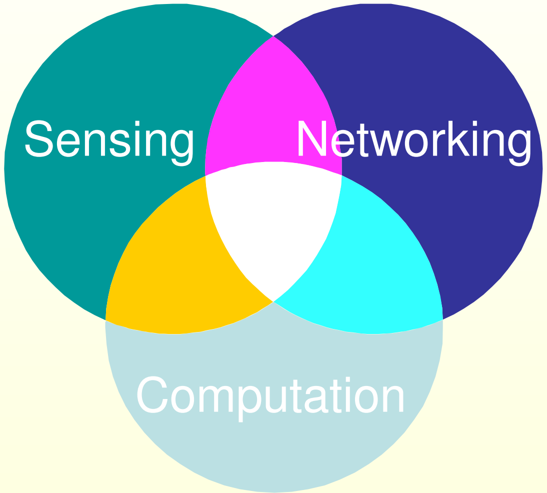
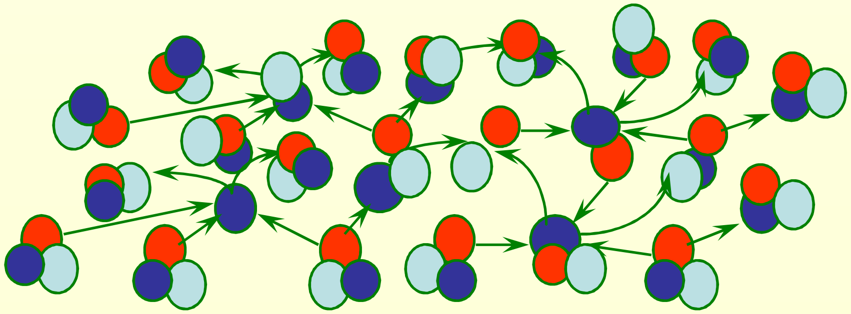
CS321: Localization I





Leonidas Guibas
Computer Science Dept.
Stanford University

Determining Sensor Locations

- Location information is important
  1. Devices need to know where they are
     - Sensor tasking: turn on the sensor near the window …
  2. We want to know the location of the data
     - A temperature reading is too hot – but where?
  3. Location helps infrastructure establishment
     - geographical routing
     - sensor coverage

GPS Not Always Feasible

- Requires view of the sky, doesn't work indoors, under foliage, etc.
- Rather expensive
    - A $10 sensor with a $100 GPS?

Localization Algorithm:

- (Optional) Some nodes (anchors or beacons) know their locations (e.g., through GPS)
- Nodes make local measurements
    - Distances or angles between two neighbors
- Communicate with others to exchange data
- Infer location information from these measurements

The Localization Problem

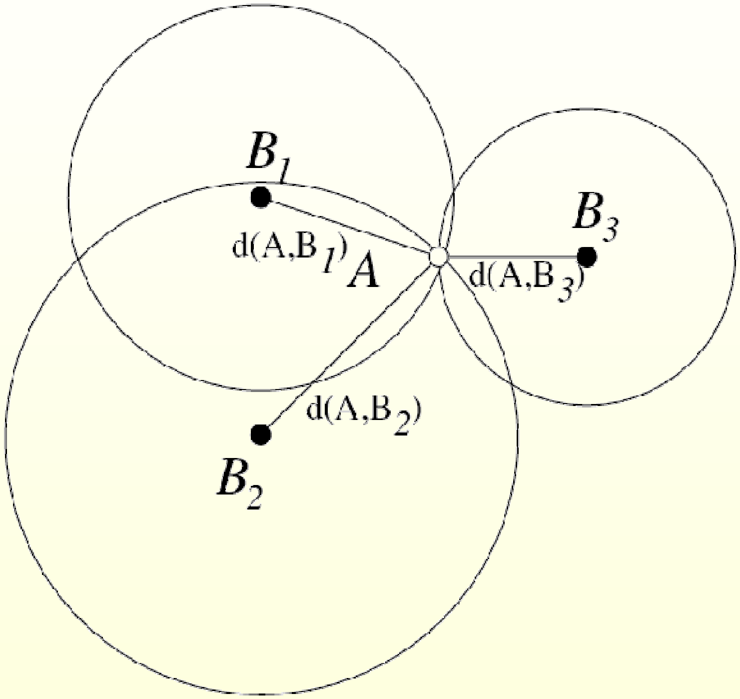
- Output: location of the sensor nodes
  - World coordinates, e.g., what GPS gives
  - Relative locations (floating global frame)
- Input:
  - Connectivity hop count (under UDG model)
    - Nodes with *k* hops away are within Euclidean distance *k*
    - Nodes without a link must be at least distance 1 away
  - Distance measurements between neighboring nodes
  - Angle measurements between neighboring nodes
  - Combinations of the above

Distance Measurements (Ranging)

- Received Signal Strength Indicator (RSSI)
  - The further away, the weaker the received signal
  - Mainly used for RF signals
  - Also possible w. ultrasound
- Time of Arrival (ToA) or Time Difference of Arrival (TDoA)
  - Signal propagation time translates to distance
  - RF, acoustic, infrared and ultrasound
  - Medium propagation speed must be estimated
  - Require clock synchronization

Time of Arrival (ToA)

- Used in GPS
- Triangulation
- Need synchronization
- Synchronization can be relaxed if round-trip time is used.



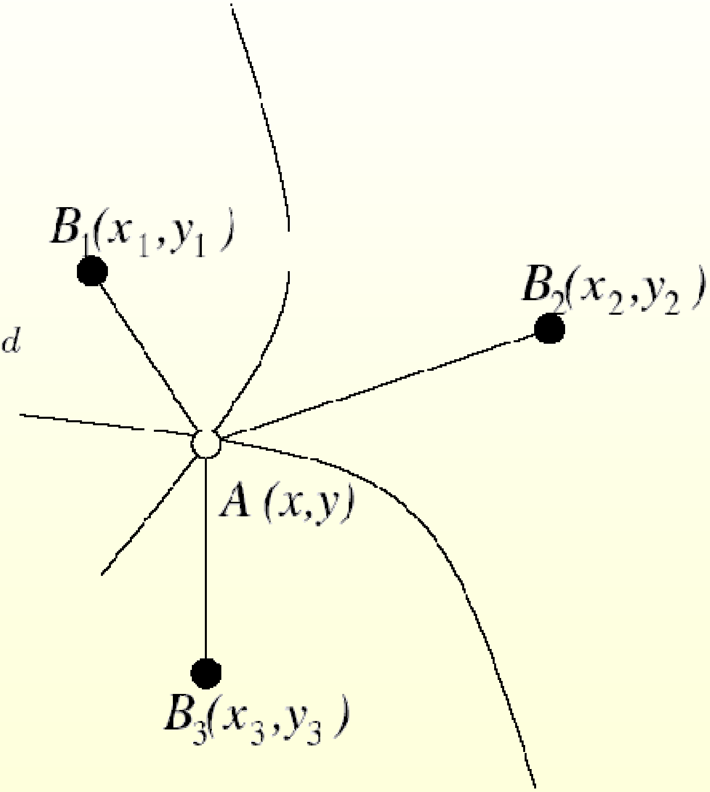
Since circles have two intersections, three measurements are needed

Time Difference of Arrival (TDoA)

- Anchors B1 and B2 send signal to A simultaneously The time difference of arrival is recorded
- A must be on the hyperbola:

$$\sqrt{(x-x_1)^2+(y-y_1)^2}-\sqrt{(x-x_2)^2+(y-y_2)^2}=\delta_d$$

- Do this for B2 and B3.
- A must be on the intersection of the hyperbolae
- If the two hyperbolae have 2 intersections, one more measurement is needed

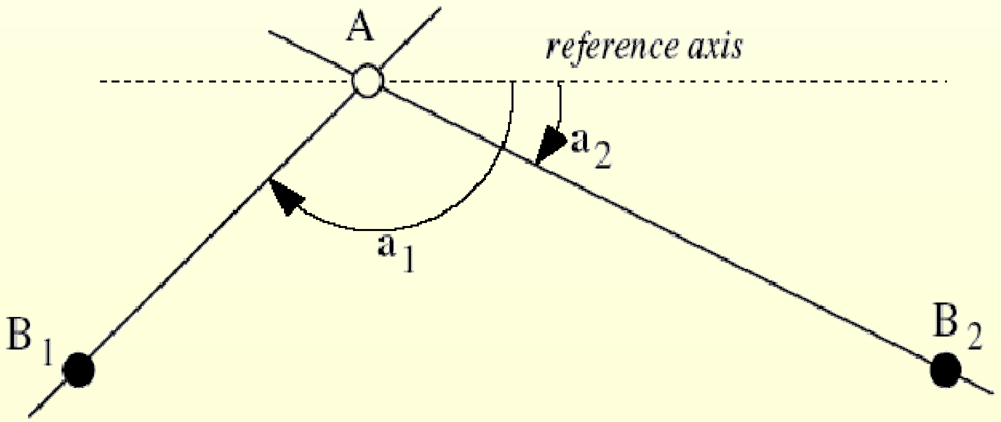


Angle Measurements

- Angle of Arrival (AoA)
  - Determine the direction of propagation of a radio-frequency wave incident on an antenna array
- Directional antennas
- Special hardware, e.g., laser transmitters and receivers

Angle of Arrival (AoA)

- A measures the direction of an incoming link by an antenna array
- By using 2 anchors, A can determine its position
- Even if no reference axis is available, A can be placed on a circle



Localization Algorithm Zoo

- Anchor-based
  - Some nodes know their locations, either by a GPS or as pre-specified
- Anchor-free
  - Relative locations only
  - A harder problem, need to solve for the global structure. Nowhere to start …

- Range-based
  - Use range information (distance estimation)
- Range-free
  - No true distance estimation, use connectivity information such as hop count, or other tools

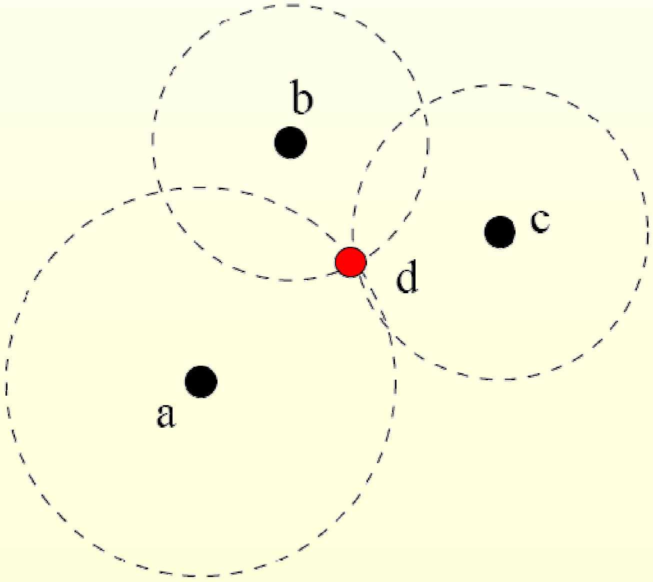
Papers

- A. Savvides, C.-C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. Proc. MobiCom 2001.
- Andreas Savvides, and Mani B. Strivastava. Distributed Fine-Grained Localization in Ad-Hoc Networks. IEEE Transactions of Mobile Computing, to appear, 2003.
- Tolga Eren, David Goldenberg, Walter Whitley, Yang Richard Yang, A. Stephen Morse, Brian D.O. Anderson and Peter N. Belhumeur, Rigidity, Computation, and Randomization of Network Localization. In Proceedings of IEEE INFOCOM, Hong Kong, China, April 2004.
- D. Moore, J. Leonard, D. Rus, S. Teller, Robust distributed network localization with noisy range measurements, Proc. ACM SenSys 2004.
- Yi Shang, Wheeler Ruml, Ying Zhang, and Markus P.J. Fromherz, Localization from Mere Connectivity, MobiHoc'03.

Multilateration:
using plane geometry

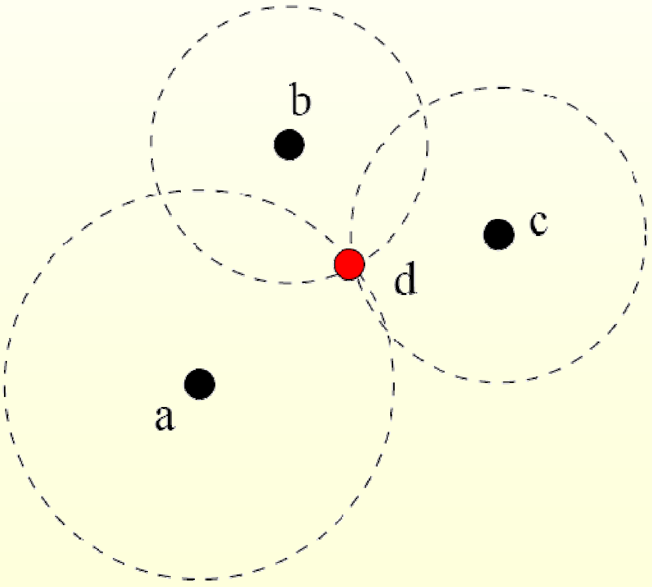
Triangulation, Trilateration

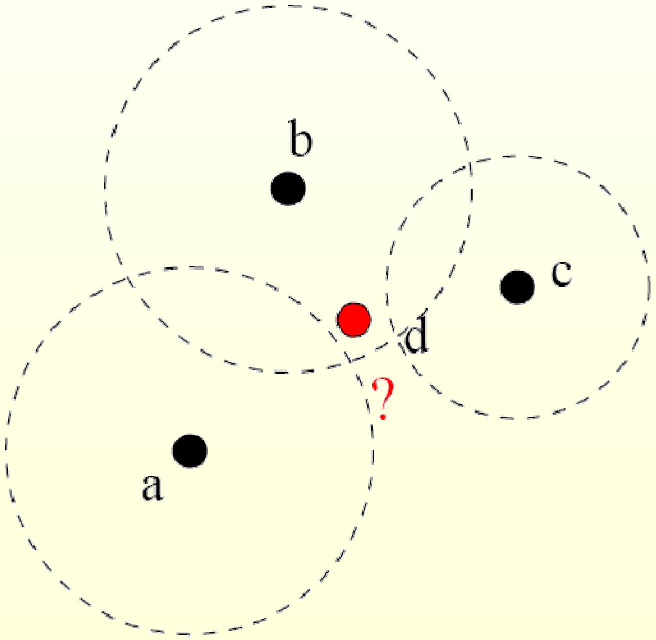
- Anchors advertise their coordinates & transmit a reference signal
- Other nodes use the reference signal to estimate distances to anchor nodes



Triangulation, Trilateration

- Distance measurements can be noisy!
- Solve an optimization problem: minimize the mean square error.





