Elective in Robotics 2014/2015

Analysis and Control of Multi-Robot Systems

Elements of Graph Theory

Dr. Paolo Robuffo Giordano
CNRS, Irisa/Inria
Rennes, France

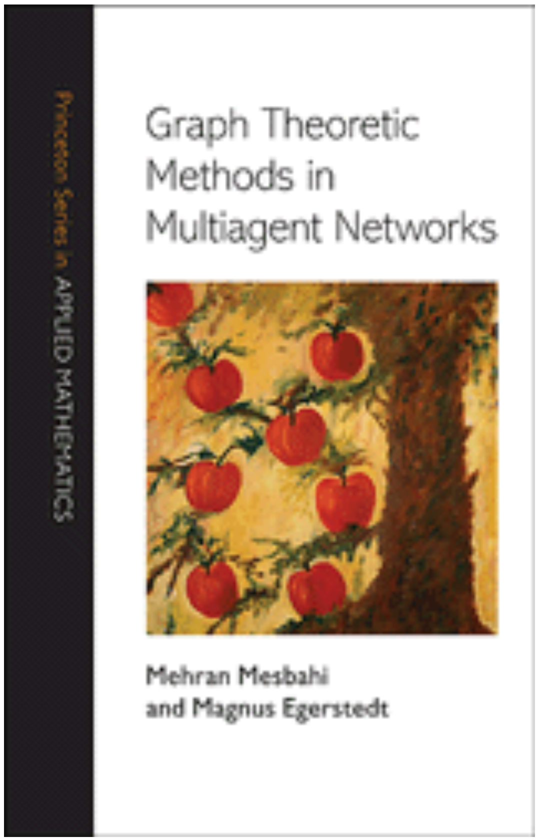
Dipartimento di Ingegneria Informatica
Automatica e Gestionale Antonio Ruberti

Sapienza
Università di Roma

INTRODUCTION TO GRAPHS

References

- Main TextBook



M. Mesbahi and M. Egerstedt
Graph Theoretic Methods in Multiagent Networks
Princeton Series in Applied Mathematics, 2010

Undirected Graphs

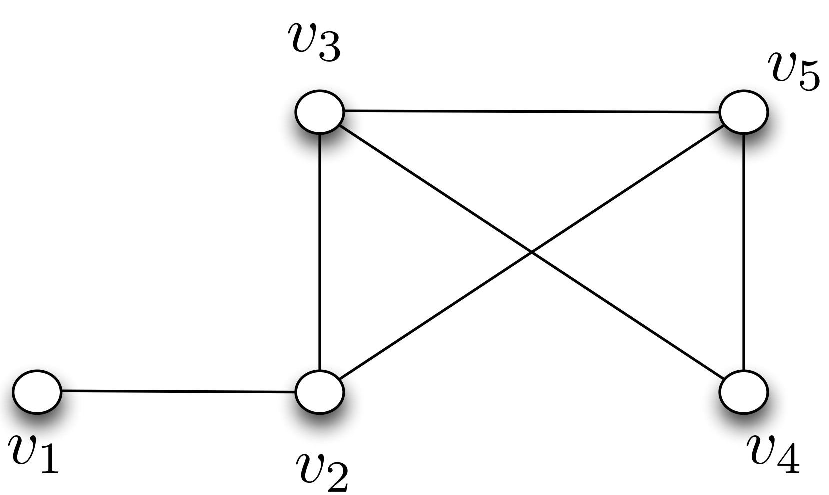
- An undirected Graph $\mathcal{G} = (\mathcal{V},\, \mathcal{E})$ is made of a Vertex Set (a finite set of elements)

$$\mathcal{V} = \{v_1,\, \ldots,\, v_N\}$$

and an Edge Set (a subset of unordered pairs of $[\mathcal{V}]^2$, the "2-element subsets" of $\mathcal{V}$)

$$[\mathcal{V}]^2 = \{(v_i,\, v_j)\},\ i = 1 \ldots N,\ j = 1 \ldots N,\ i \neq j$$

$$\mathcal{E} \subseteq [\mathcal{V}]^2 \qquad (v_i,\, v_j) \in \mathcal{E} \Rightarrow (v_j,\, v_i) \in \mathcal{E}$$



$$\mathcal{V} = \{v_1,\, v_2,\, v_3,\, v_4,\, v_5\}$$

$$\mathcal{E} = \{(v_1,\, v_2),\, (v_2,\, v_3),\, (v_2,\, v_5),\, (v_3,\, v_5),\, (v_3,\, v_4),\, (v_4,\, v_5)\}$$

Directed Graphs

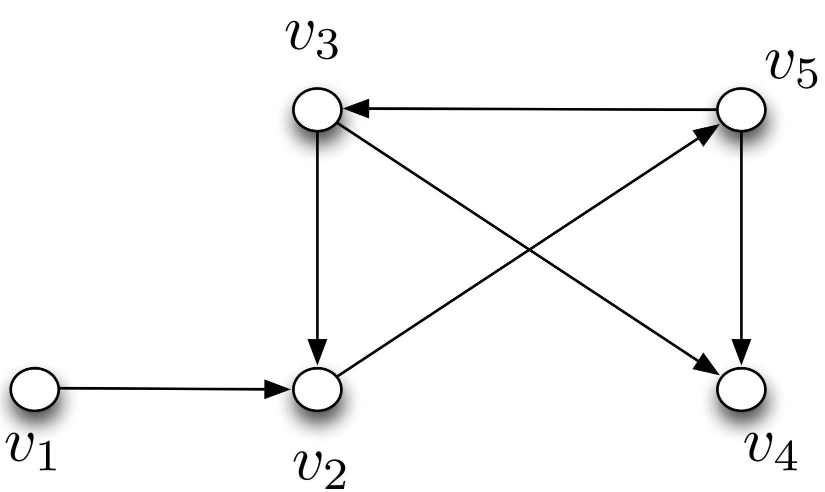
- A directed Graph $\mathcal{D} = (\mathcal{V}, \mathcal{E})$ is made of a Vertex Set (a finite set of elements)

$$\mathcal{V} = \{v_1, \ldots, v_N\}$$

and an Edge Set (a subset of ordered pairs of $[\mathcal{V}]^2$, the "2-element subsets" of $\mathcal{V}$)

$$[\mathcal{V}]^2 = \{(v_i, v_j)\}, \, i = 1 \ldots N, \, j = 1 \ldots N, \, i \neq j$$

$$\mathcal{E} \subseteq [\mathcal{V}]^2 \quad (v_i, v_j) \in \mathcal{E} \nRightarrow (v_j, v_i) \in \mathcal{E}$$



$$\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$$

$$\mathcal{E} = \{(v_1, v_2), (v_3, v_2), (v_2, v_5), (v_3, v_4), (v_5, v_3), (v_5, v_4)\}$$

Definitions

- Node v_j is said adjacent (neighbor) of v_i if $(v_j,\, v_i) \in \mathcal{E}$

- Given a node v_i, the set $\mathcal{N}_i$ is the set of all neighbors of v_i

$$\mathcal{N}_i = \{v_j \in \mathcal{V} |\ (v_j,\, v_i) \in \mathcal{E}\}$$

- The degree of a node v_i is $d_i = |\mathcal{N}_i|$ (undirected graphs)

- The in-degree of a node v_i is $d_i^{in} = |\mathcal{N}_i|$ (directed graphs)

- A path is a sequence of distinct vertexes $v_{i_0} v_{i_1} \dots v_{i_m}$ such that, $\forall k = 0, \dots, m-1$ the vertexes v_{i_k} and $v_{i_{k+1}}$ are adjacent (neighbors)

- If $v_{i_0} = v_{i_m}$ (special exception), then the path is called a cycle

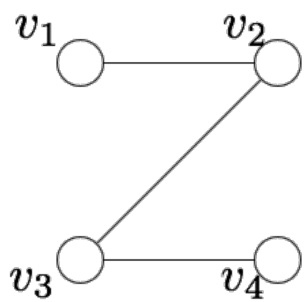
Definitions

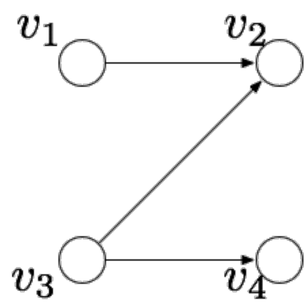
- An undirected graph is said connected if there exists a path joining any two vertexes in $\mathcal{V}$

- A directed graph is said strongly connected if there exists a (directed) path joining any two vertexes in $\mathcal{V}$

- A directed graph is said weakly connected if there exists an undirected path joining any two vertexes in $\mathcal{V}$

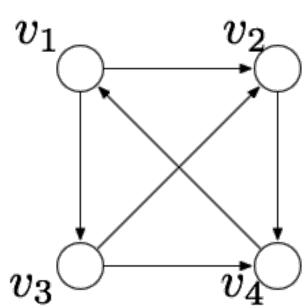
connected

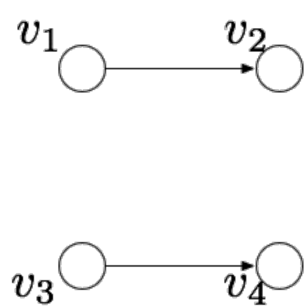




weakly connected

strongly connected

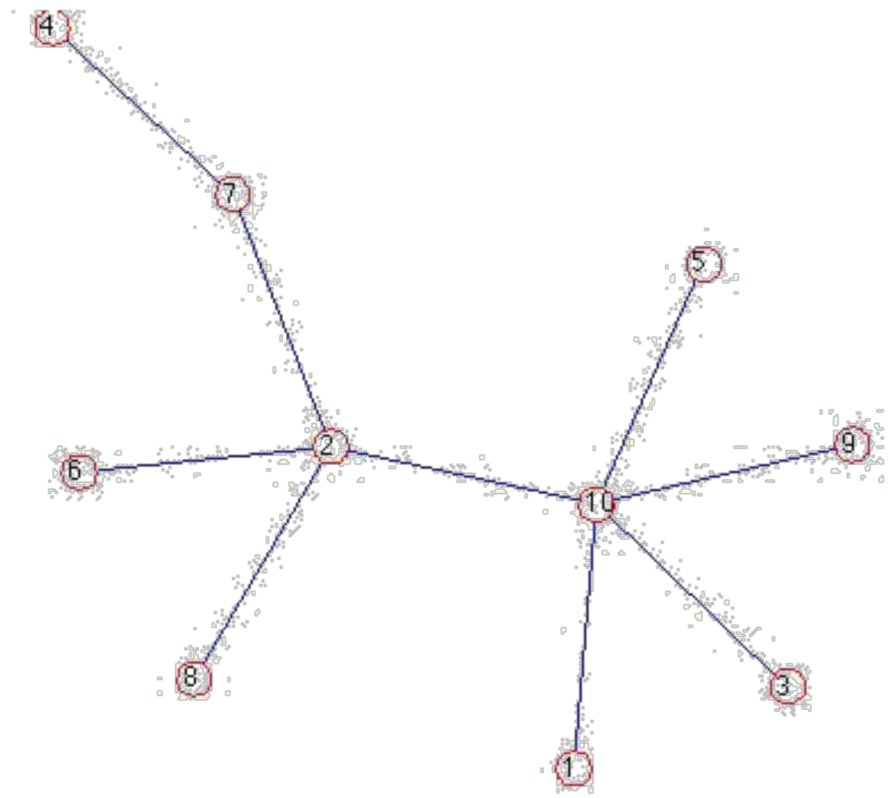




disconnected

Definitions

- A tree is a connected graph containing no cycles

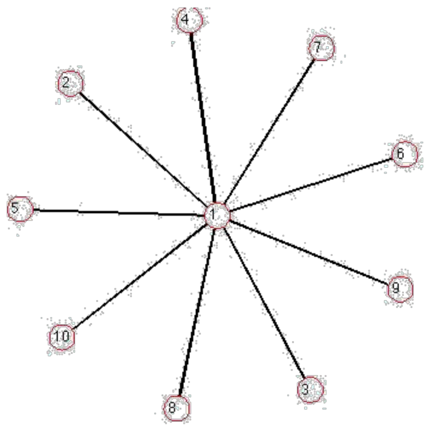


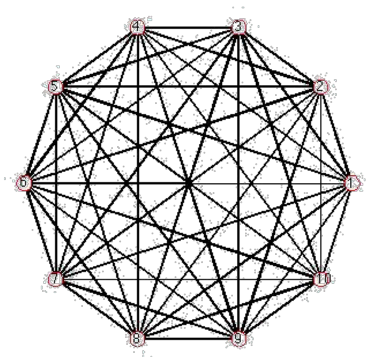
Definitions

- Other special graphs

star graph

$$S_{10}$$



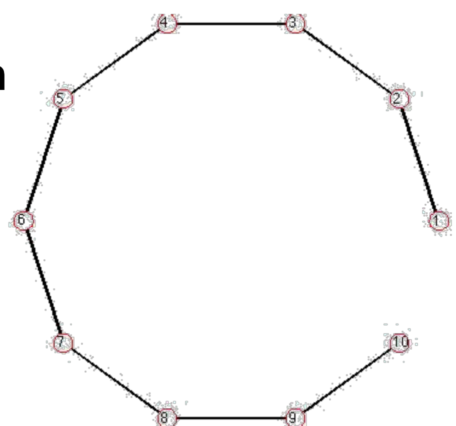


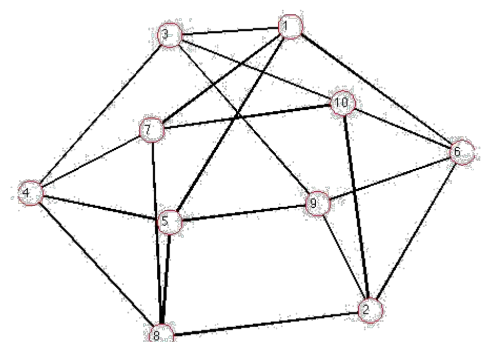
complete graph

$$K_{10}$$

path (line) graph

$$P_{10}$$





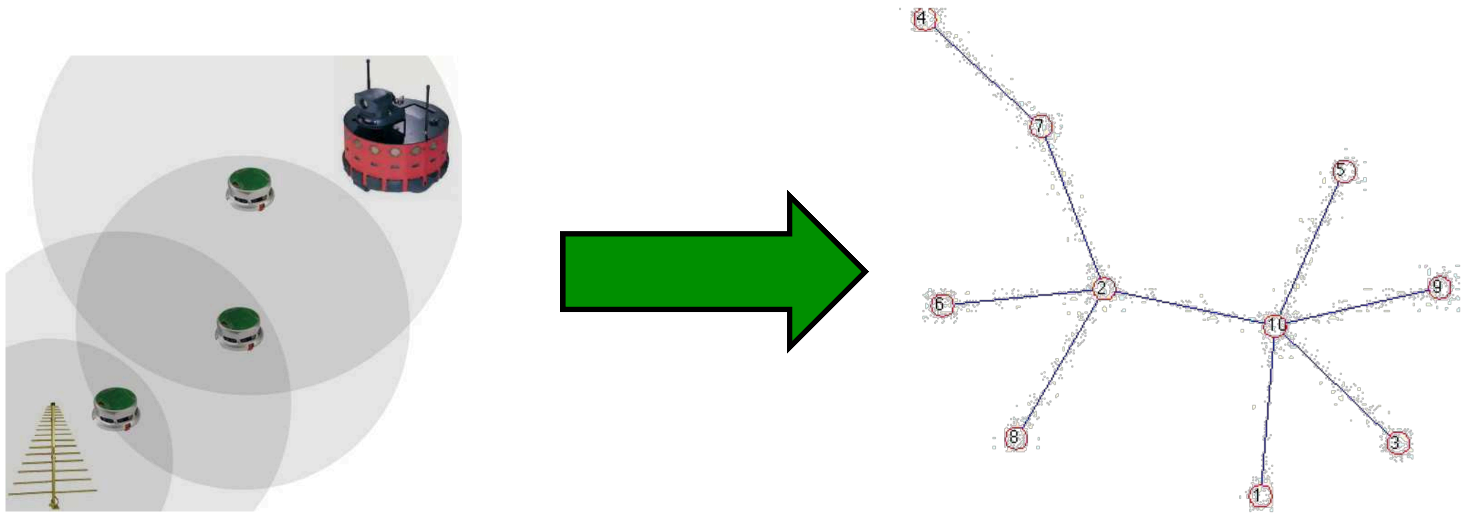
k-regular graph

4-regular

Why do we need graphs?

