Localization II

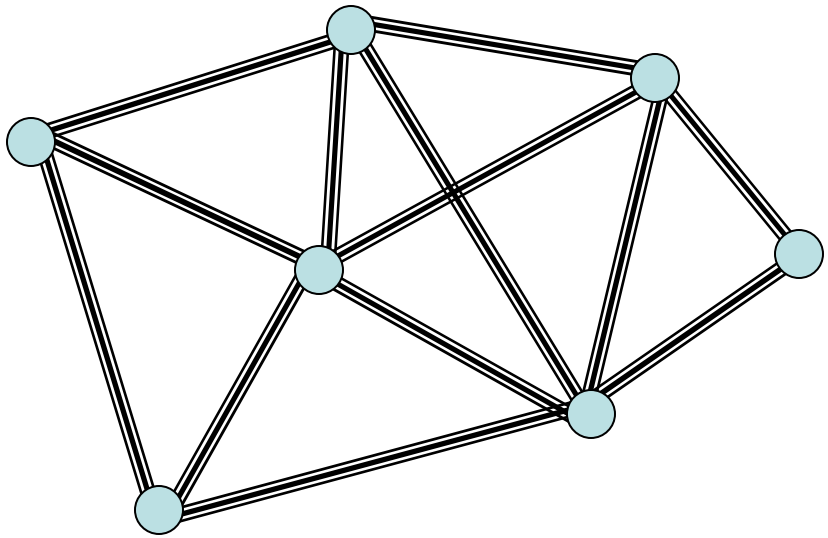
04/07/06

Improve the accumulated localization error by a global iterative algorithm ---

Mass-spring localization

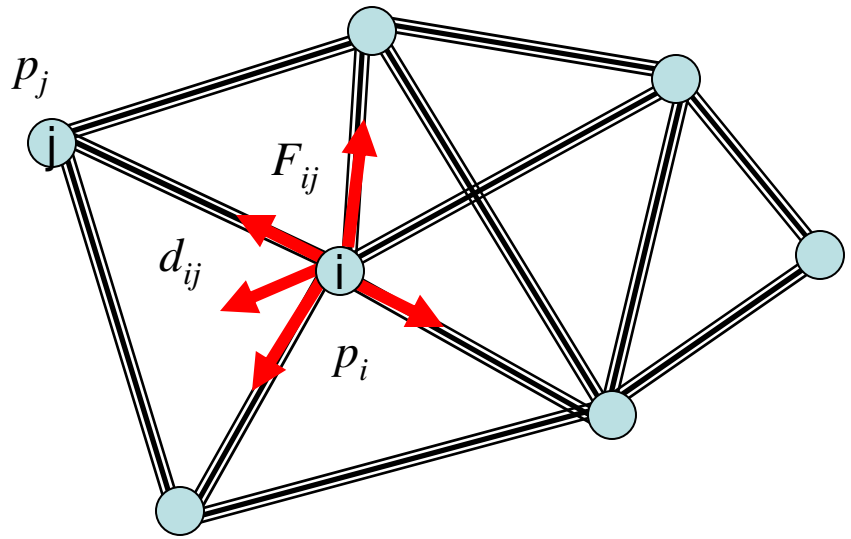
Mass-spring system

- Nodes are “masses”, edges are “springs”.
- Length of the spring equals the distance measurement.
- Springs put forces to the nodes.
- Nodes move.
- Until the system stabilizes.



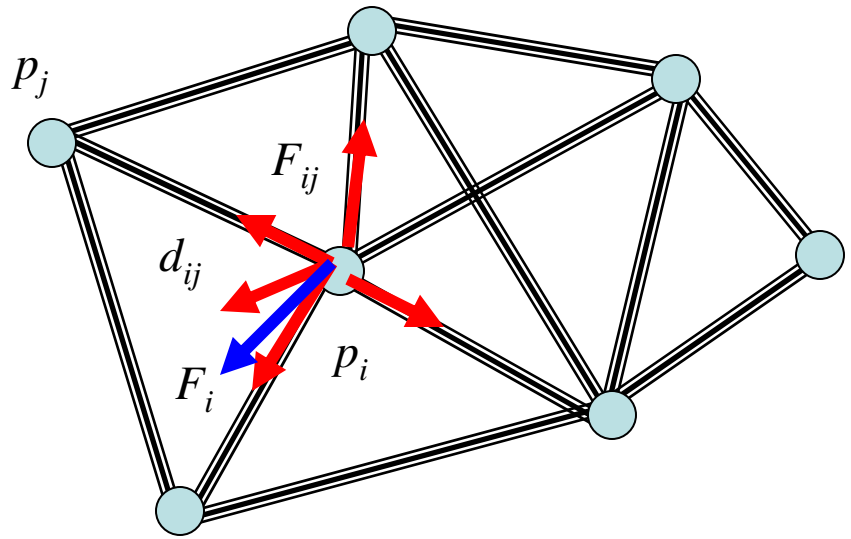
Mass-spring system

- Node n_i's current estimate of its position: p_i.
- The estimated distance d_{ij} between n_i and n_j.
- The measured distance r_{ij} between n_i and n_j.
- Force: $F_{ij} = d_{ij} - r_{ij}$, along the direction $p_i p_j$.



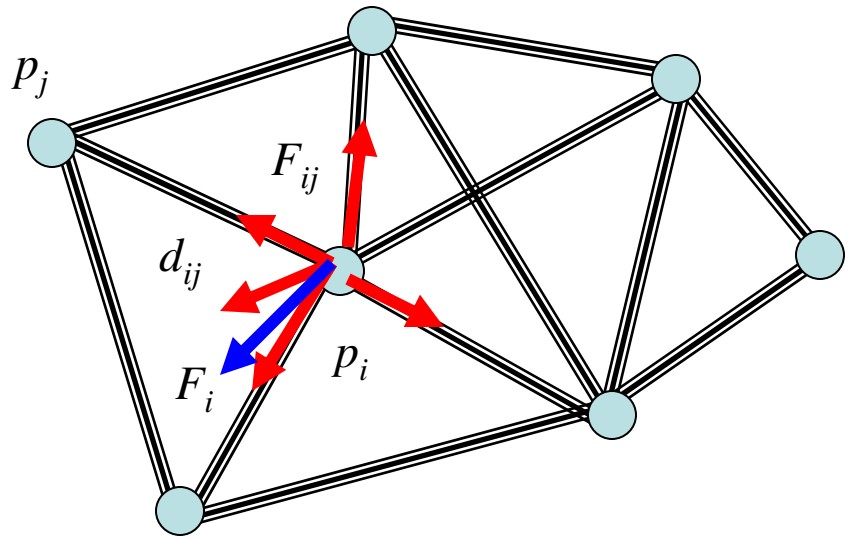
Mass-spring system

- Total force on n_i: $F_i = \Sigma\ F_{ij}$.
- Move the node n_i by a small distance (proportional to F_i).
- Recurse.