Problem Formulation

- *k* beacons at positions (x_i, y_i)
- Assume node 0 has position (x_0, y_0)
- Distance measurement between node 0 and beacon *i* is r_i
- Error:

$$f_i = r_i - \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}$$

- The objective function is

$$F(x_0, y_0) = \min \sum f_i^2$$

- This is a non-linear optimization problem

Linearization and Min Mean-Square Estimate

- Ideally, we would like the error to be 0

$$f_i = r_i - \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} = 0$$

- Re-arrange:

$$({x_0}^2 + {y_0}^2) + x_0(-2x_i) + y_0(-2y_i) - {r_i}^2 = -x_i^2 - y_i^2$$

- Subtract the *k*-th (last) equation from the *i*-th to get rid of quadratic terms

$$2x_0(x_k - x_i) + 2y_0(y_k - y_i) = r_i^2 - {r_k}^2 - x_i^2 - y_i^2 + x_k^2 + y_k^2$$

- Note that this is now linear

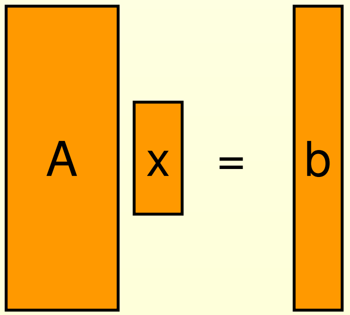
Linearization and Min Mean-Square Estimate

- In general, we will have an over-constrained linear system (*k* ≥ 3 measurements)

$$Ax = b$$

$$b = \begin{bmatrix} r_1^2 - r_k^2 - x_1^2 - y_1^2 + x_k^2 + y_k^2 \\ r_2^2 - r_k^2 - x_2^2 - y_2^2 + x_k^2 + y_k^2 \\ \vdots \\ r_{k-1}^2 - r_k^2 - x_{k-1}^2 - y_{k-1}^2 + x_k^2 + y_k^2 \end{bmatrix} \qquad A = \begin{bmatrix} 2(x_k - x_1) & 2(y_k - y_1) \\ 2(x_k - x_2) & 2(y_k - y_2) \\ \vdots & \vdots \\ 2(x_k - x_{k-1}) & 2(y_k - y_{k-1}) \end{bmatrix}$$

$$x = \begin{matrix} x_0 \\ y_0 \end{matrix}$$



Solve Using Least Squares Estimation

The linearized equations in matrix form become

$$Ax = b$$

Now we can use the least squares equation to compute the estimate

$$x = (A^T A)^{-1} A^T b$$

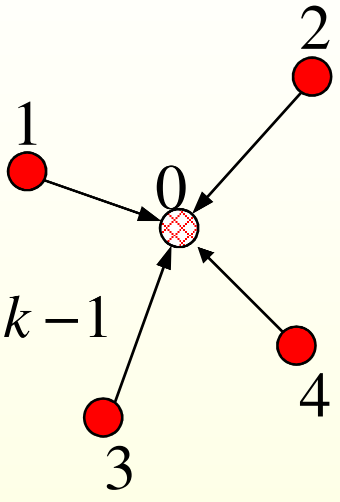
Some Issues

- Degeneracies
  - Beacon nodes must not lie on the same line
- For ToA, TDoA, how to estimate the propagation speed in the medium (e.g., sound for acoustic signals)?

Estimate Also Medium Speed

Minimize over all

$$f(x_i, x_0, s) = s t_{i0} - \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}$$



$$i = 1,2 \cdots k-1$$

This can be linearized to the form

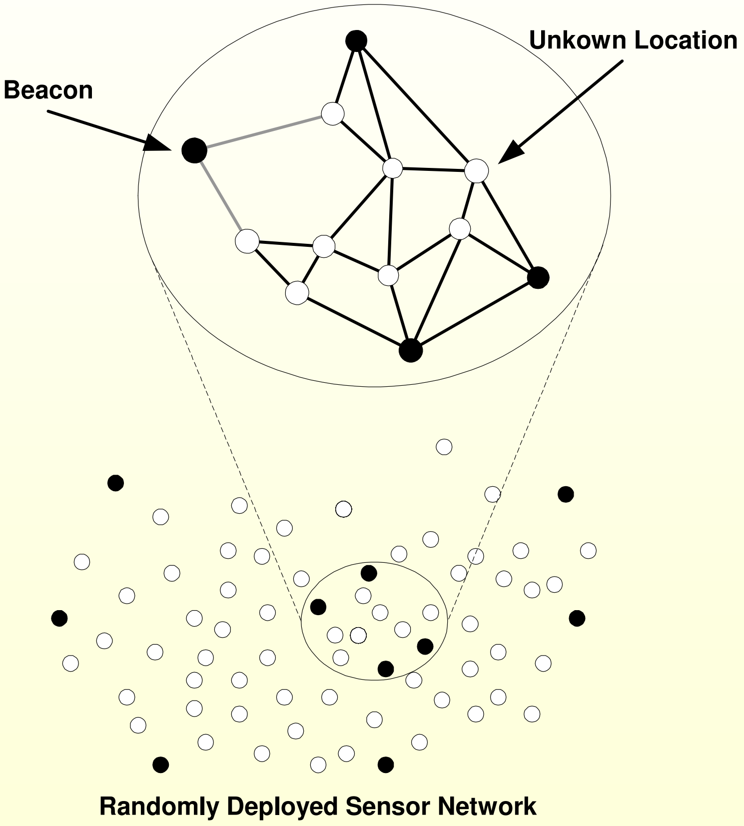
$$y = Xb$$

where

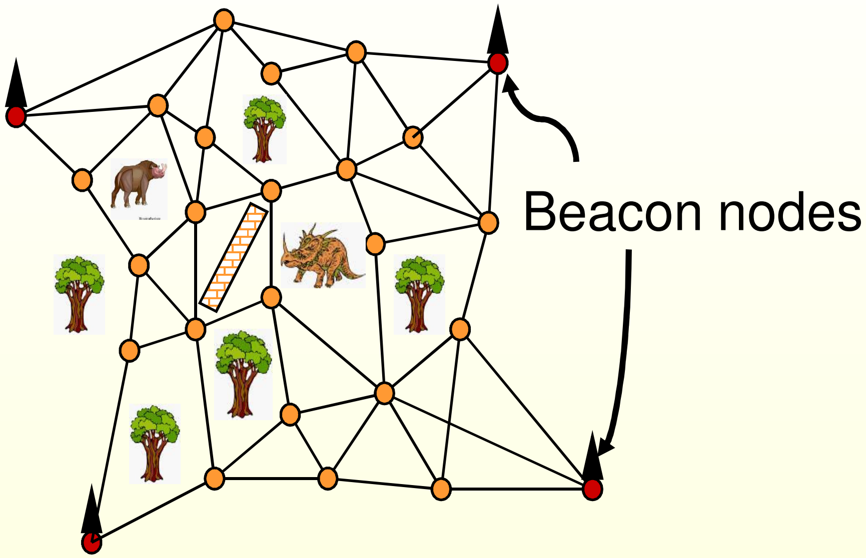
$$y = \begin{bmatrix} -x_1^2 - y_1^2 + x_k^2 + y_k^2 \\ -x_2^2 - y_2^2 + x_k^2 + y_k^2 \\ \vdots \\ -x_{k-1}^2 - y_{k-1}^2 + x_k^2 + y_k^2 \end{bmatrix} \quad X = \begin{bmatrix} 2(x_k - x_1) & 2(y_k - y_1) & t_{k0}^2 - t_{10}^2 \\ 2(x_k - x_2) & 2(y_k - y_2) & t_{k0}^2 - t_{20}^2 \\ \vdots & \vdots & \\ 2(x_k - x_{k-1}) & 2(y_k - y_{k-1}) & t_{k0}^2 - t_{(k-1)0}^2 \end{bmatrix} \quad b = \begin{bmatrix} x_0 \\ y_0 \\ s^2 \end{bmatrix}$$

MMSE Solution: $b = (X^T X)^{-1} X^T y$

The General Localization Problem



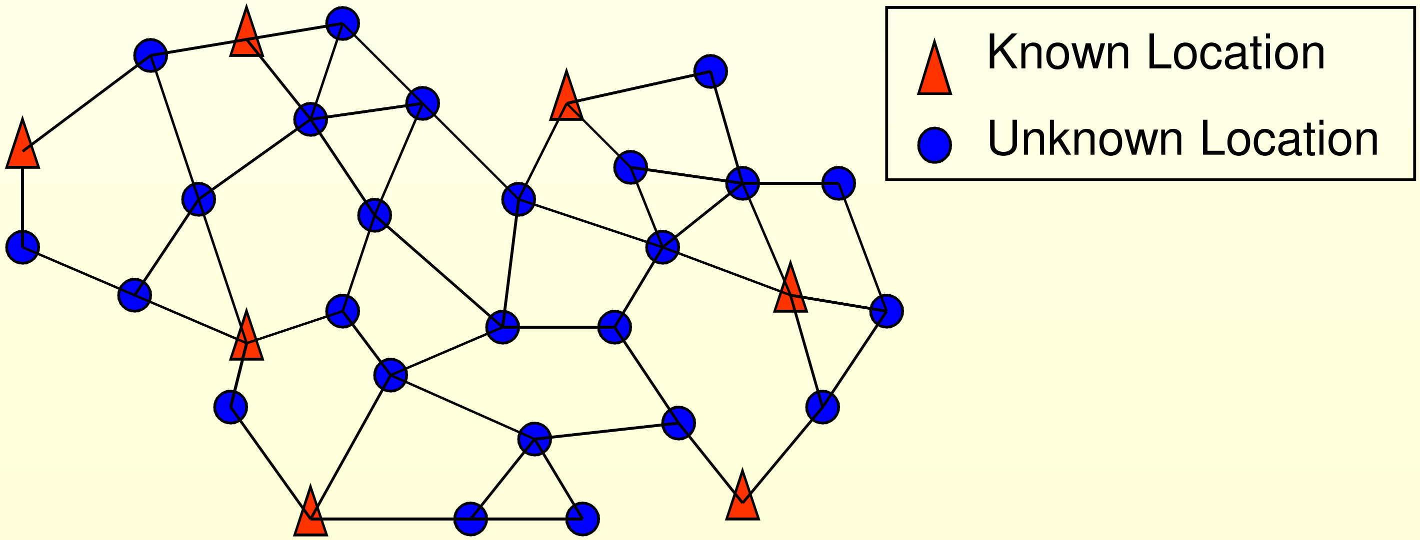
Randomly Deployed Sensor Network



- Localize nodes in an ad-hoc *multihop* network
- Based on a set of inter-node distance measurements

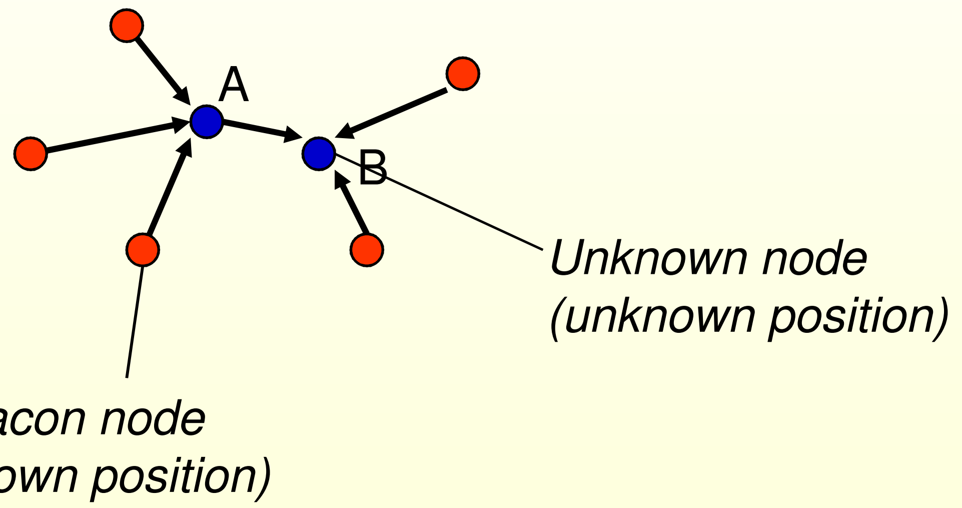
Location from Ranging

- Assume that initially a small number of nodes know their positions (base stations, with GPS, etc.) and can act as landmarks
- Other nodes will localize themselves my measuring their distances to these, and then can become beacons themselves, and so on ...



