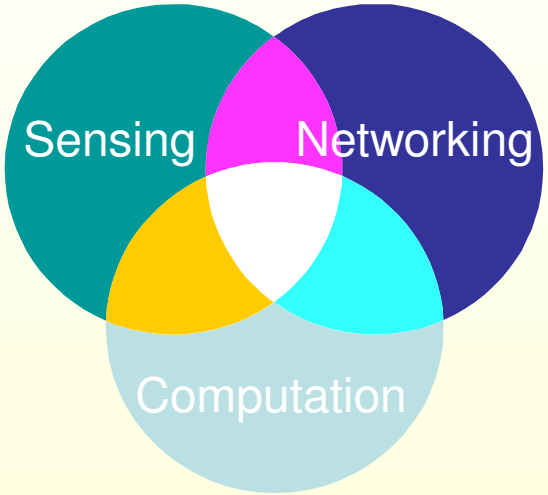
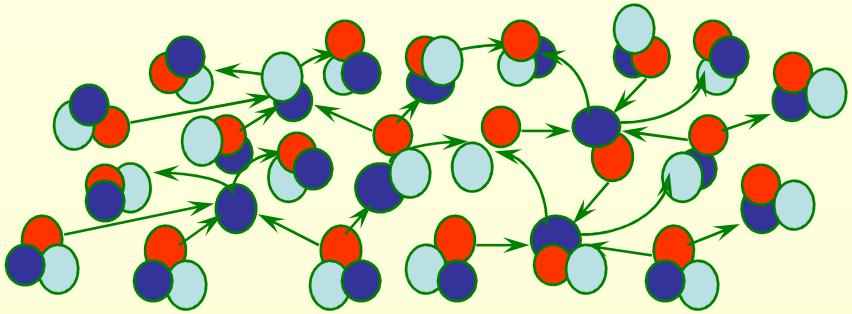
CS321: Localization II



Leonidas Guibas
Computer Science Dept.
Stanford University



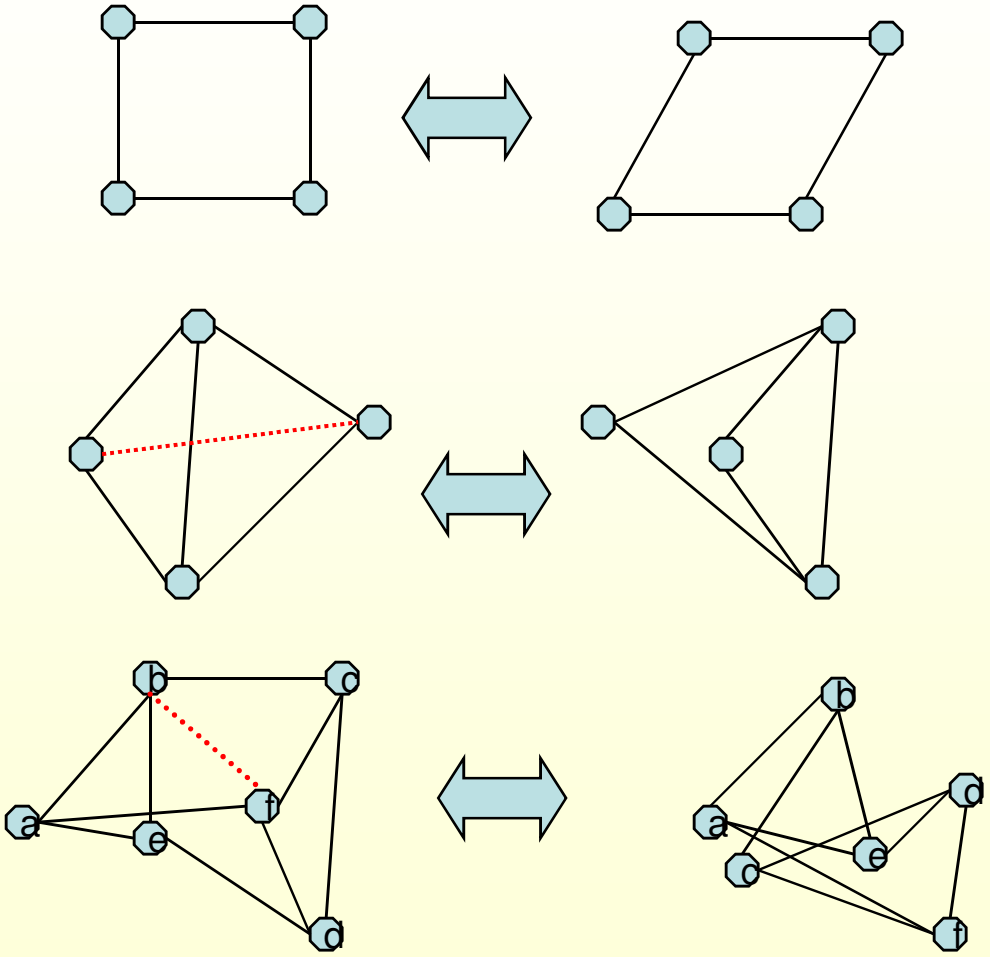
Towards Global Rigidity

Rigidity Summary

- 2D Rigidity, Laman graph

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

Global Rigidity



Solution:

G must be *rigid*

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

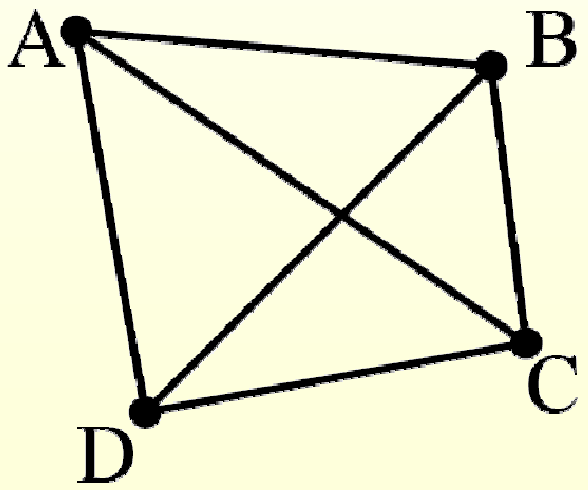
G must be *redundantly rigid:* It must remain rigid upon removal of any single edge

Two Approaches

- Local optimization:
  - Improve the robustness of iterative trilateration
- Global optimization:
  - Multi-Dimensional Scaling (MDS)
  - Related to Principal Component Analysis (PCA)

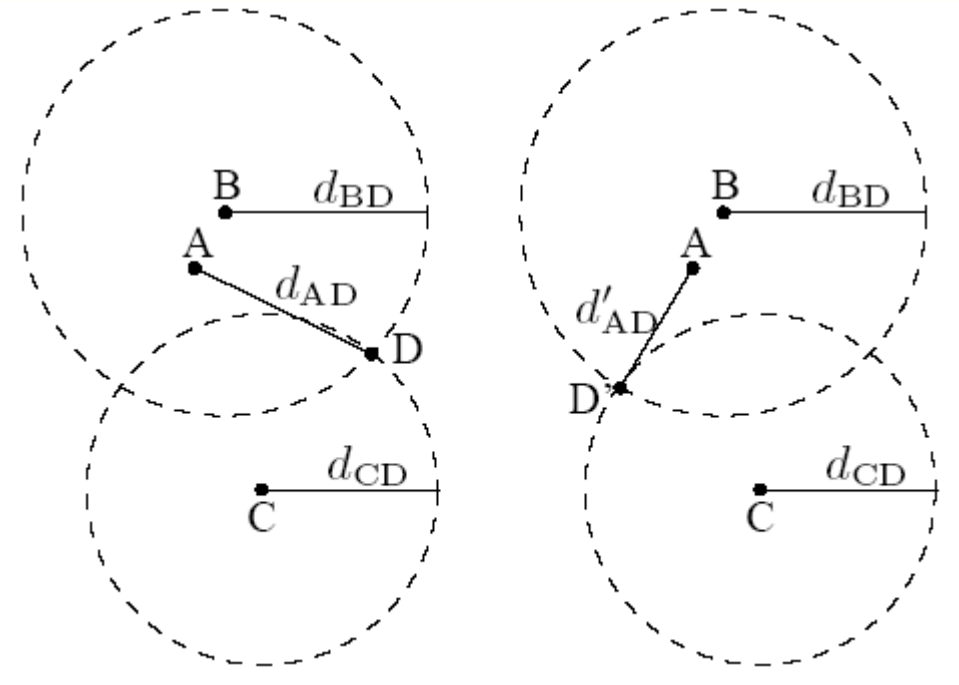
Trilateration is Based on Quadrilaterals

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



Trilateration with Noise…

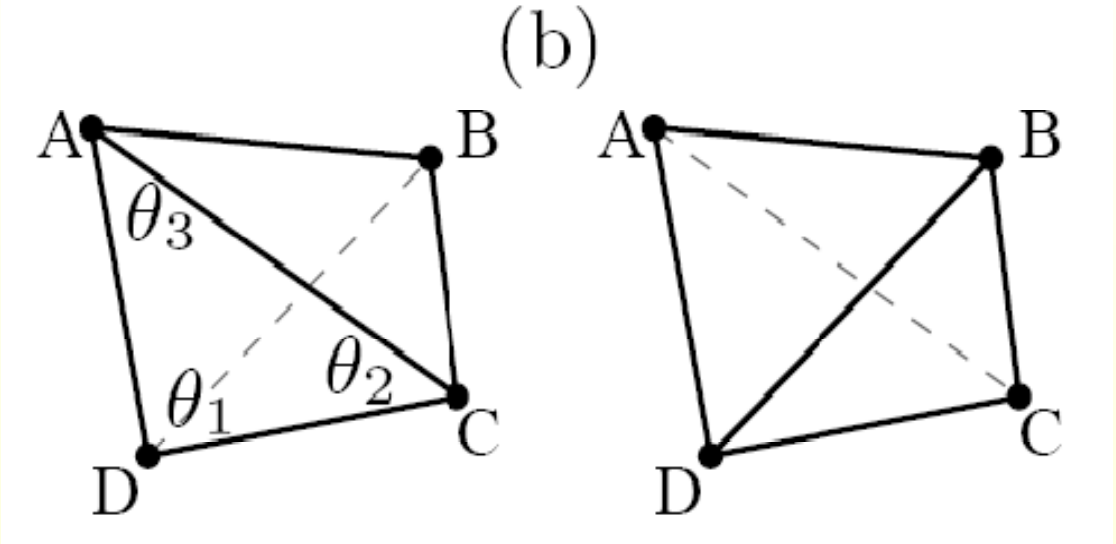
- With noisy measurements, trilateration can have flip ambiguities



Conditions for Flip Ambiguity

- There are 4 triangles in a quadrilateral

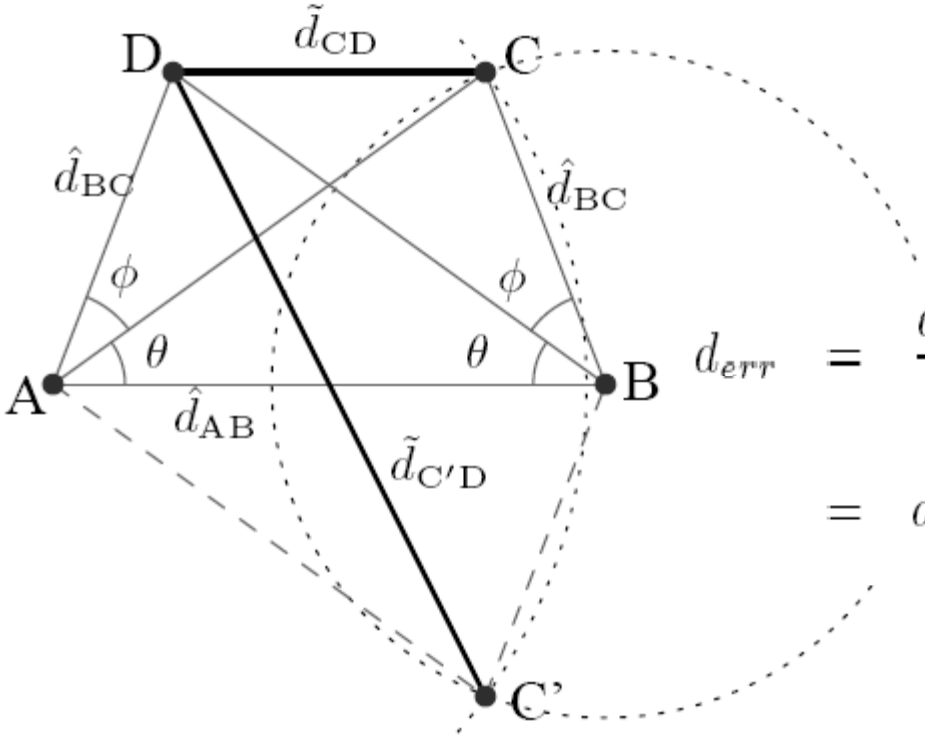
Flip ambiguity, when the smallest angle is too small!