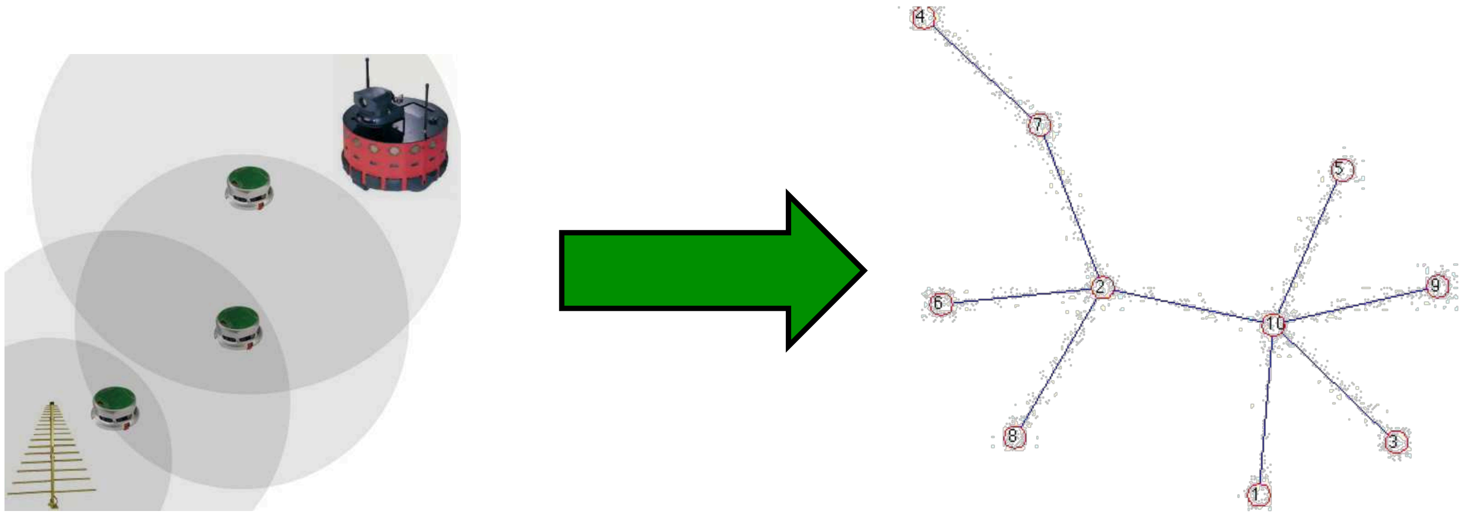
- Why are graphs important for multi-robot systems?

- Graphs are extremely powerful tools for encoding the information/action flow among the robots



- We (sometimes implicitly) assume that every robot has a limited ability to
  - perceive the environment with onboard sensors (e.g., other robots)
  - communicate information to other robots (via a communication medium)
  - elaborate information (gathered from onboard sensors or comm. medium)
  - in general, plan, act, and influence the environment (e.g., other robots)

Why do we need graphs?



- A graph naturally encodes in a compact way these limitations
- Many distinct graphs can be associated to a group of multiple robots (agents)
- Sensing graphs: for each sensors, encode what robots can be locally sensed
- Communication graphs: for each communication medium, encode with which robots a comm. link can be established (uni- or bi-directional)
- Action graphs: for each control action, encode what robots will be (locally) affected
- And so on…

Decentralization

- The issue of limited sensing/communication/action abilities (and, thus, the use of graphs) is closely related to the notion of decentralization and decentralized/distributed sensing/control

- Decentralization: every unit (robot) has
  - limited sensing/communication (information gathering)
  - limited computing power (information processing)
  - limited available memory (information storage)

- For a robot, it (typically) must elaborate the gathered information to run its local controller (making use of local computing power and memory)

- The controller complexity is bounded by the above limitations

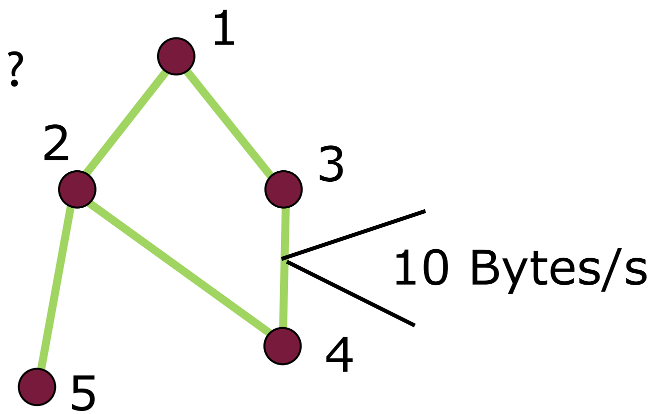
- If the whole state of all the robots is needed, the complexity (e.g., computing power) increases with the total number of robots
  - May easily become unfeasible because of the above limitations
  - And each robot would need to know the whole state...

Decentralization

- Decentralization: cope with the above limitations by designing decentralized controllers (i.e., spreading the complexity across the multiple robots)

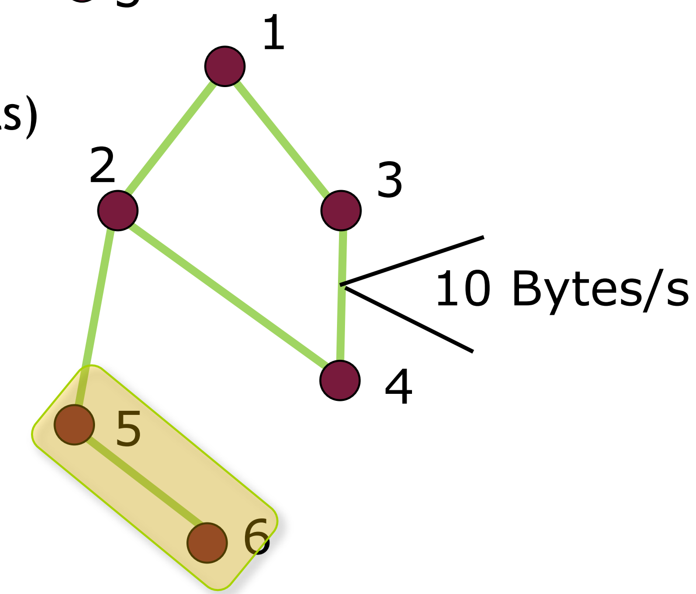
- What do we exactly mean by “decentralized controller” ?

- An example: assume graphs are used to encode the information flow among robots (sensed, communicated, elaborated)



- Decentralization: on each edge, the size of the information flow is constant (w.r.t. the number of robots)

- Example: adding node 6 does not increase the information needed by nodes 1,2,3,4



- Thus, the amount of information grows linearly with the number of neighbors

- The same applies to the used memory or computing power (<u>constant</u> per neighbor)

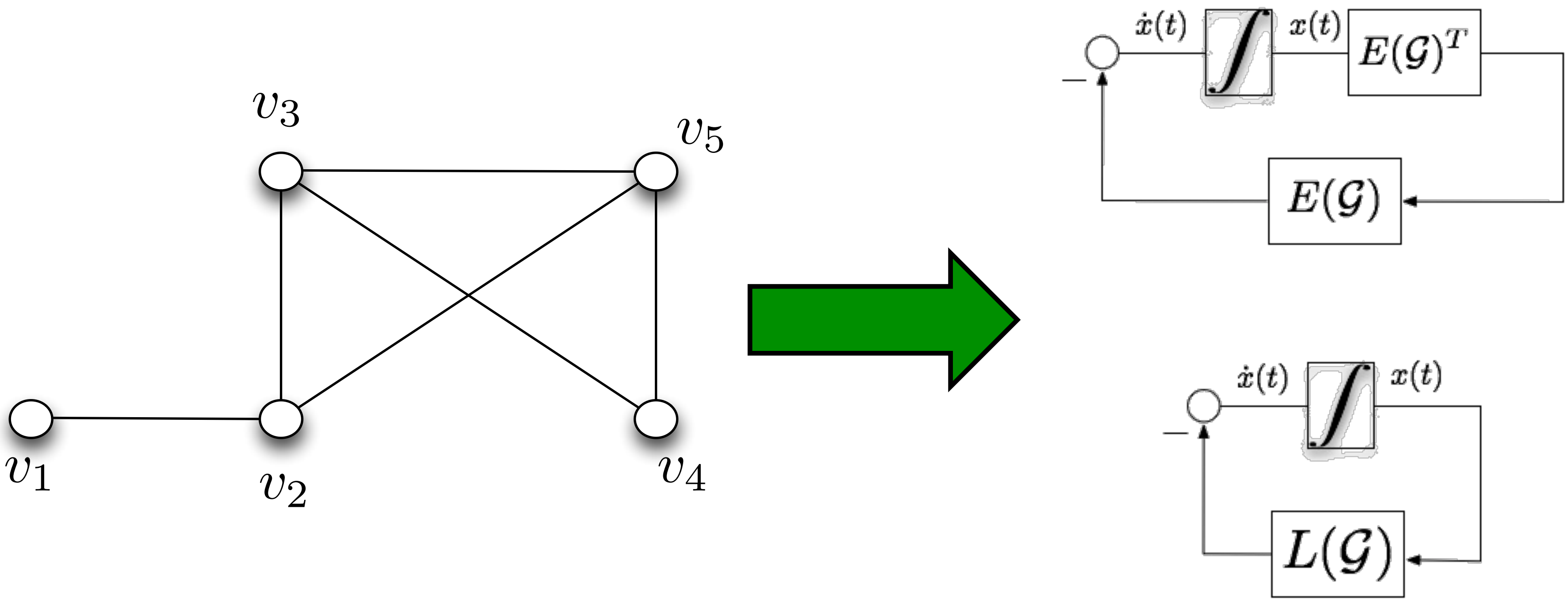
ALGEBRAIC GRAPH THEORY

Graphs and Matrixes

- Several matrixes can be associated to graphs and….
- ….several graph properties can be deduced from the associated matrixes

- Graphs + Matrixes = Algebraic Graph Theory

- The following Algebraic tools will be fundamental for linking Graph Theory to the study of multi-robot systems (when seen as a collection of dynamical systems)

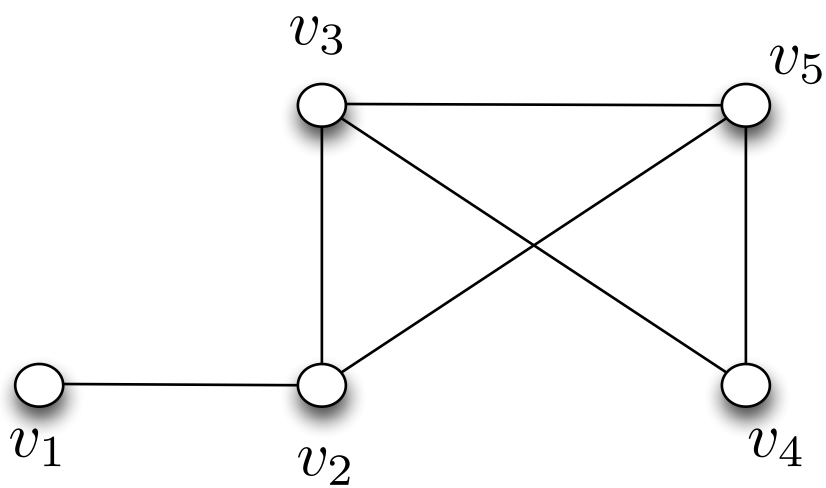


Adjacency Matrix

- Adjacency Matrix $A \in \mathbb{R}^{N \times N}$
- Square and symmetric (only for undirected graphs) matrix
- Defined so that $A_{ij} = 0$ if $(v_j,\, v_i) \notin \mathcal{E}$ and $A_{ij} = 1$ if $(v_j,\, v_i) \in \mathcal{E}$
- Note: $A_{ii} = 0$ and $A_{ij} = A_{ji}$, thus $A = A^T$
- Note: one can generalize to any positive weight $A_{ij} = w_i,\ w_i \geq 0$
- Note: for directed graphs, in general $A_{ij} \neq A_{ji}$ and thus $A \neq A^T$

Adjacency Matrix

- Example



$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Degree Matrix

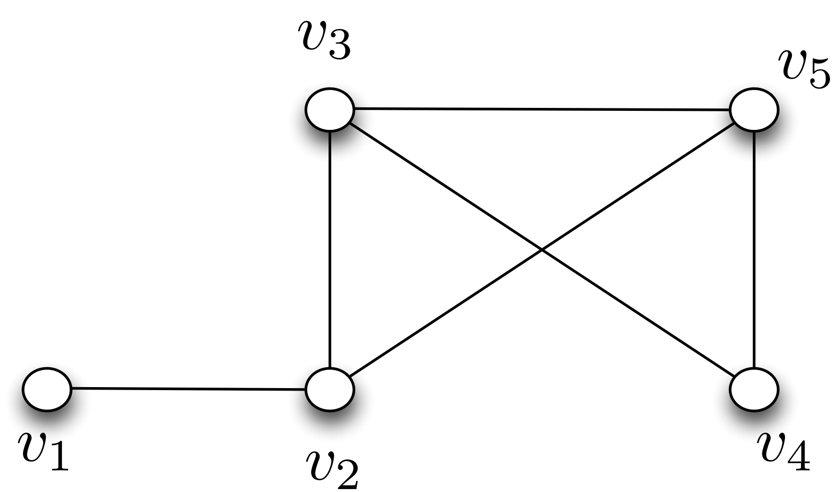
- Degree matrix $\Delta \in \mathbb{R}^{N\times N}$

- Diagonal (symmetric) matrix with the node degrees d_i as diagonal elements

$$\Delta = diag(d_i)$$

- Alternatively,

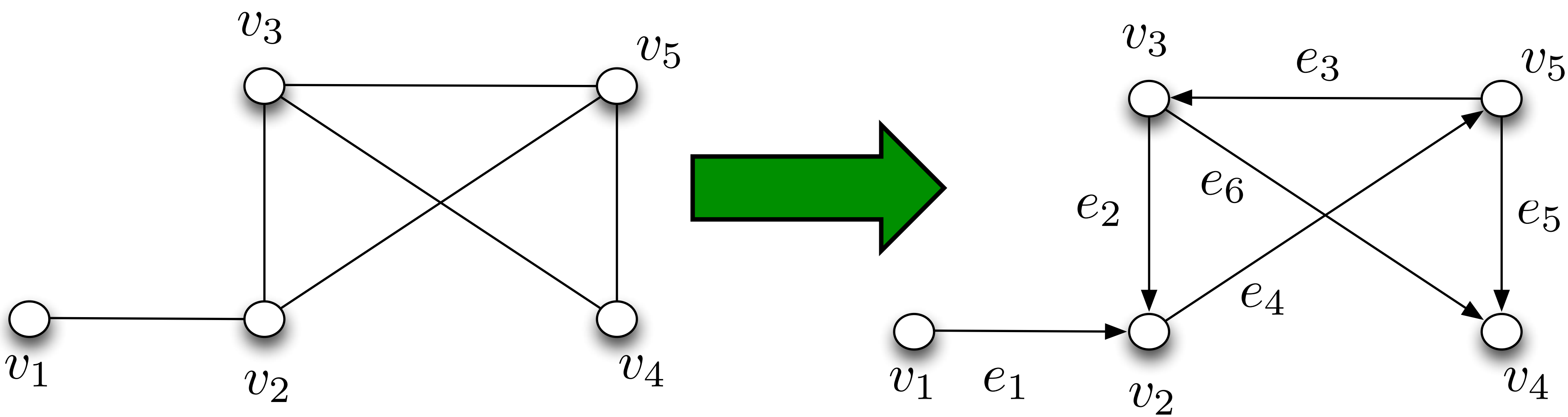
$$\Delta = diag\left(\sum_{j=1}^{N} A_{ij}\right)$$



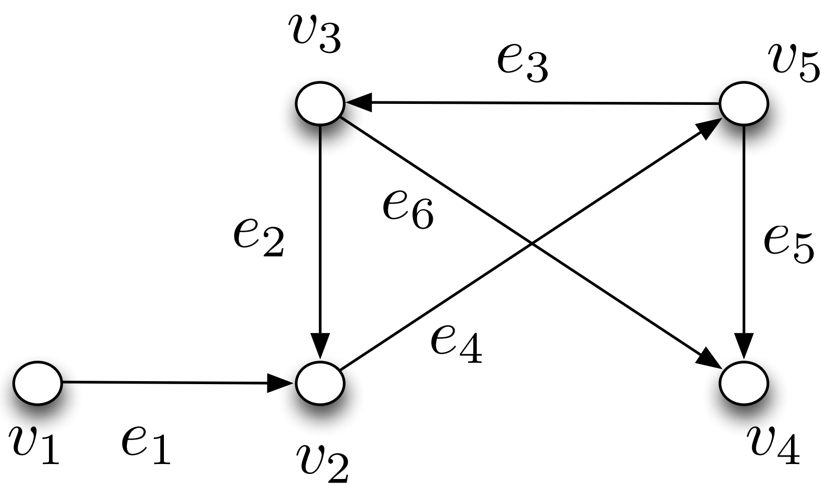
$$\Delta = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