Iterative Multilateration

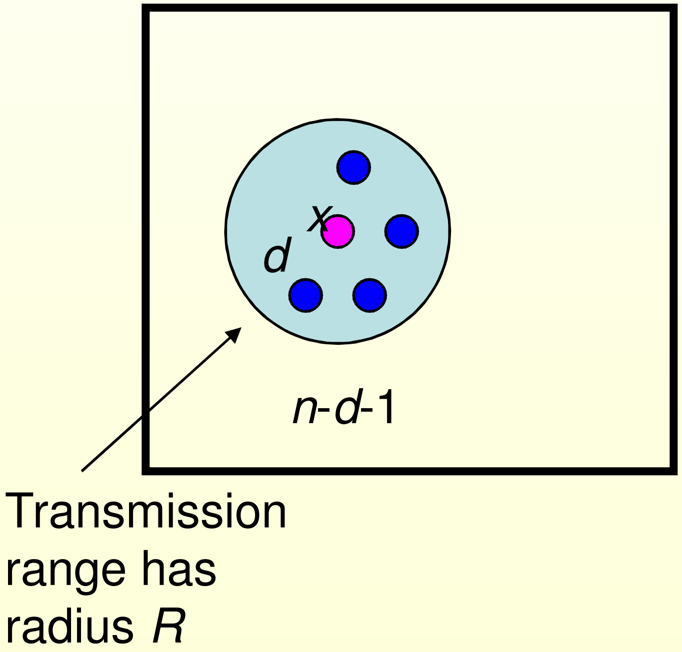
- Iterative multilateration
  - a node with at least 3 neighboring beacons estimates its position and becomes a beacon
  - Iterate until all nodes with 3 beacons are localized



Connectivity matters! Each node needs at least 3 neighbors.

Iterative Multilateration: How Many Beacons?

- *n* nodes deployed randomly in a square of side *L*
- P(*d*) = Pr{a node *x* has degree *d*} = ?



Probability that one node falls inside the transmission range of x?

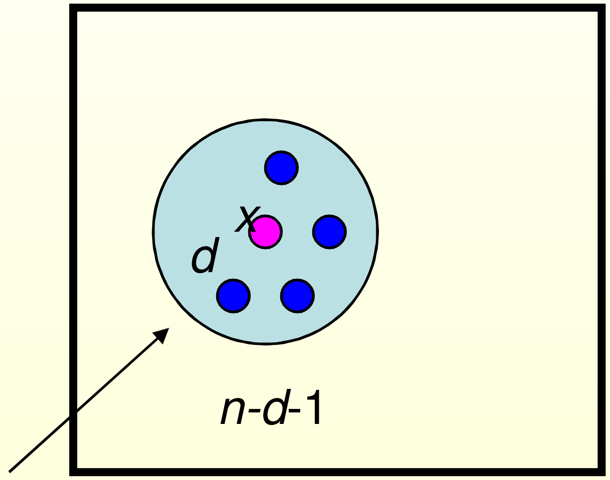
$$p = \frac{\pi R^2}{L^2}$$

Binomial distribution

$$P(d) = p^d \cdot (1-p)^{n-d-1} \cdot \binom{n-1}{d}$$

Iterative Multilateration: How Many Beacons?

- When *n* tends to infinity, the binomial distribution converges to a Poisson distribution



Transmission range has radius R

Probability that one node falls inside the transmission range of x?

$$p = \frac{\pi R^2}{L^2} \qquad \lambda = n \cdot p$$

↓ Binomial distribution tends to the Poisson distribution

$$P(d) = \frac{\lambda^d}{d!} \cdot e^{-\lambda}$$

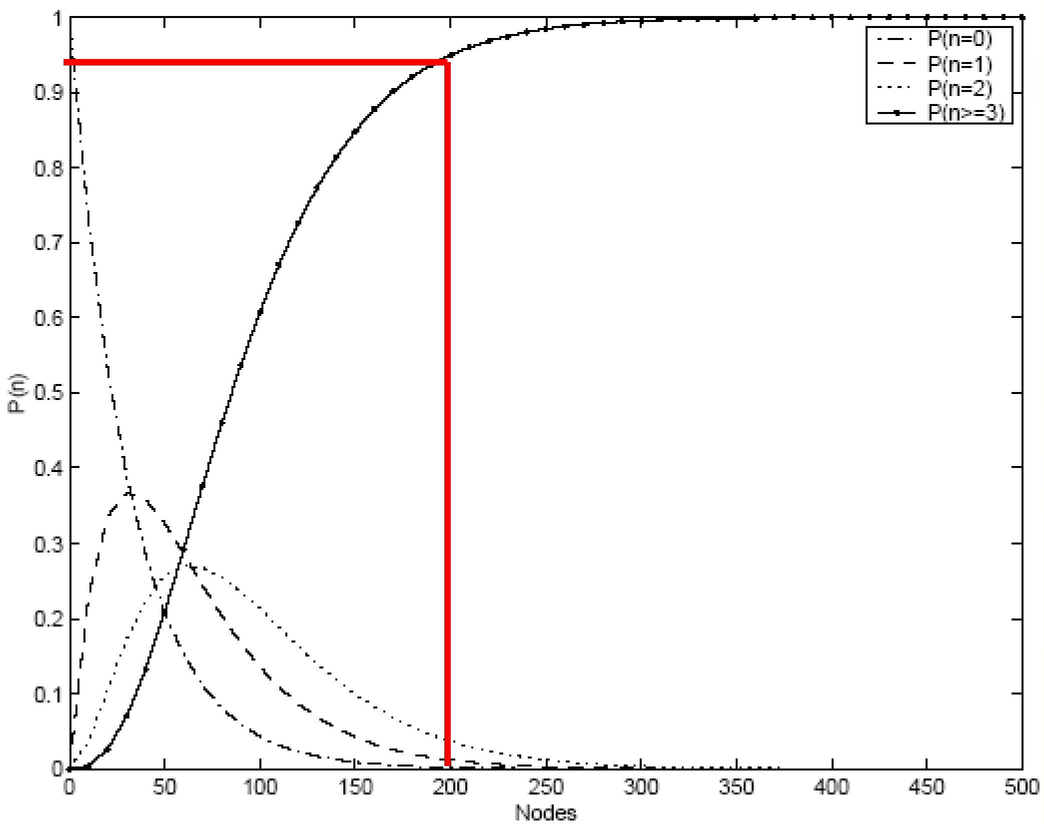
Iterative Multilateration: How Many Beacons?

$$P(d) = \frac{\lambda^d}{d!} \cdot e^{-\lambda} \qquad P(\geq d) = 1 - \sum_{i=1}^{n-1} P(i)$$

100 by 100 field
Sensor range:10

Probability of a node with
0, 1, 2, ≥ 3 neighbors.

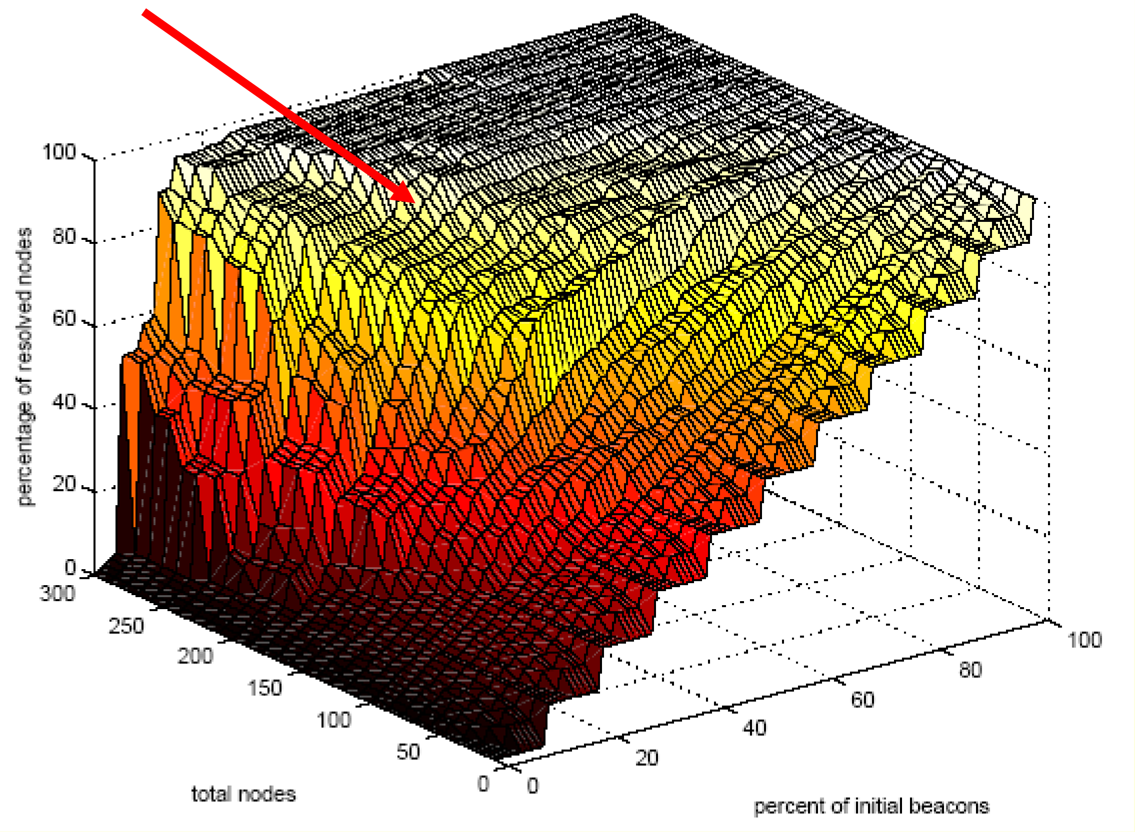
With 200 nodes,
P(≥ 3) is about 95%.



Iterative Multilateration: How Many Beacons?

Still, many beacons may be required.

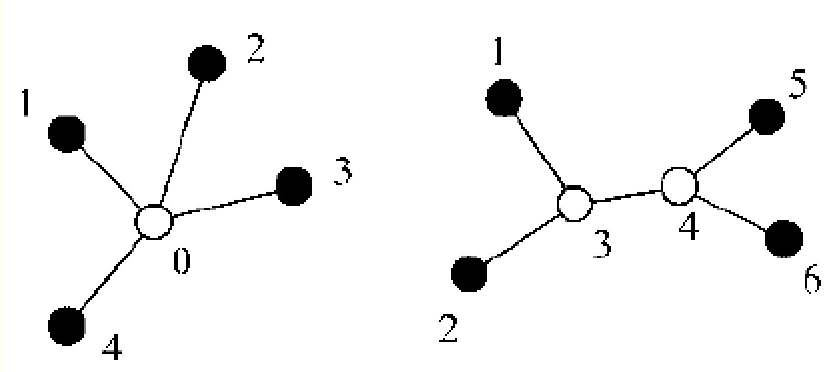
With 200 nodes, we need about 50~60 beacons to localize about 90% of the nodes. That’s ¼ of the total number of nodes.



Problems with Iterative Multilateration

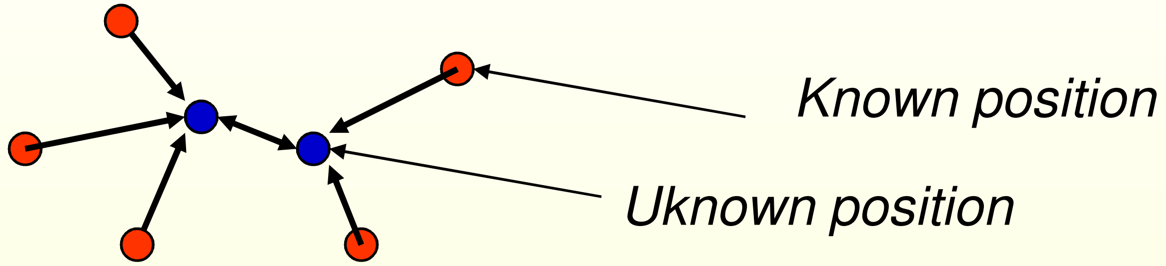
Problems

1. Requires a large fraction of beacons
2. Error accumulates ← Mass-spring optimization
3. Method gets stuck --- not all nodes with 3 or more neighbors can be located. ← Global optimization



Collaborative Mutlilateration (Savvides *et. al*., '03*)*

- All available measurements are used as constraints



- Solve for the positions of multiple unknowns simultaneously
- Catch: This is a non-linear optimization problem!
- How do we handle this?

