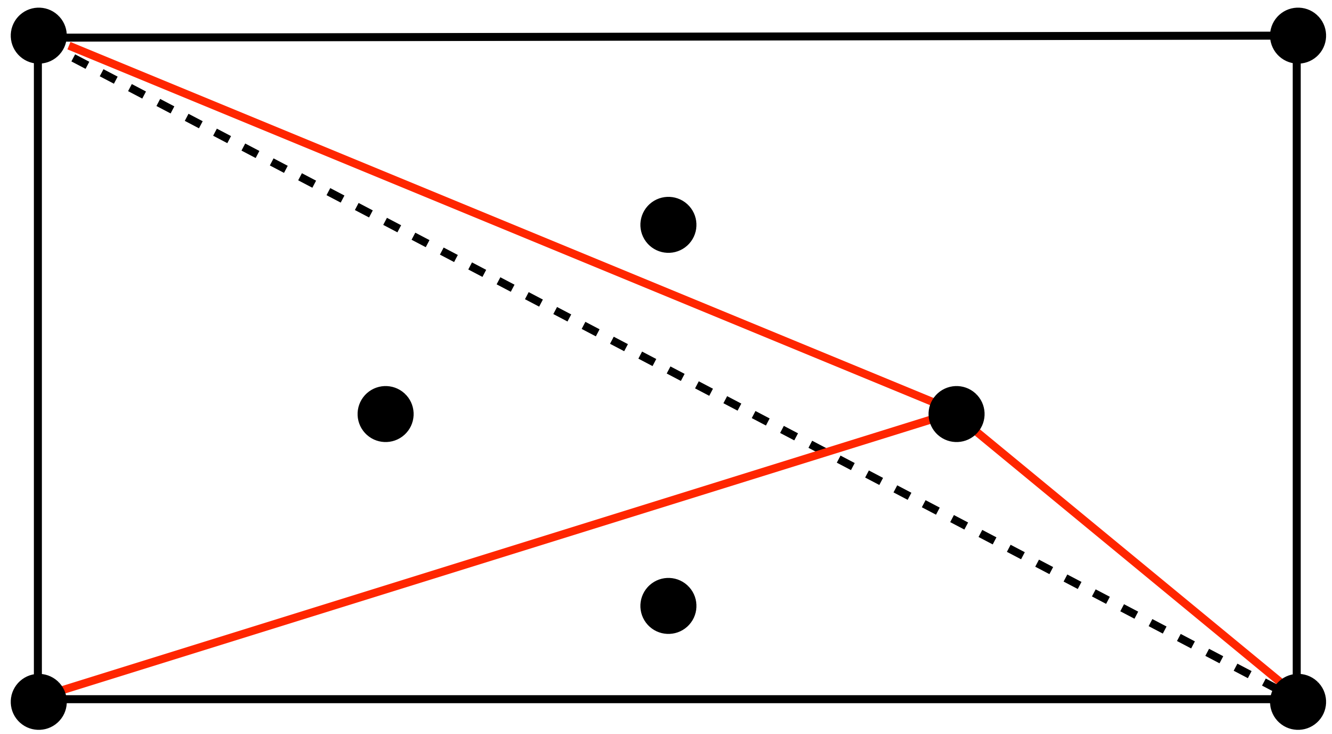
Example



Vertex Addition

Henneberg Constructions

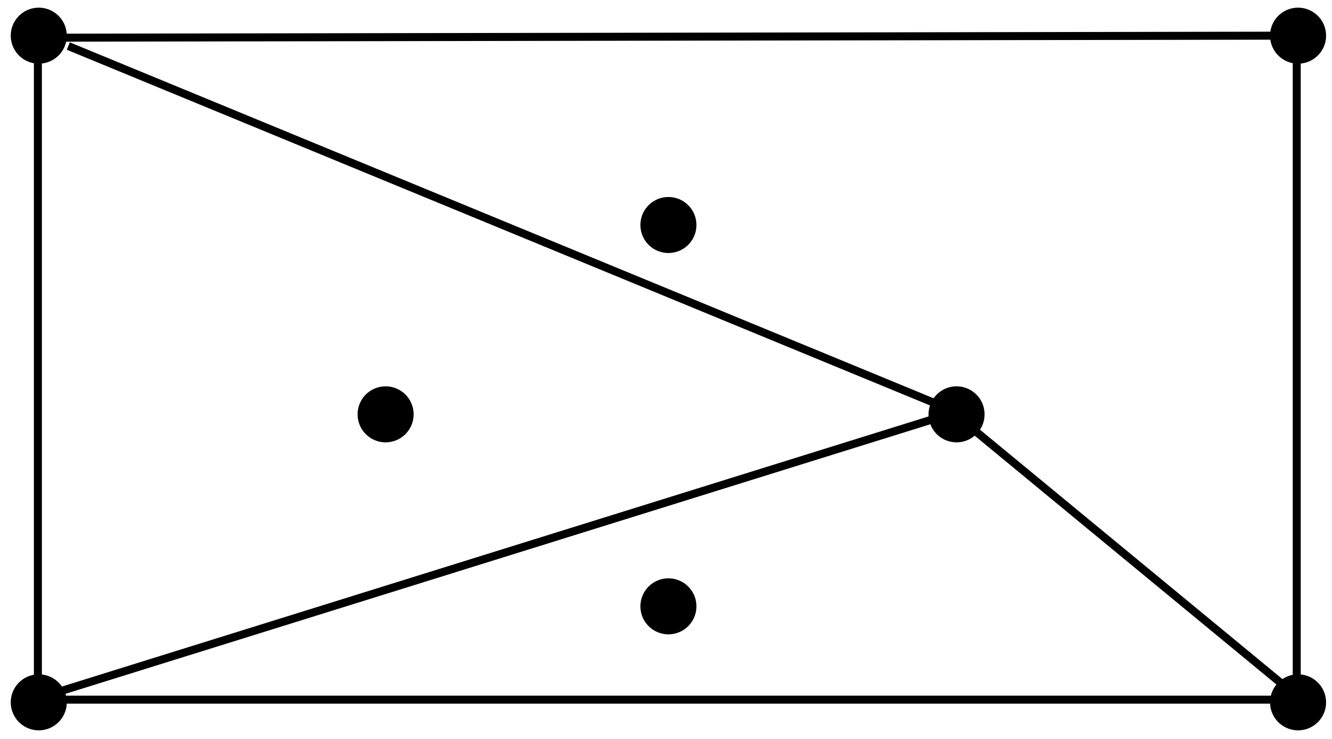
Example



Edge Splitting

Henneberg Constructions

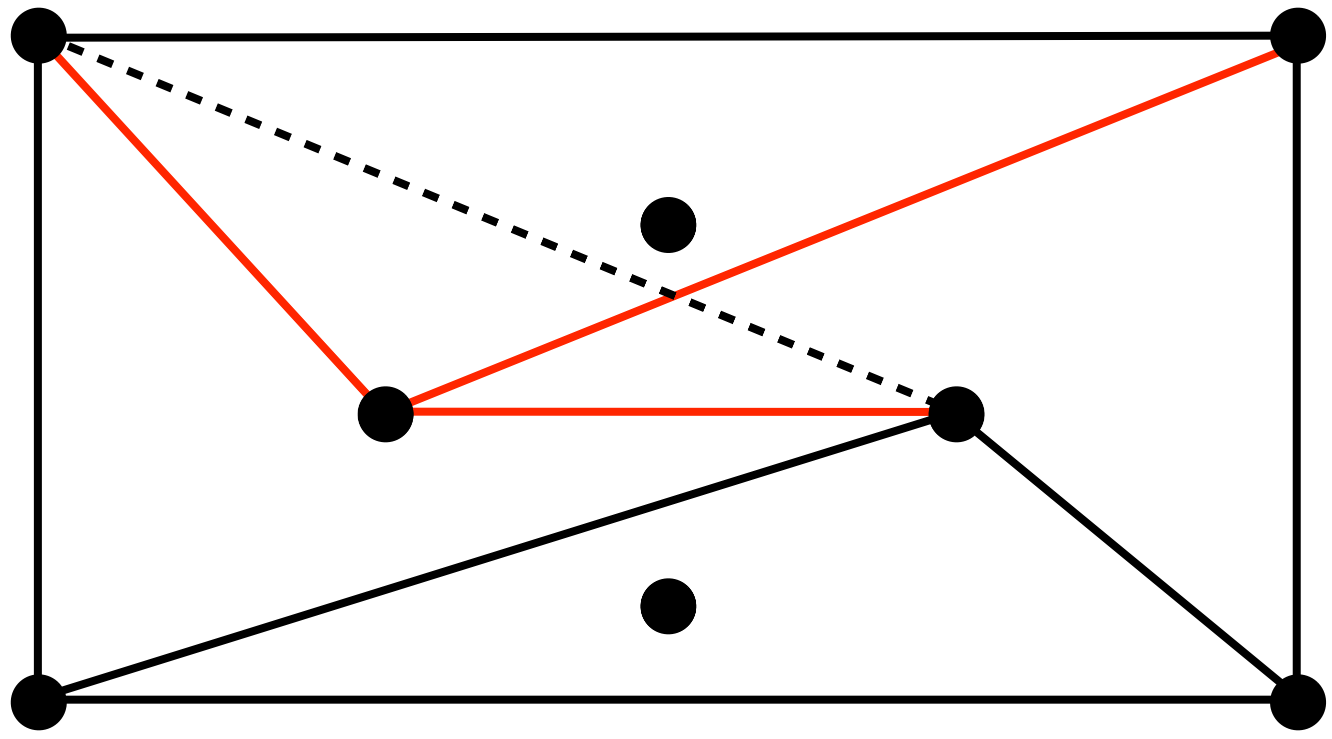
Example



Edge Splitting

Henneberg Constructions