Incidence Matrix

- Incidence matrix $E \in \mathbb{R}^{N\times|\mathcal{E}|}$
- Used to encode the incidence relationship among edges and vertexes
- Assign an arbitrary orientation and an arbitrary labeling to the edges



Incidence Matrix



- Let $E_{ij} = -1$ if vertex v_i is the tail of edge e_j
- Let $E_{ij} = 1$ if vertex v_i is the head of edge e_j
- Let $E_{ij} = 0$ otherwise

$$E = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 & -1 & 0 \end{bmatrix}$$

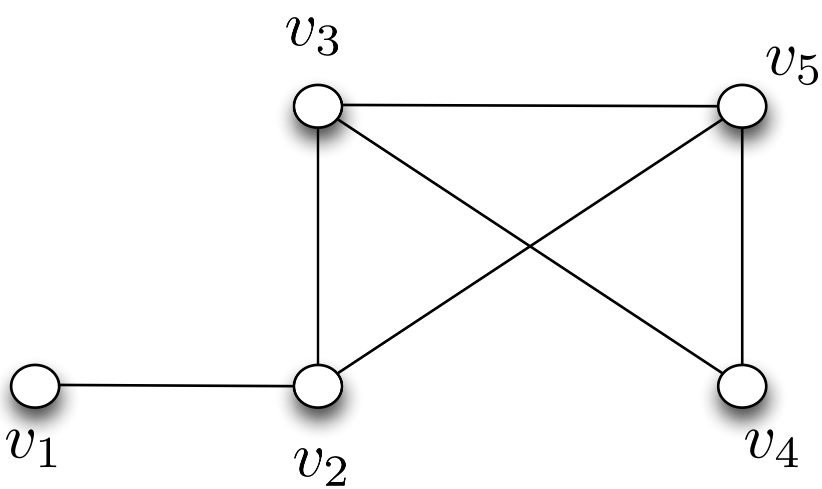
Laplacian Matrix

- Laplacian matrix $L \in \mathbb{R}^{N \times N}$

- First definition: $L = \Delta - A$

- Second definition: $L = EE^T$

- The two Defs. are equivalent, and the latter does not depend on the particular labeling and orientation chosen for the graph



$$L = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & -1 \\ 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & -1 & -1 & 3 \end{bmatrix}$$

Laplacian Matrix

- L is symmetric (from both Defs.)
- L is positive semi-definite (from Def. 2)

- $L\mathbf{1} = 0$ where $\mathbf{1}$ is a vector of all ones
  - this shows that L is actually positive semi-definite as it has a non-void null-space

- Being symmetric and positive semi-definite, all its N eigenvalues λ_i are real and non-negative

- Order them as $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N$

- **Property**: the graph $\mathcal{G}$ is **connected** if and only if $\lambda_2 > 0$

- The quantity λ_2 is referred to as connectivity eigenvalue (or Fiedler eigenvalue)

- Obviously, $\mathbf{1}$ is the eigenvector associated to λ_1 and $\mathrm{rank}(L) = N - 1$ (for connected graphs)

Laplacian Matrix

- Note also that, being L symmetric, it is $\mathbf{1}^T L = 0$

- Also, being $L = EE^T$, it is $E^T\mathbf{1} = 0$ and $\mathrm{rank}(E) = N - 1$ (for connected graphs)

- Some additional properties (among many....)

- $trace(L) = 2|\mathcal{E}|$

- Let L_i be the matrix obtained from the Laplacian L after removing the row and column indexing vertex v_i

- Then $\det L_i = t(\mathcal{G})$ for any v_i where $t(\mathcal{G})$ is the number of spanning trees of graph $\mathcal{G}$

THE CONSENSUS PROTOCOL

The Consensus Protocol

- Let us use the tools introduced so far for studying one of the most fundamental problem in multi-robots (and multi-agents) literature

- **The Consensus Protocol**

- Formulation of the problem:
  - Consider N agents with an internal state $x_i \in \mathbb{R}$
  - Consider an internal dynamics for the state evolution
    - in our case, single integrator $\dot{x}_i = u_i$
  - Consider an interaction graph $\mathcal{G}$ having the agents as vertexes

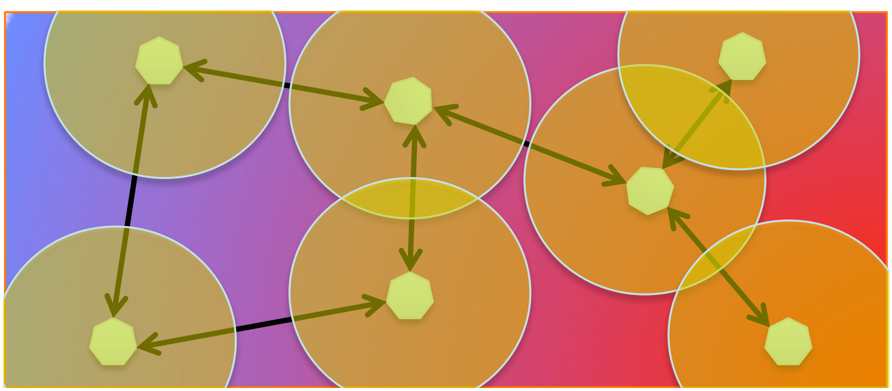
- Problem: design the control inputs u_i so that
  - all the states x_i agree on the same common value $\bar{x}$ (**unspecified**)

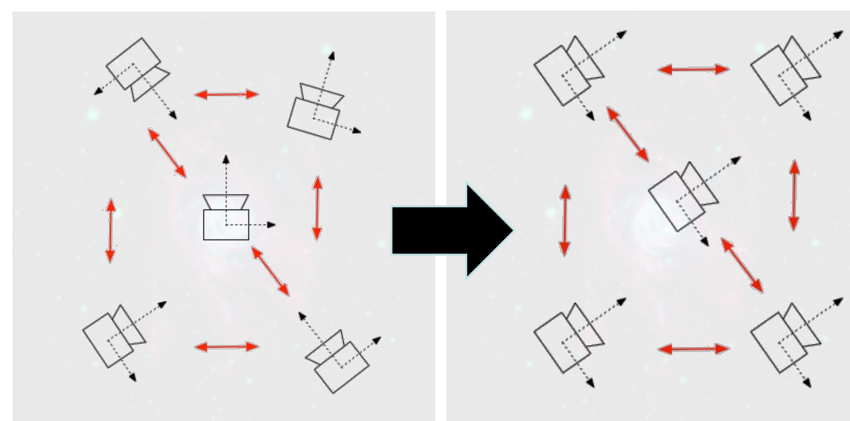
$$\lim_{t \to \infty} x_i(t) = \bar{x}, \ \forall i$$

  - by making use in u_i of only relative information w.r.t. the neghbors' state (relative sensing and decentralization)

The Consensus Protocol

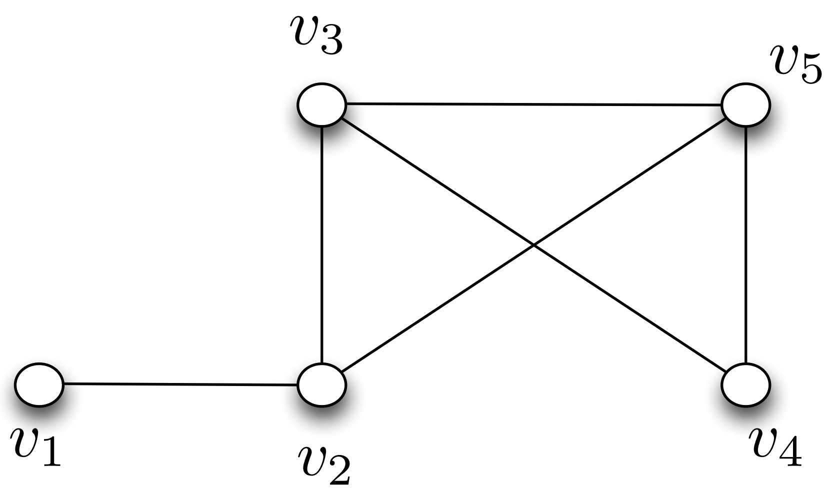
- Possible applications of the consensus protocol
  - rendezvous: meet at a common point (uniform the positions)
  - alignment: point in the same direction (uniform the angles)
  - distributed estimation: agree on the estimation of some distributed quantity (e.g., average temperature)
  - synchronization: agree on the same time (regardless of phase shifts or different rates in the clocks)





The Consensus Protocol

- Take $N = 5$ agents and the interaction graph $\mathcal{G}$



- The graph $\mathcal{G}$ models how information flows across the agents

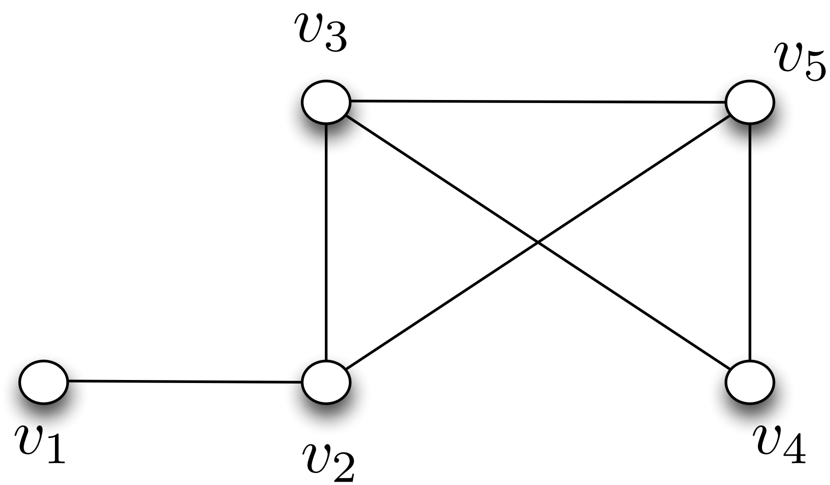
- Design $u_i = u_i(x_i - x_j) \;\; \forall j \in \mathcal{N}_i$

- Example: $u_1 = u_1(x_1 - x_2)$, $u_2 = u_2(x_1 - x_2,\, x_2 - x_3,\, x_2 - x_5)$, and so on....

- Any idea on how to solve the problem?

The Consensus Protocol

- Solution: let u_i be the sum of all the differences of the neighbors' states w.r.t. the state of agent i



$$u_1 = (x_2 - x_1)$$

$$u_2 = (x_1 - x_2) + (x_3 - x_2) + (x_5 - x_2)$$

$$u_3 = (x_2 - x_3) + (x_4 - x_3) + (x_5 - x_3)$$

$$u_4 = (x_3 - x_4) + (x_5 - x_4)$$

$$u_5 = (x_2 - x_5) + (x_3 - x_5) + (x_4 - x_5)$$

The Consensus Protocol

- Consensus protocol:
  - in compact form for agent i ➡ $u_i = \sum_{j \in \mathcal{N}_i} (x_j - x_i)$
  - in compact form for all the agents $u = -Lx$

$$u = -\begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & -1 \\ 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & -1 & -1 & 3 \end{bmatrix} x$$

  - and when closing the loop (recall that $\dot{x}_i = u_i$)

$$\dot{x} = -Lx$$

The Consensus Protocol

- Problem: under which conditions the closed-loop system

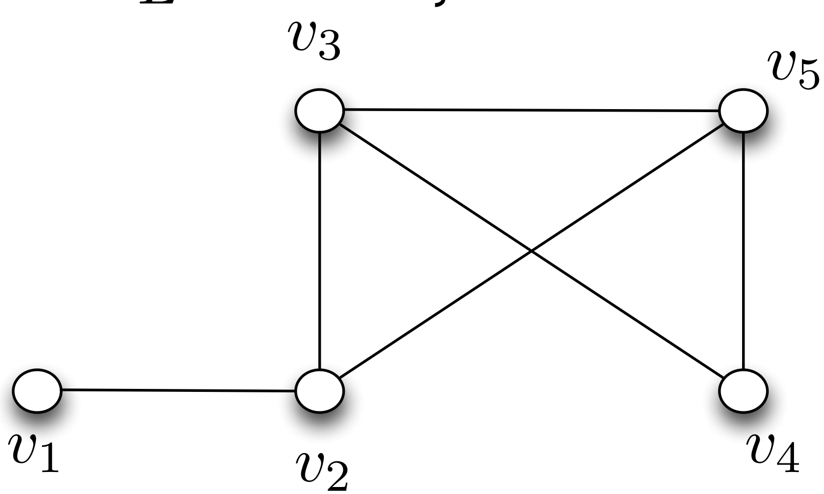
$$\dot{x} = -Lx$$

will solve the initial consensus requirement (if at all)?

$$\lim_{t\to\infty} x_i(t) = \bar{x}, \quad \forall i$$

- Convergence to an (arbitrary but common) $\bar{x}$ is related to the properties of the Laplacian L (the state-transition matrix in the closed-loop dynamics)

- Properties of the Laplacian L are directly related to the associated graph $\mathcal{G}$



The Consensus Protocol

- Main result: the consensus protocol converges if and only if graph $\mathcal{G}$ is **connected**

- **First** proof making use of the explicit solution of $\dot{x} = -Lx$

- Given an initial condition x_0, the explicit solution of the consensus dynamics (time-invariant linear system) is

$$x(t) = e^{-Lt}x_0$$

- **Fact 1**: a symmetric matrix (such as L) is always diagonalizable by an orthonormal matrix U, i.e., such that $UU^T = I$

- Therefore, $L = U\Lambda U^T$ where $\Lambda = diag(\lambda_i)$

- **Fact 2**: $e^{-U\Lambda U^T t} = Ue^{-\Lambda t}U^T$

- We then get $x(t) = Ue^{-\Lambda t}U^T x_0$

The Consensus Protocol

- Rewrite as $x(t) = u_1 u_1^T e^{-\lambda_1 t} x_0 + \sum_{i=2}^{N} u_i u_i^T e^{-\lambda_i t} x_0$
- We already know that $\lambda_1 = 0$ and $u_1 = \dfrac{\mathbf{1}}{\sqrt{N}}$
- Thus $x(t) = \dfrac{(\mathbf{1}^T x_0)\mathbf{1}}{N} + \sum_{i=2}^{N} u_i u_i^T e^{-\lambda_i t} x_0$
- If $\mathcal{G}$ is connected, then $\lambda_2 \neq 0$ and $\lambda_N \geq \ldots \geq \lambda_2 > 0$
- Therefore $\lim_{t \to \infty} x(t) = \dfrac{(\mathbf{1}^T x_0)\mathbf{1}}{N}$
- What is $\dfrac{(\mathbf{1}^T x_0)\mathbf{1}}{N}$?

The Consensus Protocol

- The term $\frac{\mathbf{1}^T x_0}{N}$ is just the average of the initial state x_0

- The post-multiplication by $\mathbf{1}$ in $\frac{(\mathbf{1}^T x_0)\mathbf{1}}{N}$ spreads this average on all the components of x

- $x_i \to \frac{\mathbf{1}^T x_0}{N}, \quad \forall i$

- Thus, what have we obtained? All the agent states x_i converge towards a common value, that is, the average of the initial state x_0

- Definition: the agreement subset $\mathcal{A} \subseteq \mathbb{R}^N = span(\mathbf{1}) = \{x|\ x_i = x_j\}$

- The consensus protocol makes the state $x(t) \to \mathcal{A}$

The Consensus Protocol

- **Second** proof: exploit Lyapunov Arguments
- Define the Lyapunov candidate $V(x) = \frac{1}{2}x^T x$
- Its evolution (along the system trajectories) is

$$\dot{V}(x) = x^T \dot{x} = -x^T L x$$

- Matrix L is positive semi-definite. Therefore $\dot{V}(x) \leq 0$
- This shows that the state trajectories are bounded since $V(x)$ does not increase over time
- To draw additional conclusions, we must resort to LaSalle's Invariance theorem
- What is the largest invariant set contained in $\dot{V}(x) = 0$?

The Consensus Protocol

- What is the set $\dot{V}(x) = x^T L x = 0$?