Problem Formulation

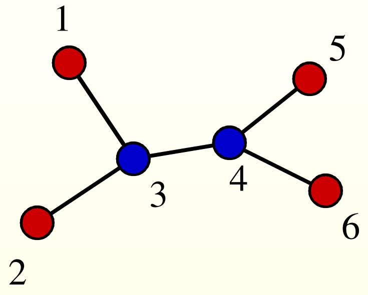
$$f_{2,3} = R_{2,3} - \sqrt{(x_2 - \hat{x}_3)^2 + (y_2 - \hat{y}_3)^2}$$

$$f_{3,5} = R_{3,5} - \sqrt{(\hat{x}_3 - x_5)^2 + (\hat{y}_3 - y_5)^2}$$

$$f_{4,3} = R_{4,3} - \sqrt{(\hat{x}_4 - \hat{x}_3)^2 + (\hat{y}_4 - \hat{y}_3)^2}$$

$$f_{4,5} = R_{4,5} - \sqrt{(\hat{x}_4 - x_5)^2 + (\hat{y}_4 - y_5)^2}$$

$$f_{4,1} = R_{4,1} - \sqrt{(\hat{x}_4 - x_1)^2 + (\hat{y}_4 - y_1)^2}$$



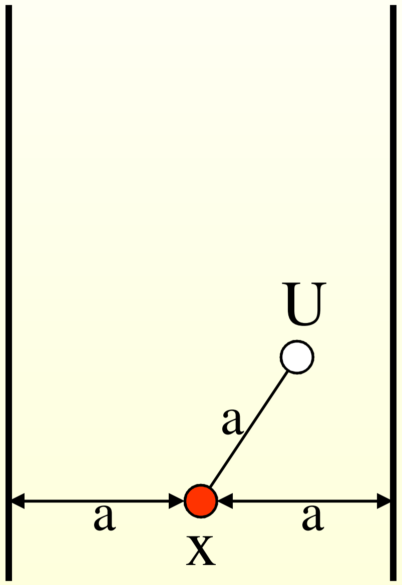
The objective function is

$$F(\hat{x}_3, \hat{y}_3, \hat{x}_4, \hat{y}_4) = \min \sum f_{i,j}^2$$

Need some decent initial estimates, then iterate using a Kalman Filter

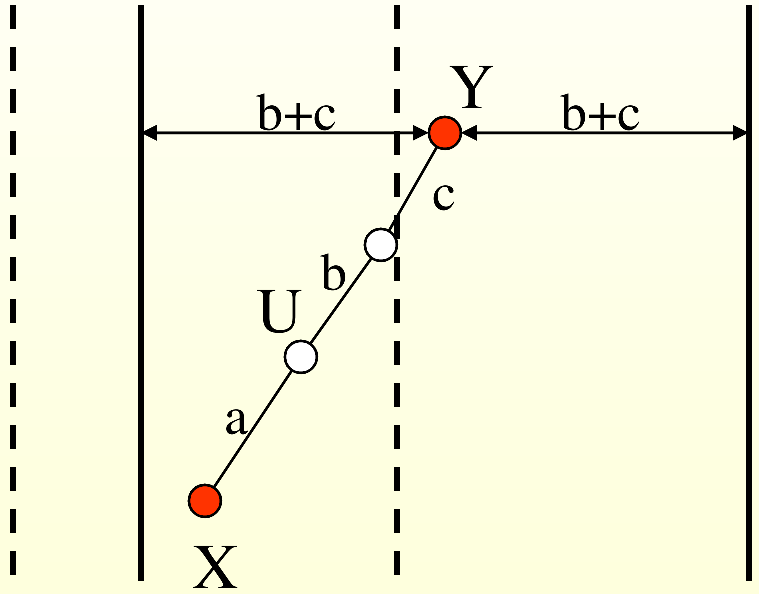
Initial Estimates

- Use the accurate distance measurements to impose constraints in the *x* and *y* coordinates – bounding box
- Use the distance to a beacon as bounds on the *x* and *y* coordinates



Initial Estimates, Con’t

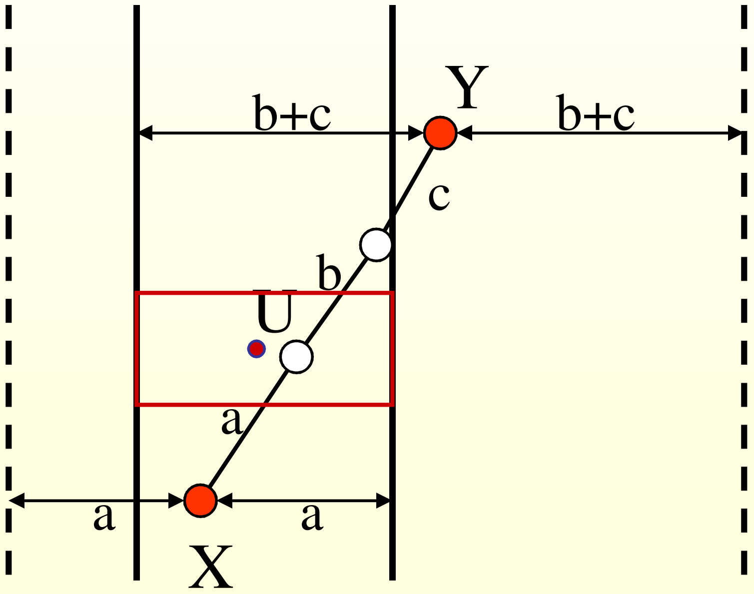
- Use the accurate distance measurements to impose constraints in the *x* and *y* coordinates – bounding box
- Use the distance to a beacon as bounds on the *x* and *y* coordinates
- Do the same for beacons that are multiple hops away
- Select the most constraining bounds



U is between [Y-(b+c)] and [X+a]

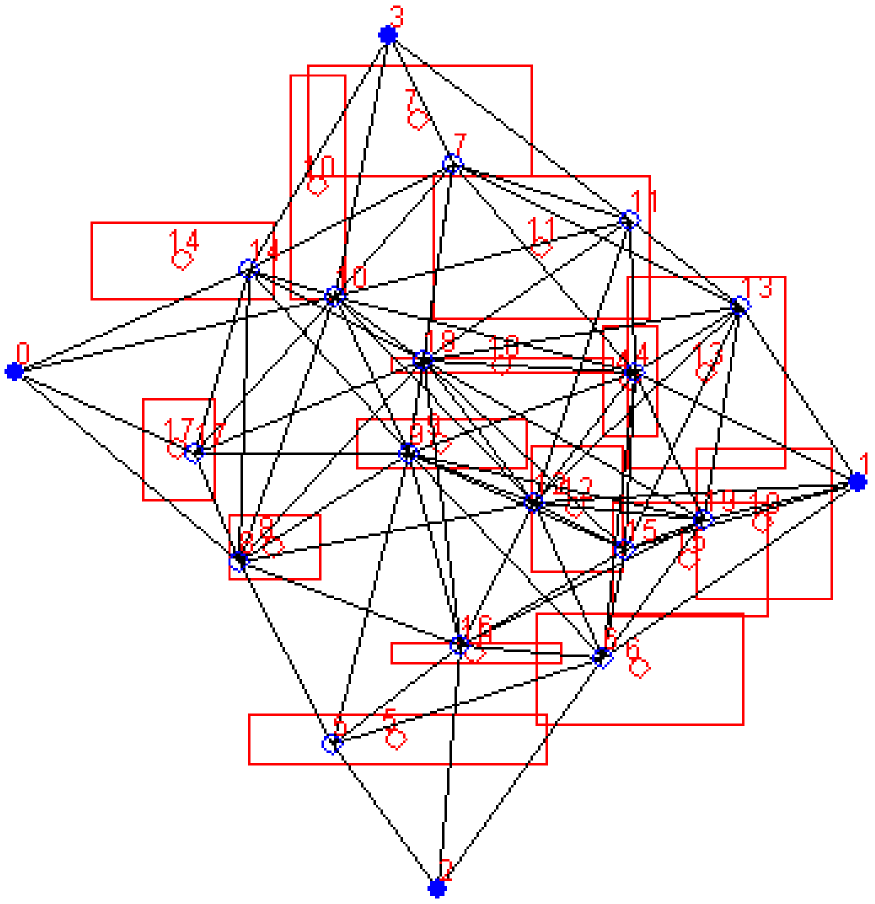
Initial Estimates, Cont'd

- Use the accurate distance measurements to impose constraints in the *x* and *y* coordinates – bounding box
- Use the distance to a beacon as bounds on the *x* and *y* coordinates
- Do the same for beacons that are multiple hops away
- Select the most constraining bounds
- Set the center of the bounding box as the initial estimate

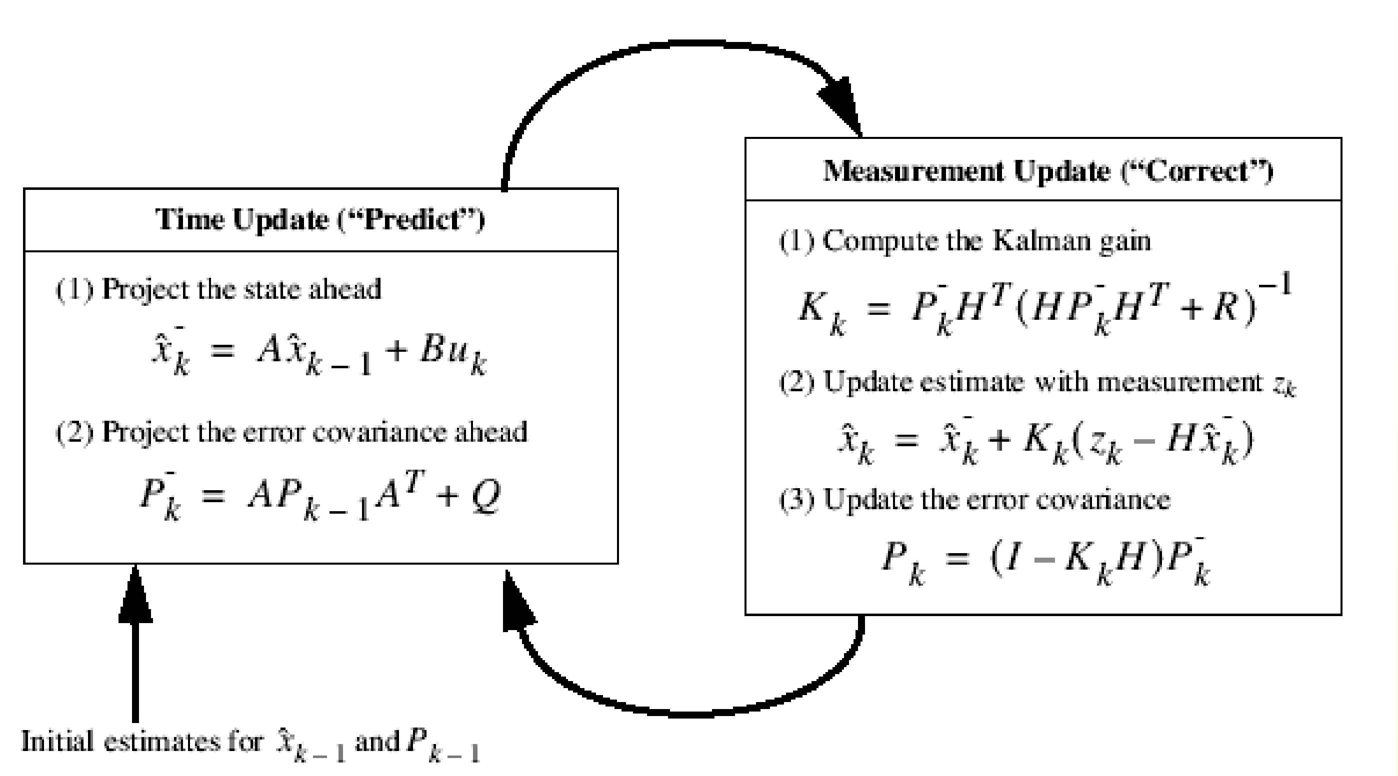


Initial Estimates, Cont'd

- Example:
  - 4 beacons
  - 16 unknowns
- To get good initial estimates, beacons should be placed on the perimeter of the network
- **Observation:** If the unknown nodes are outside the beacon perimeter then initial estimates are on or very close to the convex hull of the beacons



Kalman Filter



Time Update (“Predict”)

(1) Project the state ahead

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$$

(2) Project the error covariance ahead

$$P_k^- = AP_{k-1}A^T + Q$$

Measurement Update (“Correct”)

(1) Compute the Kalman gain

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

(2) Update estimate with measurement z_k

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

(3) Update the error covariance

$$P_k = (I - K_k H)P_k^-$$

Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

- We only use measurement update since the nodes are static
- We know *R* (ranging noise distribution)
- Artificial notion of time: sequentially introduce distance constraints