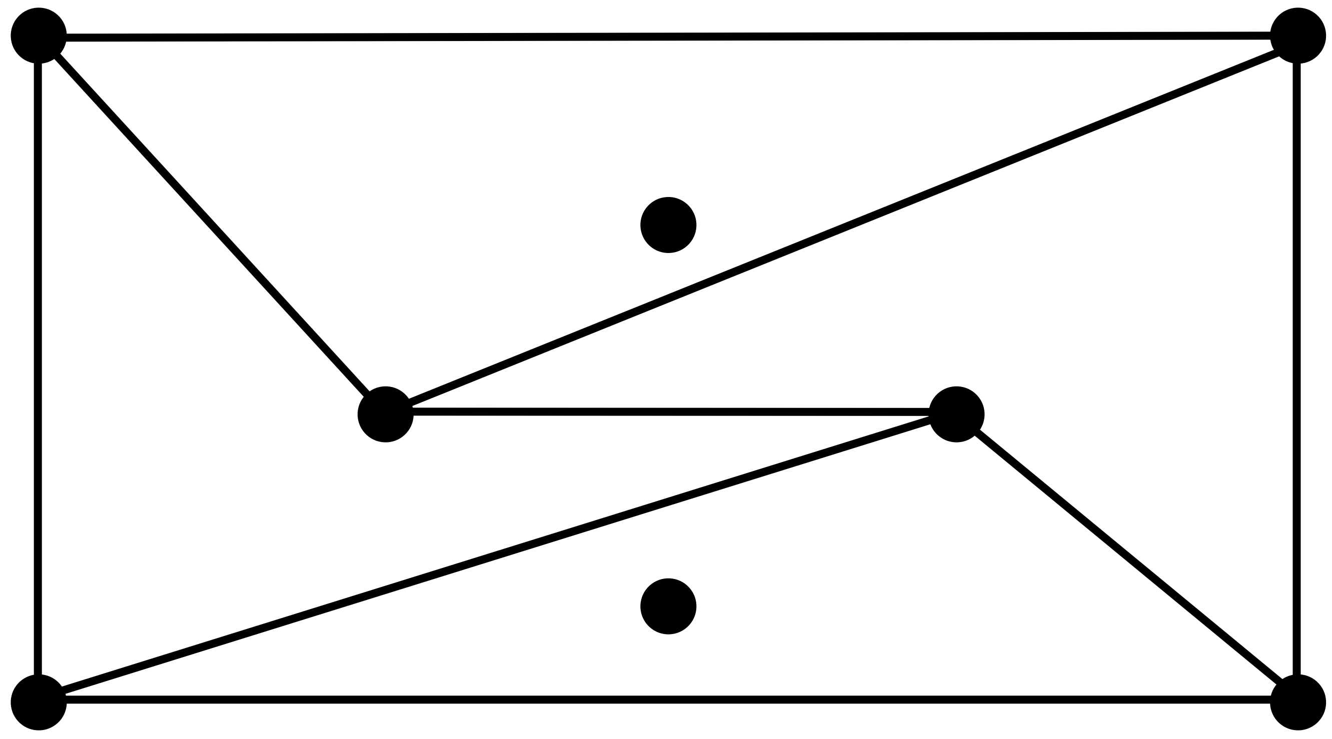
Example



Edge Splitting

Henneberg Constructions

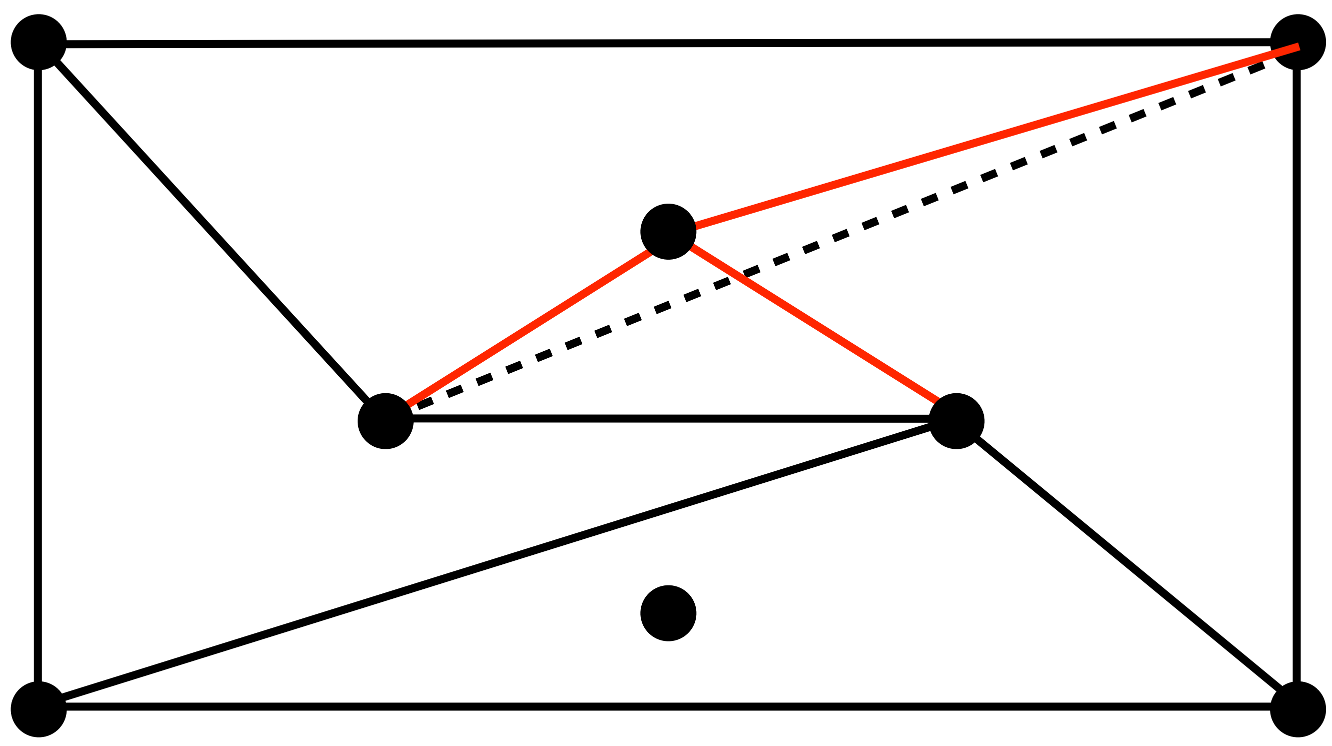
Example



Edge Splitting

Henneberg Constructions

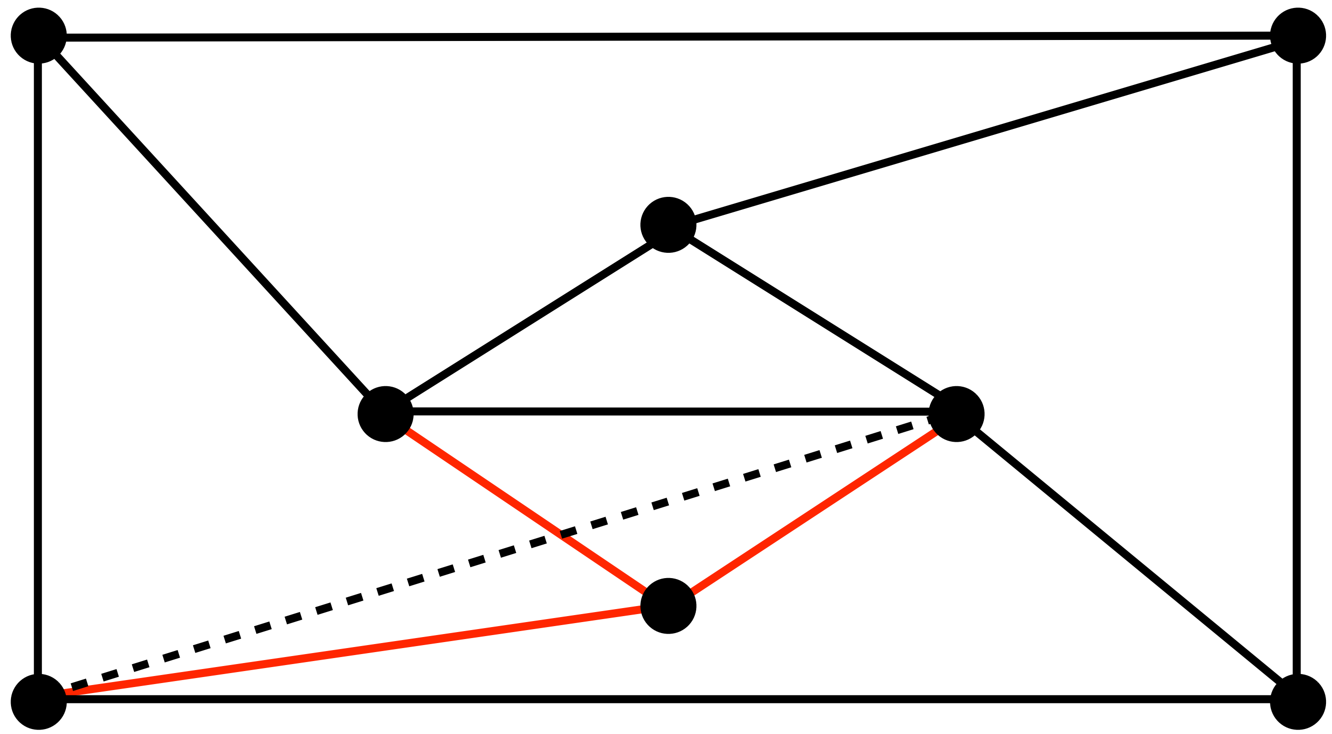
Example



Edge Splitting

Henneberg Constructions