- It is the null-space of L (remember L is symmetric)

- If the graph $\mathcal{G}$ is connected, we know that this null-space is just $\mathcal{A}$

- Therefore, $x(t) \rightarrow \mathcal{A} = span(\mathbf{1})$

- Another remark: consider the scalar quantity $\mathbf{1}^T x$. What is its time evolution under the consensus protocol?

- $\mathbf{1}^T \dot{x} = -\mathbf{1}^T L x = 0$

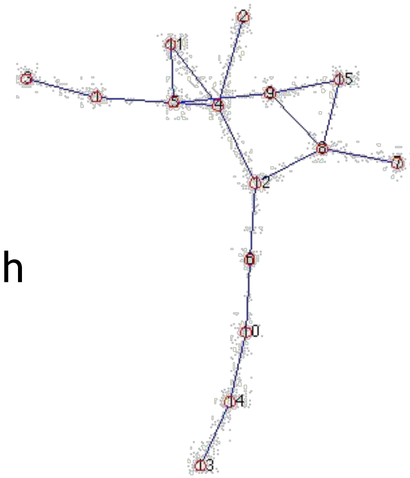
- Therefore, $\mathbf{1}^T x$ represents a constant of motion of the closed-loop system

- $\mathbf{1}^T x(t) \equiv \mathbf{1}^T x_0 = const$. The centroid of the states never changes over time

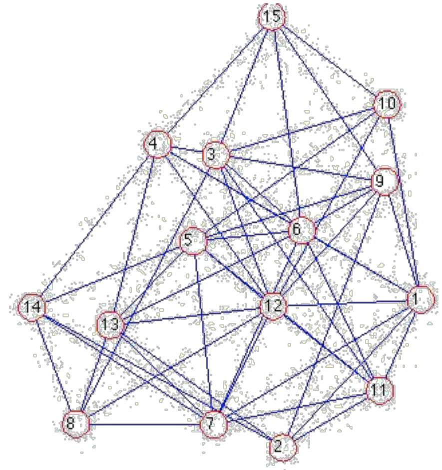
The Consensus Protocol

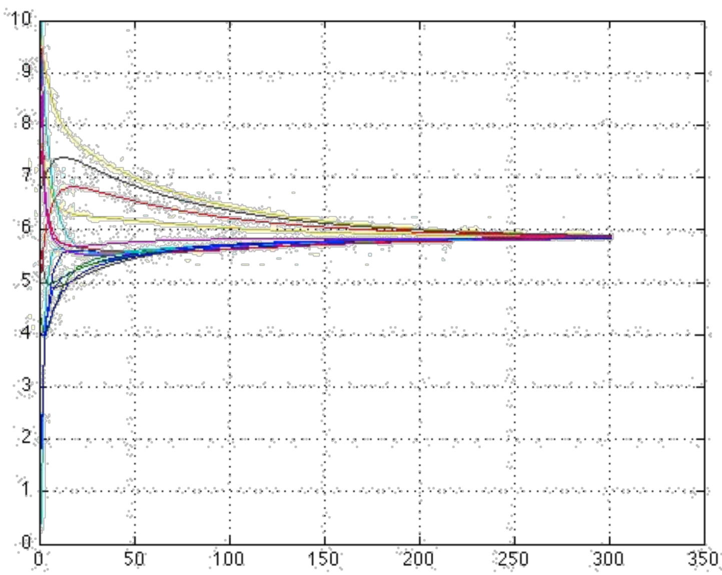
- Some simulations:

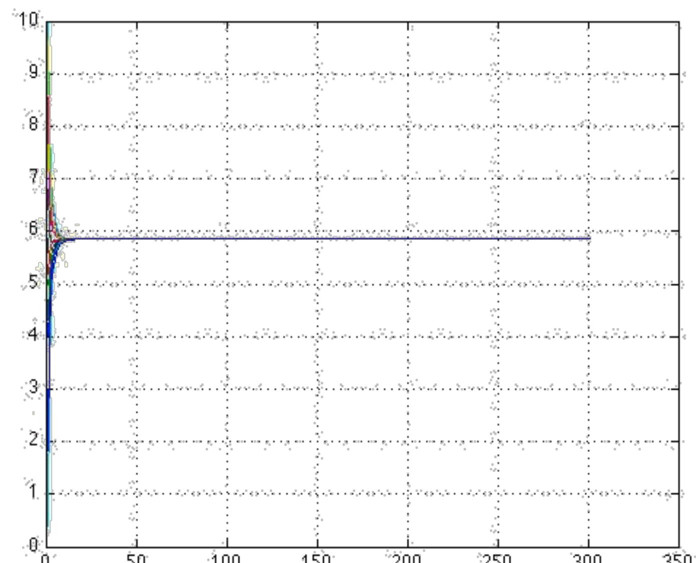
“sparse” graph



“dense” graph







The Consensus Protocol

- What dictates the rate of convergence of the consensus protocol?

- Sparse graph -> slow convergence
- Dense graph -> fast convergence

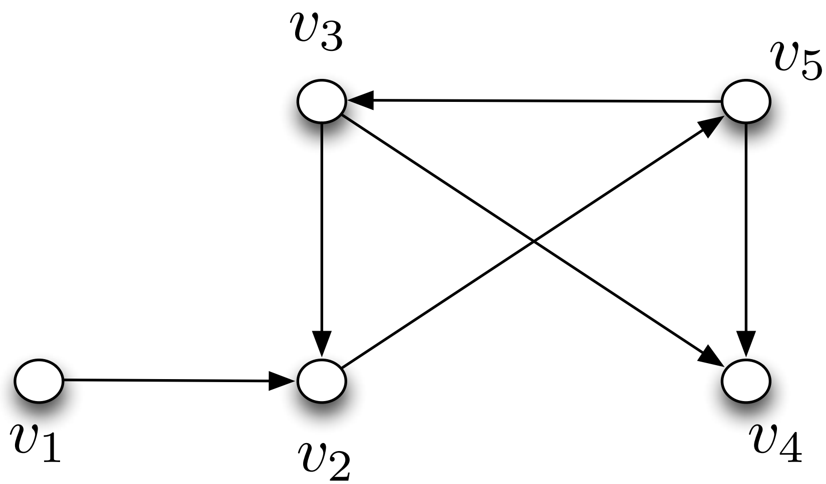
- Rate of convergence is directly related to the value of λ_2 (i.e., to the degree of connectivity of the graph)
- From $x(t) = \dfrac{(\mathbf{1}^T x_0)\mathbf{1}}{N} + \sum_{i=2}^{N} u_i u_i^T e^{-\lambda_i t} x_0$
- The value of λ_2 (smallest eigenvalue in the sum) dictates the rate of the asymptotic decay of the sum of exponential functions

- If λ_2 is large, the exponential sum will decay faster

- Therefore: the more connected the graph, the faster the consensus convergence

The Consensus Protocol

- Let us now briefly consider the case of directed graphs



- How does the consensus machinery apply to this case?

- First "**big**" difference: the graph Laplacian L is not symmetric any more

$$L = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & 0 & 0 & 1 \end{bmatrix}$$

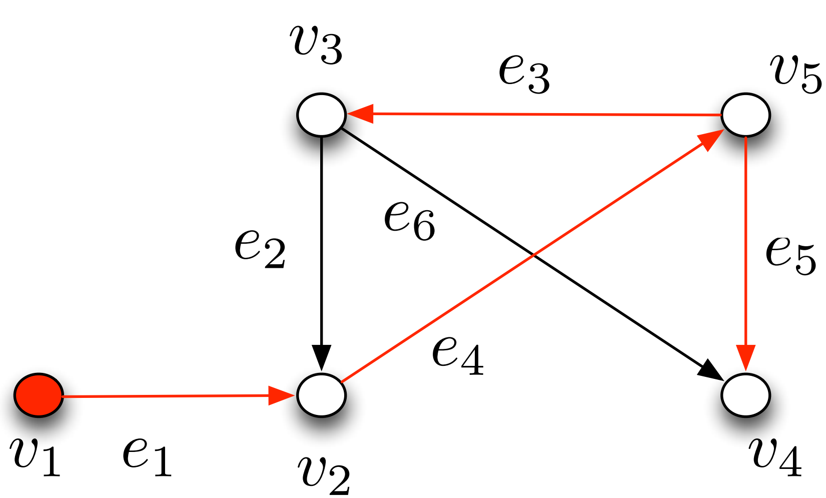
The Consensus Protocol

- How does now the system $\dot{x} = -Lx$ evolve in this situation?

- We still have $L\mathbf{1} = 0$ but in general $\mathbf{1}^T L \neq 0$

- **Fact 1:** $\mathrm{rank}(L) = N - 1$ if and only if the graph contains a rooted out-branching

- A rooted out-branching is a directed graph such that
    - it contains no cycles
    - it has a vertex (root) with a directed path to all the other vertexes



The Consensus Protocol

- If $\mathrm{rank}(L) = N - 1$ then $\mathbf{1}$ is the only vector spanning its right null-space

- **Fact 2** (application of Gersgorin Theorem): a Laplacian matrix for directed graphs has all the eigenvalues with non-negative real part (and they cannot be an imaginary pair)

$$\Re(\lambda_i) \geq 0$$

- Exploiting fact 1, it must be $\lambda_1 = 0$ and $0 < \Re(\lambda_2) \leq \ldots \leq \Re(\lambda_N)$

- Then, we can follow an argument equivalent to the undirected graph case

- Let $L = PJ(\Lambda)P^{-1}$ be the Jordan decomposition of L

with $J(\Lambda) = \begin{bmatrix} 0 & 0 & \ldots & 0 \\ 0 & J(\lambda_2) & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \ldots & 0 & J(\lambda_N) \end{bmatrix}$

The Consensus Protocol

- Expanding into the explicit solution of $\dot{x} = -Lx$ we get

$$x(t) = e^{-Lt}x_0 = (p_1 q_1^T)x_0 + P\sum_{i=2}^{N}(e^{-J(\lambda_i)t})P^{-1}x_0$$

where p_1 and q_1 are the right and left eigenvector associated to $\lambda_1 = 0$ ($p_1 = \mathbf{1}$ as we already know)

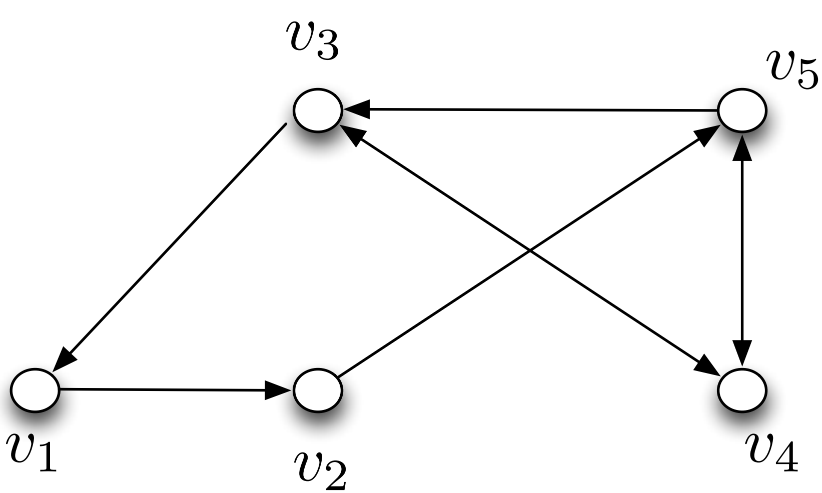
- Since $0 < \Re(\lambda_2) \leq \ldots \leq \Re(\lambda_N)$ we then obtain (normalizing $q_1^T \mathbf{1} = 1$)

$$\lim_{t\to\infty} x(t) = (q_1^T x_0)p_1 = (q_1^T x_0)\mathbf{1}$$

- Note that in general $q_1 \notin span(\mathbf{1})$
- For instance, for our example it is $q_1 = span([1\ 0\ 0\ 0\ 0]^T)$
- In general, the consensus will not converge to the average of the initial condition

The Consensus Protocol

- Is it possible to have $q_1 \in span(\mathbf{1})$ also for the directed graph case?
- This would allow for $\lim_{t\to\infty} x(t) = \frac{(\mathbf{1}^T x_0)\mathbf{1}}{N}$ also in this case
- Definition: a directed graph is called balanced if, for every vertex, the in-degree equals the out-degree
- Example



$$L = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & -1 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & 0 & -1 & 2 \end{bmatrix}$$

The Consensus Protocol

- For a balanced directed graph, it is $\mathbf{1}^T L = 0$ (in addition to $L\mathbf{1} = 0$)

- Thus, assuming existence of a rooted out-branching (as before), we have

$$\lim_{t \to \infty} x(t) = \frac{(\mathbf{1}^T x_0)\mathbf{1}}{N}$$

analogously to the undirected graph case

The Consensus Protocol

- To conclude, we draw some additional remarks on the consensus machinery

- It is straightforward to modify the consensus protocol in order to take into account suitable gains

$$u_i = k_i(t) \sum_{j \in \mathcal{N}_i} (x_j - x_i)$$

with $k_i(t) > 0$

- It is possible to generalize to a stochastic settings (agreement over Markov chains)

- It is possible to consider time-varying topologies for the graph $\mathcal{G}$

- In this case, $\mathcal{G} = (\mathcal{V}, \mathcal{E}(t))$ and $u_i = \sum_{j \in \mathcal{N}_i(t)} (x_j - x_i)$

- This case is highly relevant whenever ability to establish an edge depends on the state of the robots (e.g., maximum range for communication or occlusion of visibility)

The Consensus Protocol

- Considering a time-varying topology induces a time-varying closed-loop linear system $\dot{x} = -L(t)x$, in particular may lead to a switching dynamics

- One can still prove convergence given some looser properties of the underlying graph structure (~ the graph maintains some form of global connectivity across the switchings)

- It is possible to consider more complex linear or nonlinear dynamics in place of $\dot{x} = u_i$
  - for example second-order systems, general Lagrangian (mechanical) systems
  - but also unicycle-like (nonholonomic)

- It is possible to consider time delays and/or asynchronous communication in the information exchange (along edges)

- The "consensus paradigm" has given rise to a large number of variants
  - one example: decentralized estimation of exogenous time-varying quantities (**PI-ACE** – proportional/integral average consensus estimator)

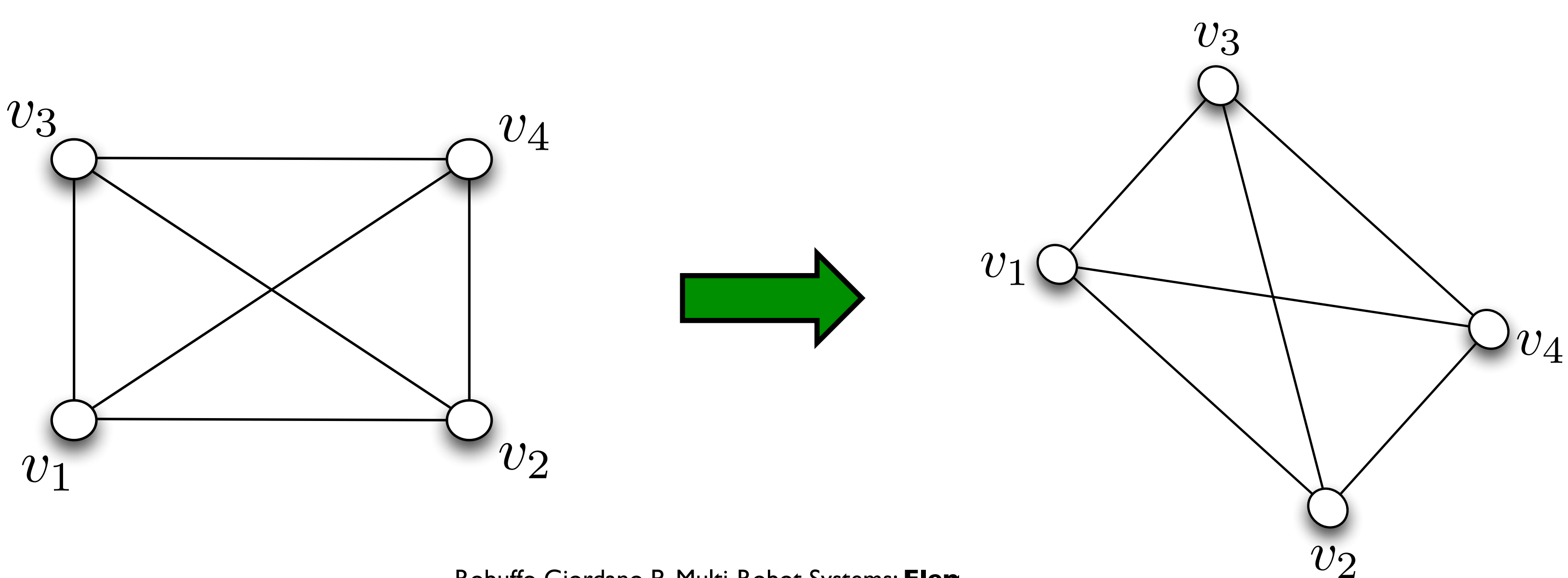
GRAPH RIGIDITY

Rigidity of Structures

- Given N agents and $M \leq N(N-1)/2$ pair-wise geometrical constraints (edges), do the constraints univocally determine the shape (spatial arrangement) of the agents ?