Global Kalman Filter

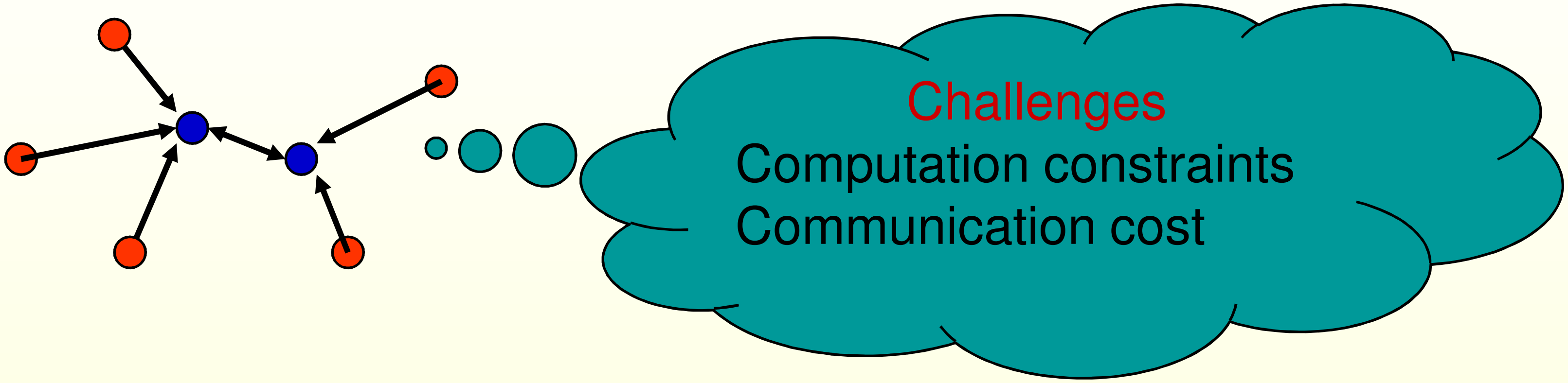
$$\hat{z}_k{}^T = \begin{bmatrix} \sqrt{(x_2 - ex_3)^2 + (y_2 - ey_3)^2} \\ \sqrt{(ex_3 - x_5)^2 + (ey_3 - y_5)^2} \\ \sqrt{(ex_3 - ex_4)^2 + (ey_3 - ey_4)^2} \\ \sqrt{(ex_4 - x_1)^2 + (ey_4 - y_1)^2} \\ \sqrt{(ex_4 - x_5)^2 + (ey_4 - y_5)^2} \end{bmatrix}$$

$$H = \left.\underbrace{\begin{bmatrix} 0 & 0 & \frac{x_2 - ex_3}{\hat{z}_k(1)} & \frac{y_2 - ey_3}{\hat{z}_k(1)} \\ \frac{ex_3 - x_5}{\hat{z}_k(2)} & \frac{ey_3 - y_5}{\hat{z}_k(2)} & 0 & 0 \\ \frac{ex_3 - ex_4}{\hat{z}_k(3)} & \frac{ey_3 - ey_4}{\hat{z}_k(3)} & \frac{ex_4 - ex_3}{\hat{z}_k(3)} & \frac{ey_4 - ey_3}{\hat{z}_k(3)} \\ \frac{ex_4 - x}{\hat{z}_k(4)} & \frac{ey_4 - y}{\hat{z}_k(4)} & 0 & 0 \\ \frac{ex_4 - x_5}{\hat{z}_k(5)} & \frac{ey_4 - y_5}{\hat{z}_k(5)} & 0 & 0 \end{bmatrix}}_{\text{\# of nodes to be located x 2}}\right\} \text{\# of edges}$$

- Matrices grow with density and number of nodes → so does computation cost
- Computation is not feasible on small processors with limited computation and memory