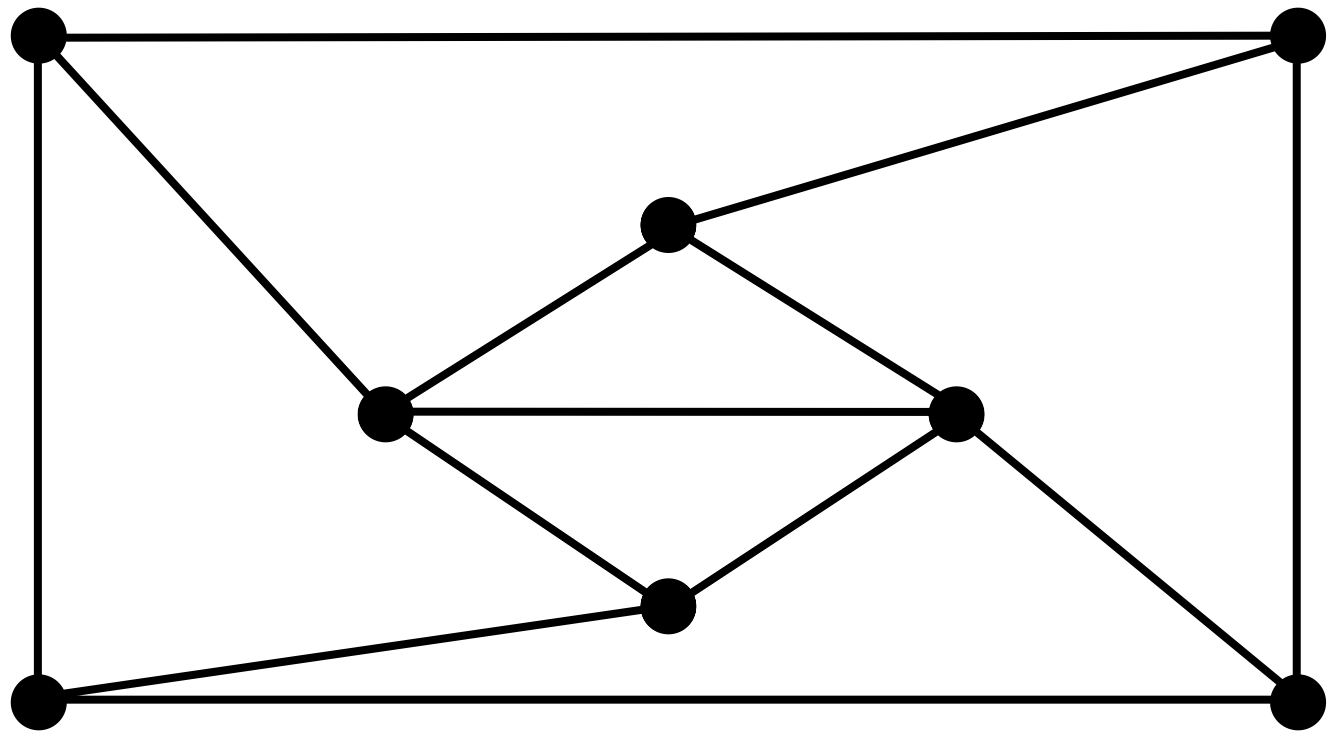
Example

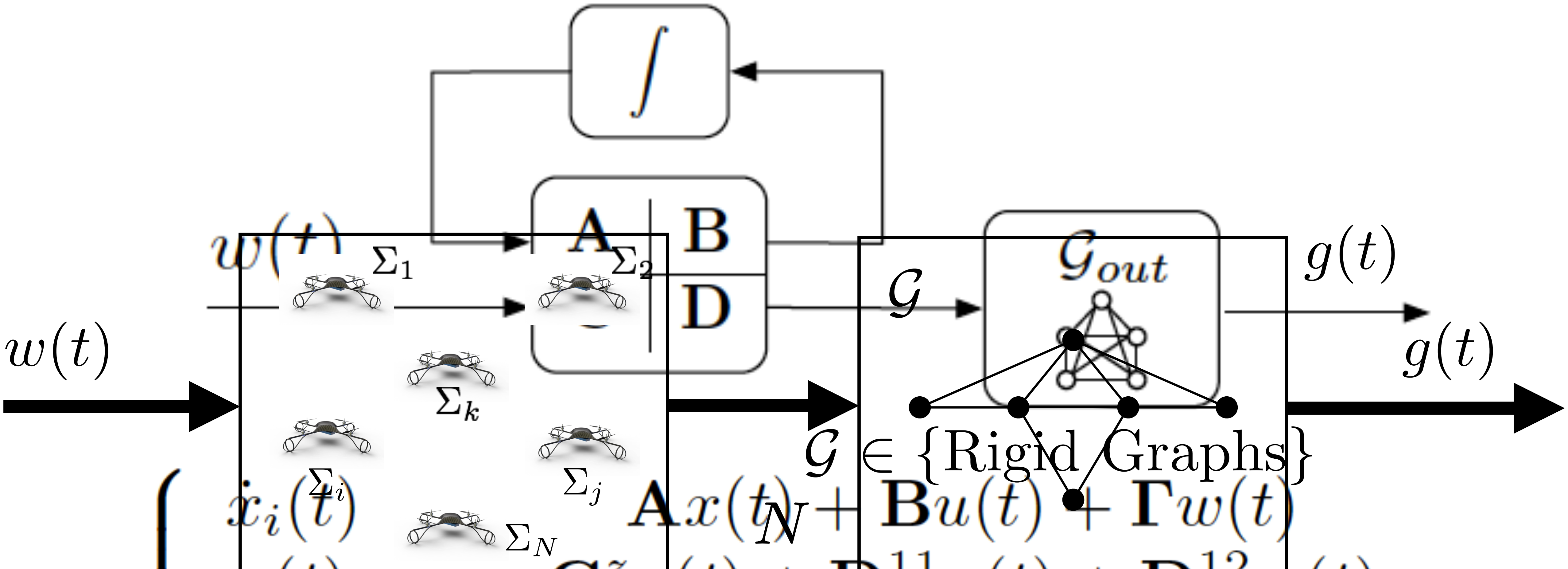


Edge Splitting

Henneberg Constructions

Example





$\mathcal{G} \in \{\text{Rigid Graphs}\}$

$$\Sigma(\mathcal{G}) \left\{ \begin{array}{rcl} \dot{x}_i(t) & = & \mathbf{A}x(t) + \mathbf{B}u(t) + \mathbf{\Gamma}w(t) \\ z(t) & = & \mathbf{C}^z x(t) + \mathbf{D}^{11}u(t) + \mathbf{D}^{12}w(t) \\ y(t) & = & \mathbf{C}^y x(t) + \mathbf{D}^{21}u(t) + \mathbf{D}^{22}w(t) \\ g(t) & = & (\mathcal{E}(\mathcal{G})^T \otimes C_x)x(t). \end{array} \right.$$