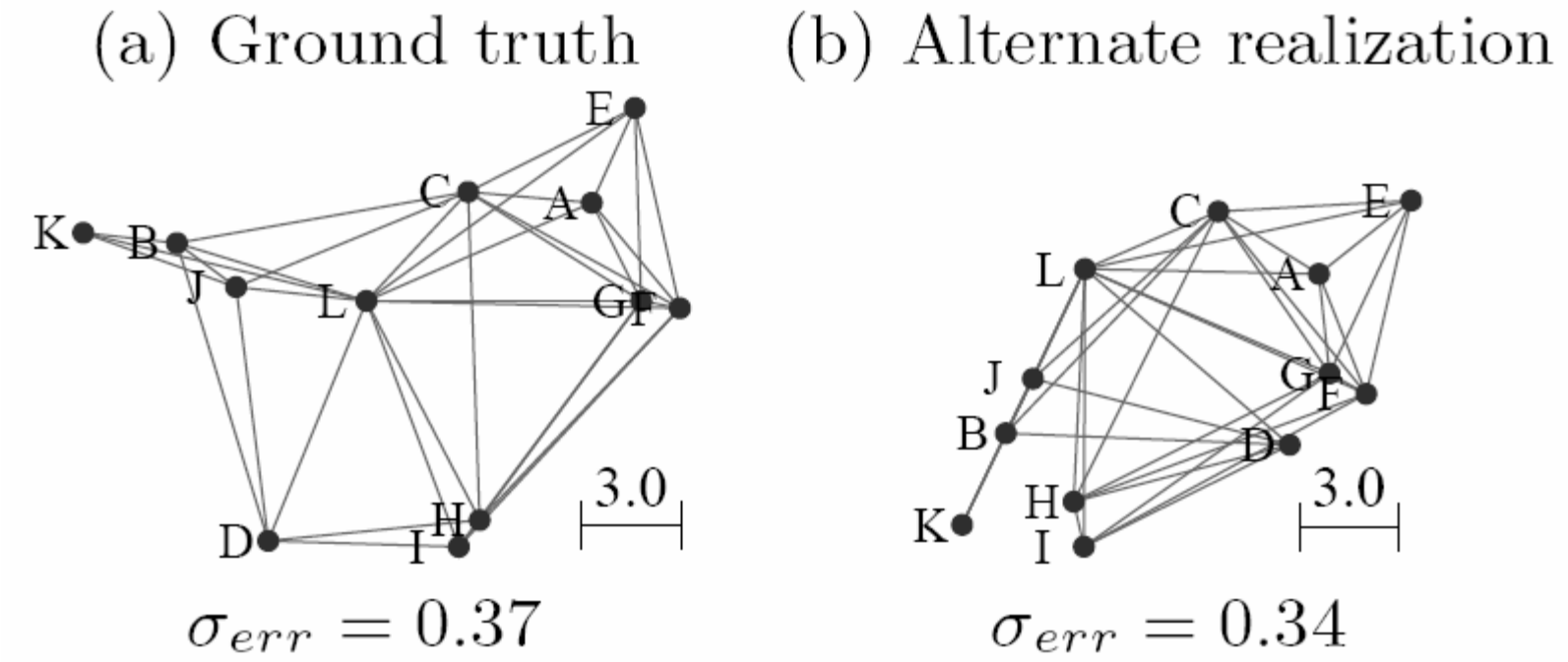
Mass-spring system

- Total energy n_i: $E_i = \Sigma\ E_{ij} = \Sigma\ (d_{ij} - r_{ij})^2$.
- Make sure that the total energy $E = \Sigma\ E_i$ goes down.
- Stop when the force (or total energy) is small enough.



Mass-spring system

- Naturally a distributed algorithm.
- Problem: may stuck in local minima.
- Need to start from a reasonably good initial estimation, e.g., the iterative multi-lateration.

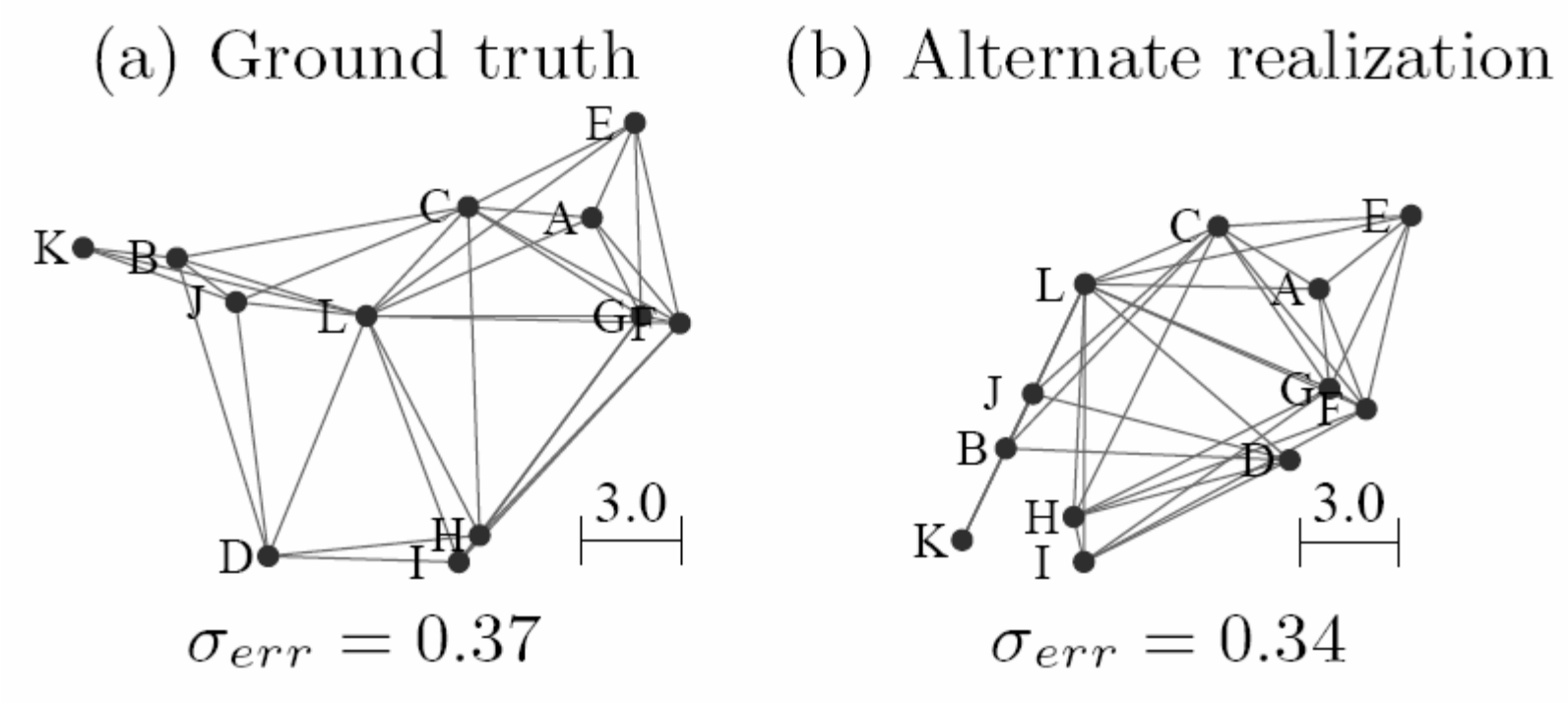


For noisy measurements, we use optimization methods…
Yet optimization does not solve ---