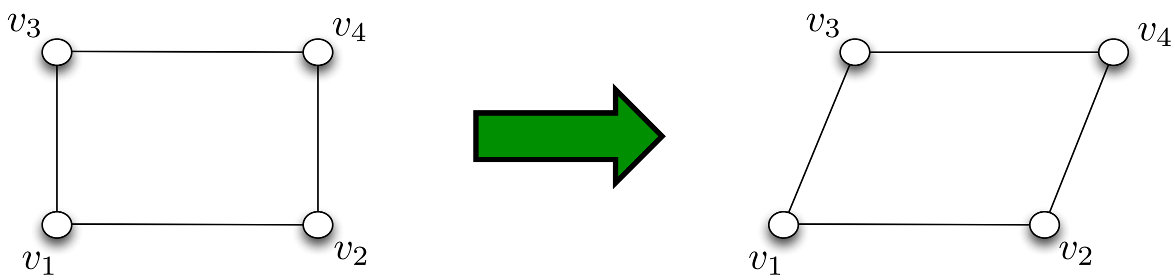
- Consider the case of distance constraints for planar agents: each edge in the graph imposes a desired distance to the incident pair

- If $M = N(N-1)/2$ (complete graph), then the shape is univocally determined (up to a rototranslation on the plane). The agents behave as a planar rigid body

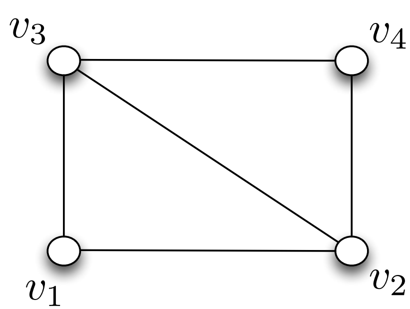


Rigidity of Structures

- If $M < N(N-1)/2$ (not the complete graph) the situation is less clear



- With these 4 edges the shape is not preserved: multiple non-congruent realizations meeting the 4 pair-wise distance constraints



- With these 5 edges the shape is instead preserved up to a rototranslation on the plane

Rigidity of Structures

- Graph rigidity: how to characterize the "flexibility" of multi-agents bound to pair-wise geometric constraints

- Needed tools: graph theory + geometry + linear algebra

- Loosely speaking: a "framework" (graph + agent poses) is rigid if the only allowed motions satisfying the constraints are those of the complete graph

- Complete graph: $N(N-1)/2$ edges, and thus need to measure/control/enforce $N(N-1)/2$ constraints (the complexity is $O(N^2)$)

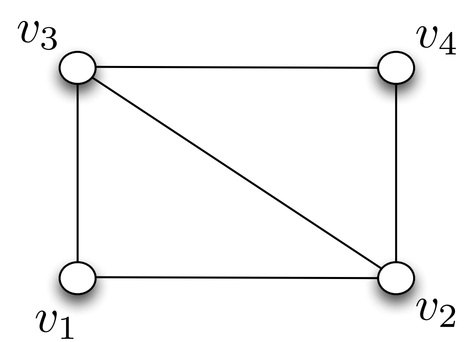
- However, framework rigidity is often possible with only a $O(N)$ set of constraints: in the previous case a minimum of $2N-3$ (properly placed) edges would be sufficient

- Comparison:

	$N(N-1)/2$	$2N-3$
$N=3$	3	3
$N=4$	6	5
$N=10$	45	17

Why rigidity matters

- If a framework is rigid then:
  - Formation control can be solved by regulating the pair-wise geometrical constraints to their desired values
    - Each agent pair controls the value of its own constraint (e.g., the distance)
    - This is enough for ensuring that the desired global shape is realized
    - And... no need to control all the possible pair-wise constraints (i.e., no need of a complete graph)
  - Relative localization can be univocally solved from the measured value of the constrains
    - Only one solution for the formation shape consistent with the pair-wise geometric constraints
    - Each agent can only be at one specific location (w.r.t. a frame attached with the formation)



Definitions

- Bar-and-joint framework: let $\mathcal{G} = (\mathcal{V},\, \mathcal{E})$ be a graph and $p : \mathcal{V} \to \mathbb{R}^d$ a function mapping each vertex to a point in $\mathbb{R}^d$

- Just the usual graph structure + a "position" associated to each node

- One could also consider mappings to full "poses" $p : \mathcal{V} \to SE(d)$

- For each edge $(i,\, j) \in \mathcal{E}$ consider a constraint function $g_{ij}(p_i, p_j)$

- In most (but not all) cases, the constraint only depends on the relative positions/poses $g_{ij}(p_i - p_j)$

- Example: in case of distances, one can take $g_{ij}(p_i - p_j) = \|p_i - p_j\|^2$

- Let then $g_{\mathcal{G}} = \{\dots g_{ij} \dots\} : \mathbb{R}^{Nd} \to \mathbb{R}^{|\mathcal{E}|}$ be the cumulative constraint function over all the edges in $\mathcal{G}$

Definitions

- A framework is rigid (w.r.t. the chosen constraint function) if there exists a neighborhood $\mathcal{U} \subset \mathbb{R}^{Nd}$ of p such that

$$\boxed{g_{\mathcal{G}}^{-1}(g_{\mathcal{G}}(p)) \cap \mathcal{U} = g_K^{-1}(g_K(p)) \cap \mathcal{U}}$$

where K_N is the complete graph

- In short: a framework $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is rigid if the only allowed motions preserving the constraints are those of the complete graph
  - i.e., removing some edges w.r.t. K_N "does not matter" for maintaining the shape
  - the value of the constraints for the "missing edges" w.r.t. K_N is univocally determined (and it is what one would have had with $\mathcal{G} = K_N$)

- Framework equivalency: two frameworks $(\mathcal{G}, p_1)$ and $(\mathcal{G}, p_2)$ are equivalent if $g_{\mathcal{G}}(p_1) = g_{\mathcal{G}}(p_2)$ (the constraints are satisfied over all the edges in $\mathcal{E}$)

- Framework congruency: two frameworks $(\mathcal{G}, p_1)$ and $(\mathcal{G}, p_2)$ are congruent if $g_K(p_1) = g_K(p_2)$ (the constraints are satisfied over all the possible edges)

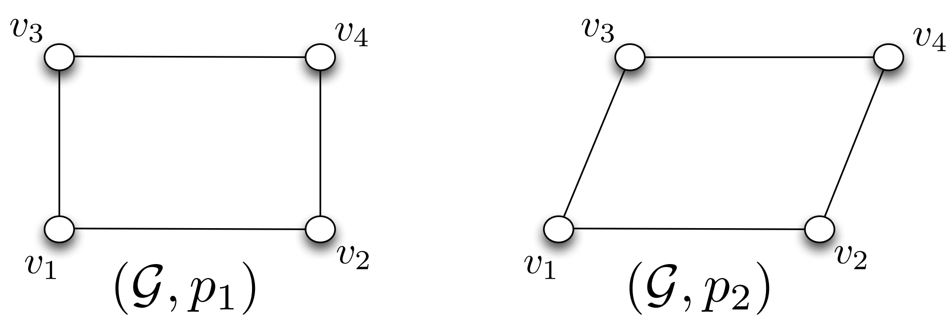
Definitions

- Alternative definition of rigidity: a framework $(\mathcal{G}, p_1)$ is rigid if all the frameworks $(\mathcal{G}, p_2)$, $p_2 \in \mathcal{U}(p_1)$, which are equivalent to $(\mathcal{G}, p_1)$ are also congruent to $(\mathcal{G}, p_1)$

- In all the above, a framework is globally rigid if $\mathcal{U} = \mathbb{R}^{Nd}$

- Finally, a framework is minimally rigid if the removal of any edge yields a non-rigid framework

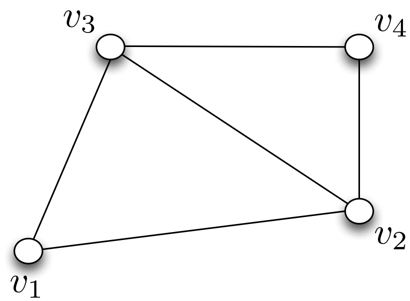
- Now some examples:



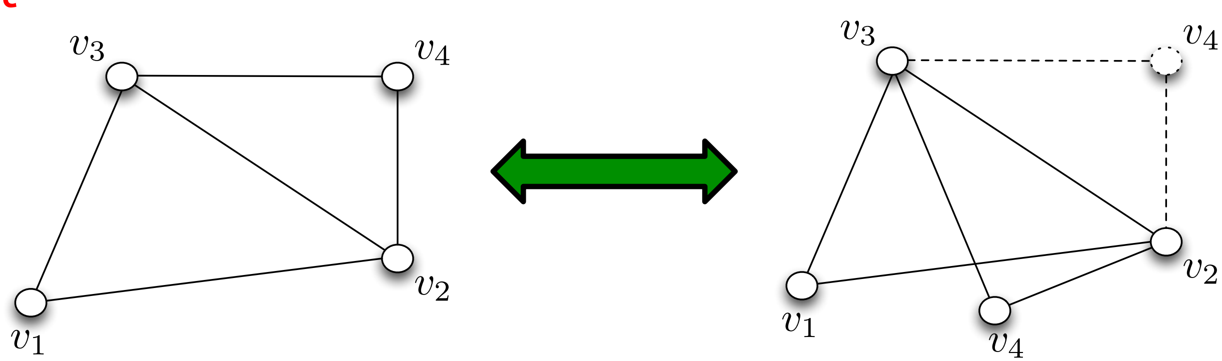
$(\mathcal{G}, p_1)$ is not rigid because one can find a framework $(\mathcal{G}, p_2)$ which is equivalent but not congruent: the constraints are met over the edges of $\mathcal{G}$ but not over all the possible edges in K_N

Definitions

- This framework is minimally rigid: by removing any edge, one gets a non-rigid framework



- However, the framework is not globally rigid: these two frameworks are equivalent but not congruent



- Note that no "smooth" motions could take the first framework to the second one. Indeed the two frameworks are (locally) rigid. Two "isolated" solutions exist for the given distance constraints

Infinitesimal Rigidity

- Infinitesimal rigidity: study the flexibility of a framework under instantaneous motions of its nodes

- Assume a smooth time dependence $p = p(t)$: what are the instantaneous motions of $p(t)$ which preserve the constraints $g_{\mathcal{G}}(p(t)) = const$?

- $g_{\mathcal{G}}(p(t)) = const \implies \dot{g}_{\mathcal{G}}(p(t)) = 0$ and using the chain rule

$$\boxed{\dot{g}_{\mathcal{G}}(p(t)) = 0 \implies \frac{\partial g_{\mathcal{G}}(p)}{\partial p}\dot{p} = R_{\mathcal{G}}(p)\dot{p} = 0}$$

- Matrix $\boxed{R_{\mathcal{G}}(p) \in \mathbb{R}^{|\mathcal{E}| \times Nd}}$ is known as the rigidity matrix

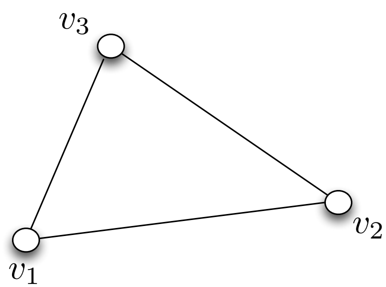
- The infinitesimal motions consistent with the constraints are then $\dot{p} \in \ker(R_{\mathcal{G}}(p))$

- A framework is infinitesimally rigid if $\ker(R_{\mathcal{G}}(p)) = \ker(R_K(p))$ or, equivalently, $\mathrm{rank}(R_{\mathcal{G}}(p)) = \mathrm{rank}(R_K(p))$
- Usual definition involving the complete graph K_N

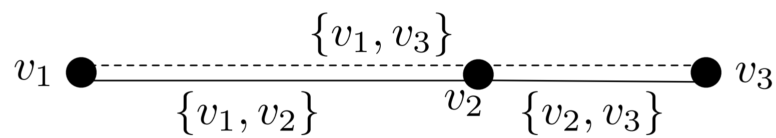
Infinitesimal Rigidity

- Infinitesimal rigidity implies rigidity, but the converse is not always true

- Indeed, the rigidity matrix can lose rank because of "non-generic" agent positions that involve special alignments



$(\mathcal{G}, p_1)$



$(\mathcal{G}, p_2)$

- $(\mathcal{G}, p_1)$ is infinitesimally rigid, and therefore rigid. However, $(\mathcal{G}, p_2)$ is not infinitesimally rigid, but it is rigid (same set of constraints over the edges)
  - the problem is the alignment of agents $v_1,\ v_2,\ v_3$ which causes the rigidity matrix to (point-wise) lose rank. Any perturbation of this alignment would allow to regain infinitesimal rigidity

- A point $\bar{p}$ is a regular point if $\mathrm{rank}(R_{\mathcal{G}}(\bar{p})) = \max_{p}(\mathrm{rank}(R_{\mathcal{G}}(p)))$
  - Infinitesimal rigidity = rigidity + $\bar{p}$ is a regular point (~ no special alignments)

Infinitesimal Rigidity

- The Rigidity matrix is a fundamental tool for control and estimation purposes
  - It establishes a link between agent motion and constraint variations
  - Its null-space $\ker(R_{\mathcal{G}}(p))$ describes all the motions preserving the constraints
  - Rigidity of a framework is equivalent to a rank condition on $R_{\mathcal{G}}(p)$. This allows to exploit spectral tools (e.g., eigenvalues, singular values) for checking or enforcing rigidity)

- The rank condition allows to also determine the minimum number of edges in a graph $\mathcal{G}$ for being rigid

- Let $\mathrm{rank}(R_{K_N}(p)) = r < Nd$. A framework is rigid if $\mathrm{rank}(R_{\mathcal{G}}(p)) = \mathrm{rank}(R_{K_N}(p))$

- Since $R_{\mathcal{G}}(p) \in \mathbb{R}^{|\mathcal{E}| \times Nd}$, this implies presence of at least $|\mathcal{E}| = r$ in the edge set of $\mathcal{G}$
  - However, not any collection of $|\mathcal{E}| = r$ edges would be good ! One needs the "right ones"

Infinitesimal Rigidity

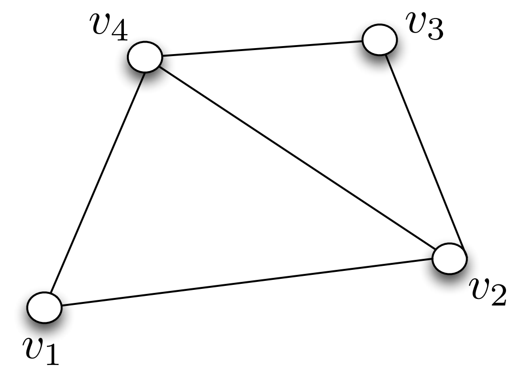
- For distance constraints in $\mathbb{R}^2$ the complete graph allows 3 collective motions: 2 translations on the plane + 1 rotation (those of a rigid body on the plane)

- Therefore, for a rigid graph, $\dim\ker(R_{\mathcal{G}}(p)) = 3$ and $\mathrm{rank}(R_{\mathcal{G}}(p)) = 2N - 3$

- One needs at least $2N - 3$ edges (connecting the "correct" agent pairs)
    - Note the linearity w.r.t. N (instead of $O(N^2)$ as in the complete graph)

- Similar arguments hold for embeddings in $\mathbb{R}^3$, $SE(2)$ and $SE(3)$

Infinitesimal Rigidity



- Let us consider this graph
- What is the associated rigidity matrix ?
- Start with the constraint function $g(p) = \begin{bmatrix} \|p_1 - p_2\|^2 \\ \|p_1 - p_4\|^2 \\ \|p_2 - p_3\|^2 \\ \|p_2 - p_4\|^2 \\ \|p_3 - p_4\|^2 \end{bmatrix}$
- Being $R_{\mathcal{G}}(p) = \dfrac{\partial g_{\mathcal{G}}(p)}{\partial p}$ one obtains

$$R_{\mathcal{G}}(p) = \boxed{\begin{bmatrix} p_1^T - p_2^T & p_2^T - p_1^T & 0 & 0 \\ p_1^T - p_4^T & 0 & 0 & p_4^T - p_1^T \\ 0 & p_2^T - p_3^T & p_3^T - p_2^T & 0 \\ 0 & p_2^T - p_4^T & 0 & p_4^T - p_2^T \\ 0 & 0 & p_3^T - p_4^T & p_4^T - p_3^T \end{bmatrix}}$$

Infinitesimal Rigidity

- At generic positions (i.e., without "special" alignments), one has $\text{rank}(R_{\mathcal{G}}(p)) = 5 = 2N - 3$ (the framework is rigid)

- What is a basis for the (3-dimensional) $\ker(R_{\mathcal{G}}(p))$?