- Assume noise has a normal distribution
- Only accept quadrilaterals with sufficient large min angle

Robust Quadrilaterals

- Assume we do trilateration to locate C
- First ignore D, then C and C' are possible locations
- If CD and C'D are quite different, then we can verify which of C, and C' is correct.



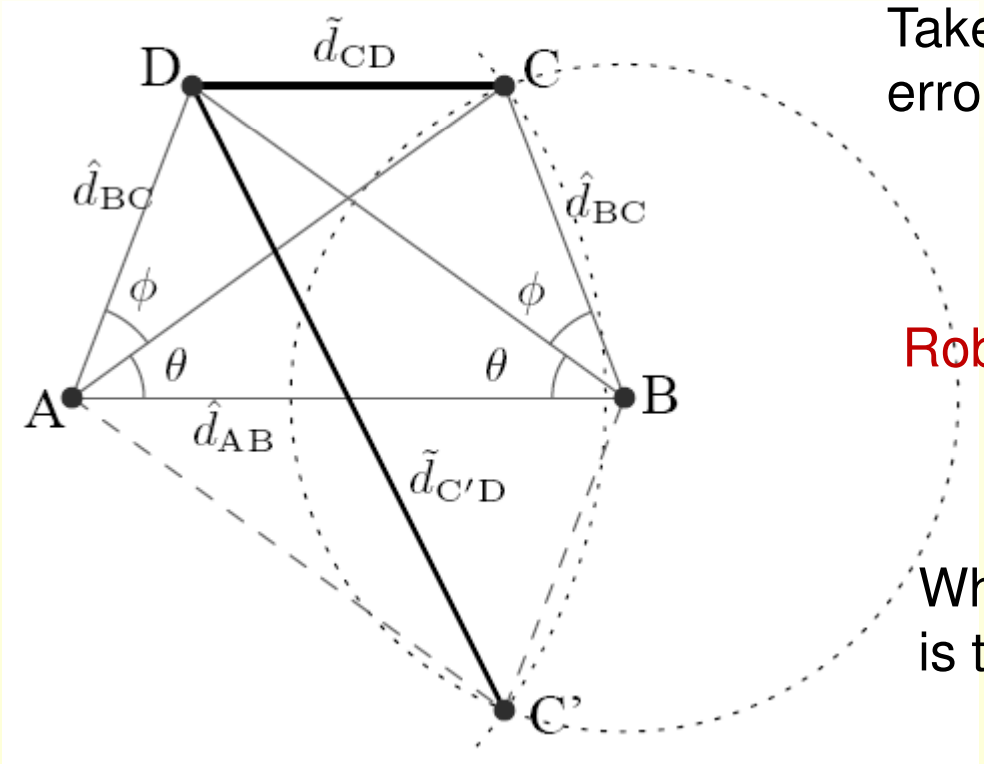
To have an ambiguity, the error in measuring distances CD, C'D must be at least

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

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

Robust Quadrilaterals

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



Take derivative to minimize the error: when $\phi=\pi/2-2\theta$.

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

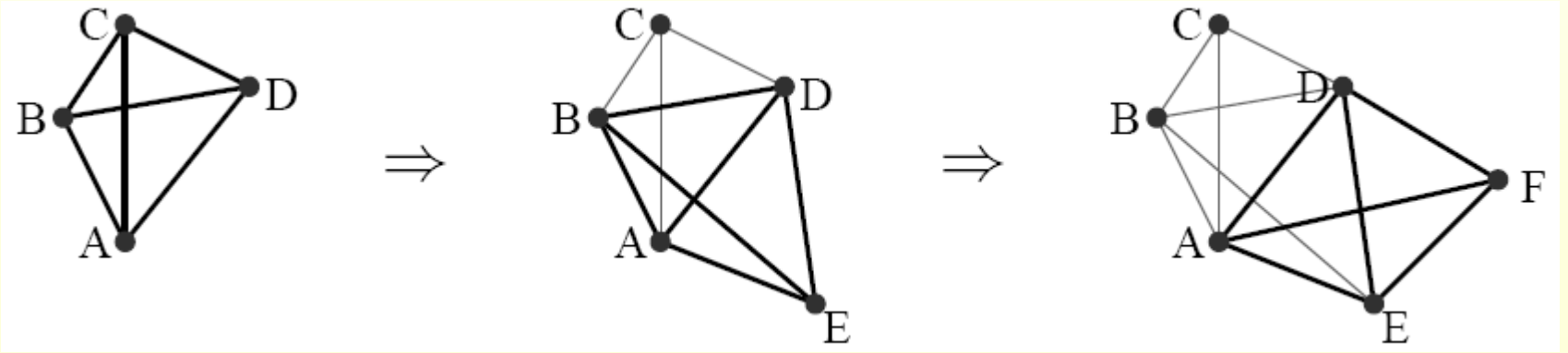
Robust quadrilateral satisfies

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

Where *b* is the shortest side and θ is the smallest angle in the quad

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



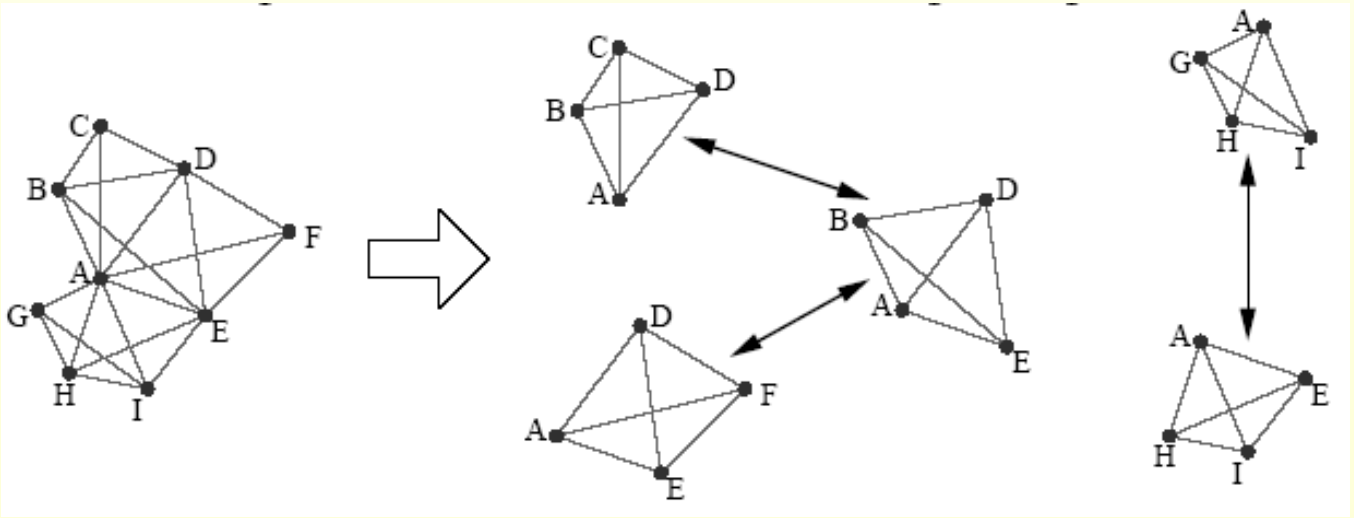
The Algorithm

- 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
  - Transform to a global coordinate system.

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*

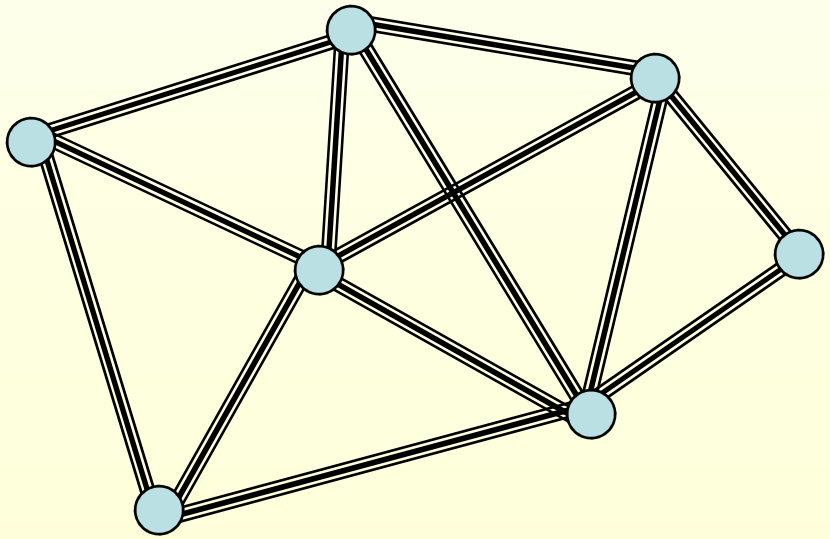


Phase II: Cluster Optimization

- For each cluster around node *x*, refine the position estimates, for example, by mass-spring relaxation
- Optional
- Reduces accumulated error