Ambiguity in localization

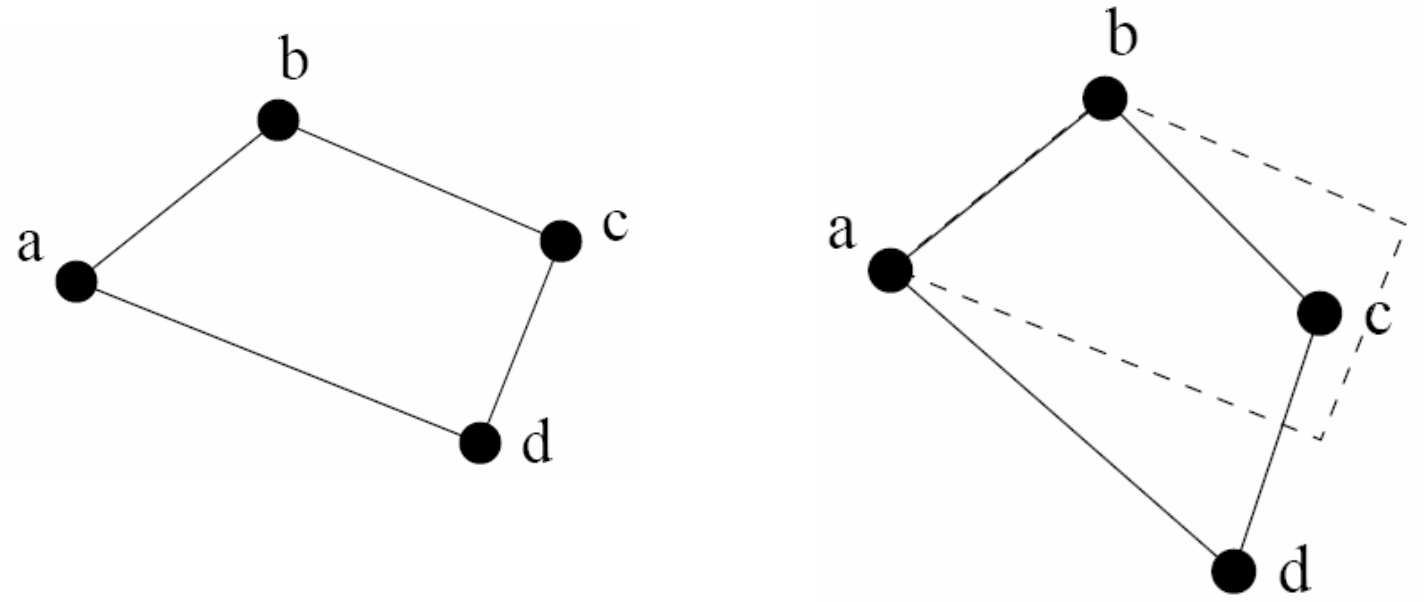
Ambiguity in localization

- Same distances, different realization.



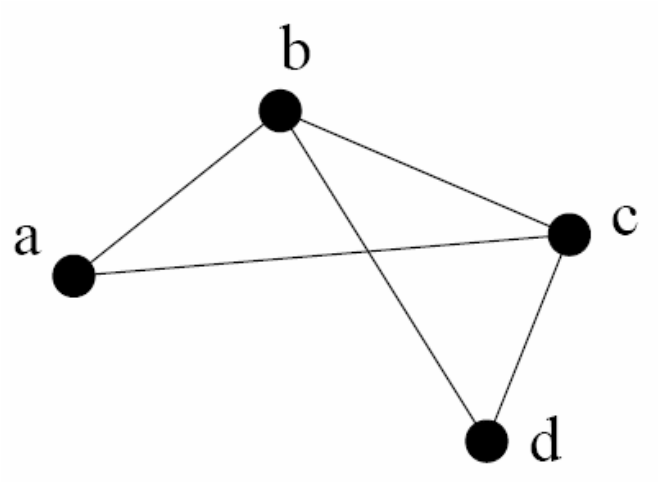
Continuous deformation

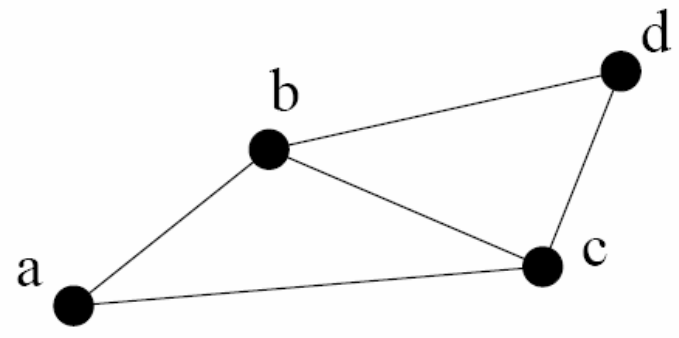
- Nodes move continuously without violating the distance constraints.



Flip

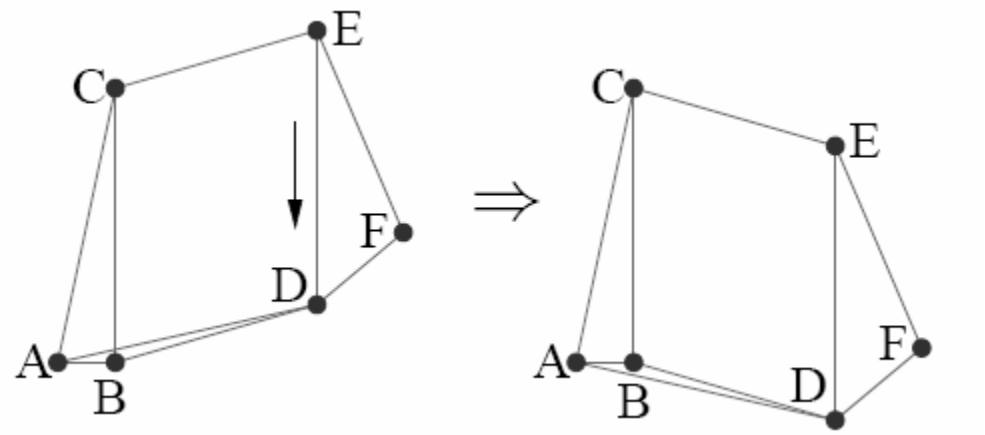
- No continuous deformation, but subjects to global flipping.





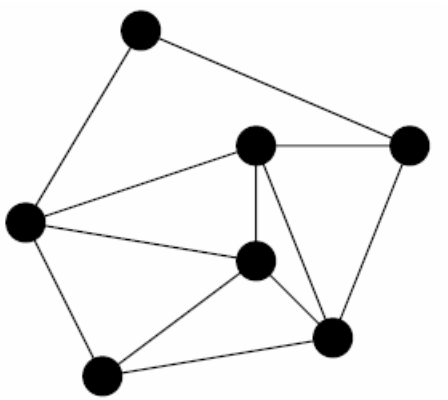
Discontinuous flex ambiguity

- Remove AD, flip ABD up, insert AD.
- No continuous deformation in between.
- But both are valid realization of the distances.



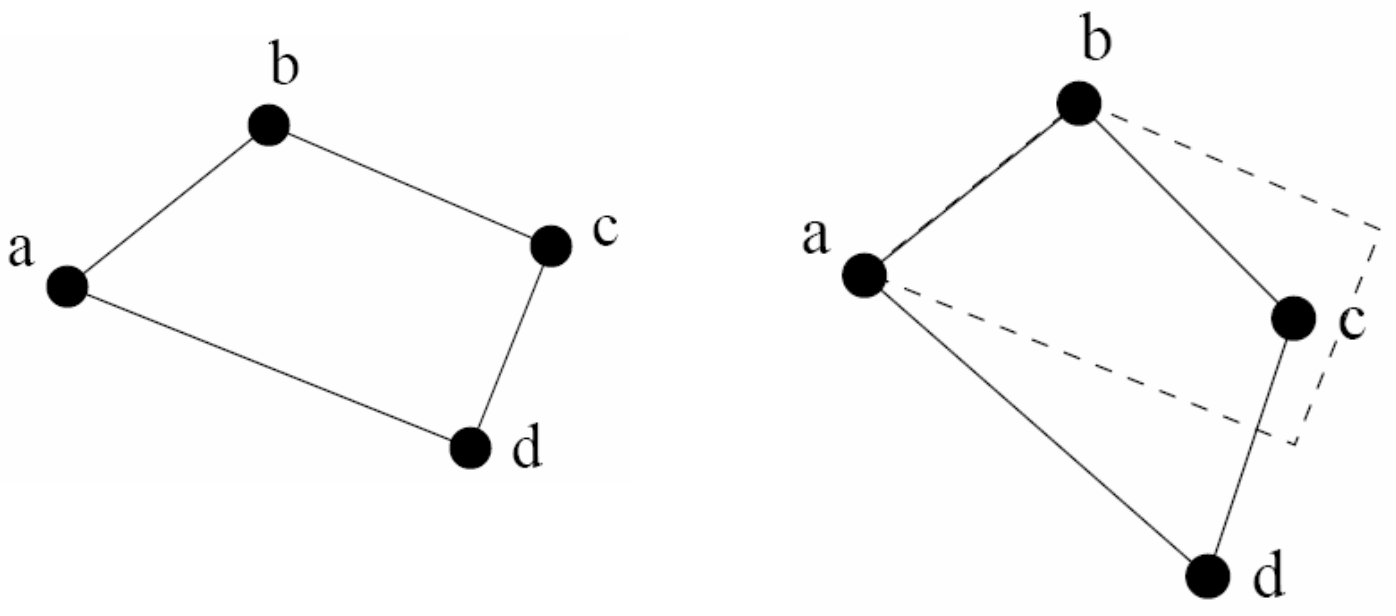
Rigidity theory

Given a system of rigid bars and hinges in 2D, does it have a continuous deformation? Multiple realizations?