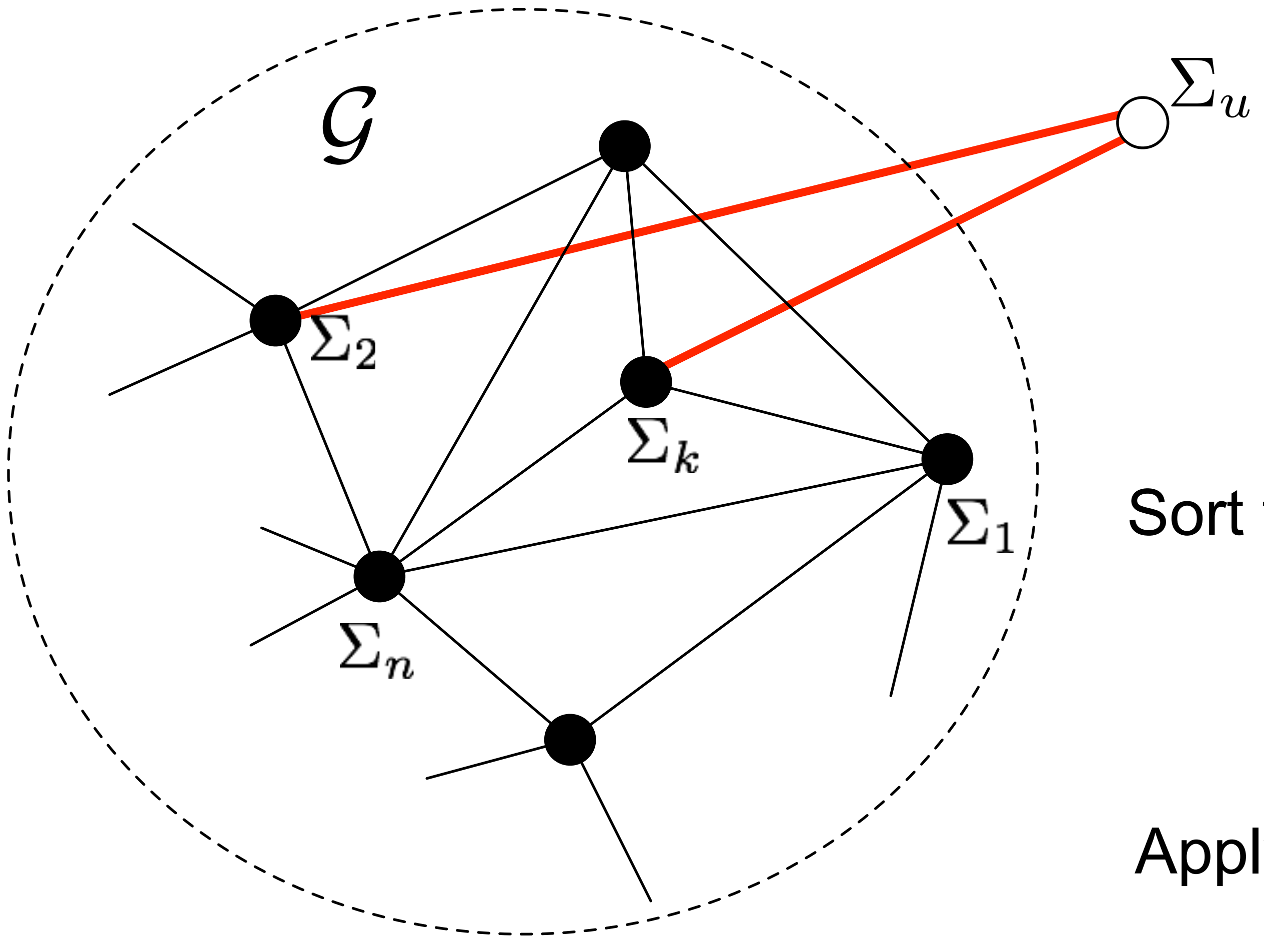
$$\|\Sigma(\mathcal{G})\|_2^2 = \sum_{i=1}^{N} d_i \|\Sigma_i\|_2^2$$

[Zelazo TAC-11]

Theorem : The $\mathcal{H}_2$ optimally rigid graph is minimally rigid.

Optimal Henneberg Construction

Proposition ($\mathcal{H}_2$ Optimal Vertex Addition)



Sort the degree-weighted norms of all nodes:

$$d_{\sigma(1)}\|\Sigma_{\sigma(1)}\|_2^2 \leq \cdots \leq d_{\sigma(N)}\|\Sigma_{\sigma(N)}\|_2^2$$