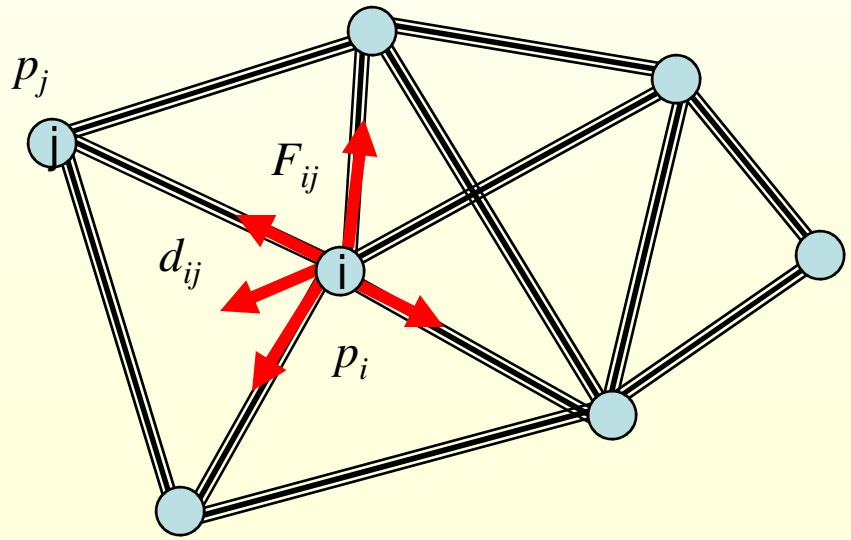
Mass-Spring Systems

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



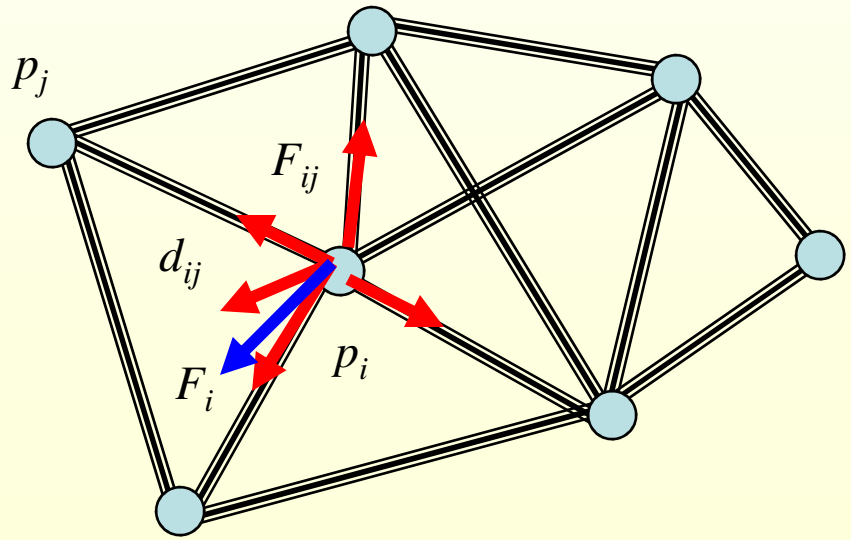
Mass-Spring Systems

- 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 $p_i p_j$



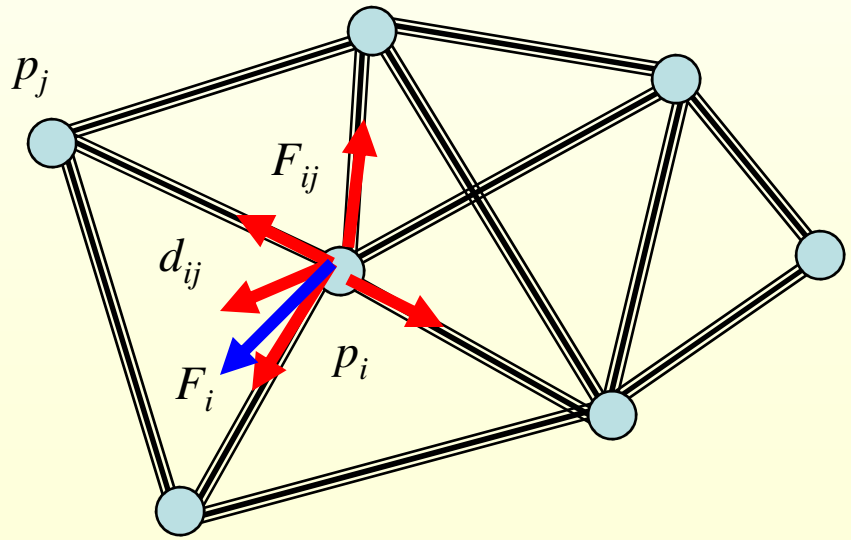
Mass-Spring Systems

- 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 Systems

- 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 Systems

- Advantage: Naturally a distributed algorithm
- Problem 1: may stuck in local minima
  - Need to start from a reasonably good initial estimation, e.g., the iterative multi-lateration
  - Typically not used alone
- Problem 2: not robust to outliers
  - If one measurement is off too much, the error gets distributed everywhere in the system

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

Rigid Alignment, Given Correspondences

- We are given two sets of corresponding points x_1, x_2, ..., x_n and y_1, y_2, ..., y_n in E^2. We wish to compute the rigid transform T that best aligns x_1 to y_1, x_2 to y_2, ..., and x_n to y_n.
- We define the error to be minimized by

$$\min_T \sum_{i=1}^{n} ||T(x_i) - y_i||^2$$

- Old Problem:
  - Known and solved as the *orthogonal Procrustes problem* in Factor Analysis (Statistics) [Shönemann, 1966]
  - Known and solved as the *absolute orientation problem* in Photogrammetry [Horn, 1986]
  - Also in robotics, graphics, medical image analysis, statistical theories of shape, etc ...

SVD-based Solution

- A rigid motion T is a combination of a translation a and a rotation R, so that $T(x) = R(x) + a$.
- If we place the origin of our coordinate system at the mean of the x_i's, then the quantity to be minimized simplifies to (up to some constants):

$$\min_{a,R} \left(\sum_{i=1}^{n} |y_i - a|^2 - 2 \sum_{i=1}^{n} \langle R(x_i), y_i \rangle \right)$$

- Note that the translational and rotational parts factor. The translational part a clearly has to be

$$a = \frac{1}{n} \sum_{i=1}^{n} y_i$$

The centroids of the two point sets have to be aligned!

Translation Calculation

$$\min_a \sum (x_i + a - y_i)^2$$

$$\frac{d}{da}\left[nt^2 + 2\sum (x_i - y_i)a + \sum (x_i - y_i)^2\right] = 0$$

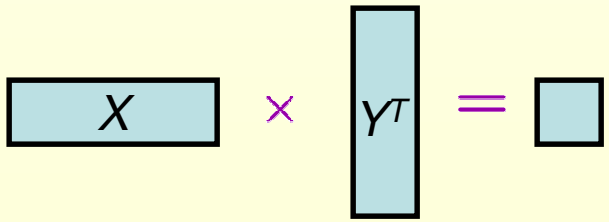
$$a = \frac{\sum (y_i - x_i)}{n}$$

The Rotation Part

- Define

$$\begin{aligned} \overline{x} &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ \overline{y} &= \frac{1}{n}\sum_{i=1}^{n} y_i \\ X &= [x_1 - \overline{x}, \ldots x_n - \overline{x}]^T \\ Y &= [y_1 - \overline{y}, \ldots y_n - \overline{y}]^T \end{aligned}$$

- Here *X* and *Y* are 2 by *n* matrices.



- Now compute the SVD*

$$XY^T = UDV^T \quad (2 \times 2)$$

- *U* and *V* are 2 by 2 orthogonal matrices, and *D* is a diagonal matrix with decreasing non-negative entries along the diagonal (the singular values).
- Define *S* by

$$S = \begin{cases} I, & \text{if } \det U \det V = 1 \\ \text{diag}(1, \ldots, 1, -1), & \\ \text{otherwise} & \end{cases}$$

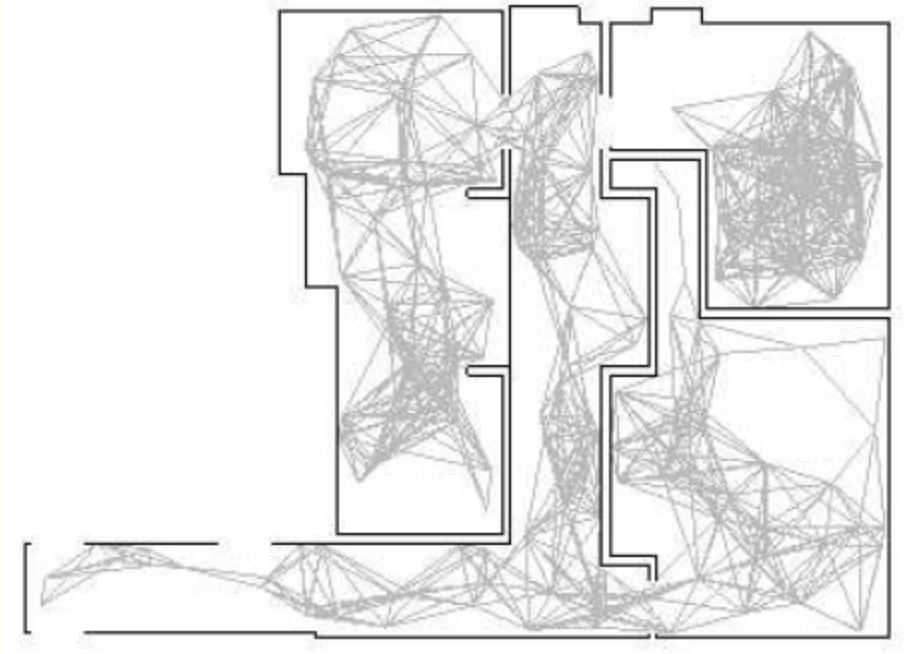
- Then $\boxed{R = USV^T}$

O(*n*) algorithm!

*SVD = singular value decomposition

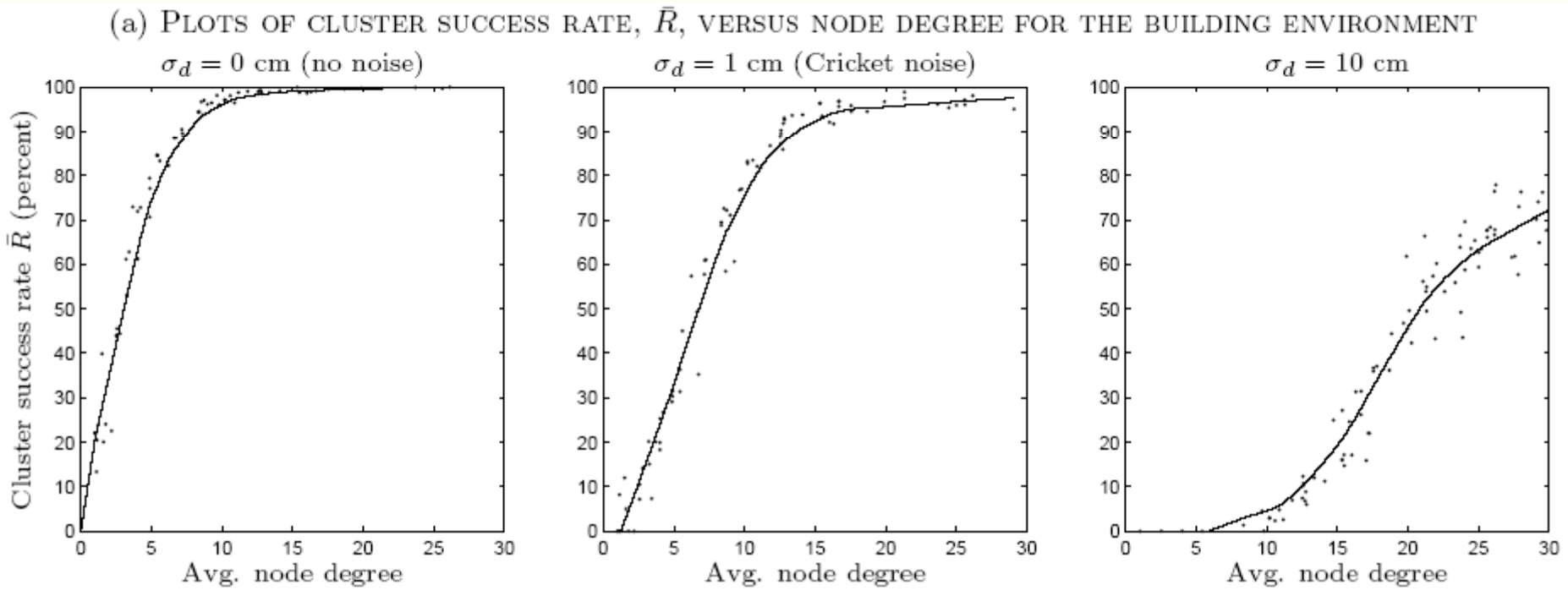
Simulations

- 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 must be ≥ 15 to achieve 100% localization
- Not good for sparse networks
- The avg degree $\cong 6$ for best throughput of the network

Observations

- Localization algorithm performs poorly when the graph is sparse
- Negative constraints are not used --- non-neighbors should be sufficiently far.