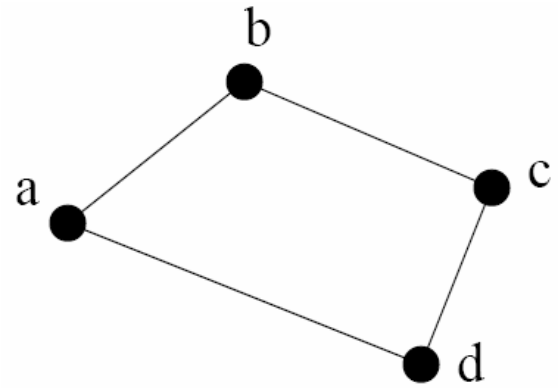
Rigidity theory

- Given a set of rigid bars connected by hinges, rigidity theory studies whether you can move them continuously.

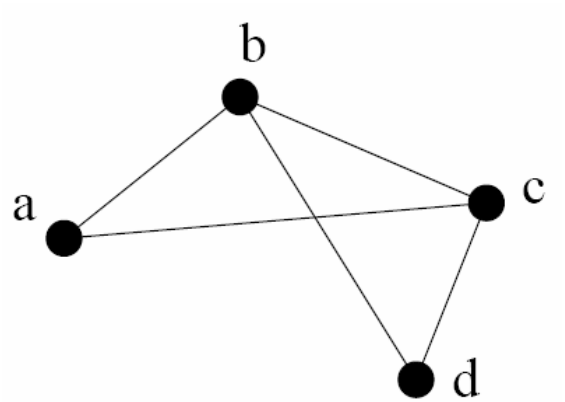


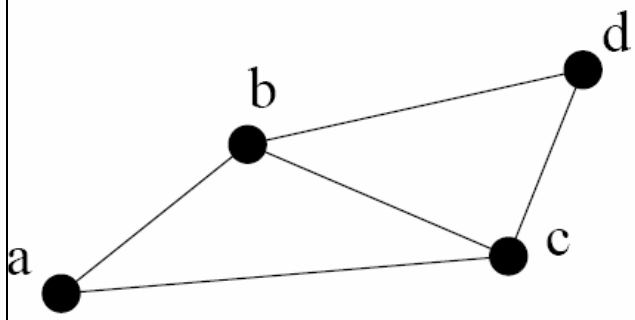
Rigidity and global rigidity

Not rigid

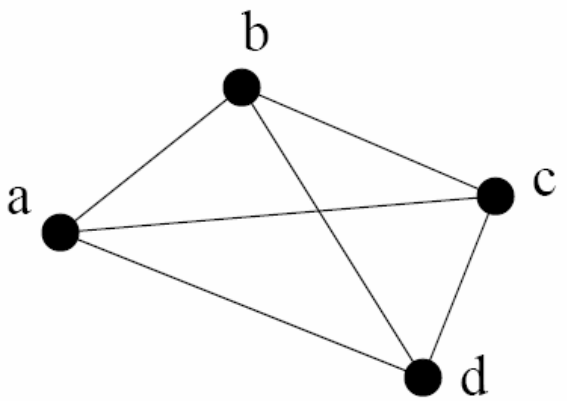


Rigid=
No continuous deformation





Globally rigid=
unique realization

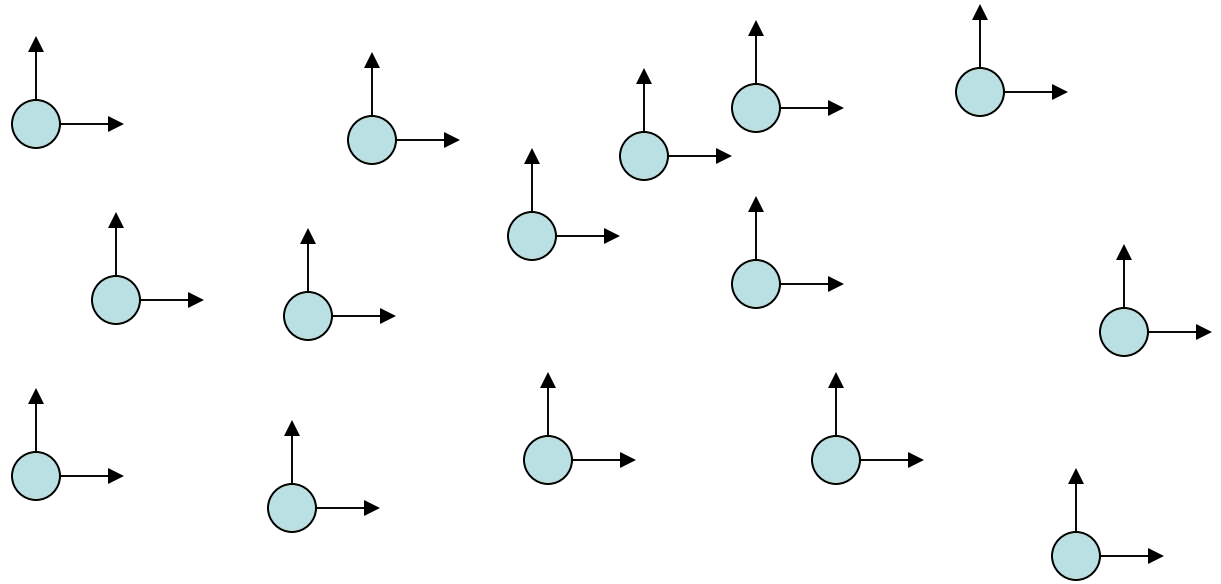


Intuition

How many distance constraints are necessary to limit a framework to only trivial motion?

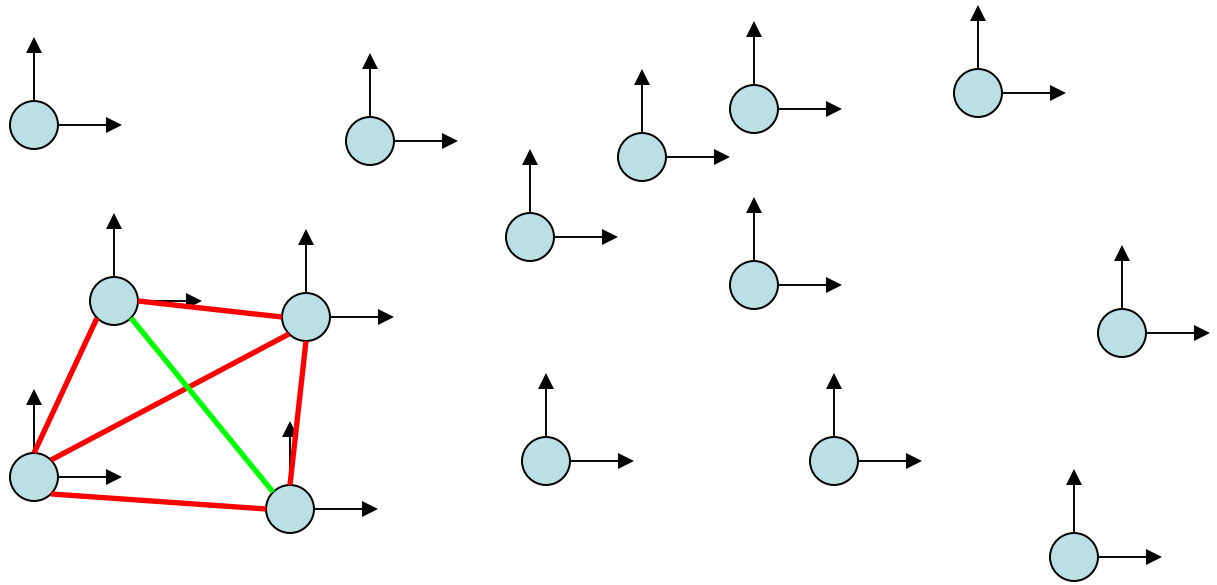
==

How many edges are necessary for a graph to be rigid?