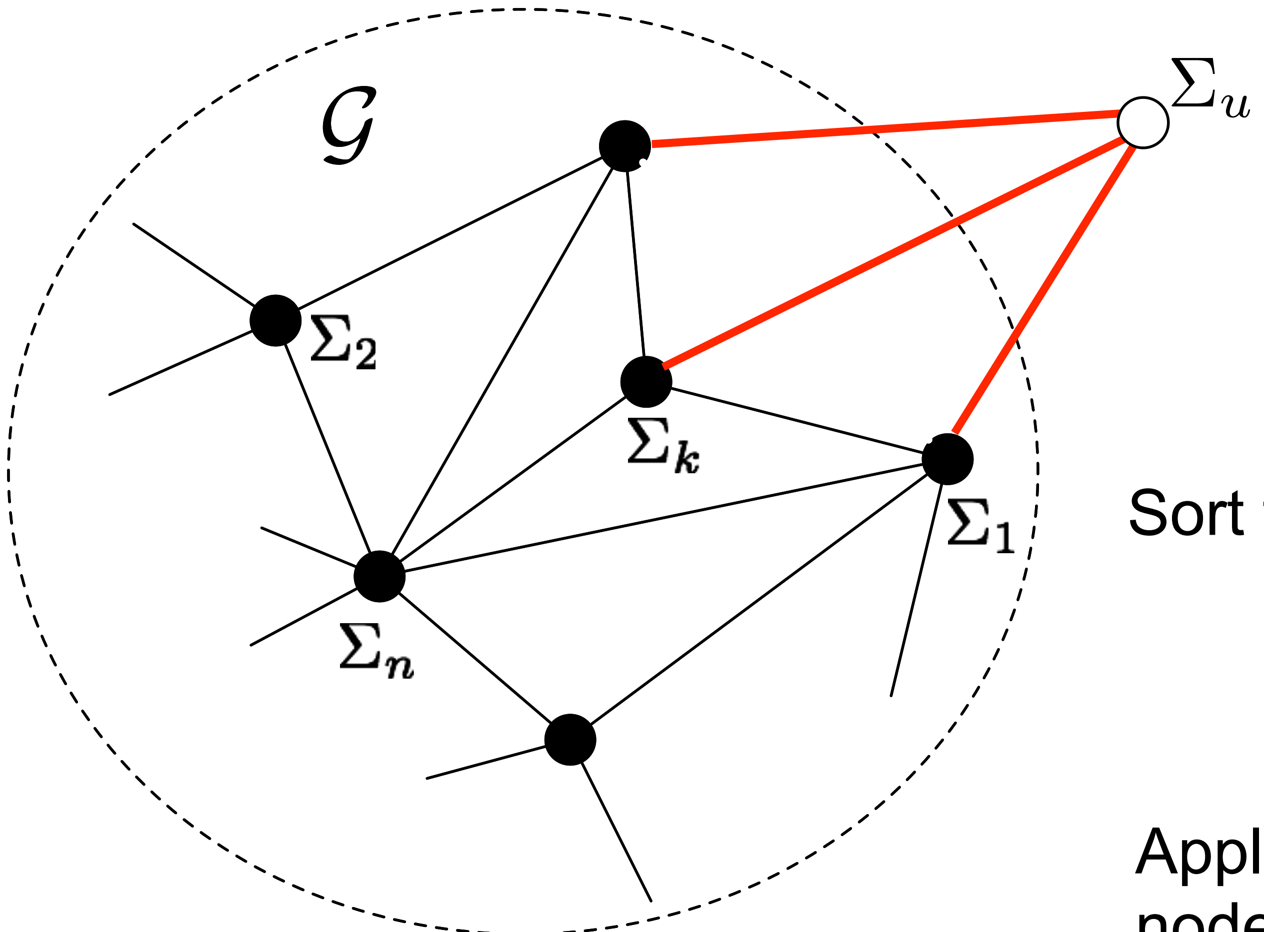
Apply *Vertex Addition* step to “smallest” weights

$$e_1 = (v_i, v_u) \;\; e_2 = (v_j, v_u)$$

$$\|\Sigma(\mathcal{G} \cup \{e_1, e_2\})\|_2^2 = \|\Sigma(\mathcal{G})\|_2^2 + 2\|\Sigma_u\|_2^2 + \|\Sigma_i\|_2^2 + \|\Sigma_j\|_2^2$$

Optimal Henneberg Construction

Proposition ($\mathcal{H}_2$ Optimal Edge Splitting)



Sort the degree-weighted norms of all nodes:

$$d_{\sigma(1)}\|\Sigma_{\sigma(1)}\|_2^2 \leq \cdots \leq d_{\sigma(N)}\|\Sigma_{\sigma(N)}\|_2^2$$

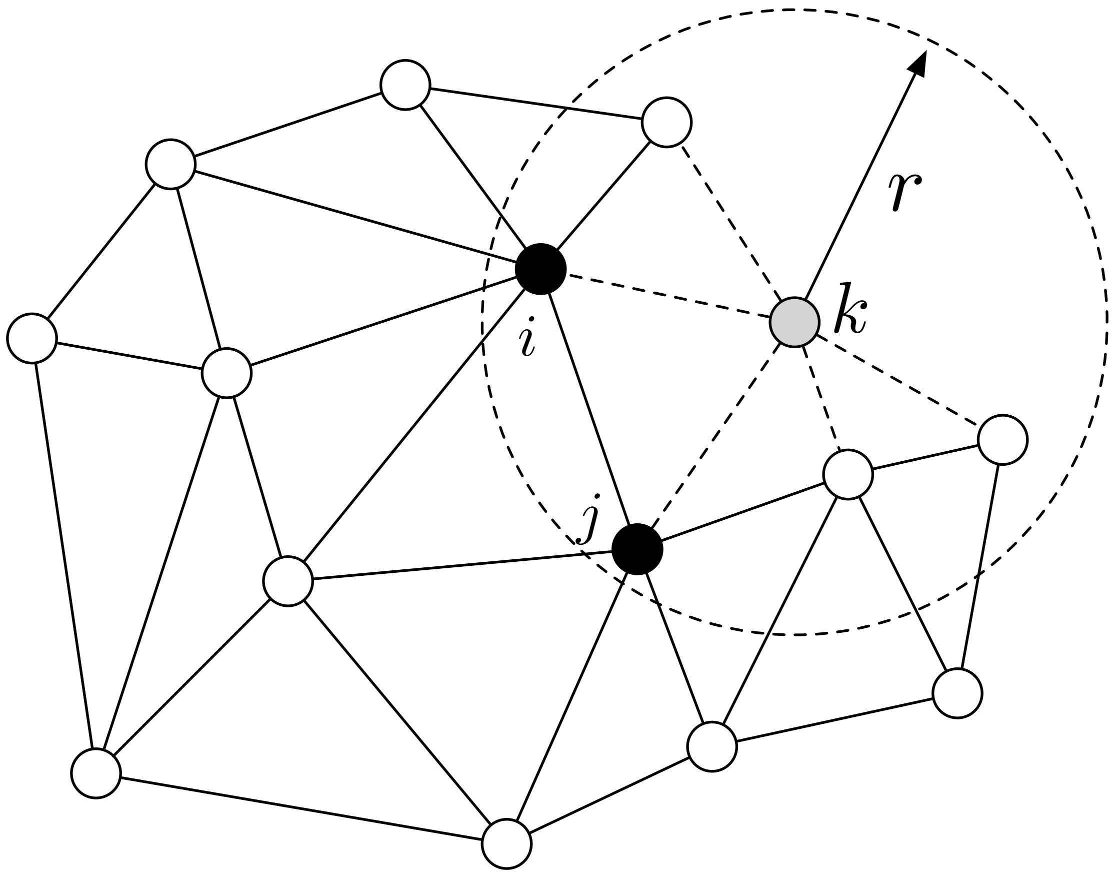
Apply *Edge Splitting* step with "smallest" weighted node and any other connected pair of nodes

$$e_1 = (v_i, v_u) \;\; e_2 = (v_j, v_u), \;\; e_3 = (v_k, v_u)$$

$$\|\Sigma(\mathcal{G} \cup \{e_1, e_2, e_3\})\|_2^2 = \|\Sigma(\mathcal{G})\|_2^2 + 3\|\Sigma_u\|_2^2 + \|\Sigma_k\|_2^2$$

Sub-Optimal Henneberg Construction

Optimal Vertex Addition and Edge Splitting steps
can be implemented "locally"



Growing Optimally Rigid Graphs

an algorithm...