- Unit disk graph embedding is NP-hard
  - Given measurements of edges of a unit disk graph, it is NP-hard to find a realization in the plane

Two Approaches

- Local optimization:
  - Improve the robustness of iterative trilateration
- Global optimization:
  - Multi-Dimensional Scaling (MDS)
  - Related to Principal Component Analysis (PCA)

Multi-Dimensional Scaling (MDS)

- Input:
  - A distance matrix *M* on *n* nodes (actually squared distances)
- Output:
  - Embedding of nodes in R^d, s.t. their inter-distances approximate entries in *M*
- Observations
  - If the distances are accurate, MDS recreates the configuration.
  - Also works when the distances are not Euclidean -- then MDS recovers the “best fit”
  - Widely used in social sciences for visualization and similarity-based clustering

MDS Formulation

- M represents pair-wise squared distances between *n* input points from a high-dimensional space

$$M = \left(\text{dist}^2(\mathbf{x_i}, \mathbf{x_j})\right)_{n \times n}$$

- Now, find *n* points in low-dimensional space R^d, so that their distance matrix is as close as possible to M.

MDS – the Math Details

We look for X',

$$X' = \begin{pmatrix} | & & | \\ \mathbf{x_1}' & \cdots & \mathbf{x_n}' \\ | & & | \end{pmatrix} \in R^{d \times n}$$

such that $\| M' - M \|$ is as small as possible, where

$$M' = \left(\mathrm{dist}^2(\mathbf{x_i}', \mathbf{x_j}') \right) = \left(\left\| \mathbf{x_i}' - \mathbf{x_j}' \right\|^2 \right) \in R^{n \times n}$$

M' is the Euclidean distances matrix for points $\mathbf{x_i}'$

MDS – the Math Details

Ideally,

we want:

$$M' = \left(\left\| \mathbf{x}_i' - \mathbf{x}_j' \right\|^2 \right) = M$$

$$\left(< \mathbf{x}_i' - \mathbf{x}_j', \ \mathbf{x}_i' - \mathbf{x}_j' > \right) = M$$

$$\left(\| \mathbf{x}_i' \|^2 + \| \mathbf{x}_j' \|^2 - 2 < \mathbf{x}_i', \mathbf{x}_j' > \right) = M$$

$$\begin{pmatrix} \| \mathbf{x}_1' \| & \| \mathbf{x}_1' \| & \dots & \| \mathbf{x}_1' \| \\ \| \mathbf{x}_2' \| & \| \mathbf{x}_2' \| & \dots & \| \mathbf{x}_2' \| \\ & & \vdots & \\ \| \mathbf{x}_n' \| & \| \mathbf{x}_n' \| & \dots & \| \mathbf{x}_n' \| \end{pmatrix} \quad \begin{pmatrix} \| \mathbf{x}_1' \| & \| \mathbf{x}_2' \| & & \| \mathbf{x}_n' \| \\ \| \mathbf{x}_1' \| & \| \mathbf{x}_2' \| & & \| \mathbf{x}_n' \| \\ \vdots & \vdots & \dots & \vdots \\ \| \mathbf{x}_1' \| & \| \mathbf{x}_2' \| & & \| \mathbf{x}_n' \| \end{pmatrix}$$

want to get rid of these

$$\begin{pmatrix} — & \mathbf{x}_1' & — \\ & \vdots & \\ — & \mathbf{x}_n' & — \end{pmatrix} \begin{pmatrix} | & & | \\ \mathbf{x}_1' & \dots & \mathbf{x}_n' \\ | & & | \end{pmatrix}$$

$X'^T \qquad X'$

MDS – the Math Details

Trick: use the “magic matrix” J :

$$J = \begin{pmatrix} 1 & -\frac{1}{n} & \cdots & -\frac{1}{n} \\ -\frac{1}{n} & 1 & -\frac{1}{n} & -\frac{1}{n} \\ \vdots & & \ddots & \vdots \\ -\frac{1}{n} & \cdots & -\frac{1}{n} & 1 \end{pmatrix}_{n \times n}$$

$$\begin{pmatrix} a & a & \cdots & a \end{pmatrix} \cdot J = 0$$

$$J \cdot \begin{pmatrix} b \\ b \\ \vdots \\ b \end{pmatrix} = 0$$

MDS – the Math Details

Cleaning the system:

$$\times J \quad \left(\begin{array}{cccc} \|\mathbf{x}_1'\| & \|\mathbf{x}_1'\| & \dots & \|\mathbf{x}_1'\| \\ \|\mathbf{x}_2'\| & \|\mathbf{x}_2'\| & \dots & \|\mathbf{x}_2'\| \\ & & \vdots & \\ \|\mathbf{x}_n'\| & \|\mathbf{x}_n'\| & \dots & \|\mathbf{x}_n'\| \end{array} \right) + \left(\begin{array}{ccccc} \|\mathbf{x}_1'\| & \|\mathbf{x}_2'\| & & & \|\mathbf{x}_n'\| \\ \|\mathbf{x}_1'\| & \|\mathbf{x}_2'\| & & & \|\mathbf{x}_n'\| \\ \vdots & \vdots & & \dots & \vdots \\ \|\mathbf{x}_1'\| & \|\mathbf{x}_2'\| & & & \|\mathbf{x}_n'\| \end{array} \right) - 2X'^T X' = M \quad \times J$$

$$-2X'^T X' = JMJ$$

$$X'^T X' = -\tfrac{1}{2} JMJ =: B$$

$$\boxed{X'^T X' = B}$$

How to Find X'

We will use the spectral decomposition of B:

$$X'^T X' = B = \begin{pmatrix} | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_n \\ | & & | \end{pmatrix} \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} \begin{pmatrix} | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_n \\ | & & | \end{pmatrix}^T$$

$$X'^T X' = \underbrace{\begin{pmatrix} | & & | & \vdots & | \\ | & & | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_d & \vdots & \mathbf{v}_n \\ | & & | & & | \\ | & & | & \vdots & | \end{pmatrix}}_{n \times d} \underbrace{\begin{pmatrix} \sqrt{\lambda_1} & & & & \\ & \ddots & & & \\ & & \sqrt{\lambda_d} & & \\ & & & \ddots & \\ & & & & \sqrt{\lambda_n} \end{pmatrix}}_{d \times d} \underbrace{\begin{pmatrix} \sqrt{\lambda_1} & & & & \\ & \ddots & & & \\ & & \sqrt{\lambda_d} & & \\ & & & \ddots & \\ & & & & \sqrt{\lambda_n} \end{pmatrix}^T \begin{pmatrix} | & & | & \vdots & | \\ | & & | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_d & \vdots & \mathbf{v}_n \\ | & & | & & | \\ | & & | & \vdots & | \end{pmatrix}^T}_{X'}$$

$$X'^T \qquad\qquad X'$$

How to Find X'

So we find X' by throwing away the last $n-d$ eigenvalues

$$X' = \begin{pmatrix} \cdots\ \sqrt{\lambda_1}\,\mathbf{v_1}\ \cdots \\ \cdots \quad \cdots \quad \cdots \\ \cdots\ \sqrt{\lambda_d}\,\mathbf{v_d}\ \cdots \end{pmatrix}_{d \times n}$$

$$X' = \arg\min_{X'} \left\| X'^T X' - B \right\|_{L^2}$$

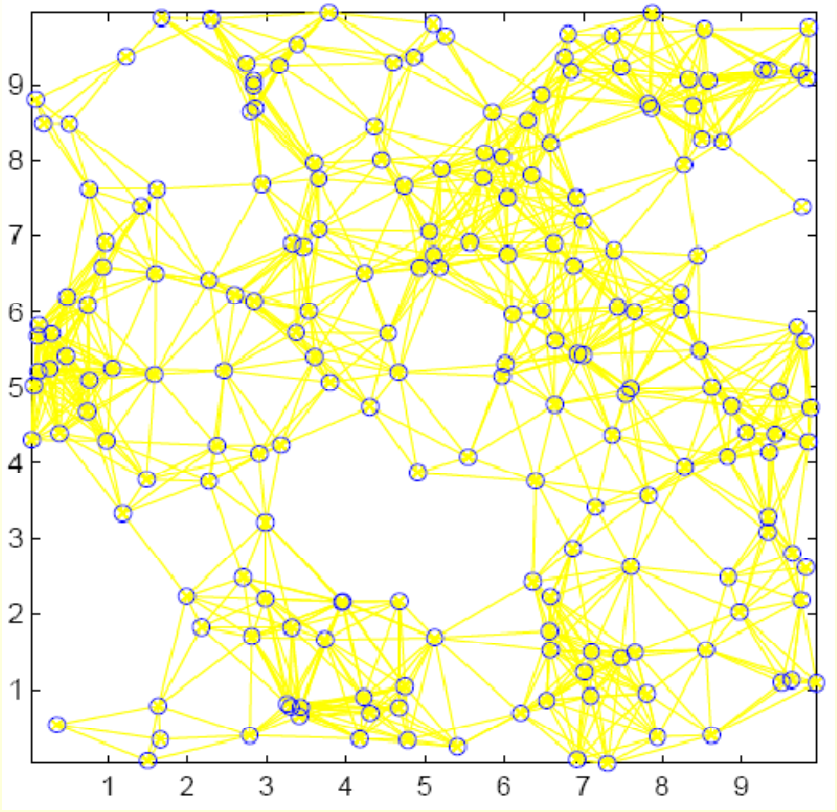
$$\left\| A \right\|_{L^2} = \sqrt{\sum_{i,j} A_{ij}^{\ 2}}$$

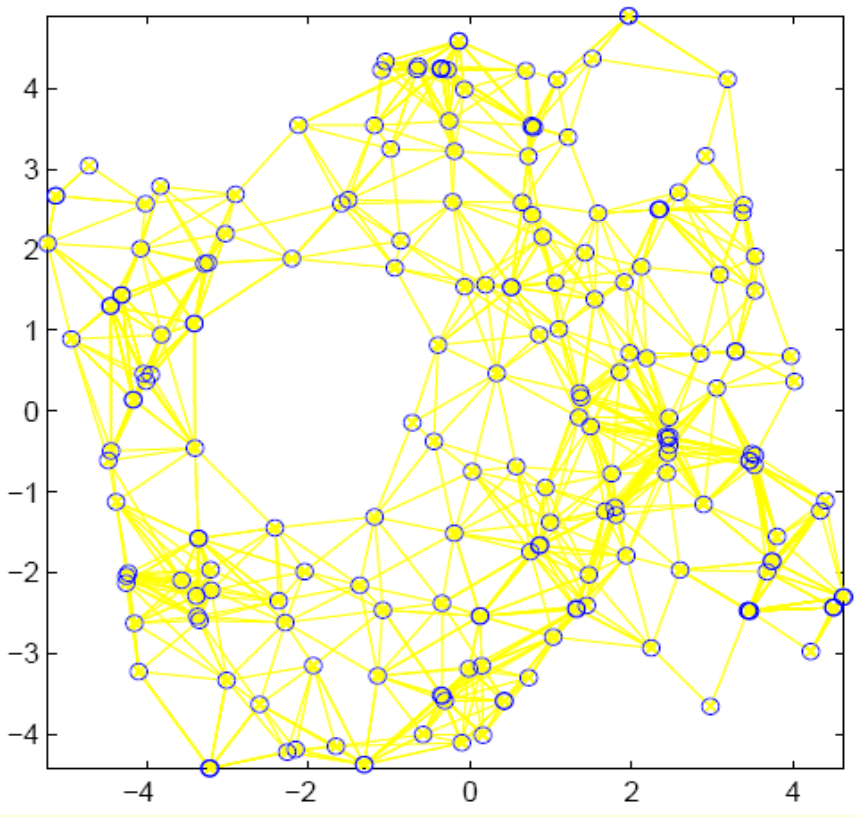
MDS for Sensor Nets

1. Compute all pairs shortest path lengths, using best distances available
2. Apply MDS on the matrix *M*
3. Retain the largest 2 eigenvalues and eigenvectors to find a 2D map