- Two vectors can be identified as $n_{1,2} = \mathbf{1}_n \otimes I_2$: these represent the two planar translations along the x and y directions

- A third vector can be identified as $n_3 = (I_N \otimes S)(p - p^*)$ with $S = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ and p^* an arbitrary point on the plane

- This represents a collective rotation around the "pivot point" p^*

- Proof: the k-th element of $R_{\mathcal{G}}(p)n_{1,2}$ is just $(p_i^T - p_j^T) - (p_i^T - p_j^T) = 0$

- the k-th element of $R_{\mathcal{G}}(p)n_3$ is $(p_i^T - p_j^T)S(p_i - p^*) - (p_i^T - p_j^T)S(p_j - p^*) =$ $= p_j^T S p_i + p_i^T S p_j = 0$ since $S = -S^T$

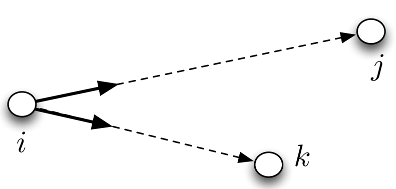
Infinitesimal Rigidity

- Note: any other linear combination of the three vectors $(n_1,\, n_2,\, n_3)$ would also be a valid solution for $\ker(R_{\mathcal{G}}(p))$
- However, the set $(n_1,\, n_2,\, n_3)$ has a clear geometrical interpretation
  - by, e.g., setting $\dot{p} = \alpha_1 n_1 + \alpha_2 n_2 + \alpha_3 n_3$ one could steer the whole formation by individually actuating the three dofs: 2D translation and rotation around p^*

- By embedding in $\mathbb{R}^3$ one obtains $\dim \ker(R_{\mathcal{G}}(p)) = 6$ for a rigid graph
  - The constraint-preserving motions are the 3 translations and 3 rotations around an arbitrary p^* (the motions of a rigid body in 3D space)

- Note: so far we have dealt with distance constraints. However, another very popular application of rigidity theory is in the case of bearing constraints

- Bearing vector: unit vector (direction) from one agent to another
  - Interesting because it is what can be measured from, e.g., perspective cameras



Bearing Rigidity

- Bearing constraint: keep a desired bearing vector (i.e., a "set of angles") w.r.t. neighboring pairs
  - Note: the distance constraint is a scalar constraint in any dimension
  - The bearing constraint is a $(n-1)$-dimensional constraint in $\mathbb{R}^n$ (more stringent constraint)

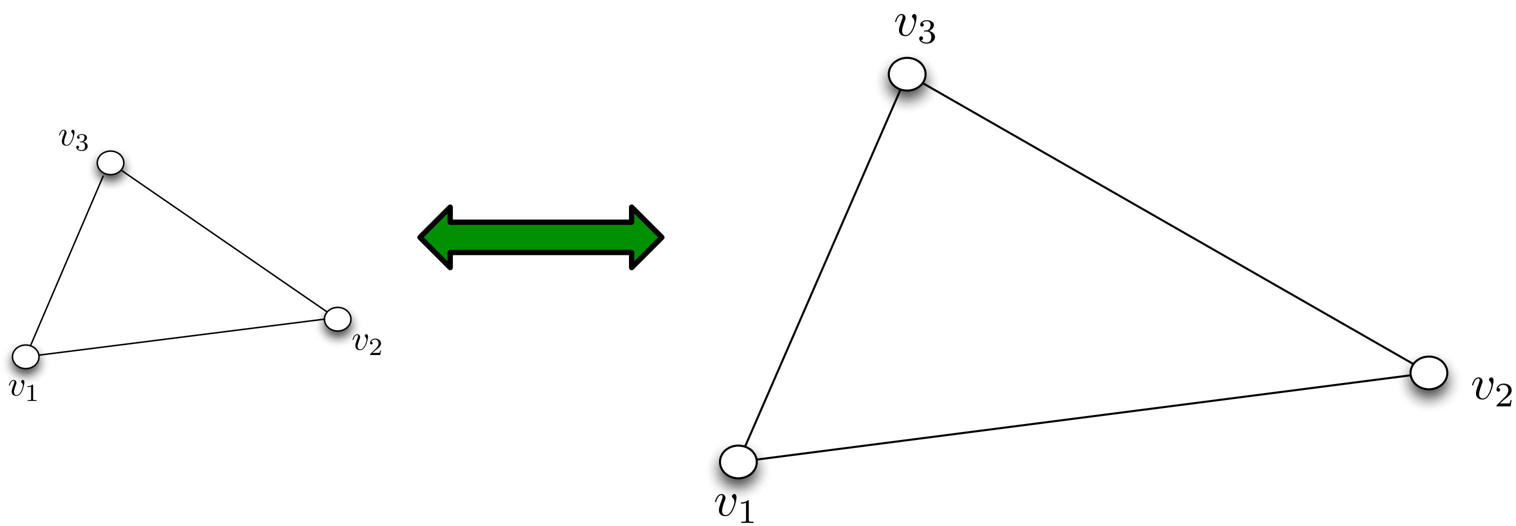
- Examples of relative bearings:
  - Absolute bearing: $\beta_{ij} = \dfrac{p_j - p_i}{\|p_j - p_i\|} \in \mathbb{S}^{n-1}$ pointing vector expressed in a common frame
  - Body-frame bearing: $\beta_{ij} = R^i \dfrac{p_j - p_i}{\|p_j - p_i\|} \in \mathbb{S}^{n-1}$ pointing vector expressed in the local frame of agent i

- The analysis becomes slightly more complex than for the distance case. However, the same general reasoning applies

Bearing Rigidity

- For instance, in case of absolute bearings $\beta_{ij} = \dfrac{p_j - p_i}{\|p_j - p_i\|} \in \mathbb{S}^{n-1}$ one talks about "parallel rigidity"

- The only allowed motions of the complete graph K_N on the plane are the usual 2D translations and an expansion/retraction (but no rotation!)



- Thus, in $\mathbb{R}^2$ one has $\mathrm{rank}(R_{\mathcal{G}}(p)) = 2N - 3$. Same rank as for the previous distance constraints, but different kernel !! (in particular, n_3 is different)

- When dealing with bearing constraints, the scale is never fixed. Not surprising since we are constraining "relative angles" between pairs of agents

Rigidity-based control and localization

- We will quickly review why rigidity (and, in particular, the rigidity matrix) are important for formation control and localization

- We will only consider the case of distance constraints (however, similar ideas apply, mutatis mutandis, for the bearing case)

- Assume that we want to stabilize the pose $p \in \mathbb{R}^{Nd}$ of N agents to a pose congruent with a desired p_d
  - in other words, we only care about the final shape, and not of where the shape will be placed on the plane

- Neighboring agent pairs can only sense the constraint value $g_{ij}(p)$ (i.e., they can only measure their relative distance)

- Let $g_d = g_{\mathcal{G}}(p_d)$ be the constraint value at the desired pose: find a feedback controller which zeros the "constraint error" $g_d - g_{\mathcal{G}}(p)$

- If the framework is rigid, we are guaranteed that $g_d = g_{\mathcal{G}}(p)$ implies congruency with the desired p_d

Rigidity-based control and localization

- Define the usual scalar error function $e = \frac{1}{2}\|g_d - g_{\mathcal{G}}(p)\|^2 = \frac{1}{2}\sum_{(i,j)\in\mathcal{E}} (g_{ij}(p_d) - g_{ij}(p))$
  - sum over the edges of the squared constraint violations
- What is $\frac{\partial e}{\partial p}$? $\frac{\partial e}{\partial p} = -(g_d^T - g_{\mathcal{G}}^T(p))\frac{\partial g_{\mathcal{G}}(p)}{\partial p} = -(g_d^T - g_{\mathcal{G}}^T(p))\boxed{R_{\mathcal{G}}(p)}$
- The error function can be minimized by following its negative gradient, i.e.,

$$\dot{p} = R_{\mathcal{G}}^T(p)(g_d - g_{\mathcal{G}}(p)) \quad (\blacksquare)$$

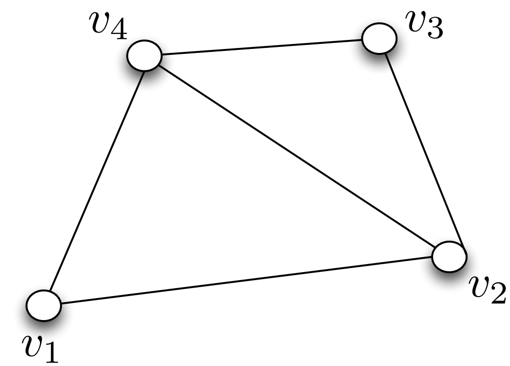
- This is a nice result because (■) is inherently decentralized. This is because of the decentralized structure of the rigidity matrix $R_{\mathcal{G}}(p)$
- Indeed, the explicit expression of (■) for the i-th agent is $\dot{p}_i = -\sum_{j\in\mathcal{N}_i} (\|e_{ij}\|^2 - d_{ij}^2)e_{ij}$ where $e_{ij} = p_j - p_i$ and $d_{ij} = \|p_{j,d} - p_{i,d}\|$
- Additional feature: the centroid $p^o = \frac{1}{N}\sum p_i$ is invariant under (■), i.e., $\boxed{\dot{p}^o = 0}$

Rigidity-based control and localization

- Recall the example

$$
R_{\mathcal{G}}(p) = \begin{array}{c} \\ \left[\begin{array}{cccc} p_1^T - p_2^T & p_2^T - p_1^T & 0 & 0 \\ p_1^T - p_4^T & 0 & 0 & p_4^T - p_1^T \\ 0 & p_2^T - p_3^T & p_3^T - p_2^T & 0 \\ 0 & p_2^T - p_4^T & 0 & p_4^T - p_2^T \\ 0 & 0 & p_3^T - p_4^T & p_4^T - p_3^T \end{array} \right] \end{array}
$$

Agent 3



- The i-th column of $R_{\mathcal{G}}(p)$ (associated to agent i) only depends on p_i and $p_j,\ j \in \mathcal{N}_i$
- The rigidity matrix has a decentralized structure
- Conceptually analogous results can be obtained for the bearing-rigidity case

Rigidity-based control and localization

- A similar reasoning can be applied to the (dual) localization problem

- Assume N agents can measure a set of relative distances according to some measurement graph $\mathcal{G}$

- Is it possible to univocally localize the agent positions from the measured distances ? Localize = find correct agent positions in some "common frame"

- Assume $(\mathcal{G},\, p)$ is a rigid framework and let $\hat{p}$ an estimation of the agent positions

- Because of the framework rigidity, if $\hat{p}$ agrees with the measurements, i.e., if $g_{\mathcal{G}}(\hat{p}) = g_{\mathcal{G}}(p)$, then $\hat{p}$ can only be a rigid rototranslation of the real p

- Therefore, $\hat{p}$ represents a correct localization of the agents in "some frame" (which can be different from the frame where p is expressed!). However:
    - all the agents will obtain an estimation of their position w.r.t. a unique common frame
    - and, this is achieved by only exploiting measured distances !

Rigidity-based control and localization

- The localization problem can be solved as before: define $e = \frac{1}{2}\|g_{\mathcal{G}}(p) - g_{\mathcal{G}}(\hat{p})\|^2$. Note that we now consider $p = const$ and minimize w.r.t. $\hat{p}$

- One has $\frac{\partial e}{\partial \hat{p}} = -(g_{\mathcal{G}}^T(p) - g_{\mathcal{G}}^T(\hat{p}))R_{\mathcal{G}}(\hat{p})$. Therefore, an update law for $\hat{p}$ is

$$\boxed{\dot{\hat{p}} = R_{\mathcal{G}}^T(\hat{p})(g_{\mathcal{G}}(p) - g_{\mathcal{G}}(\hat{p}))}$$

- As before, decentralized structure….

- It is also possible to enforce additional constraints on the estimated positions $\hat{p}$ for fixing the final roto-translation ambiguity

- For instance, one could add constraints to fix the origin of the underlying common frame by fixing the estimated position of one of the agents

- If, for instance, one sets $\hat{p}_1 = 0$, then all the remaining $\hat{p}_i$ will represent relative positions w.r.t. the position of agent 1
    - This essentially removes the translational ambiguity in $\ker(R_{\mathcal{G}})$

Rigidity-based control and localization

- Similarly, one could fix the orientation of the common frame by fixing the direction of one of its edges connecting two agents
- For instance, one can enforce $\hat{p}_1 - \hat{p}_k = p_1 - p_k$ with $k \in \mathcal{N}_1$. Force $\hat{p}_k$ to lie on the direction of the real $p_1 - p_k$
- This removes the last rotational ambiguity in $\ker(R_{\mathcal{G}})$

- All these constraints can be embedded in a single cost function

$$e = \frac{1}{2}\|g_{\mathcal{G}}(p) - g_{\mathcal{G}}(\hat{p})\|^2 + \frac{1}{2}\|\hat{p}_1\|^2 + \frac{1}{2}\|p_1 - p_k - (\hat{p}_1 - \hat{p}_k)\|^2$$

which leads to the update law

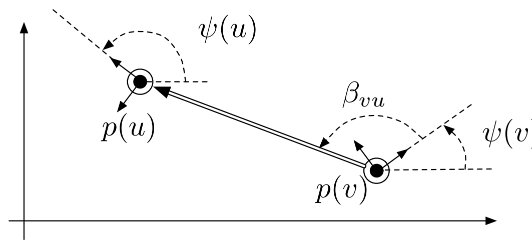
$$\dot{p}_i = -\sum_{j \in \mathcal{N}_i} (\|\hat{p}_i - \hat{p}_j\|^2 - d_{ij}^2)(\hat{p}_i - \hat{p}_j) - \delta_{i1}\hat{p}_1 - \delta_{ik}(\hat{p}_1 - \hat{p}_k - (p_i - p_k))$$

where δ_{ij} is the Kroenecker delta

- This law is, again, decentralized

Rigidity-based control and localization

- Consider the case of body-frame bearings $\beta_{ij} = R^i \dfrac{p_j - p_i}{\|p_j - p_i\|} \in \mathbb{S}^{n-1}$
  - Set of relative angles expressed in the local frame of sensing agents
  - What one can retrieve from onboard cameras



- We consider a planar problem in which the vertexes of graph $\mathcal{G}$ are mapped to a pose $(p_i,\, \psi_i) \in SE(2)$

- Each node consists of a position on the plane and an orientation w.r.t. some global frame
  - The configuration space $(p_1,\, \psi_1,\, \ldots p_N,\, \psi_N)$ has then dimension $3N$

- The associated bearing-rigidity matrix will then have dimensions $R_{\mathcal{G}} \in \mathbb{R}^{|\mathcal{E}| \times 3N}$

Bearing-based localization and control

- For the complete graph K_N there exist 4 allowed motions:
  - 2D translation
  - expansion/contraction
  - coordinated rotation about a pivot point p^*

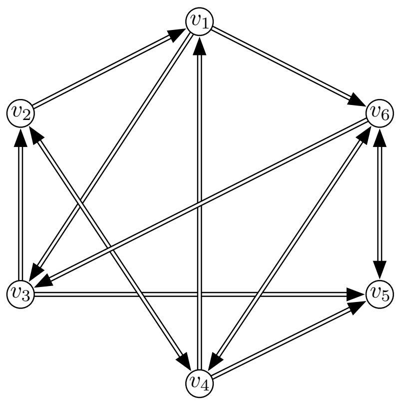
- Therefore, for an infinitesimally rigid framework one has $\mathrm{rank}(R_{\mathcal{G}}) = 3N - 4$
  - Need of at least $3N - 4$ edges in the edge set $\mathcal{E}$ for being bearing-rigid