Total degrees of freedom: 2n

How many edges are necessary to make a graph of *n* nodes rigid?

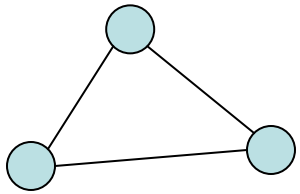


Each edge can remove a single degree of freedom

Rotations and translations will always be possible, so at least *2n-3* edges are necessary for a graph to be rigid.

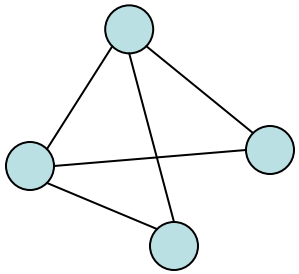
Are 2n-3 edges sufficient?

$n = 3$, $2n\text{-}3 = 3$



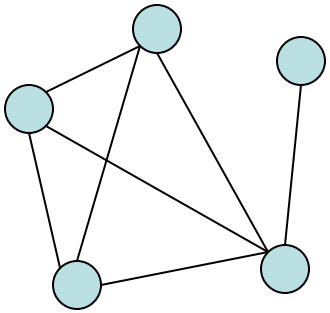
yes

$n = 4$, $2n\text{-}3 = 5$



yes

$n = 5$, $2n\text{-}3 = 7$



no

Further intuition

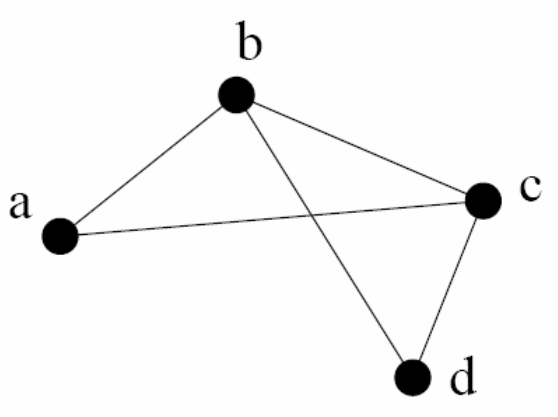
- Need at least *2n-3* “well-distributed” edges.

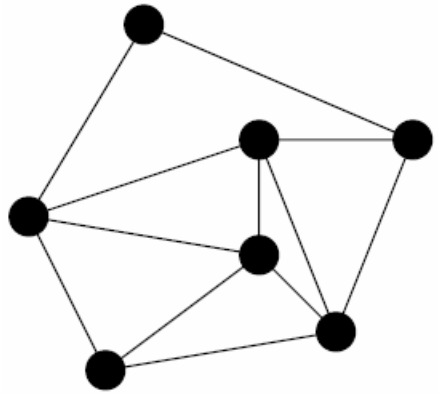
- If a subgraph has more edges than necessary, some edges are redundant.
- Non-redundant edges are independent.

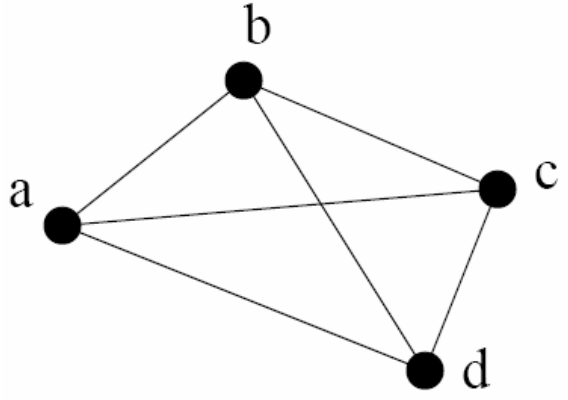
- Each independent edge removes a degree of freedom.
- Therefore, *2n-3* independent edges guarantee rigidity.

Laman condition

Laman condition:
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.







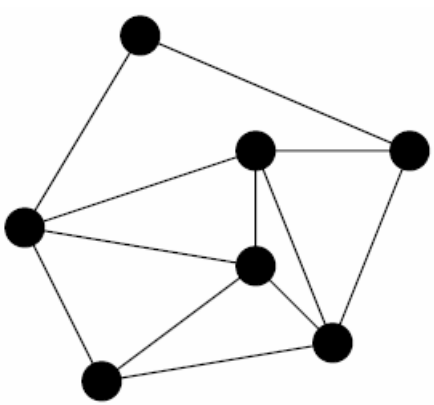
Henneberg constructions

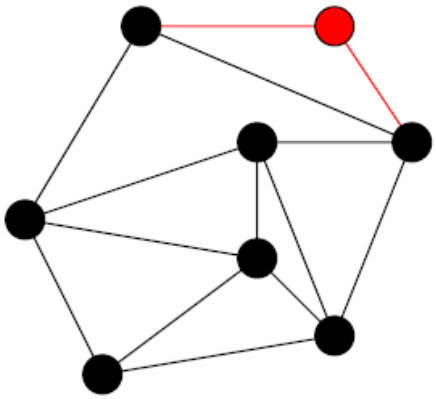
- **Henneberg constructions** (Tay-Whiteley): a Laman graph can be constructed inductively by adding one vertex at a time:

- Start with an edge
- At each step, add a new vertex
- Type I step: join the vertex to two old vertices via two 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 the three endpoints.

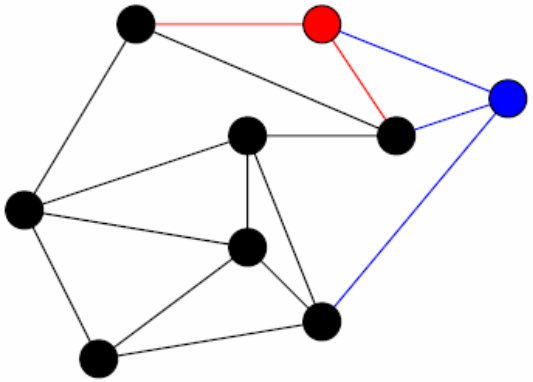
Henneberg constructions

- Type I step: join the vertex to two old vertices via two 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 the three endpoints.