Overview: Collaborative Multilateration

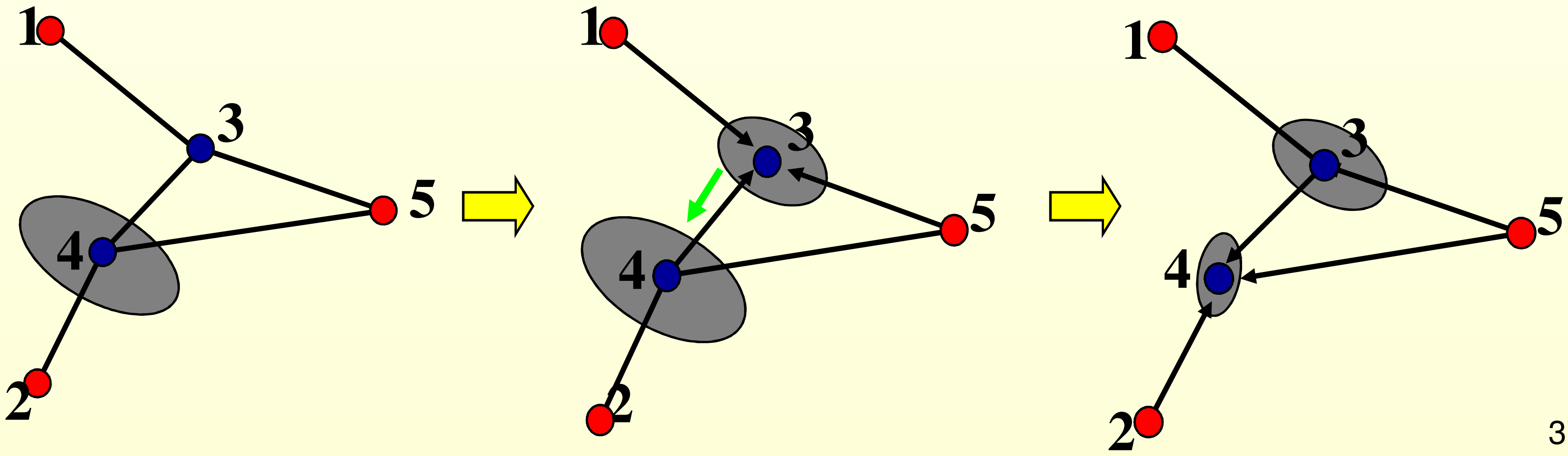
Collaborative Multilateration



Challenges

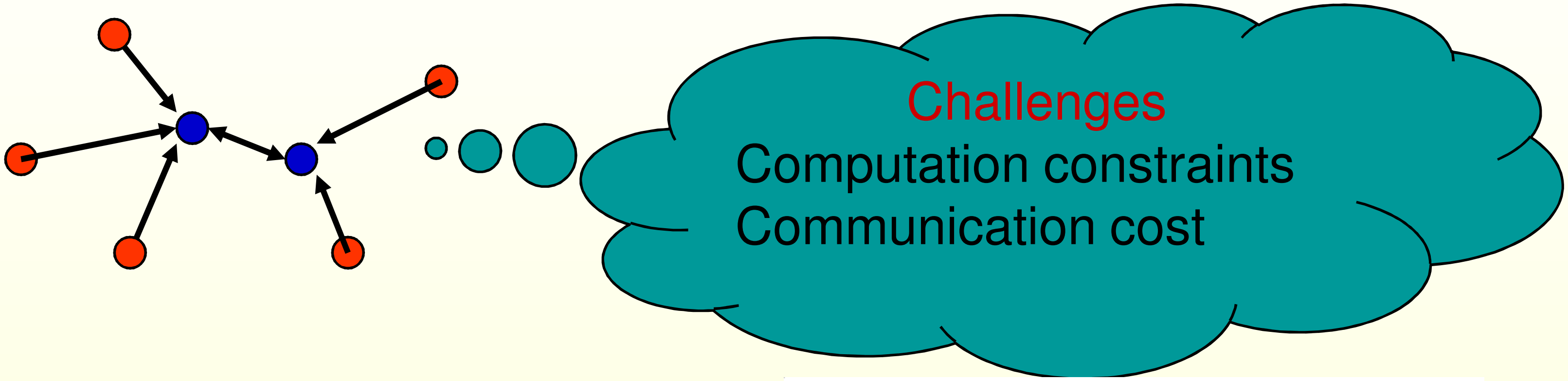
Computation constraints

Communication cost



Overview: Collaborative Multilateration

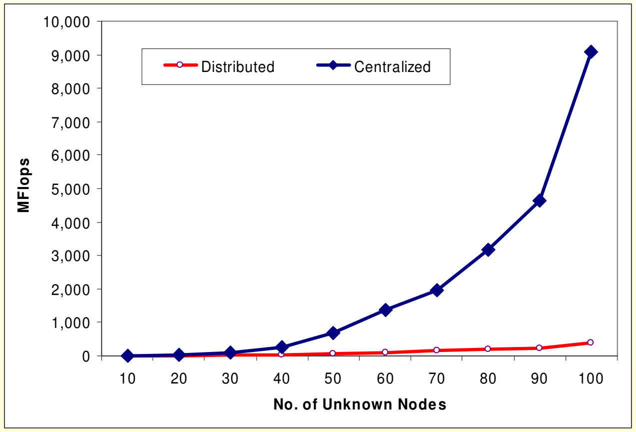
Collaborative Multilateration

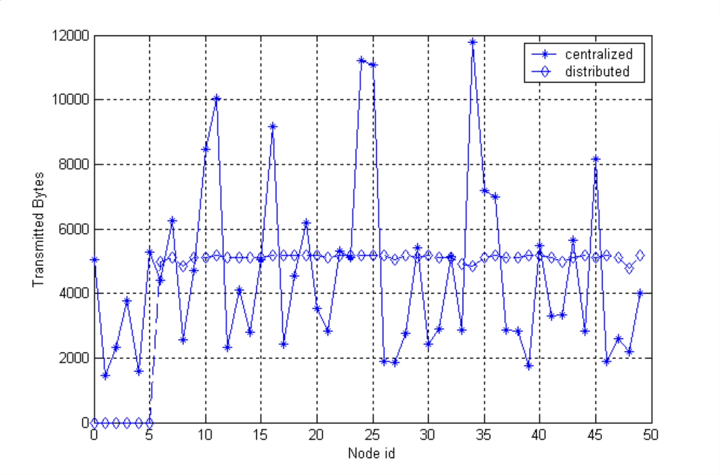


Challenges

Computation constraints

Communication cost

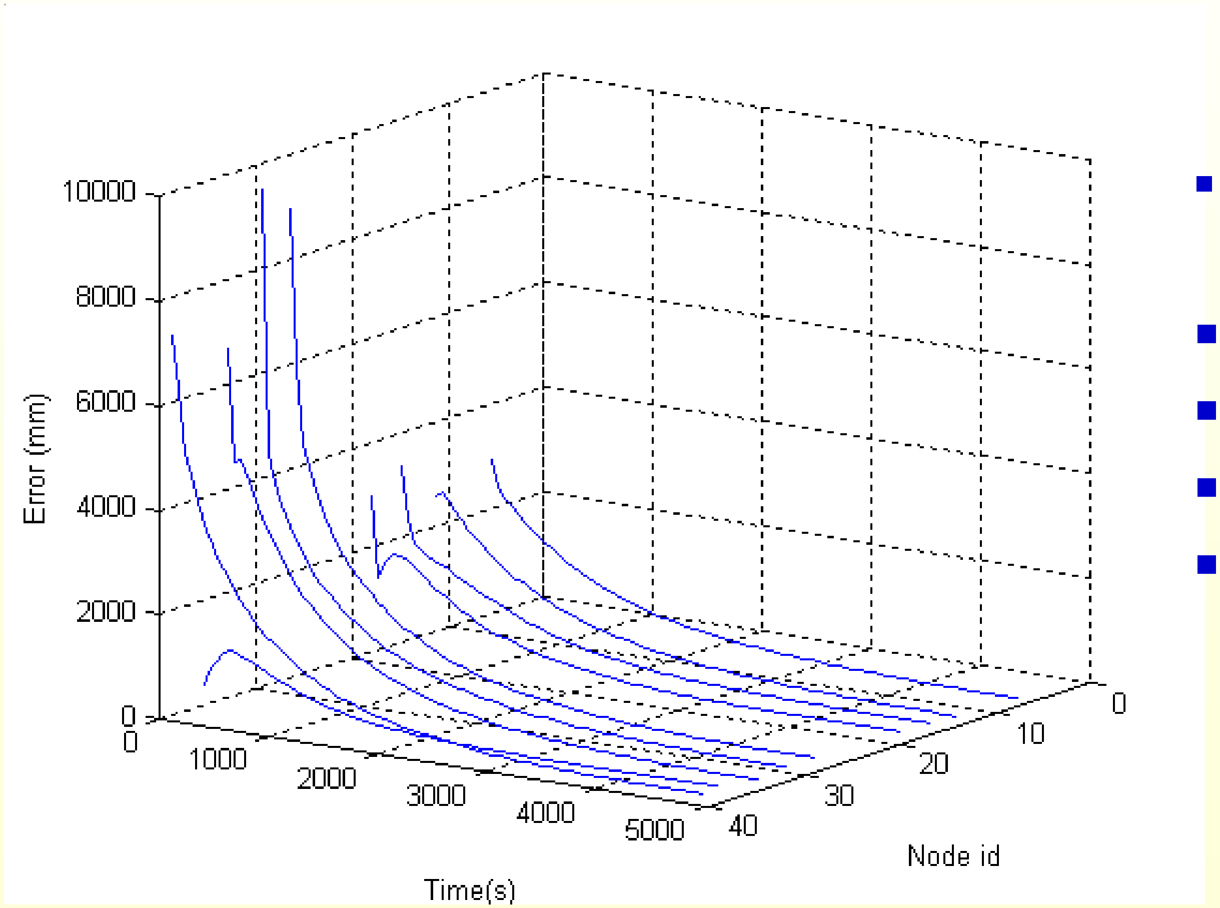




Distributed has reduced cost

Even sharing of communication cost

Satisfy Global Constraints with Local Computation



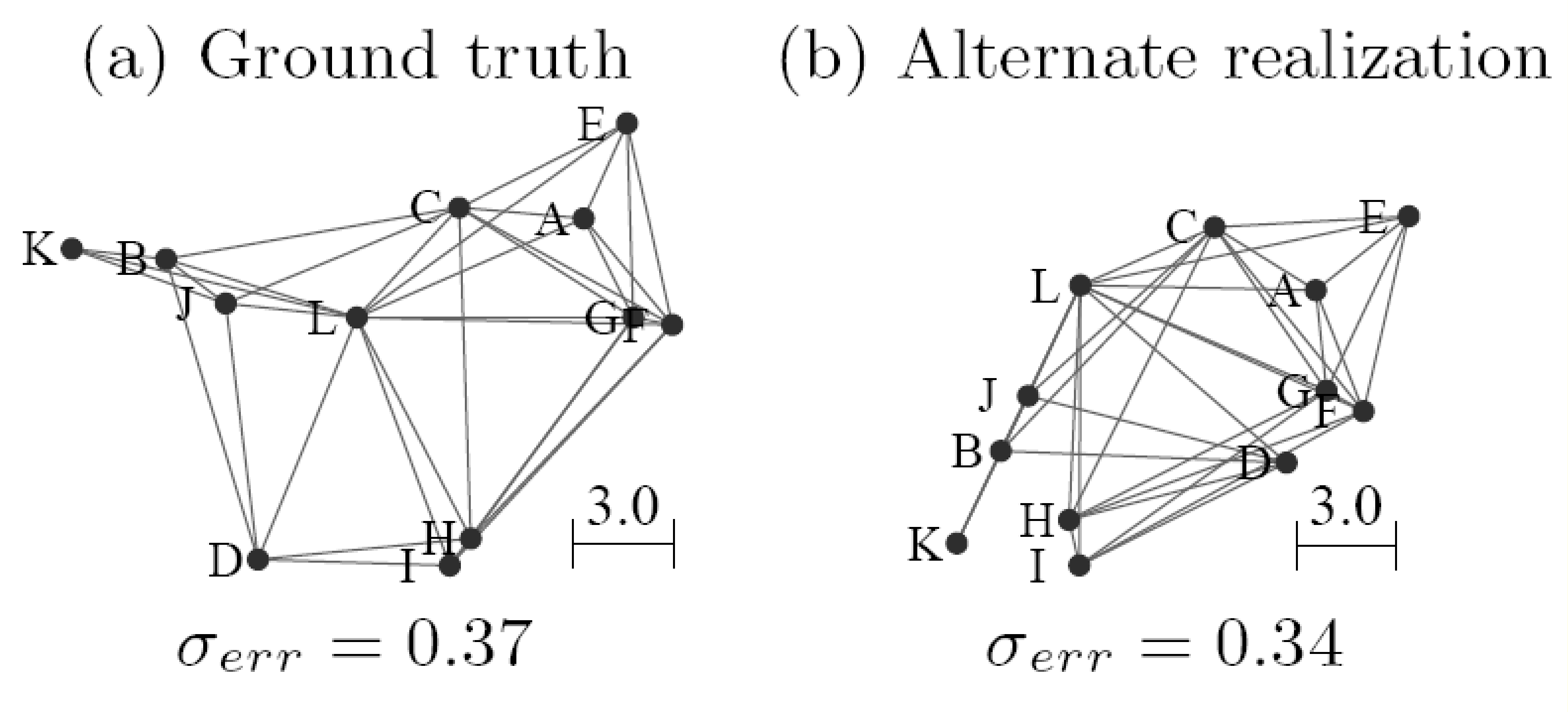
- From SensorSim simulation
- 40 nodes, 4 beacons
- IEEE 802.11 MAC
- 10Kbps radio
- Average 6 neighbors per node

However, optimization does address:

Ambiguities in localization:
when do we have enough distance constraints to localize?

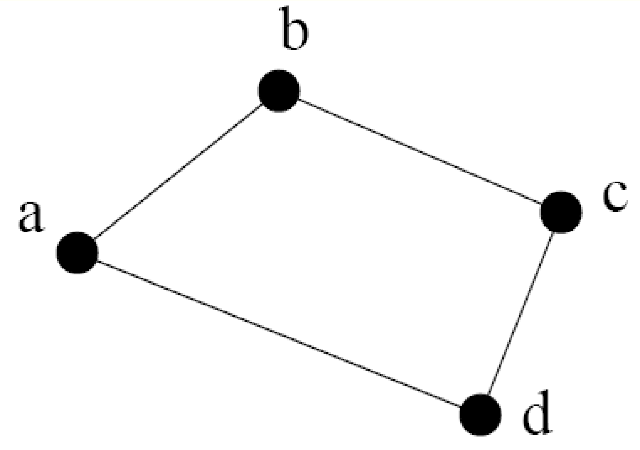
Ambiguity in Localization

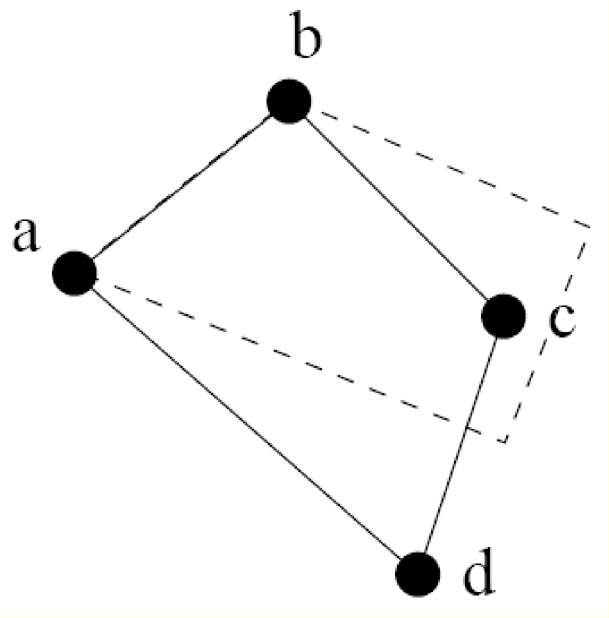
- Same distances, different realizations



Non-Rigidity: Continuous Deformation

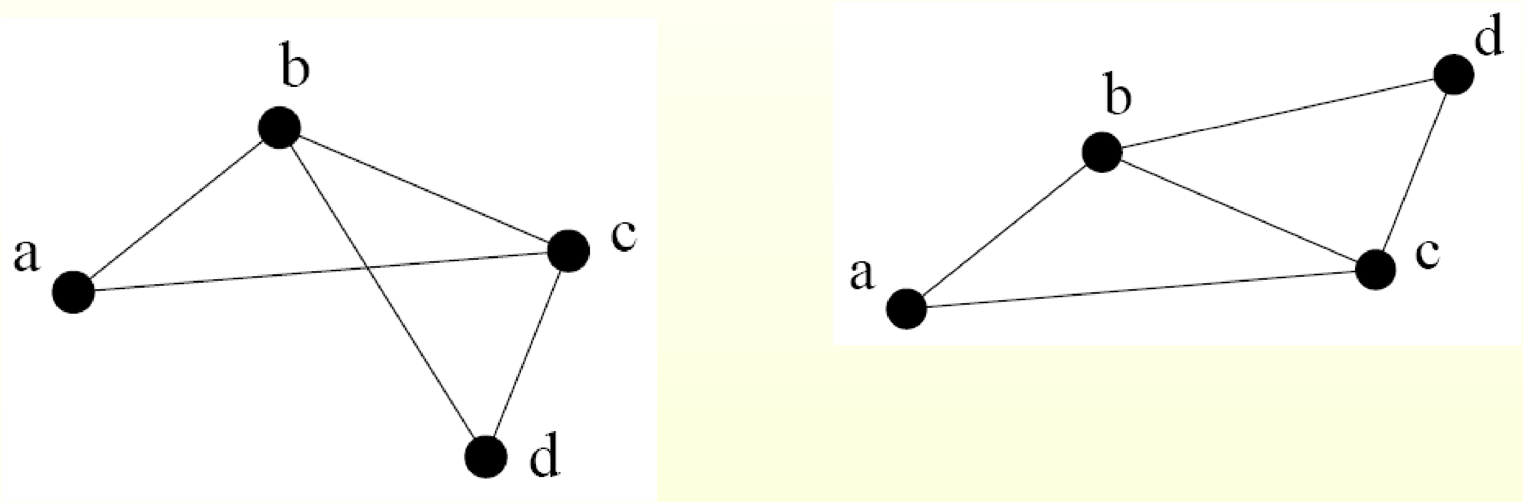
- Nodes can move continuously without violating the distance constraints





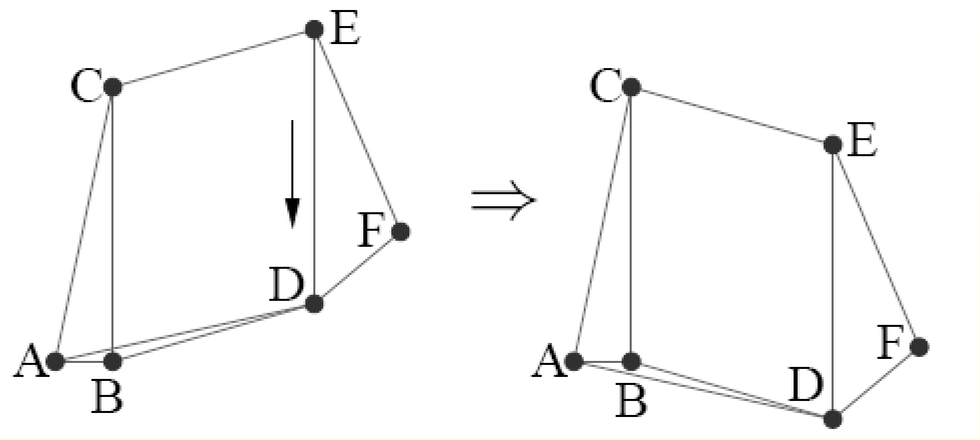
Non-Rigidity: Flip

- No continuous deformation, but subject to global flips



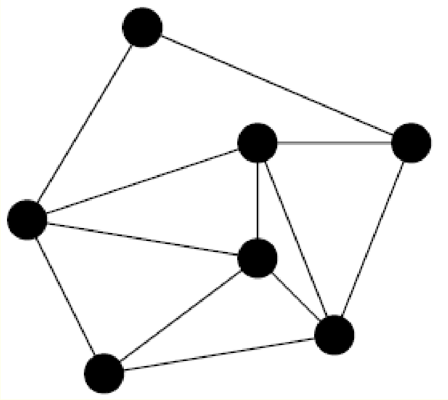
Discontinuous Flex Ambiguity

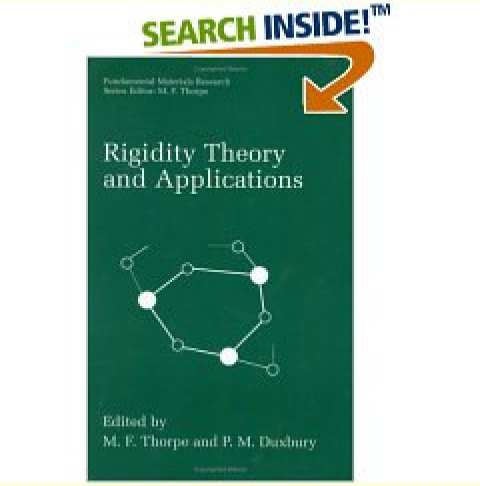
- Remove AD, flip ABD up, insert AD
- No continuous deformation in between
- But both are valid realization of the distances



Rigidity Theory

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





SEARCH INSIDE!™

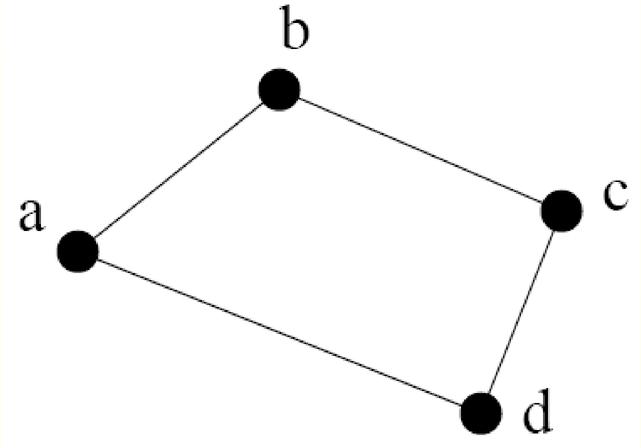
Rigidity Theory
and Applications

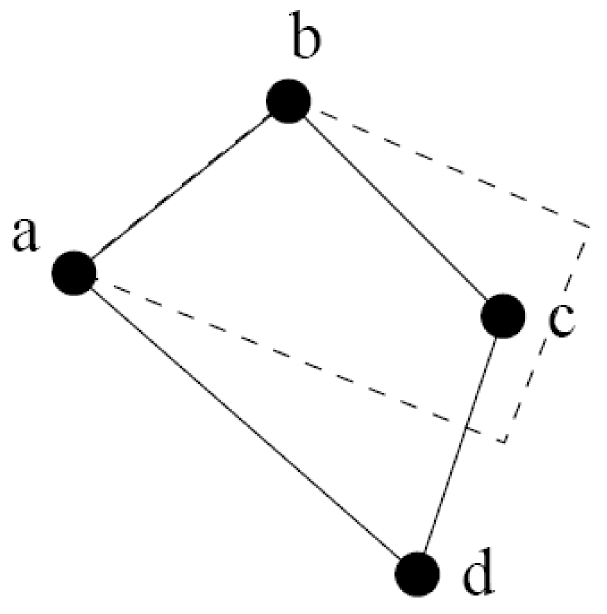
Edited by
M. F. Thorpe and P. M. Duxbury

Rigidity Theory and Applications
by M.F. Thorpe (Editor), P.M. Duxbury (Editor)
Plenum Publishers