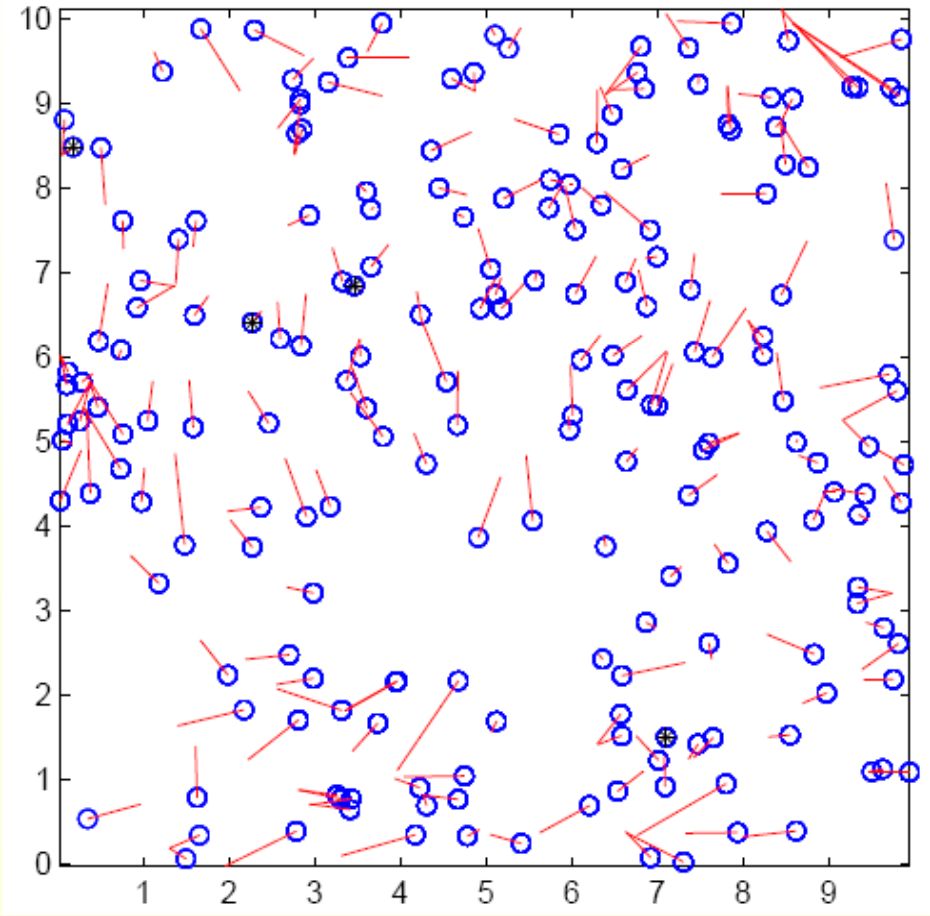
Simulations

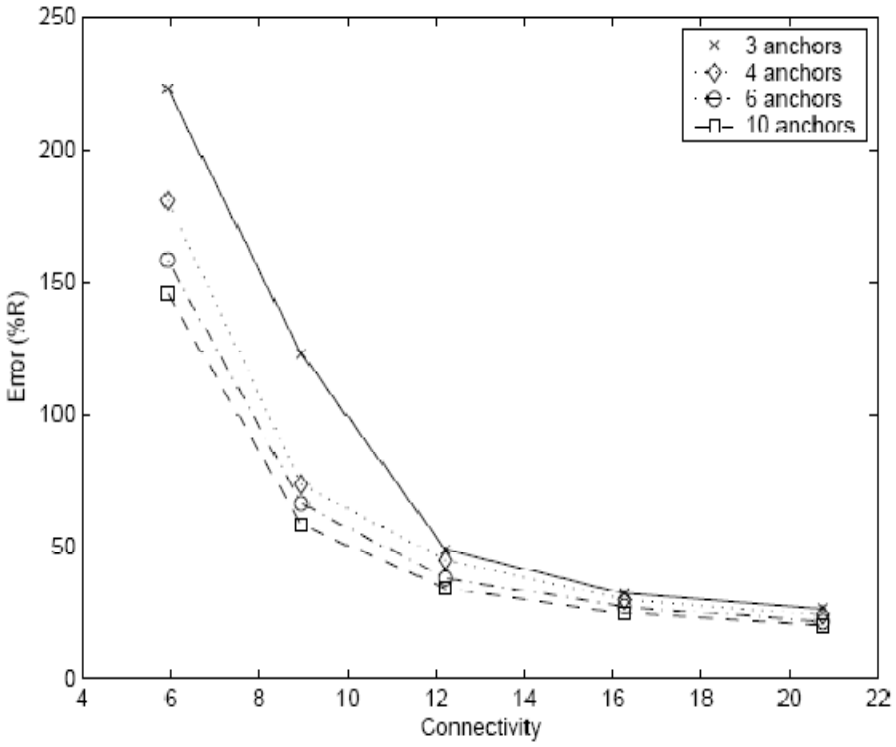
- Random placement





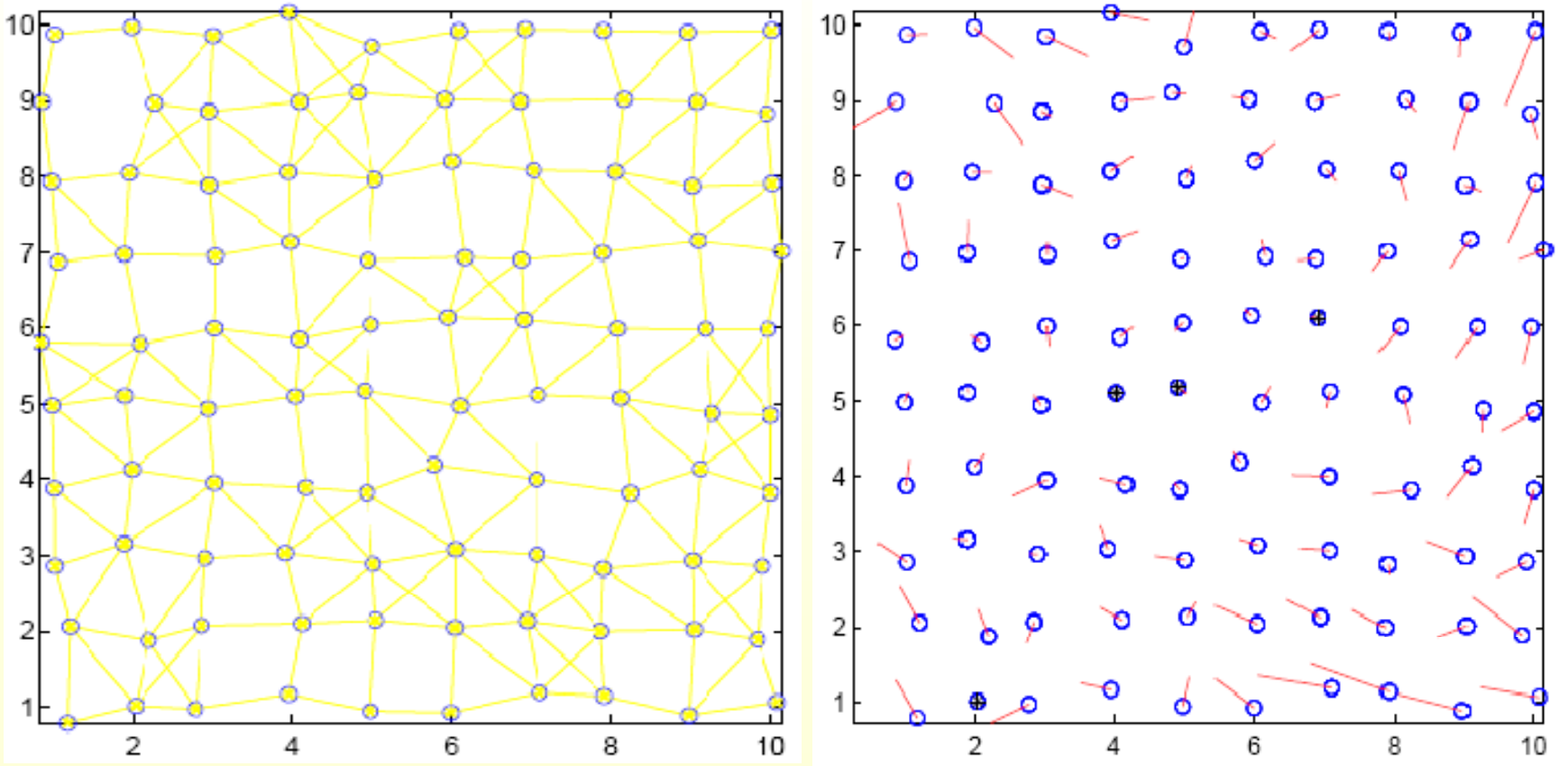
Simulations





Simulations

- Grid placement with 10% error.



MDS Approach

- Experimentally most accurate
- Centralized approach
- Computationally expensive (can't be executed at a sensor node)

Papers

- He, T., Huang, C., Blum, B. M., Stankovic, J. A., and Abdelzaher, T. Range-free localization schemes for large scale sensor networks. In Proceedings of the 9th Annual I**nternational Conference on Mobile Computing and Networking** (San Diego, CA, USA, September 14 - 19, 2003). MobiCom '03.
- Koen Langendoen, Niels Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. **Computer Networks** 43 (2003) 499–518.

Using long distance estimates

Three-phase approach

1. Determine distance to beacon nodes
   (communication)

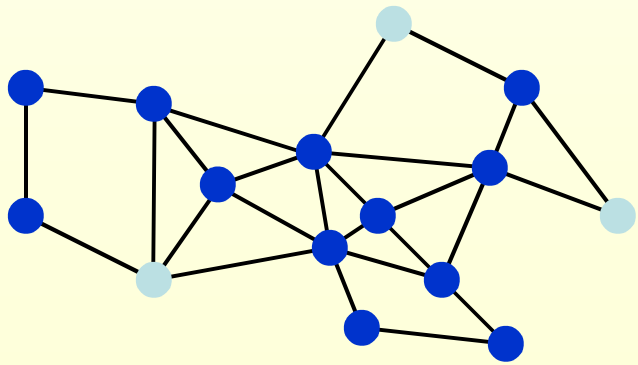
2. Establish position estimates
   (computation)

3. Iteratively refine positions using additional range measurements
   (both)

Phase 1: Distance to Beacons

- Three algorithms
  - Sum-dist [Savvides et al.]
  - DV-Hop [Niculescu et al., Savarese et al.]
  - Euclidean [Niculescu et al.]