Algorithm 1: $\mathcal{H}_2$ Optimally Rigid Graph Algorithm

Data: A set of N dynamic agents of form (1), indexed by the set $\mathcal{V} = \{v_1, \dots, v_n\}$. Each agent has $\mathcal{H}_2$ norm $\|\Sigma_i\|_2$ and identical sensing radius r.

Result: An $\mathcal{H}_2$ optimally rigid graph.

begin

·Sort and relabel each agent according to their $\mathcal{H}_2$ norm such that $\|\Sigma_1\|_2^2 \leq \|\Sigma_2\|_2^2 \leq \cdots \leq \|\Sigma_N\|_2^2$

·Assign weights, sort, and label candidate edges† such that $w(e_1) \leq \cdots \leq w(e_{|\mathcal{E}|})$, where $e_i = (v_k, v_l) \in \mathcal{E}$ and $w(e_i) = \|\Sigma_k\|_2^2 + \|\Sigma_l\|_2^2$.

·Set $\mathcal{G}^* := (\mathcal{V}^*, \mathcal{E}^*)$ with $\mathcal{V}^* = \{v_a, v_b\}$, $\mathcal{E}^* = \{e_1 = (v_a, v_b)\}$.

while $\mathcal{V}^* \neq \mathcal{V}$ **do**

  ·Set $\Omega = \{v \in \mathcal{V} \,|\, |\mathcal{V}^* \cap \mathcal{N}(v,t)| \geq 2\}$ and select the node $u = \arg\min_{i \in \Omega} \|\Sigma_i\|_2^2$

  if $|\mathcal{N}(u,t)| = 2$ **then**

    ·**do** $\mathcal{H}_2$ Optimal Vertex Addition (new edges e_a, e_b)

    ·Set $\mathcal{G}^* = (\mathcal{V}^* \cup \{u\}, \mathcal{E}^* \cup \{e_a, e_b\})$

  else

    ·Evaluate (7) for candidate edges

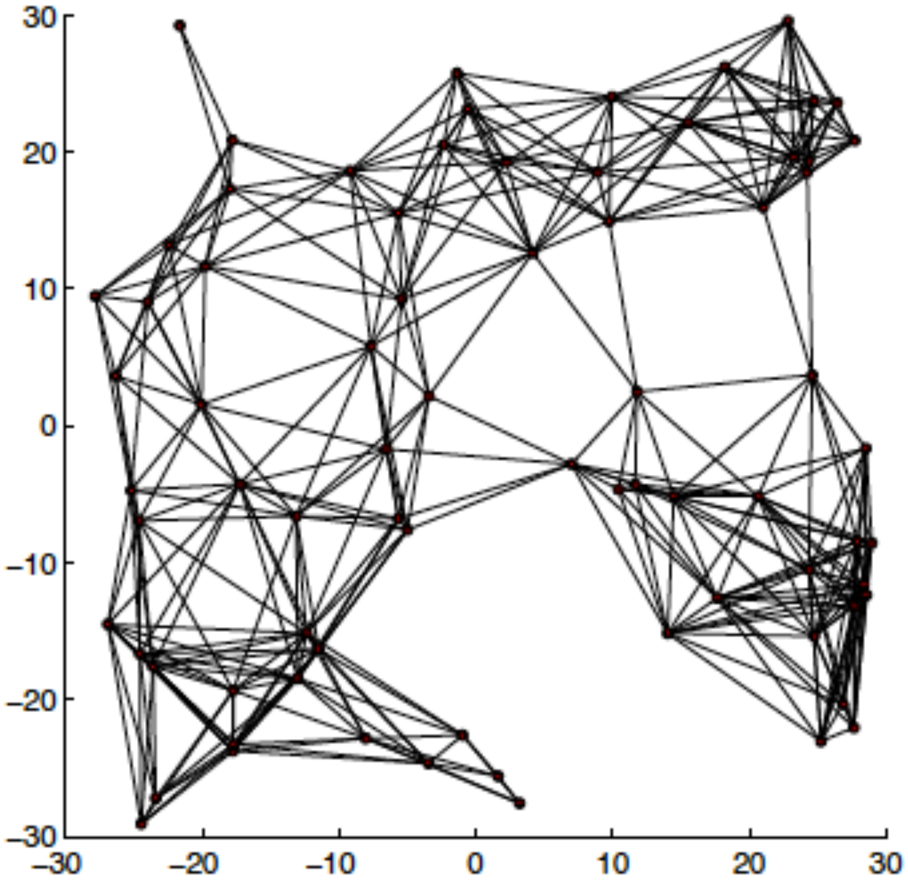
    ·**do** $\mathcal{H}_2$ Optimal Vertex Addition or $\mathcal{H}_2$ Optimal Edge Splitting based on (7) (new edges $\{e_a, e_b, e_c\}$ and deleted edge e_d)

    ·Set $\mathcal{G}^* = (\mathcal{V}^* \cup \{u\}, \mathcal{E}^* \cup \{e_a, e_b\})$ or $\mathcal{G}^* = (\mathcal{V}^* \cup \{u\}, \mathcal{E}^* \cup \{e_a, e_b, e_c\} - e_d)$