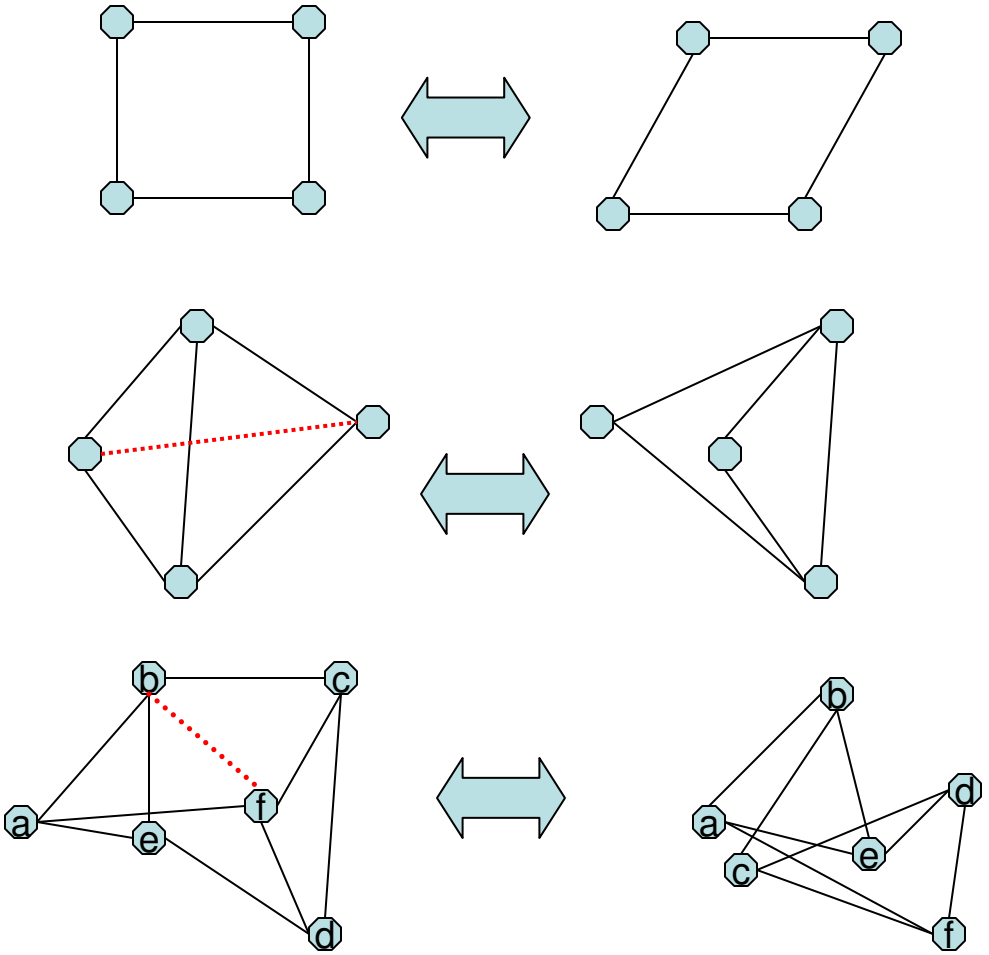


Type I



Type II

Global rigidity



Solution:

G must *rigid*

G must be 3-connected

G must be *redundantly rigid:*
It must remain rigid upon removal of any single edge

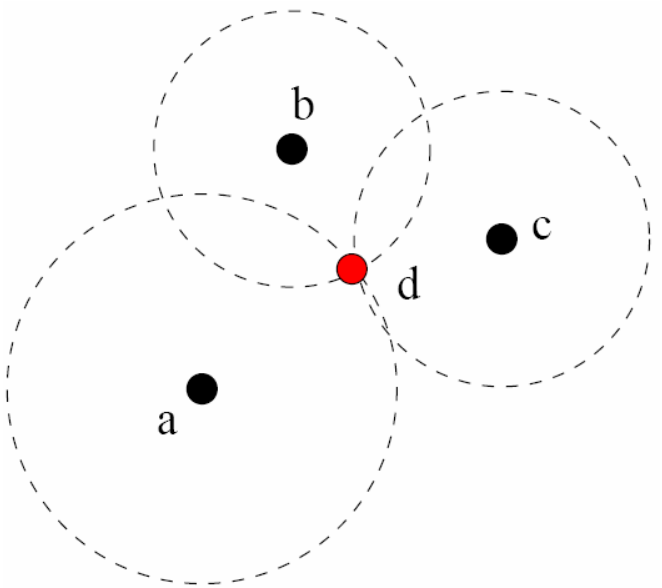
Papers

- D. Moore, J. Leonard, D. Rus, S. Teller, **Robust distributed network localization with noisy range measurements**, Proc. ACM SenSys 2004.

- Anchor-free method.
- Rigidity-aware.

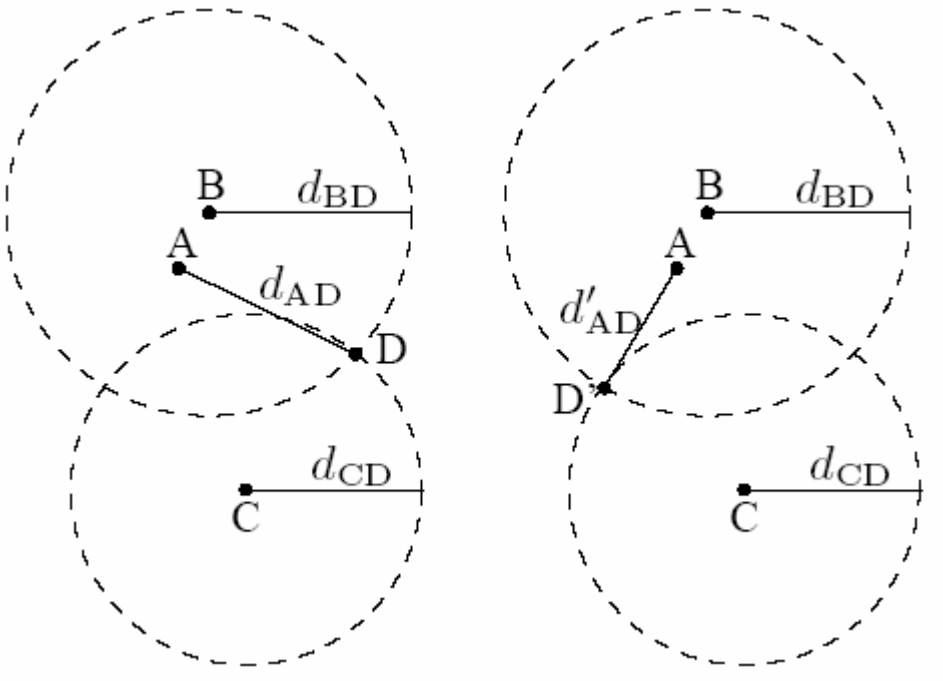
Trilateration without noise

- If three anchors are not on the same line, trilateration with accurate distance measurements gives a unique location.



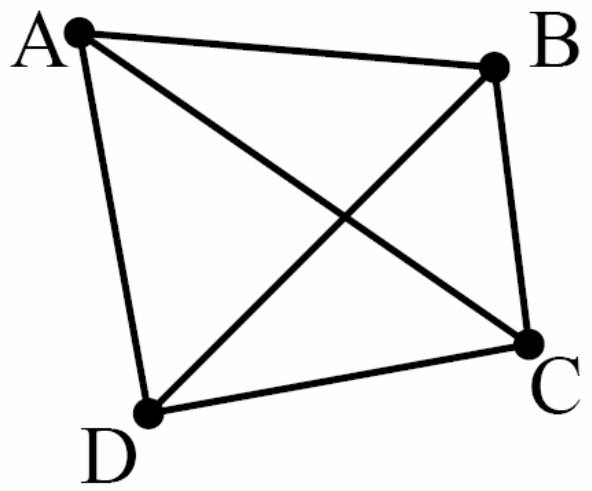
Trilateration with noise…

- With noisy measurements, trilateration can have flip ambiguity.