- beacons flood network with their known positions



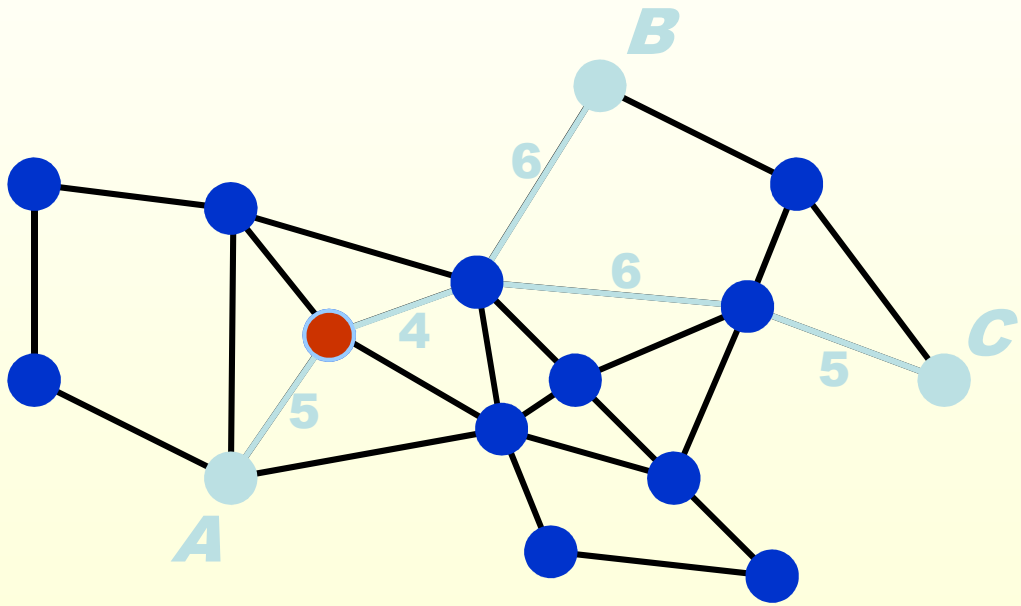
Phase 1: Sum-dist

Beacons

- flood network with known position

Nodes

- add hop distances
- requires range measurement



A: 5
B: 6+4 = 10
C: 5+6+4 = 15

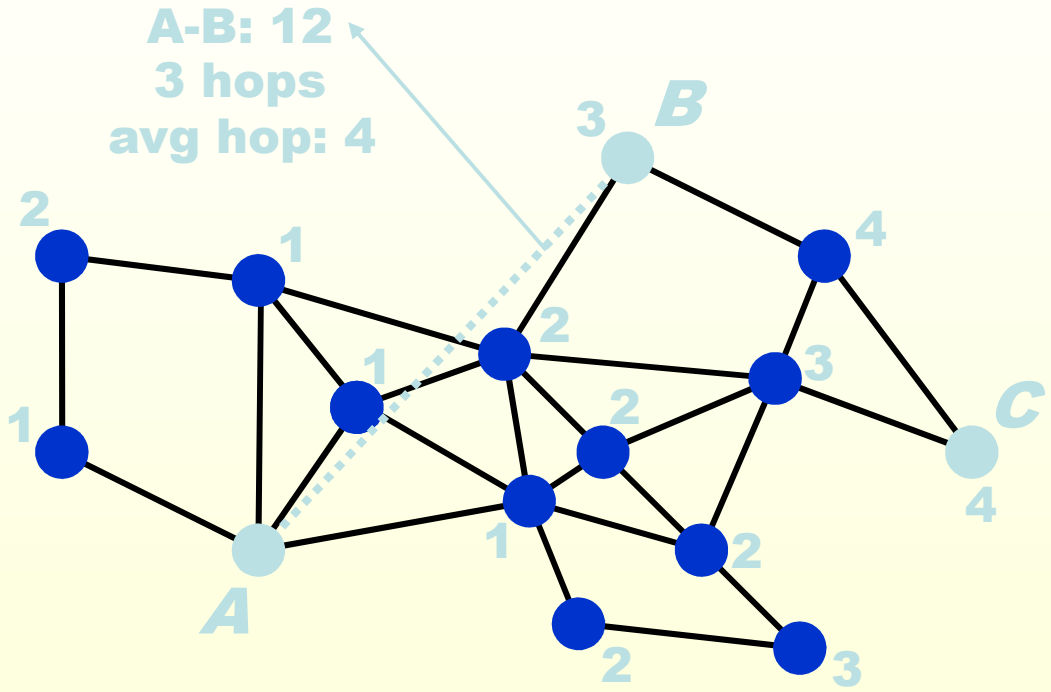
Phase 1: DV-hop

Beacons

- flood network with known position
- flood network with avg hop distance

Nodes

- count # of hops to anchors
- multiply with avg hop distance



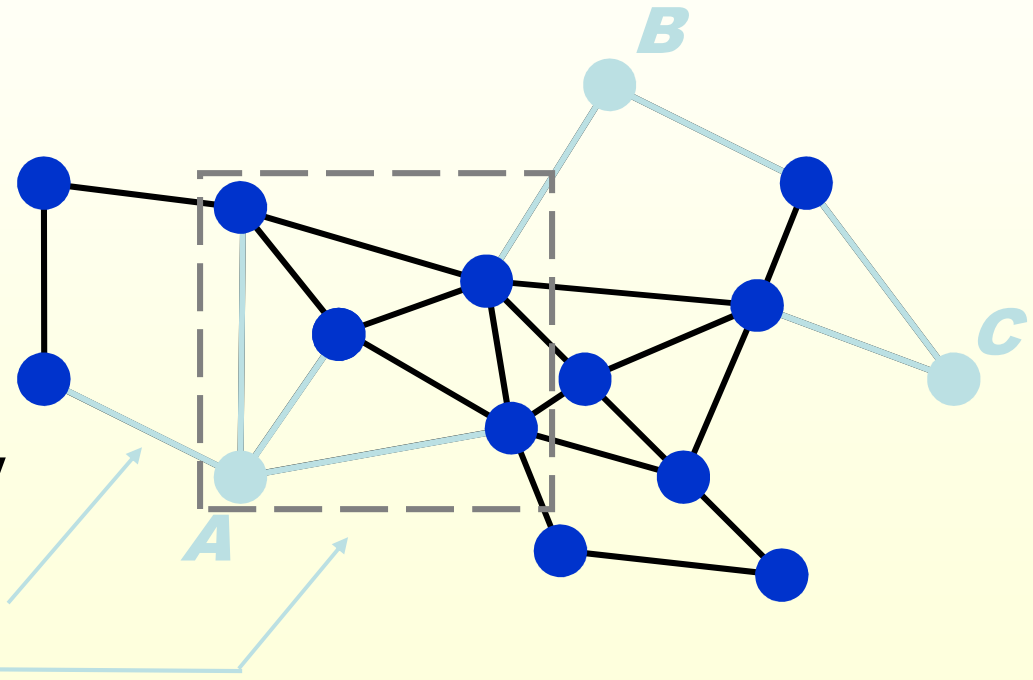
Phase 1: Euclidean

Beacons

- flood network with known positions

Nodes

- determine distance by
  1. range measurement
  2. geometric calculation
- require range measurement



Phase 1: Euclidean (2)

- Wanted:

  Distance A-G

Using AEG*F*:

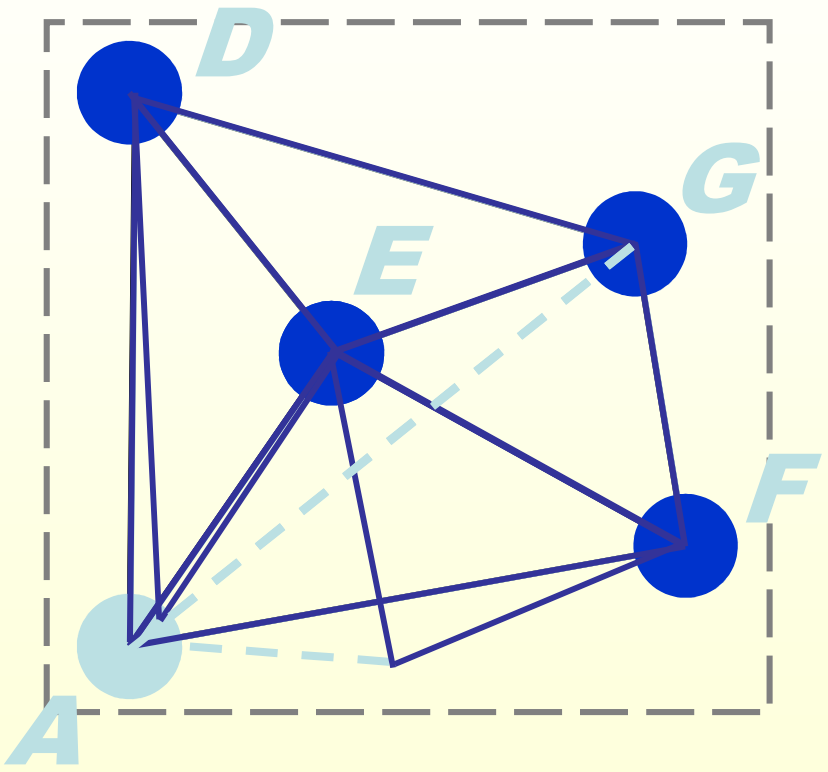
A-G = 8 ...or 3

Using AEG*D*:

A-G = 8 ...or 0.5

↓

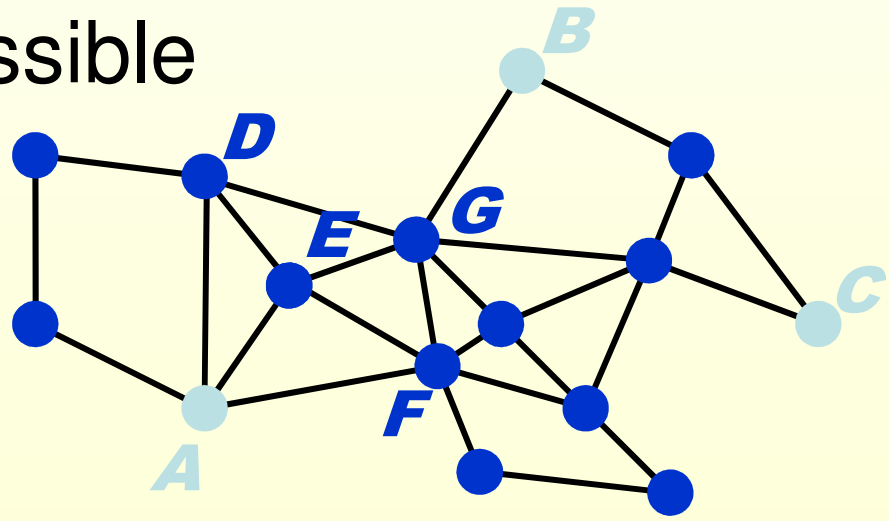
A-G = 8



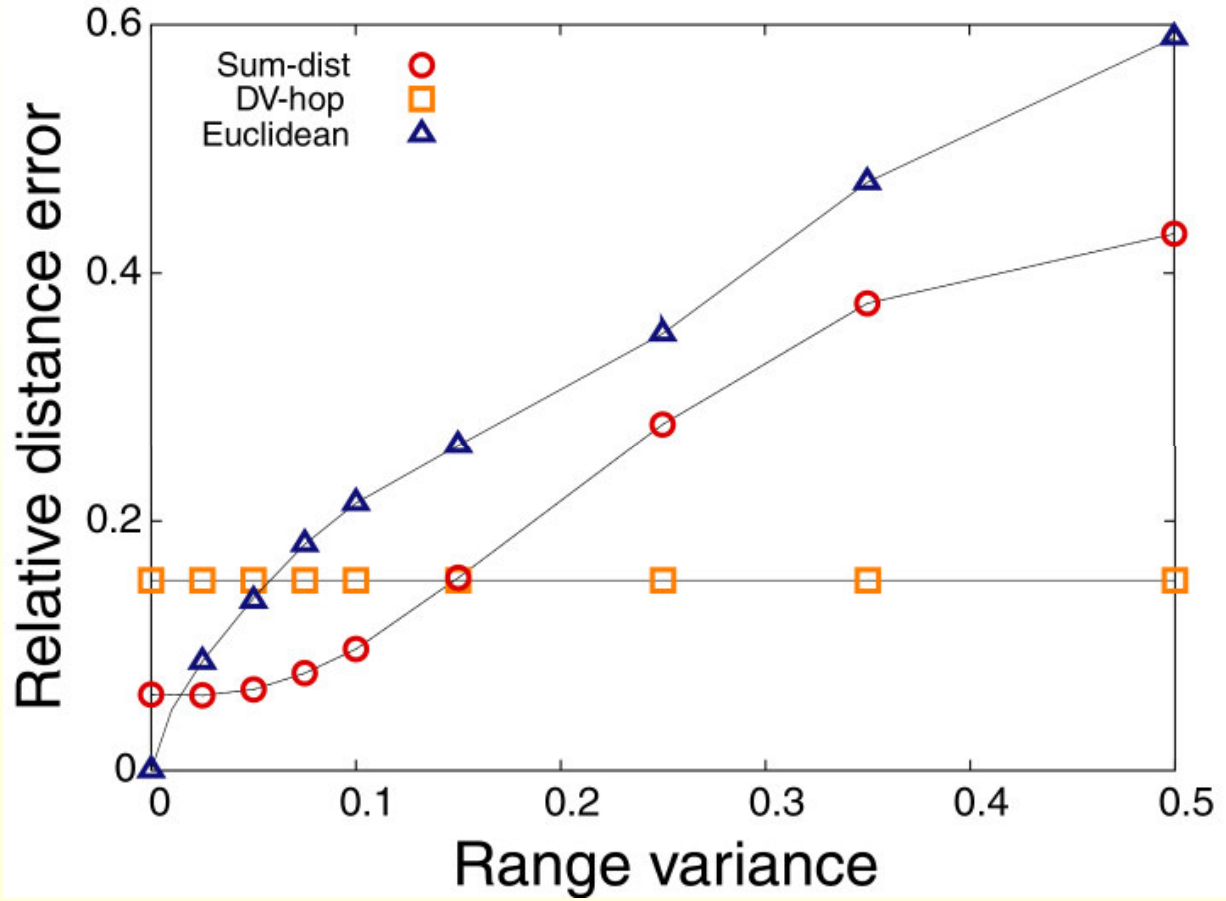
Phase 1: Euclidean (3)

- Needs high connectivity
- Error prone (selecting wrong distance)

- Perfect accuracy possible



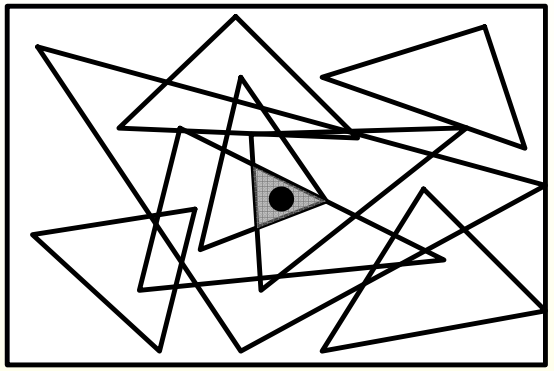
Phase 1: Comparison



- Range measurement
  - Very accurate: Euclidean
  - Reasonable: Sum-dist
  - None / very bad: DV-hop

Getting by with only distance comparisons

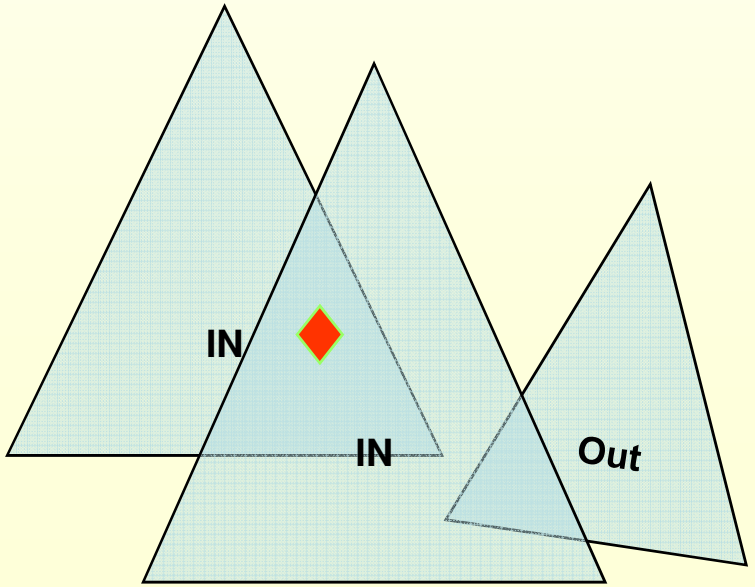
APIT: a Method Using Only Distance Comparisons



- APIT employs a novel *area-based* approach. Beacons divide the field into triangular regions

- A node's presence inside or outside of these triangular regions allows a node to narrow the area in which it can potentially reside.

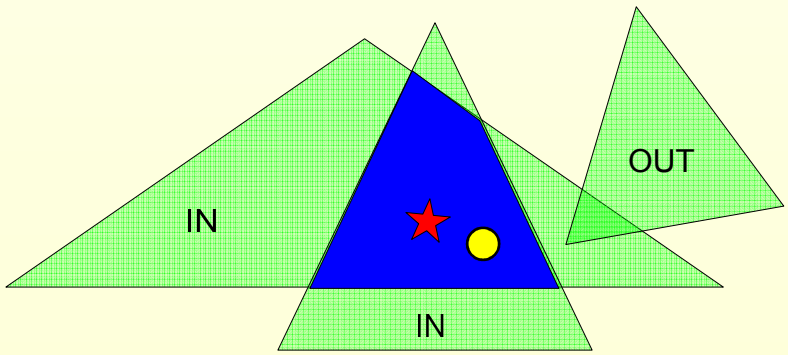
- The method to do so is called Approximate Point In Triangle Test (APIT).



APIT Main Algorithm

For each node

- Get Beacon Locations
- Individual APIT Test
- Triangle Aggregation
- Center of Gravity Estim.



Pseudo Code:

Receive locations (X_i, Y_i) from N beacons

N beacons form $\binom{N}{3}$ triangles.

For (each triangle $T_i \in \binom{N}{3}$)

{

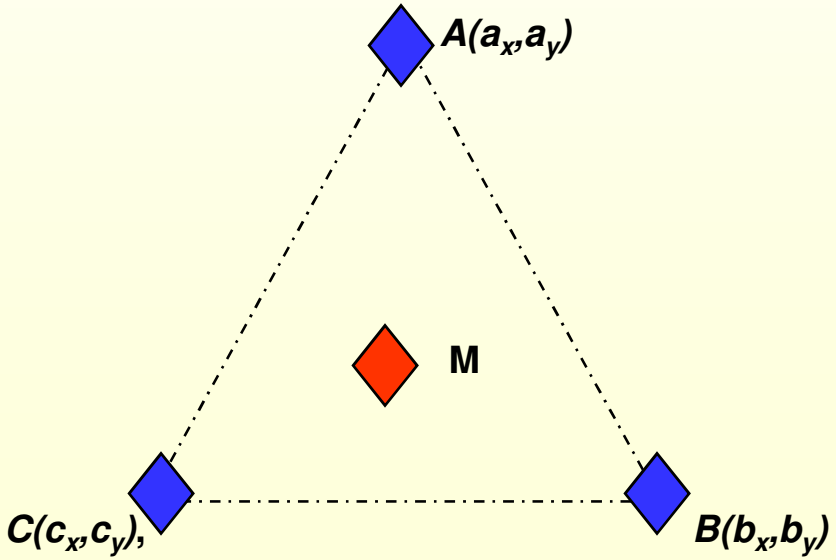
InsideSet ← Point-In-Triangle-Test (T_i)

}

Position = $CoG\ (\cap T_i \in InsideSet)$;

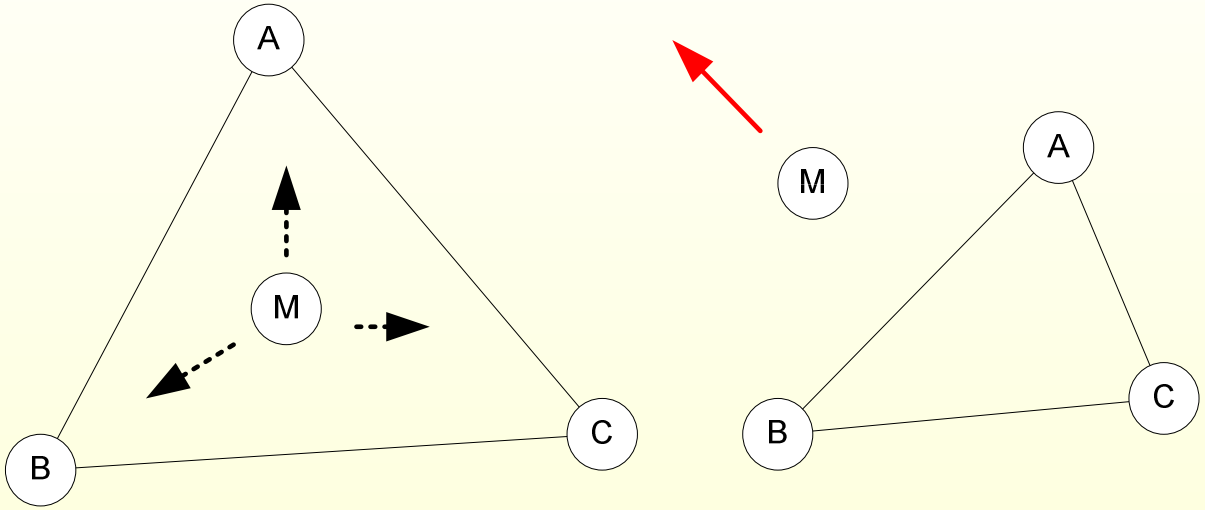
Point-In-Triangle-Test

- *For three beacons with known positions: $A(a_x,a_y)$, $B(b_x,b_y)$, $C(c_x,c_y)$, determine whether a point M with an unknown position is inside triangle ΔABC or not.*



Perfect P.I.T Theory

- *If there exists a direction in which M gets further from points A, B, and C simultaneously, then M is outside of ΔABC. Otherwise, M is inside ΔABC.*