Rigidity Theory

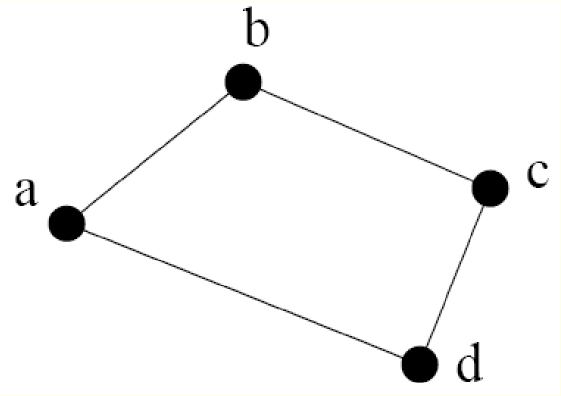
- Given a set of rigid bars connected by hinges, rigidity theory studies whether they can be moved continuously



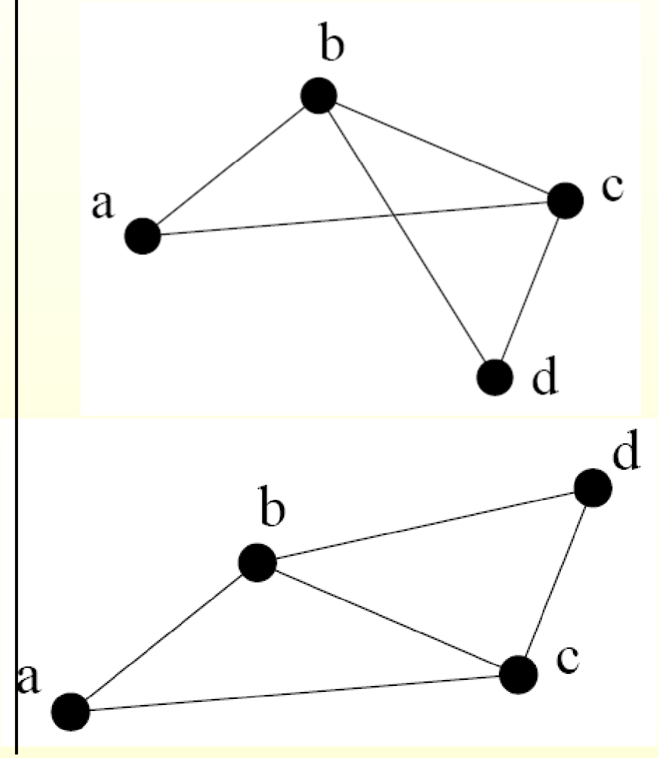


Types of Rigidity

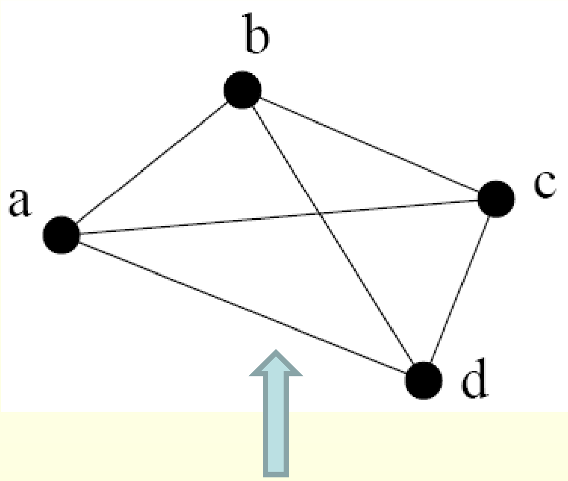
Not rigid



Rigid=
No continuous
deformation



Globally rigid=
unique realization



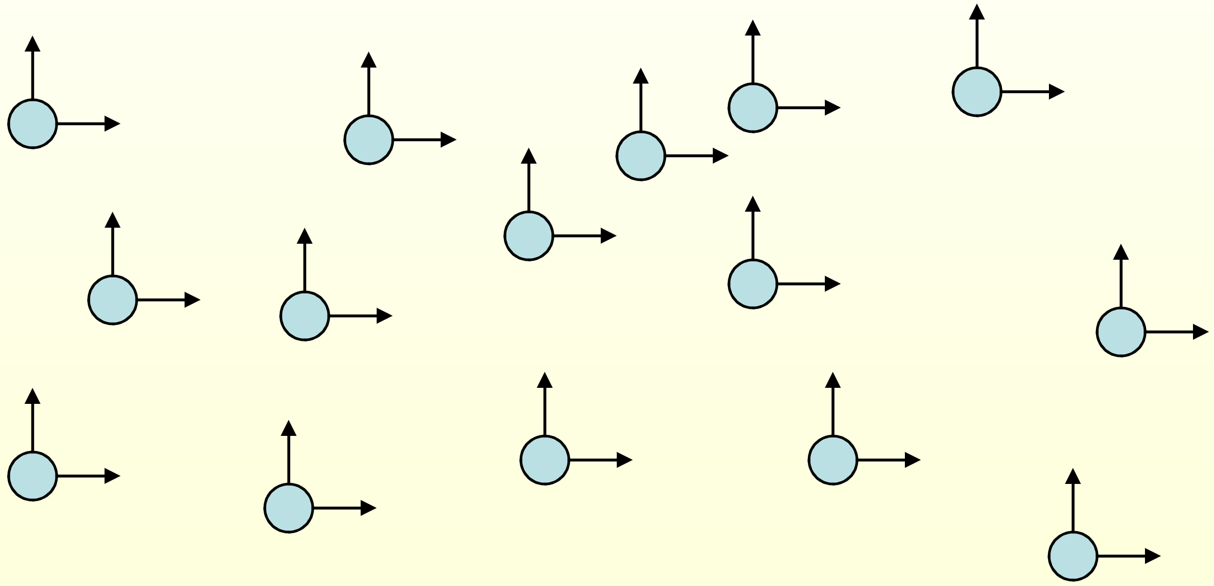
For localization,
this is what we want!

Degrees of Freedom

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

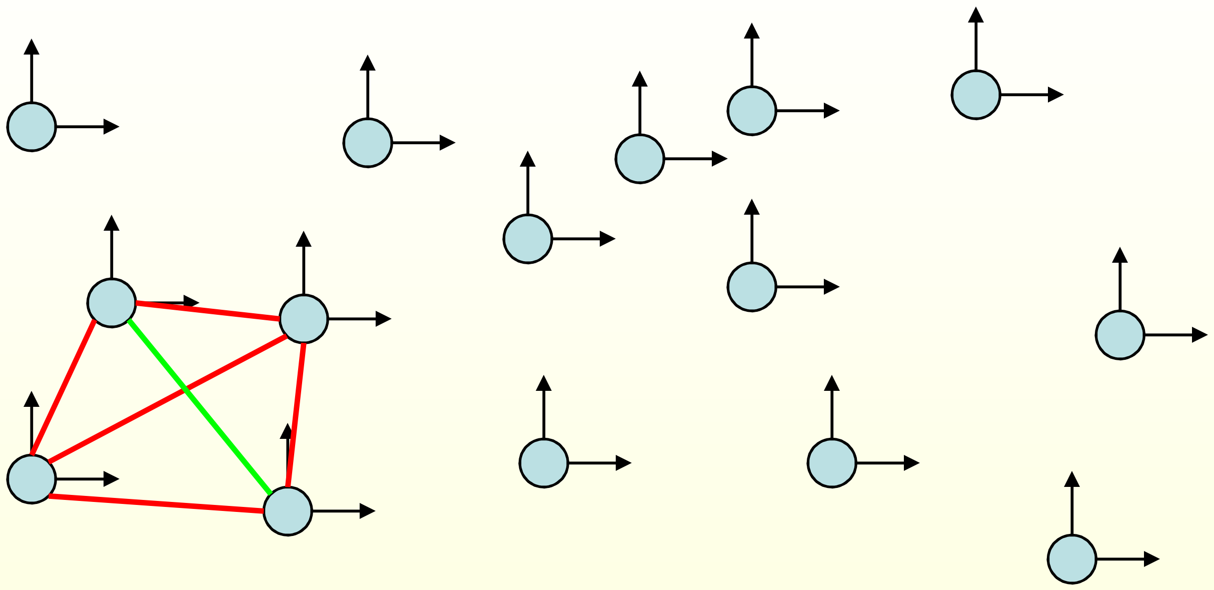
==

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



Total degrees of freedom: $2n$

Edge Counting

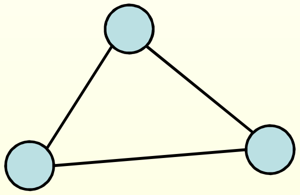


Each edge can remove a single degree of freedom

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

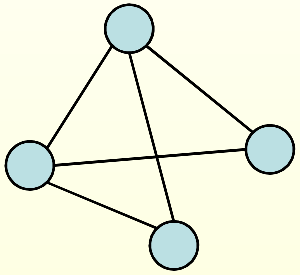
Are 2*n*-3 Edges Sufficient?

$n = 3,\ 2n-3 = 3$



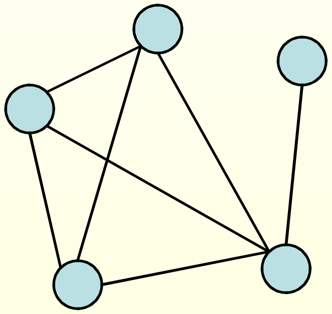
yes

$n = 4,\ 2n-3 = 5$



yes

$n = 5,\ 2n-3 = 7$



no

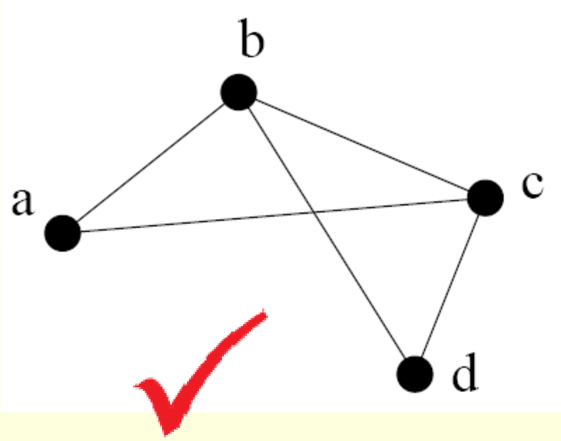
Further Intuition

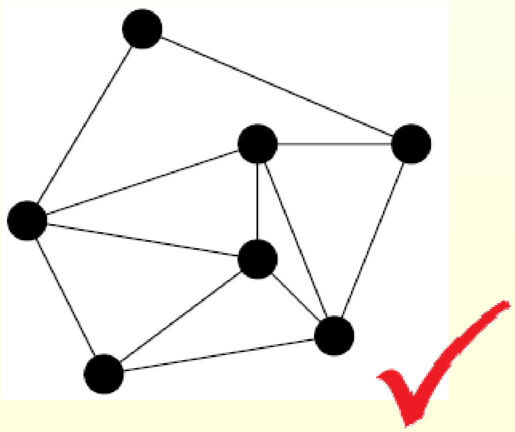
- Need at least *2n-3* well-distributed edges
- If a subgraph has more edges than necessary, some edges are redundant
- Non-redundant edges are independent, i.e., they remove a different degree of freedom each
- Therefore, 2*n*-3 independent edges guarantee rigidity

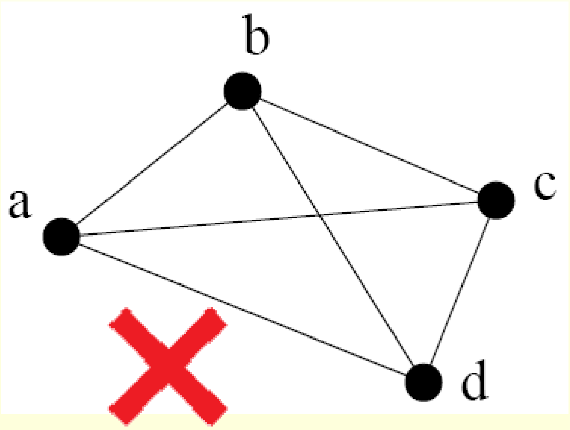
Laman Condition

Laman Graph: it has $2n-3$ edges and no subgraph of k vertices has more than $2k-3$ edges.

Laman Condition: A graph is rigid if it contains a Laman graph.







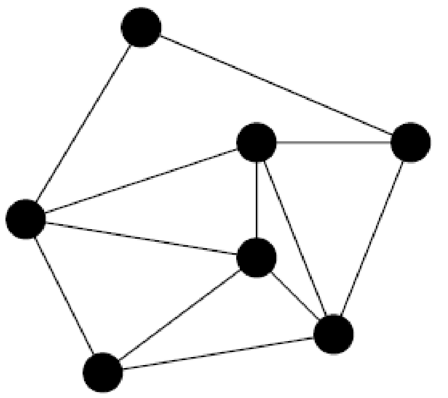
How does a Laman graph look like?

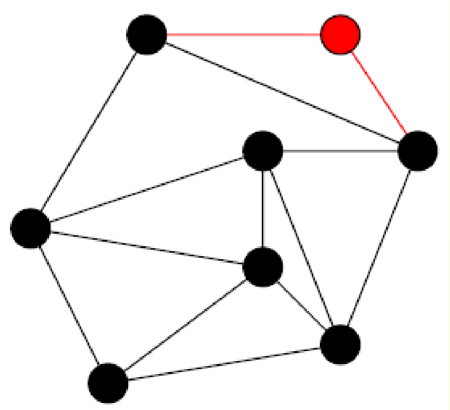
Henneberg Constructors

- Henneberg constructions (Tay-Whiteley): inductive, add 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 them, via three edges. Remove an old edge between the three endpoints.