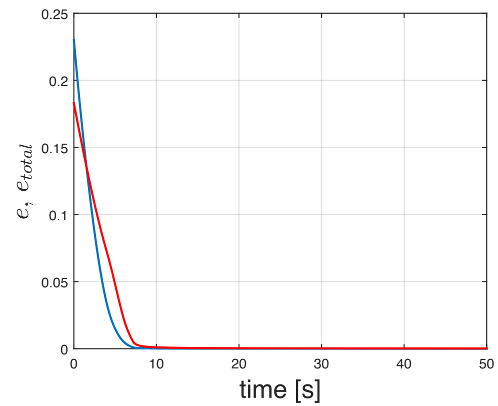
- Let us consider the case of formation control and of localization

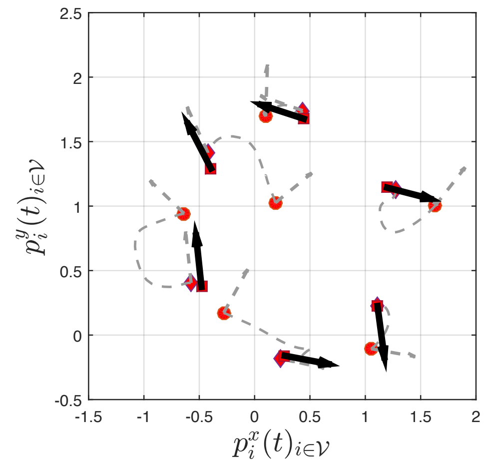
- Note that, because of the structure of $\ker(R_{\mathcal{G}})$ these two problems can only be solved up to a global roto-translation and scaling on the plane

Bearing-based localization and control

- Let us consider the case of bearing formation control

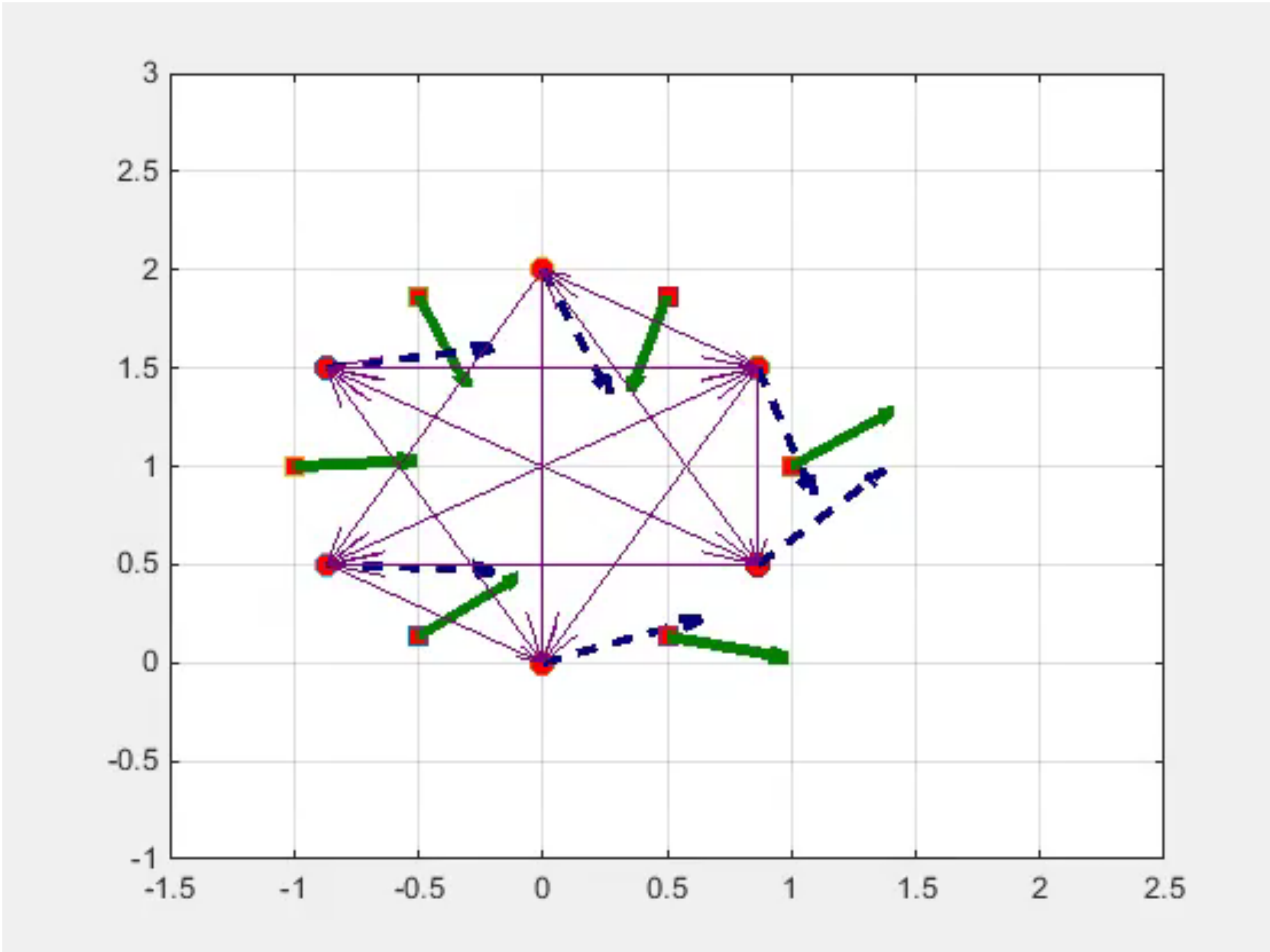






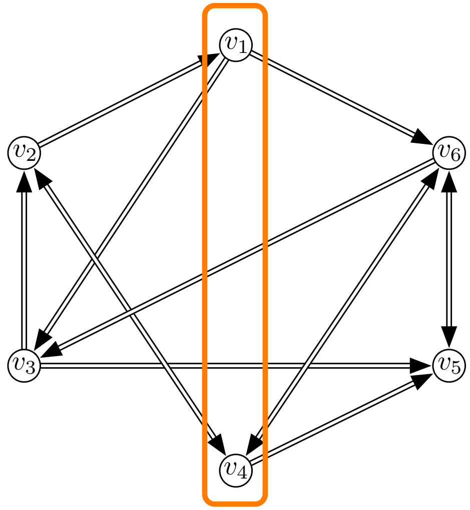
- $N = 6$ agents and $|\mathcal{E}| = 14$ edges (bearing measurements/constraints)
- The framework is minimally infinitesimally rigid
- The "usual" gradient controller steers the formation to a configuration congruent with the desired one
- Features of the controller: the centroid $\bar{p} = \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} p_i$ and "scale" $\bar{s}_p = \dfrac{1}{N}\sqrt{\displaystyle\sum_{i=1}^{N} \|p_i - \bar{p}\|^2}$ are invariant

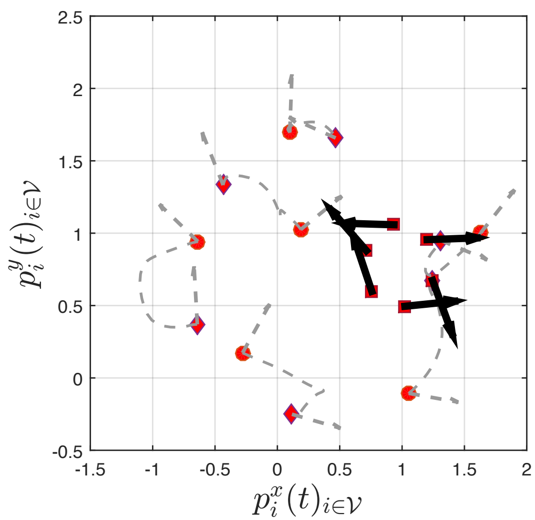
Bearing-based localization and control

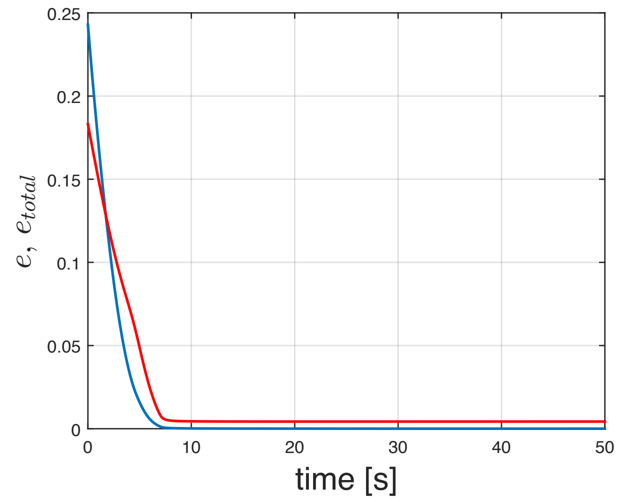


Bearing-based localization and control

- By removing one edge (edge $(4,\ 1)$) bearing rigidity is lost



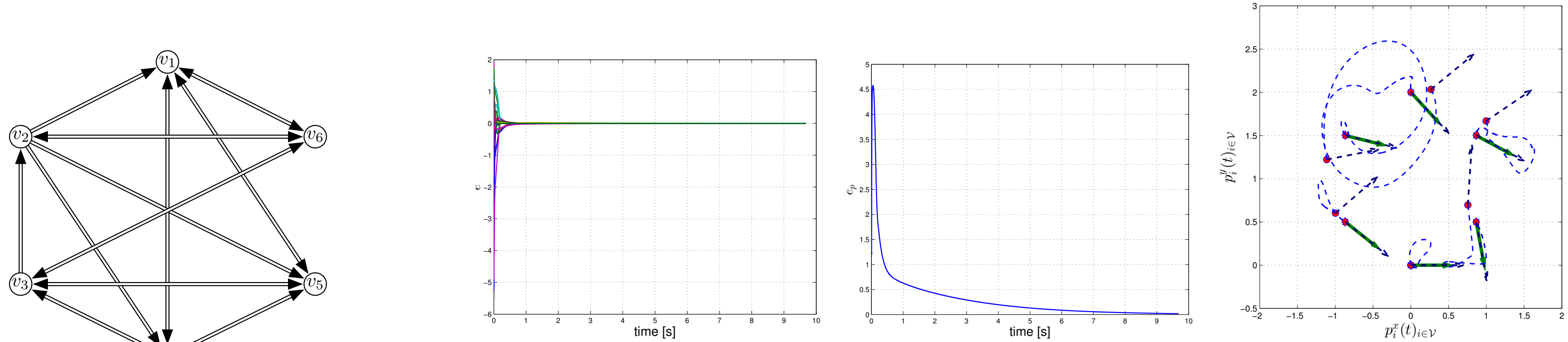




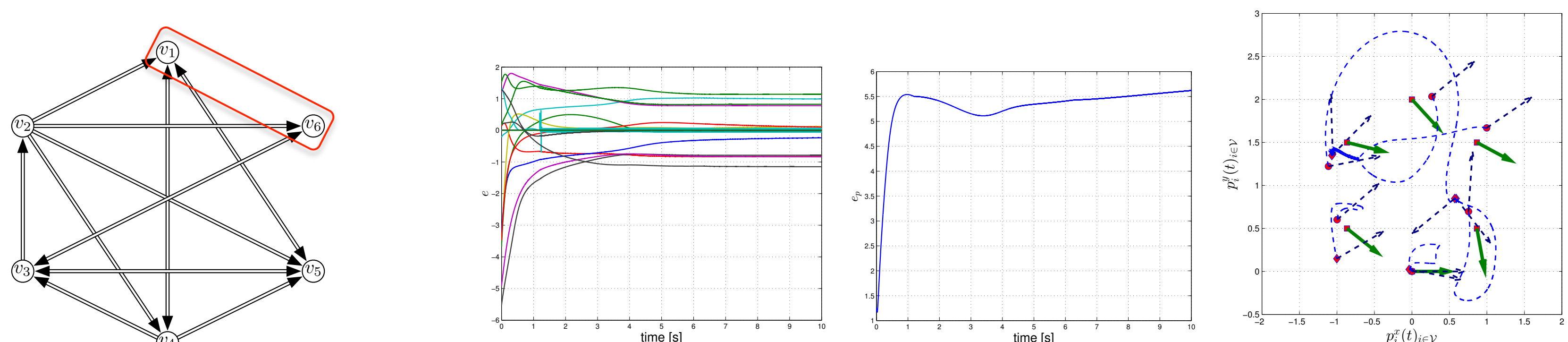
- The agents converge to a formation equivalent but not congruent with the desired on