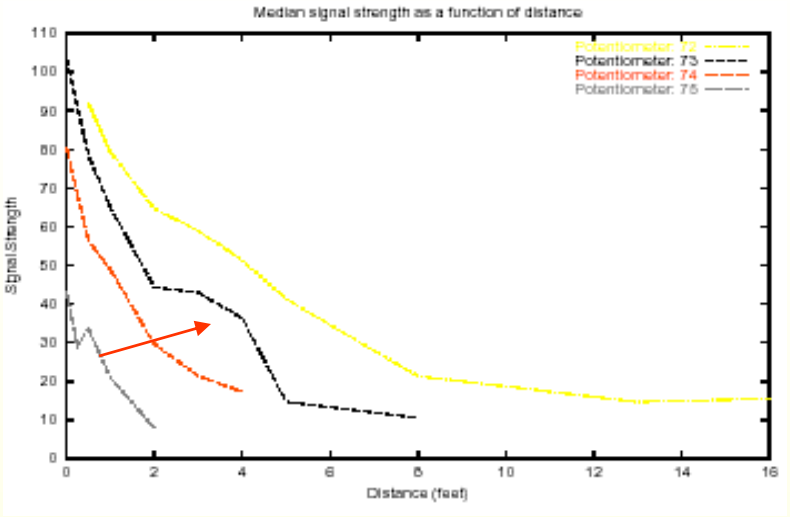


- Require approximation for practical use
  - Nodes cannot move, how to recognize direction of departure (moving away)
  - Exhaustive test on all directions is impractical

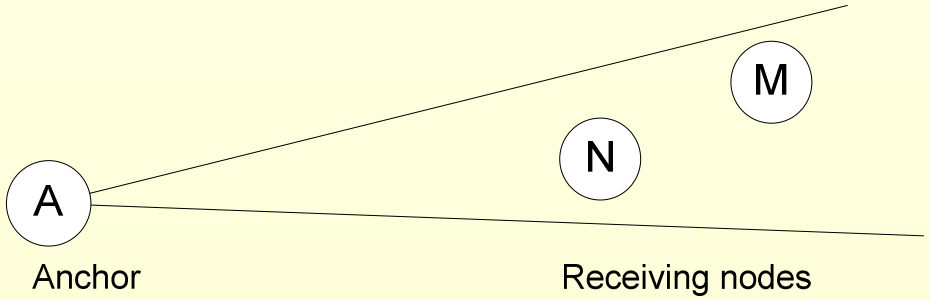
Distance Test

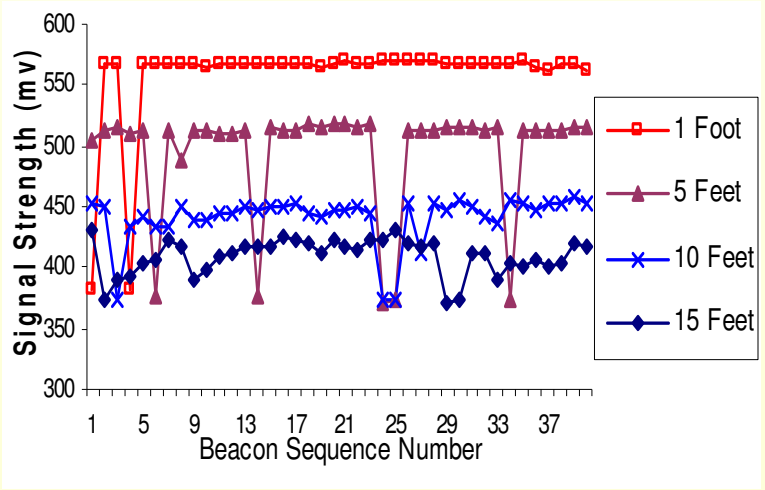
Recognize directions of departure (moving away) via neighbor exchange

1. Receiving Power Comparison Smoothed Hop Distance Comparison



Experiment Result from Berkeley



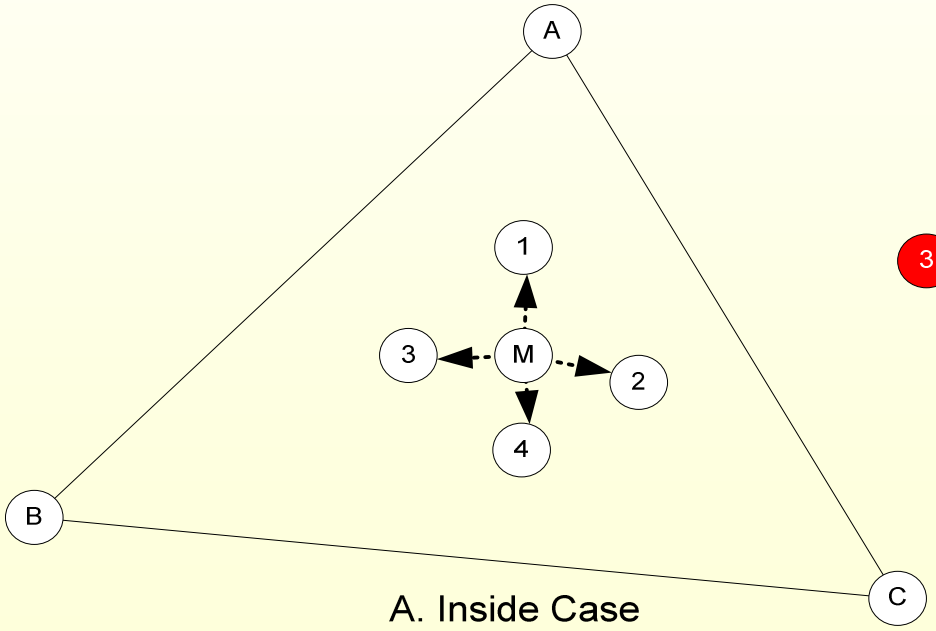


Experiment Result from UVA

A.P.I.T. Test

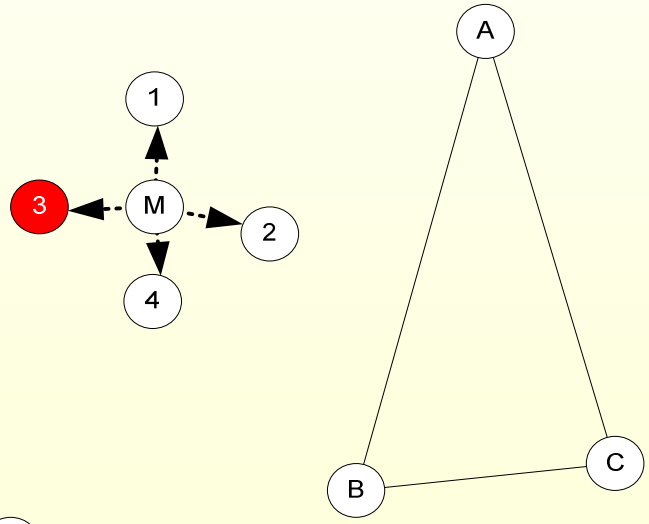
Approximation: Test only directions towards neighbors

- Error in individual test exists , however is relatively small and can be masked by APIT aggregation.



A. Inside Case

APIT(A,B,C,M) = IN

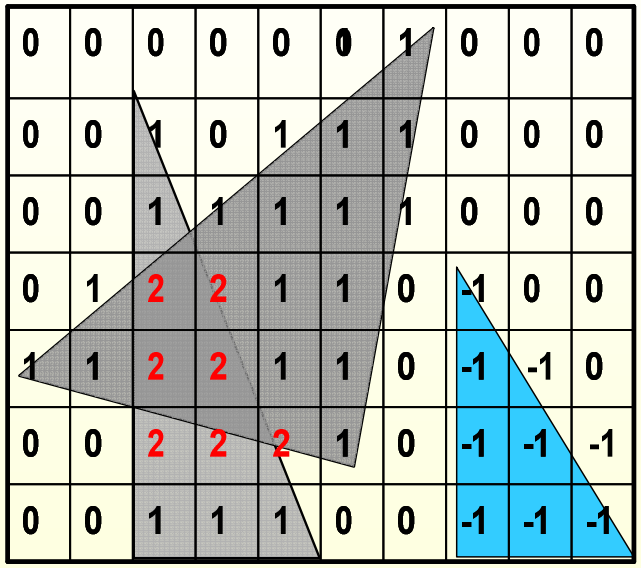


B. OutSide Case

APIT(A,B,C,M) = OUT

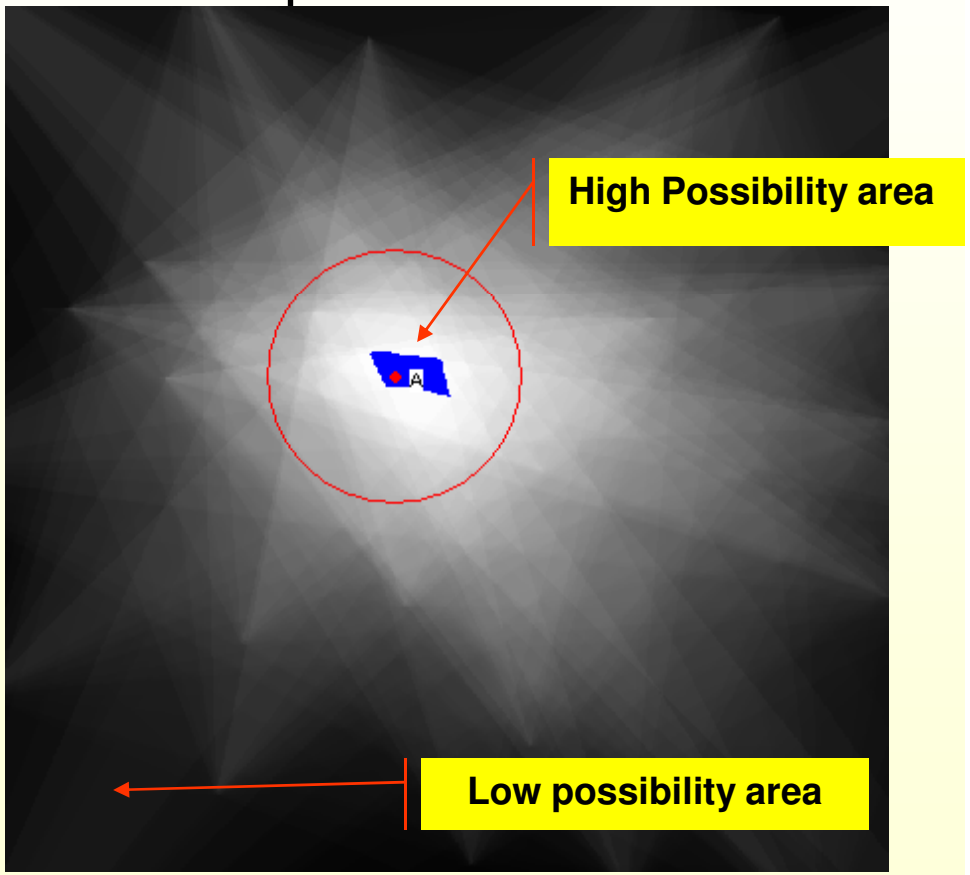
APIT Aggregation

- Aggregation provides a good accuracy, even results by individual tests are coarse and error prone.



Grid-Based Aggregation

With a density 10 nodes/circle, Average
92% A.P.I.T Test is correct
Average 8% A.P.I.T Test is wrong



Localization Simulation example

Does All This Solve the LD Problem?

- No! Several other challenges
- Solution depends on
  - Problem setup
    - Infrastructure assisted (beacons), fully ad-hoc & beaconless, hybrid
  - Measurement technology
    - Distances vs. angles, acoustic vs. rf, connectivity based, proximity based
    - The underlying measurement error distribution changes with each technology
- The algorithm will also change
  - Fully distributed computation or centralized
  - How big is the network and what networking support do you have to solve the problem?
  - Mobile vs. static scenarios
  - Many other possibilities and many different approaches

Summary

- Location discovery is a central but difficult problem in sensor networks
- Most localization methods are based on ranging (distance estimates)
- Localization algorithms can be demanding for small nodes
- Localization errors need to be taken into account
- Location services are needed so that location information becomes globally available

The End