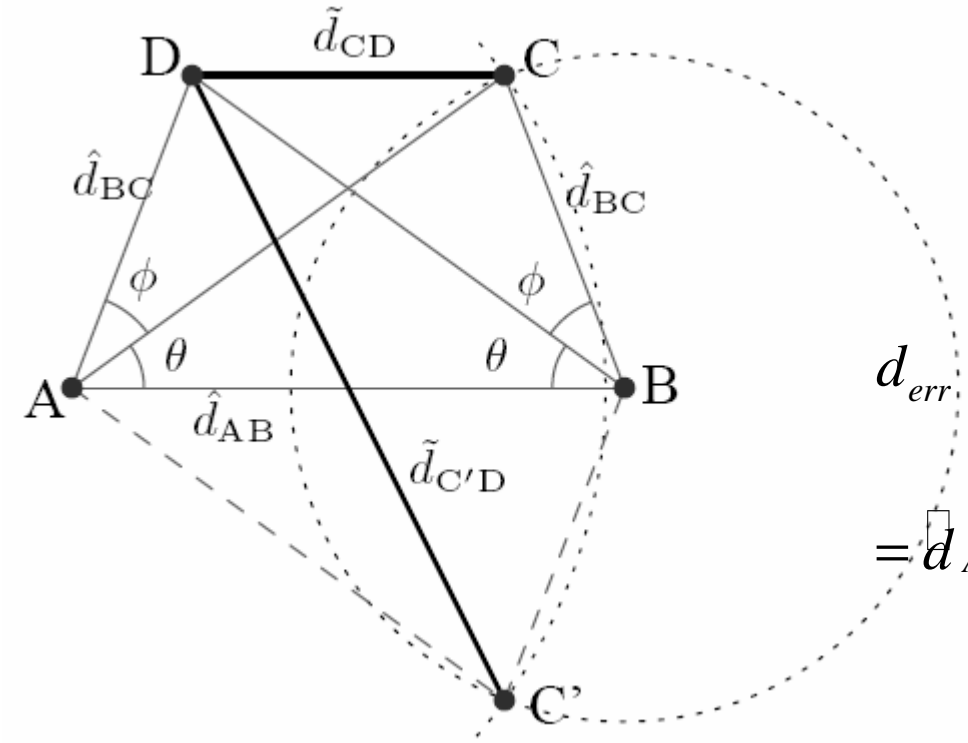
Use quadrilaterals

- Four nodes with fixed pair-wise distances
- It is the smallest 3-connected redundantly rigid graph ➔ globally rigid.



Robust quadrilaterals

- If measurement noise is bounded, the quadrilateral has no incorrect flip.



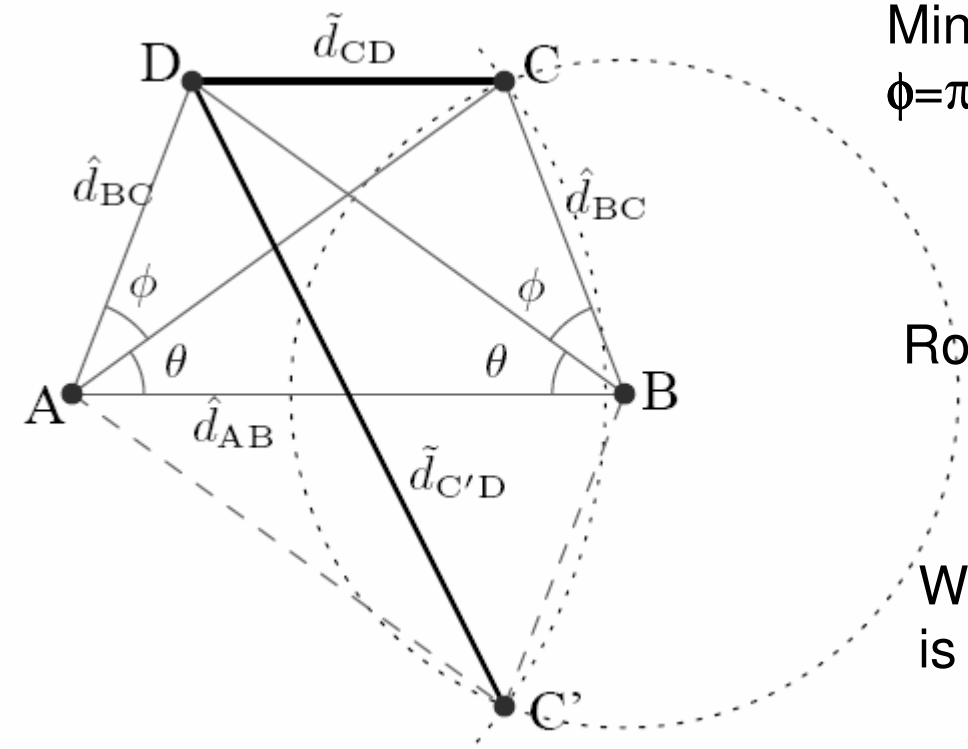
Incorrect flip: can't verify whether C or C' is correct.

$$d_{err} = \frac{\tilde{d}_{C'D} - \tilde{d}_{CD}}{2}$$

$$= \hat{d}_{AB} \frac{\sqrt{\sin^2\phi + 4\sin^2(\theta+\phi)\sin^2\theta} - \sin\phi}{2\sin(2\theta+\phi)}$$

Robust quadrilaterals

- If measurement noise is bounded, the quadrilateral has no incorrect flip.



Minimize the error by choosing $\phi=\pi/2-2\theta$.

$$d_{err} = \hat{d}_{AB} \sin^2 \theta$$

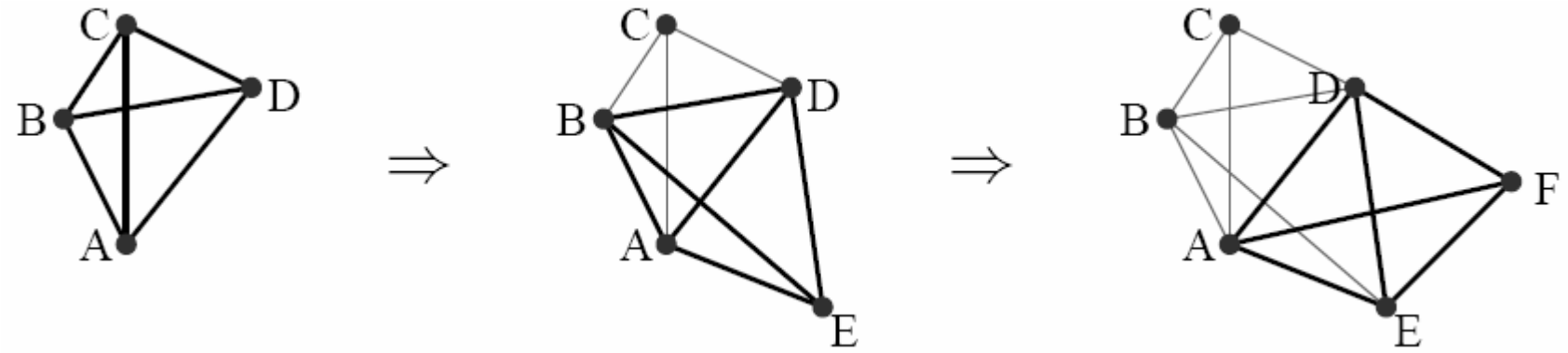
Robust quadrilateral satisfies

$$d_{err} < b \sin^2 \theta$$

Where b is the shortest side and θ is the smallest angle.

Clusters

- Robust quads that share three nodes can be merged into clusters.
- The cluster is still a 3-connected redundantly rigid graph ➔ globally rigid.
- Actually it has 3n-6 edges.