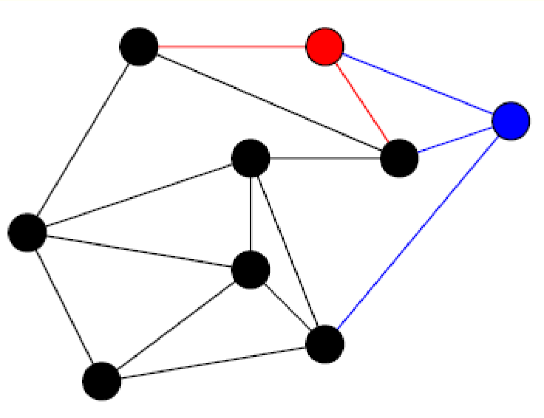
Henneberg Constructors

- 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 them, via three edges. Remove an old edge between the three endpoints.





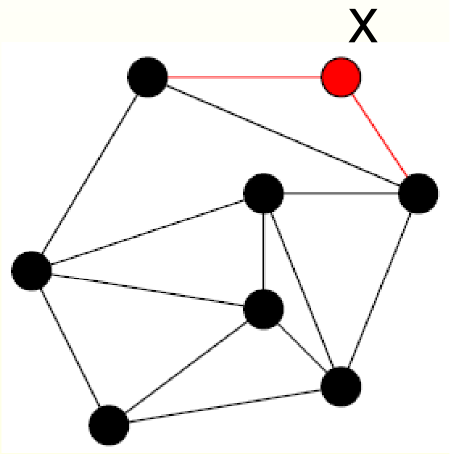
Type I



Type II

Henneberg → Laman

Claim: A graph constructed by Henneberg steps is Laman.



Proof: By induction. Suppose the current graph G is Laman with n vertices, 2n-3 edges.

Type I: Add node x. Now n+1 vertices, and 2n-3+2=2(n+1)-3 edges.

Similarly, for a subgraph with k nodes, if it does not include x, by the induction hypothesis, there are ≤ 2k-3 edges.

If the subgraph includes x, for the other k-1 nodes, there are at most 2(k-1)-3 edges between them (induction hypothesis), in total there are ≤ 2(k-1)-3 +2 = 2k-3 edges

Henneberg → Laman

Type II: Add node x. We have n+1 vertices, and 2n-3+3-1=2(n+1)-3 edges.

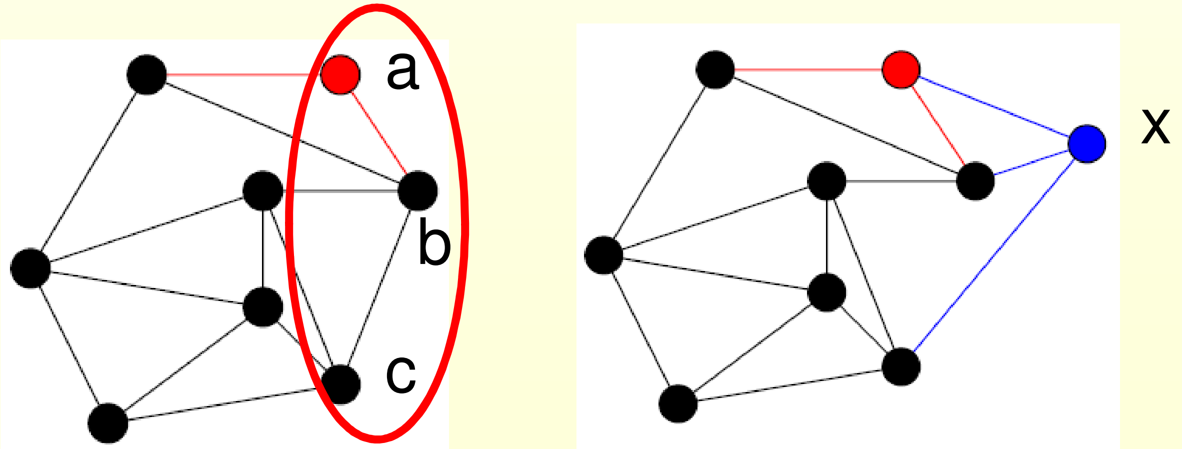
For a subgraph with k nodes, if it does not include x, by the induction hypothesis, there are ≤ 2k-3 edges.

If the subgraph includes x, for the other k-1 nodes, there are at most

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

Add x, for case 1, there are ≤ 2(k-1)-3 +2 = 2k-3 edges.

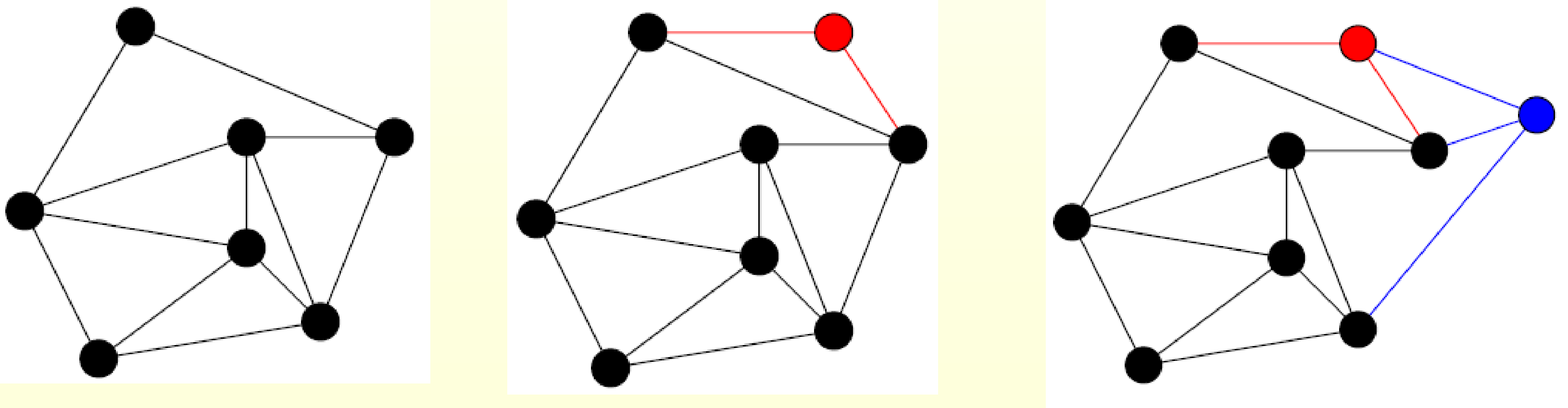
For case 2, there are ≤ 2(k-1)-4 +3 = 2k-3 edges. #



Laman ➔ Henneberg

Claim: Each Laman graph has a Henneberg construction.

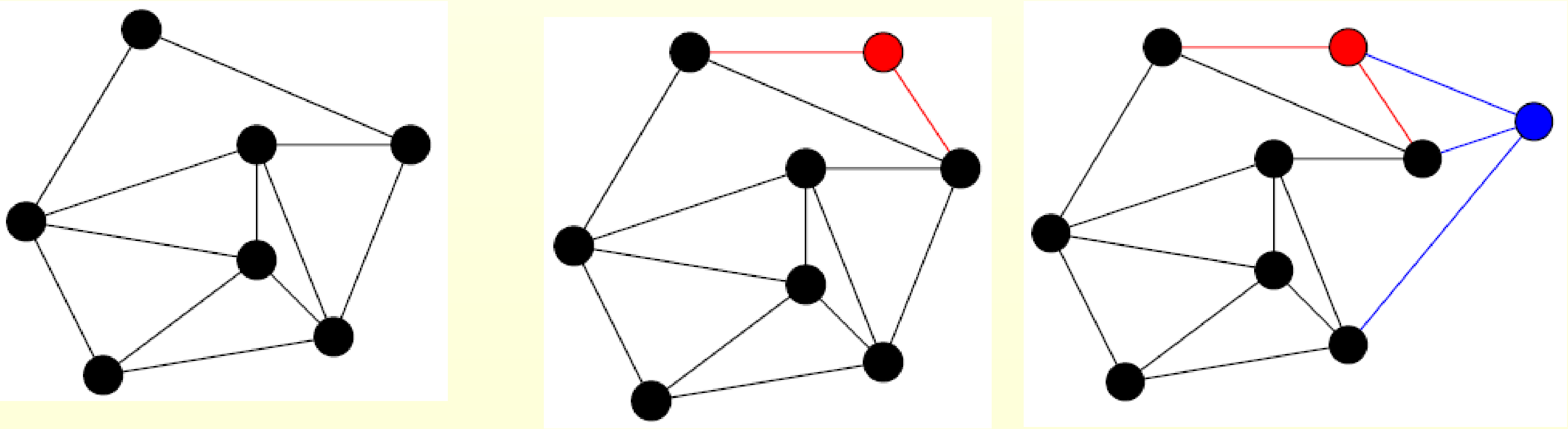
- If m=2n-3, there exists at least one vertex of degree 2 or 3.
- Otherwise, all nodes have degree 4. Thus we have at least 4n/2=2n edges. ➔ contradiction.



Laman → Henneberg

Claim: Each Laman graph has a Henneberg construction.

- If degree 2: remove the vertex and its adjacent edges (Type I step in reverse)
- If degree 3: remove the vertex and the edges to its three neighbors {a, b, c}. They can't span all three edges (else violate 2k-3 for k=4, e.g., {a, b, c, x}). Put one edge between them. (Type II step in reverse).
- Laman still holds, so we can continue.



Hennerberg Construction Implies …

- The subgraph examined by iterative multi-lateration is rigid
  - Start with three nodes (with known locations)
  - Add 1 new node with 3 edges to existing nodes
- Such a graph is named “trilateration graph”

Side Note: Laman Theorem in 3D?

Laman condition in 3D?

A graph is generically minimally rigid in 3D if and only if it has *3n-6* edges and no subgraph of *k* vertices has more than *3k-6* edges?

Unfortunately, the condition is necessary but not sufficient.

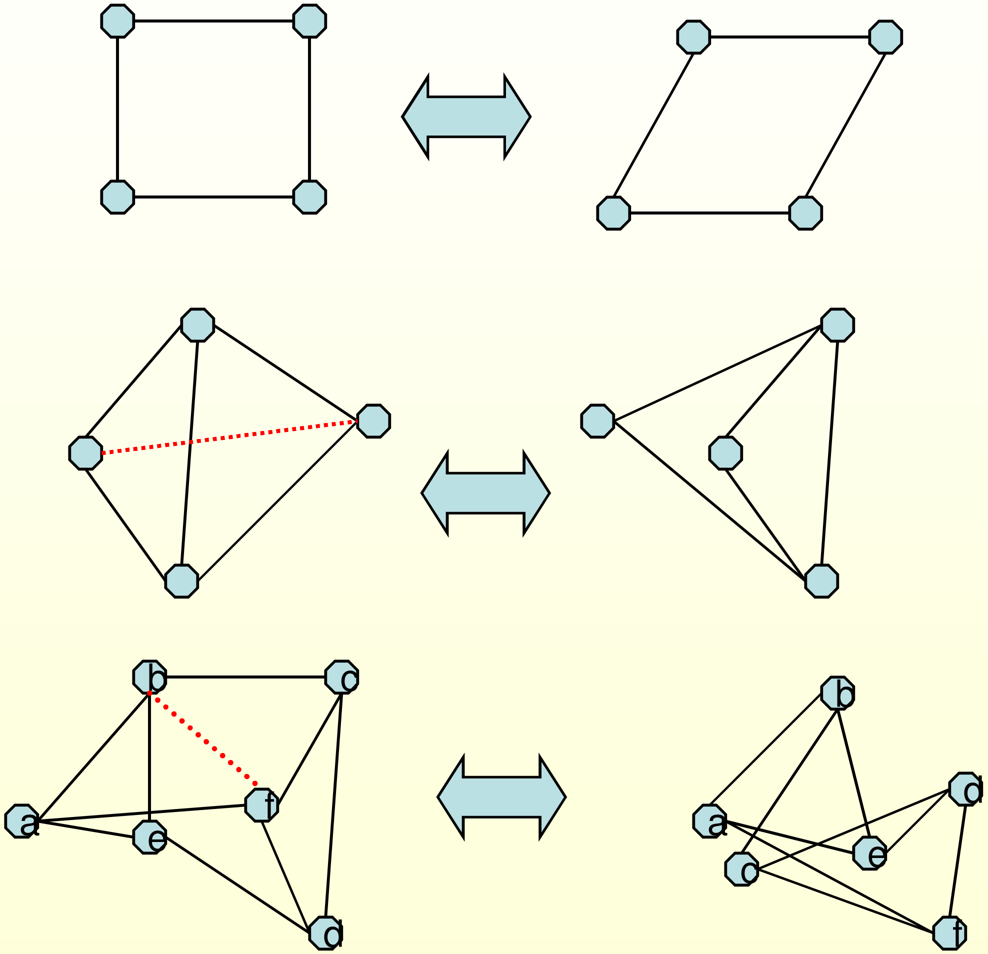
It's a long-standing open problem to detrmine combinatorial conditions for rigidity in 3D

Rigidity Summary

- 2D Rigidity, Laman graph

- But, rigidity does not mean global rigidity
- For localization, we really want global rigidity

Global Rigidity: Three Conditions



Solution:

G must be *rigid*

G must be 3-connected, i.e. Connected after removal of 2 vertices.

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

Two approaches

- Local optimization:
  - Avoid flip ambiguities of iterative trilateration
- Global optimization:
  - Multi-dimensional scaling

The End