An example of bearing-based localization

- Similar results for the localization case

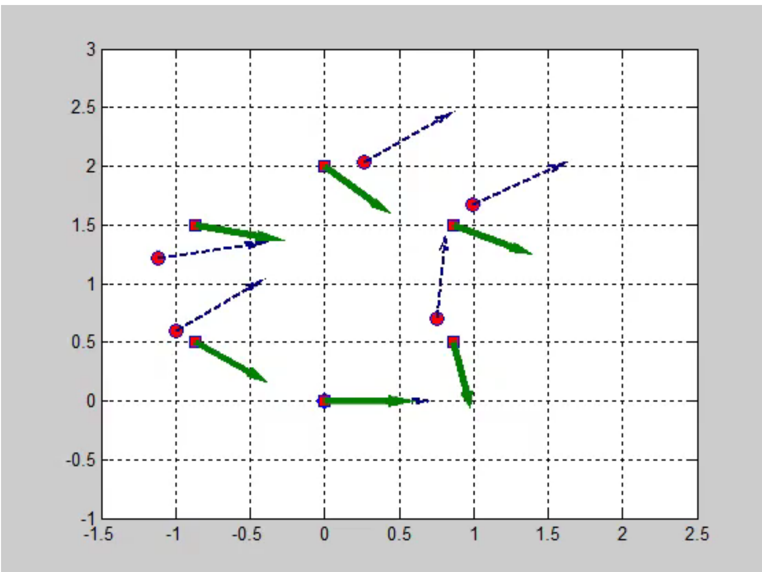


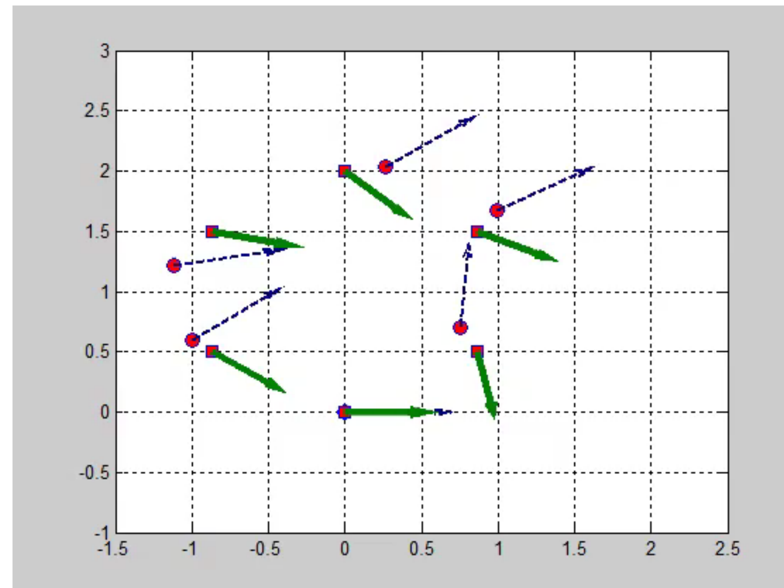
Rigid framework

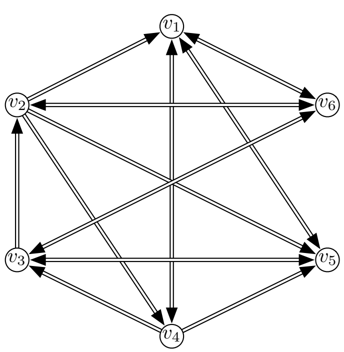


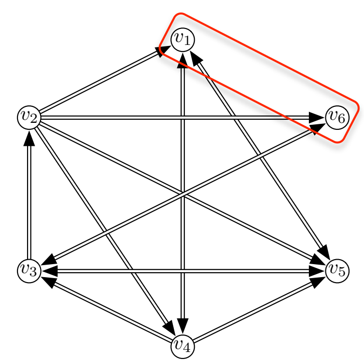
Non-rigid framework

An example of bearing-based localization







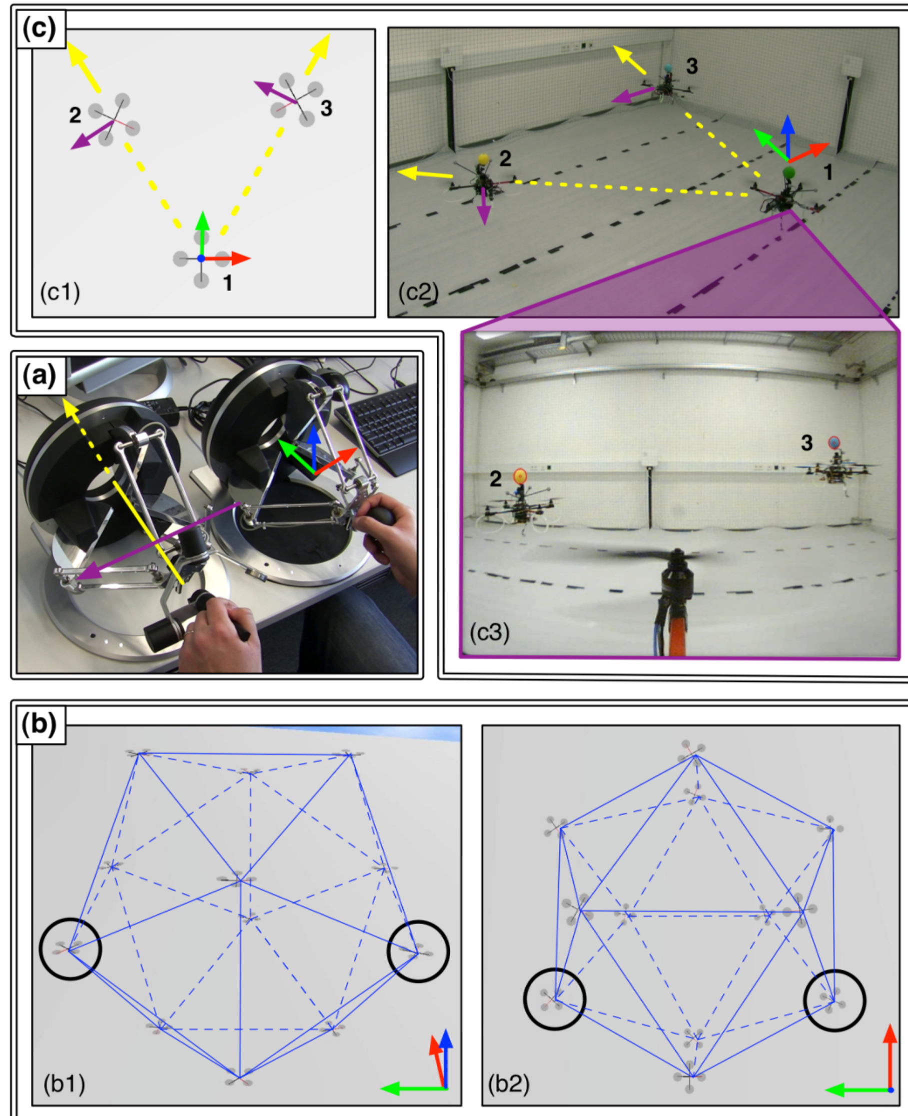


Bearing Formation Control for Quadrotors

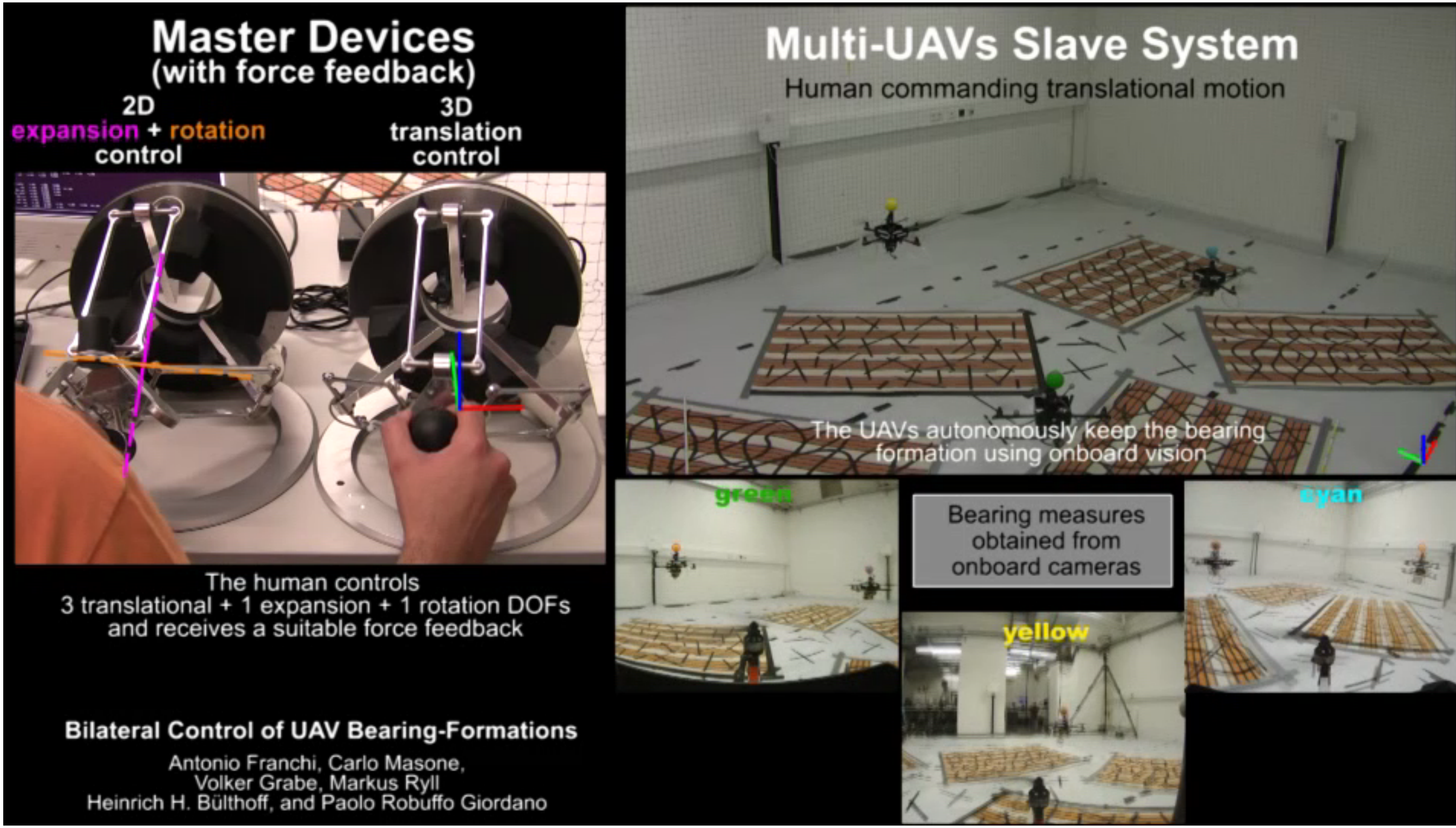
- Use relative bearings (unit vectors in 3D) for formation control
- Relative bearings can be directly retrieved from onboard cameras
- Lack of metric (distance) measurements

- The spatial formation is defined up to **5 dofs**:
  - Collective translation $\boldsymbol{\nu} \in \mathbb{R}^3$
  - Synchronized expansion $s \in \mathbb{R}$
  - Synchronized rotation $w \in \mathbb{R}$

- The human operator controls these **5 dofs** with **2 haptic devices**
  - Force feedback: mismatch between the desired and actual commands

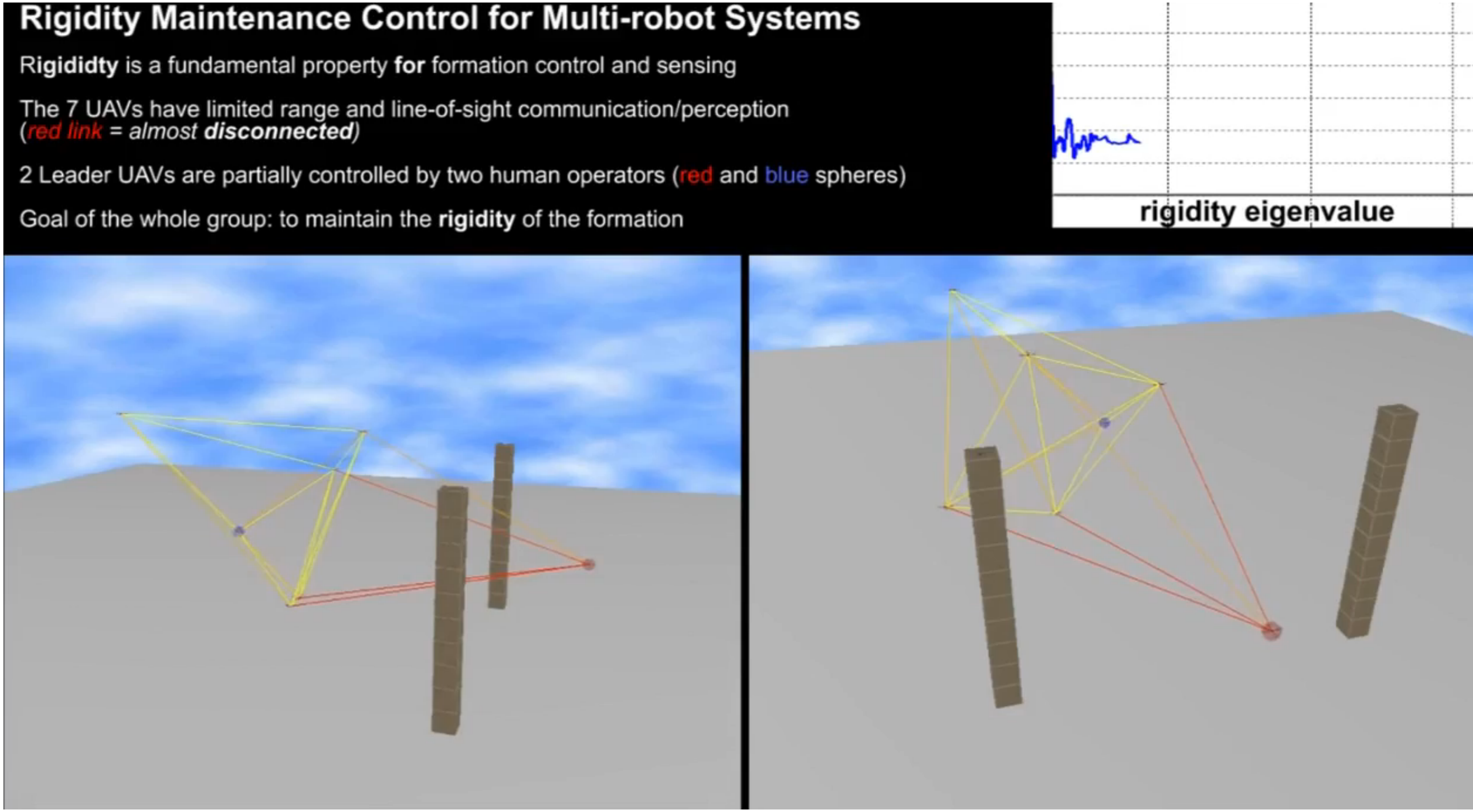


Bearing Formation Control for Quadrotors



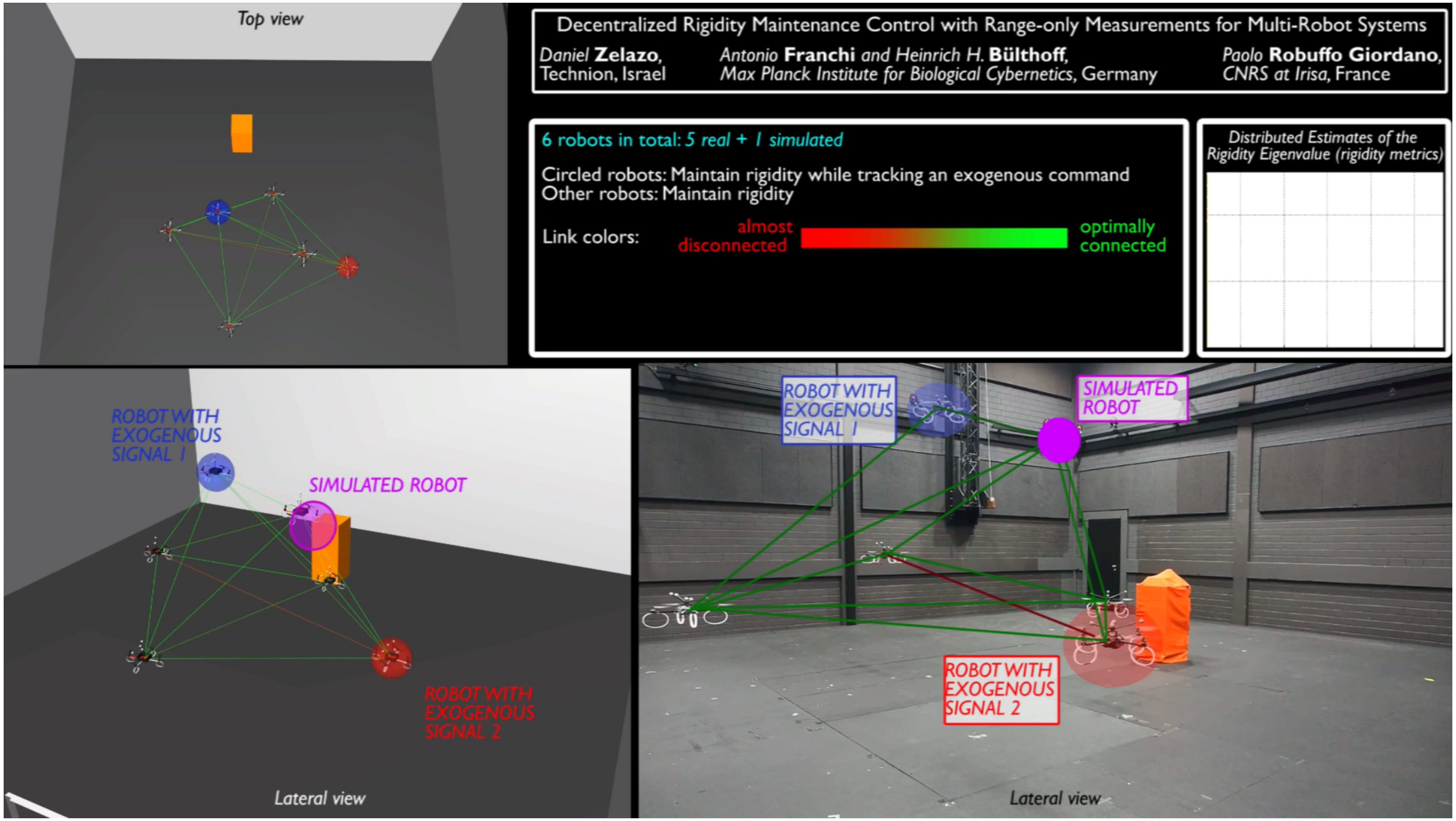
- The free dofs of a formation of UAVs are controlled by a human operator
- The instantaneous mismatch between commands (in terms of changes in formation shape) and actual motion becomes a force cue

Rigidity Maintenance with Distance Constraints



In collaboration with  D. Zelazo Technion, Isreal

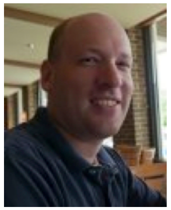
Rigidity Maintenance with Distance Constraints



IJRR 2014

- The quadrotors are maintaining formation rigidity
- This allows them to run a decentralized estimator able to obtain relative positions out of measured relative distances
- Relative positions are then needed by the rigidity controller

In collaboration with



D. Zelazo
Technion,
Isreal

Final remarks

- Many more extensions to the rigidity theory

- Here, only a sketch of the basics

- For instance:
    - How to maintain rigidity in a robust way (possibility to lose/regain links)
    - How to determine, in a decentralized way, whether a given framework is rigid
    - How to characterize rigidity in a pure combinatorial way (i.e., only looking at the graph $\mathcal{G}$)
    - How to characterize the stability and equilibria of the proposed control/localization schemes based on the Rigidity Matrix
    - How to grow rigid framework from a starting rigid framework
    - How to split a rigid framework into two rigid frameworks
    - How to join two rigid frameworks into a single rigid framework (with the minimum
    - And so on