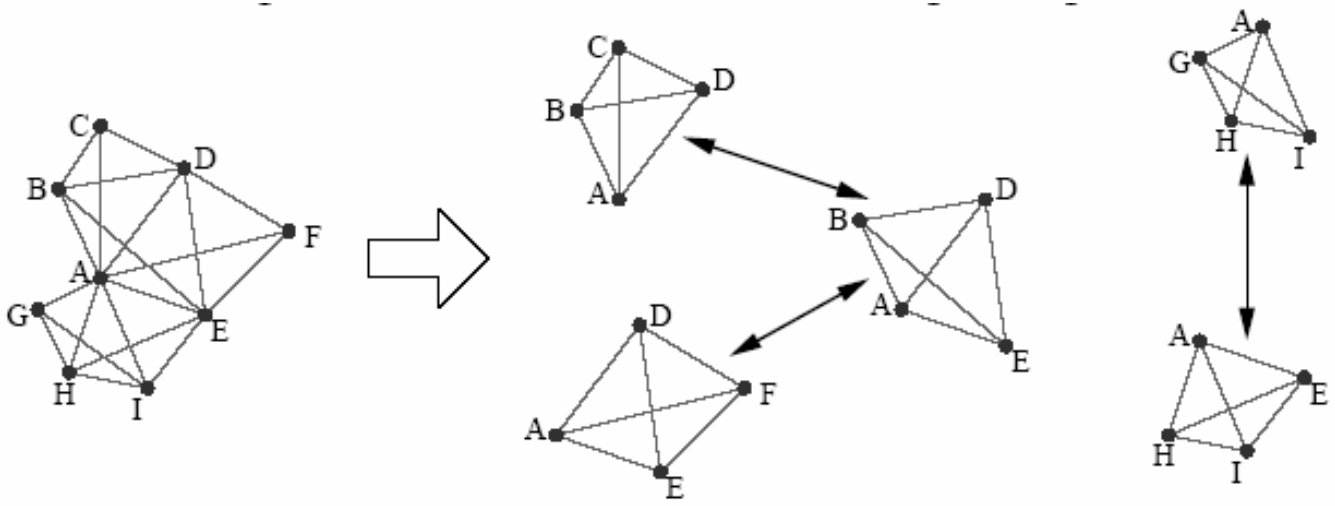
Three phases

- Cluster localization
  - Each node x find its local robust quadrilaterals.
  - Merge them to a robust cluster around x.
- (optional) cluster optimization
  - Refine the nodes' location inside a cluster.
- Cluster transformation
  - Glue the local quadrilaterals together
  - Transformation to a global coordinate system.

Algorithm phase I: cluster localization

1. Each node x gets the distance measurements between each pair of 1-hop neighbors.
2. Identify the set of robust quadrilaterals.
3. Merge the quads if they share 3 nodes.
4. Estimate the positions of as many nodes as possible by iterative trilateration.

- Note: Local coordinate system rooted at x.



Algorithm phase II: cluster optimization

- For each cluster around node x, refine the position estimates, for example, by mass-spring relaxation.
- Optional.

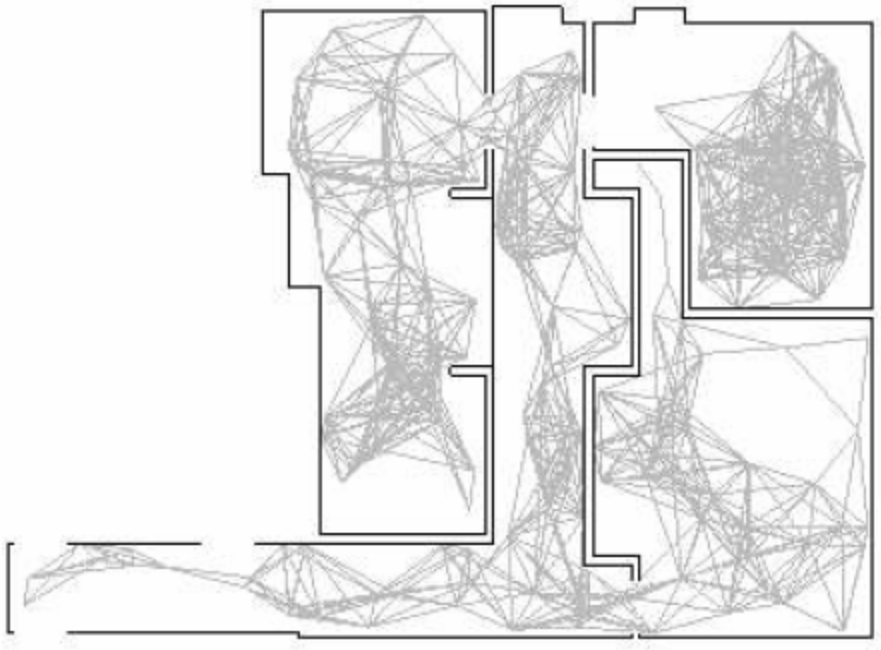
Algorithm phase III: cluster transformation

- Align neighboring local coordinates systems.

- Find the set of nodes in common between two clusters.
- Compute the translation, rotation that best align them.

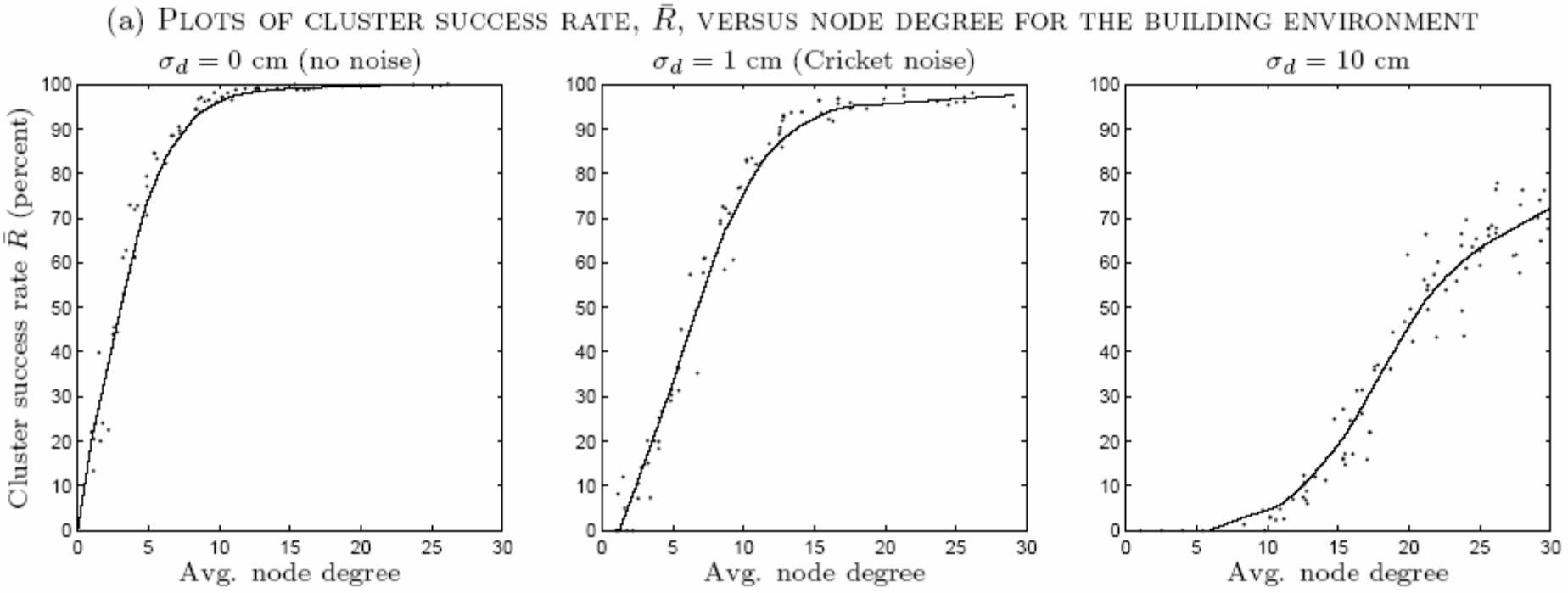
Simulations

- 183 nodes uniformly inside a building.
- Connectivity is only between nodes not obstructed by walls.



Simulations

- Cluster success rate v.s. node degree.
- Each plot represents a simulation run.



Algorithm properties

- Nodes not included in the robust quadrilaterals are not localized.
  - A wrong location is worse than no location.
- Even as noise goes to 0, avg degree ≥ 10 to achieve 100% localization.
- Not good for sparse networks.
- The avg degree $\cong 6$ for best throughput of the network.

Demo

- Localize mobile nodes
- Show a video clip

Observations

- Localization algorithm performs poorly when the graph is sparse.

- Rigidity is an issue in anchor-free methods, or anchor-based methods with noisy measurements.