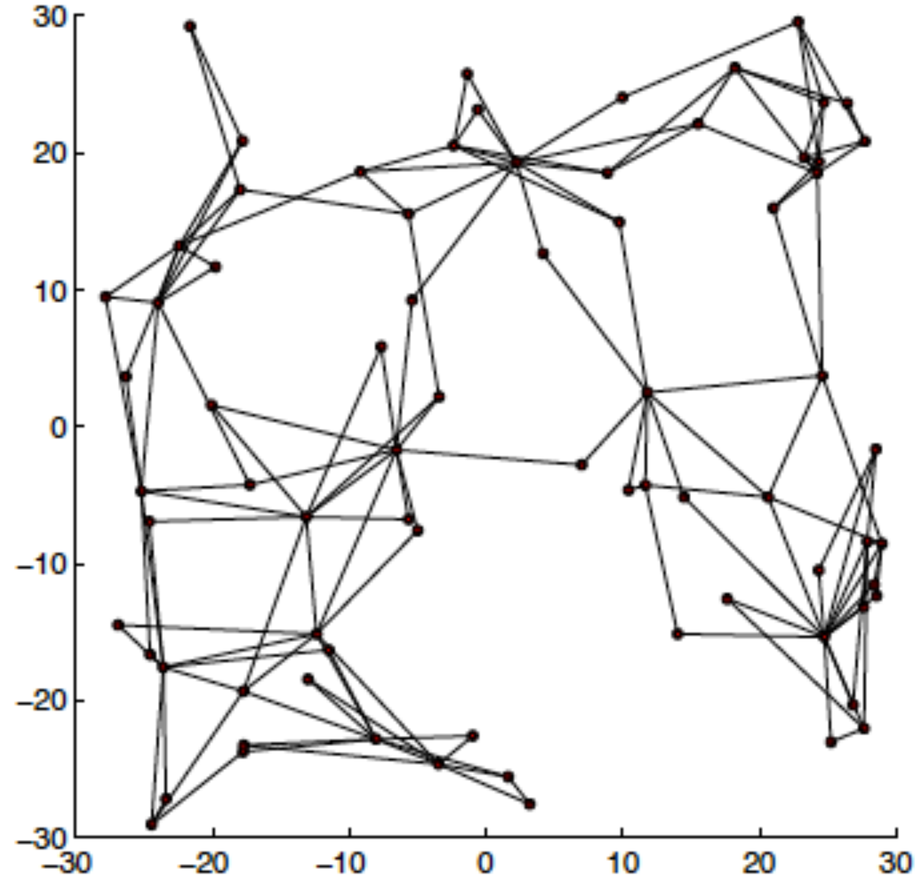
† The candidate edges are all possible edges an agent can establish within its sensing range.

Growing Optimally Rigid Graphs

simulation example...

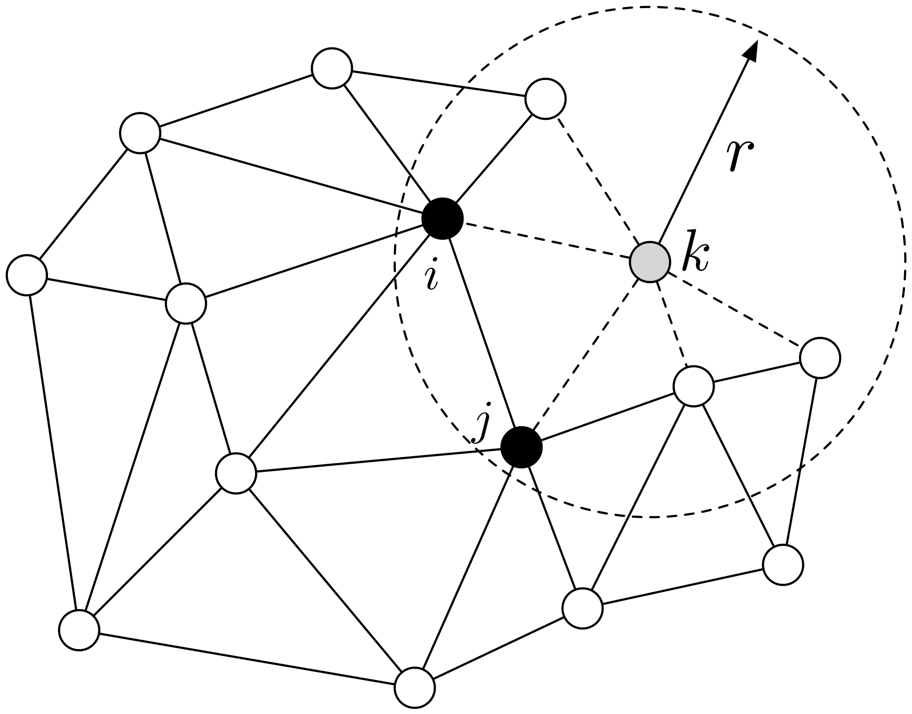


(a) All possible edges.

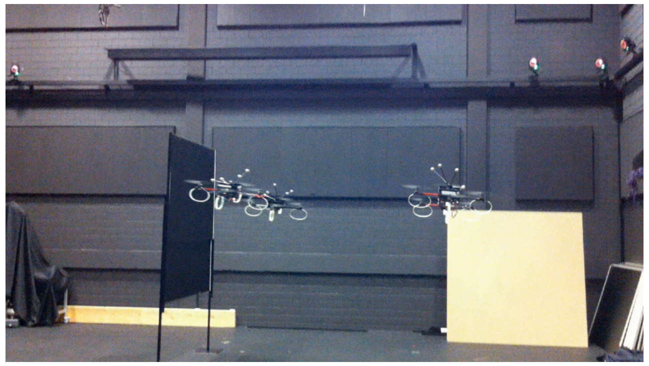


(b) The $\mathcal{H}_2$ optimally rigid graph.

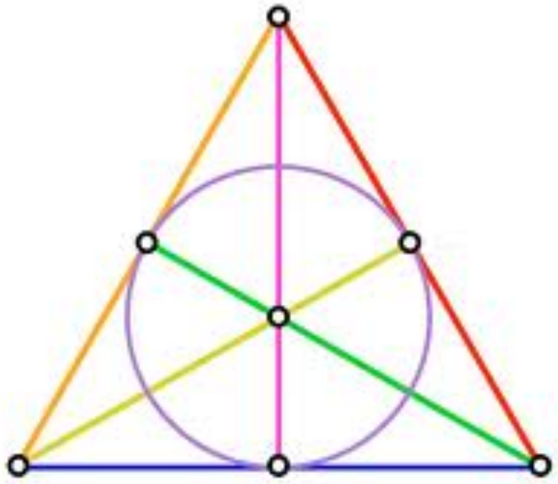
Future Outlook



- full distributed implementations
- formation specification and trajectory tracking
- optimality
- rigidity matroids
- sub-modular optimization
- sensor fusion and localization
- ...



$$f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y)$$



Acknowledments

Prof. Frank Allgöwer

Dr. Paolo Robuffo Giordano

Dr. Antonio Franchi