Manifold Learning Algorithms for Localization in Wireless Sensor Networks

Neal Patwari and Alfred O. Hero III

University of Michigan

Dept. of Electrical Engineering & Computer Science

http://www.engin.umich.edu/~npatwari

ICASSP’04 Presentation

May 19, 2004

Sensor Localization in Large Scale Apps

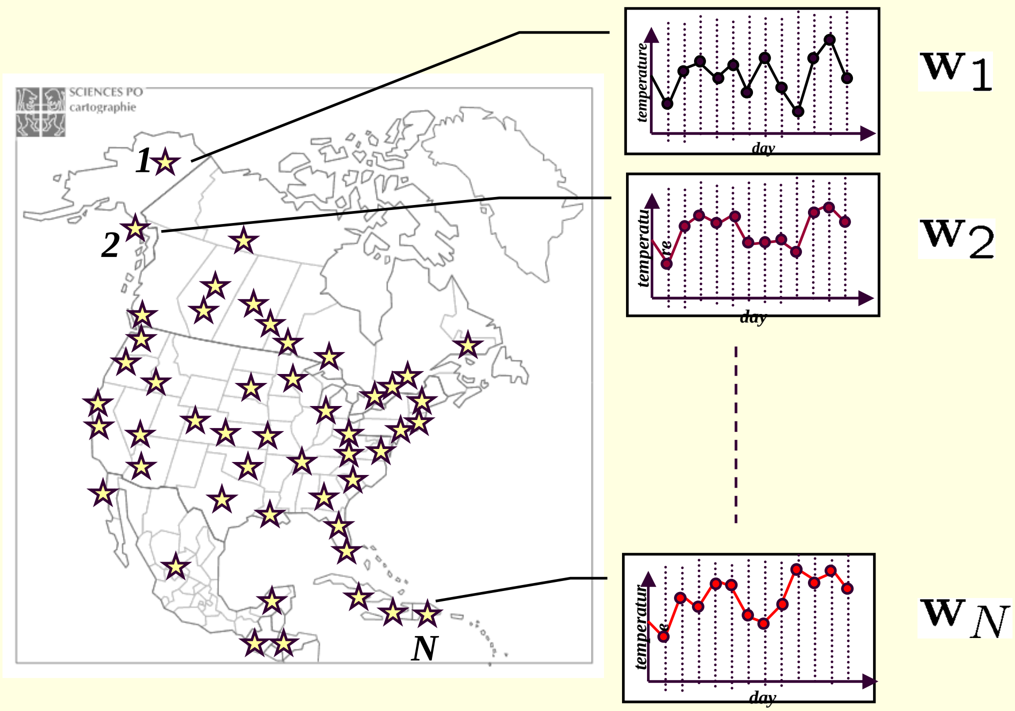
- 1000s to millions of devices
- Device cost is 1st priority (10¢)
- Range measurement can add cost, consume energy
- Sensor data is recorded anyway – can it be used for localization?

Outline of Presentation

- Sensor Data is High-Dimensional Location
- Manifold Learning for Sensor Localization
- Simulation Experiments
  - Random Field Model
  - Results
- Current and Future Work

Data from a Space-Time Sensor Field

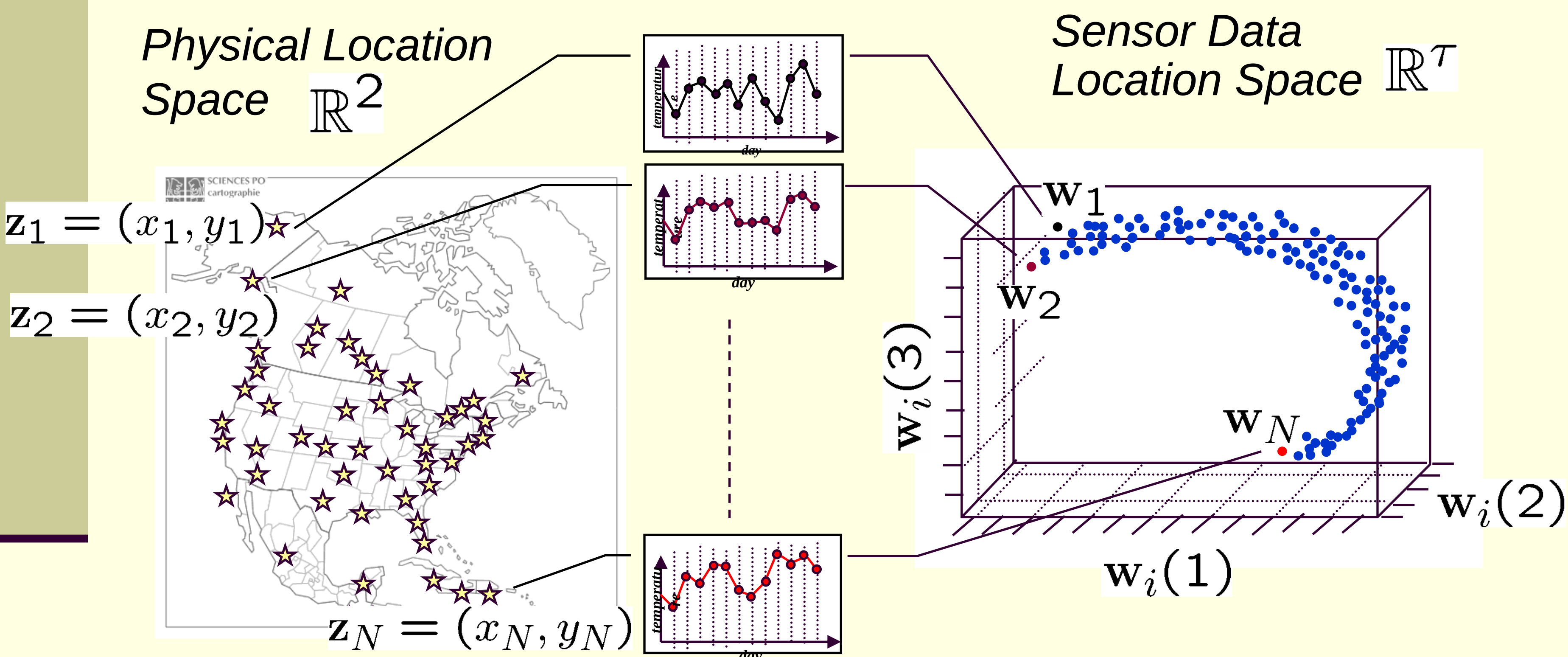
- Ex: Average daily temp; Soil moisture & chemistry



- Record data at sensors $1 \ldots N$
- Keep time history from $1..\tau$

Sensor Data Location Space

- Data vectors $\mathbf{w}_i$ serve as a ‘location’ in a τ-dim space



Example: $\tau = 3$

Estimation Problem Statement

- Estimate:
  - Coordinates of n unknown-location devices:

$$\theta = [\mathbf{z}_1^T, \ldots, \mathbf{z}_n^T]$$

- Given:
  - *a priori* known coordinates of m devices:

$$\{\mathbf{z}_i\}_{i=n+1}^{N} \qquad \mathbf{z}_i = [x_i, y_i]^T \qquad N = n + m$$

  - Sensor measurements:

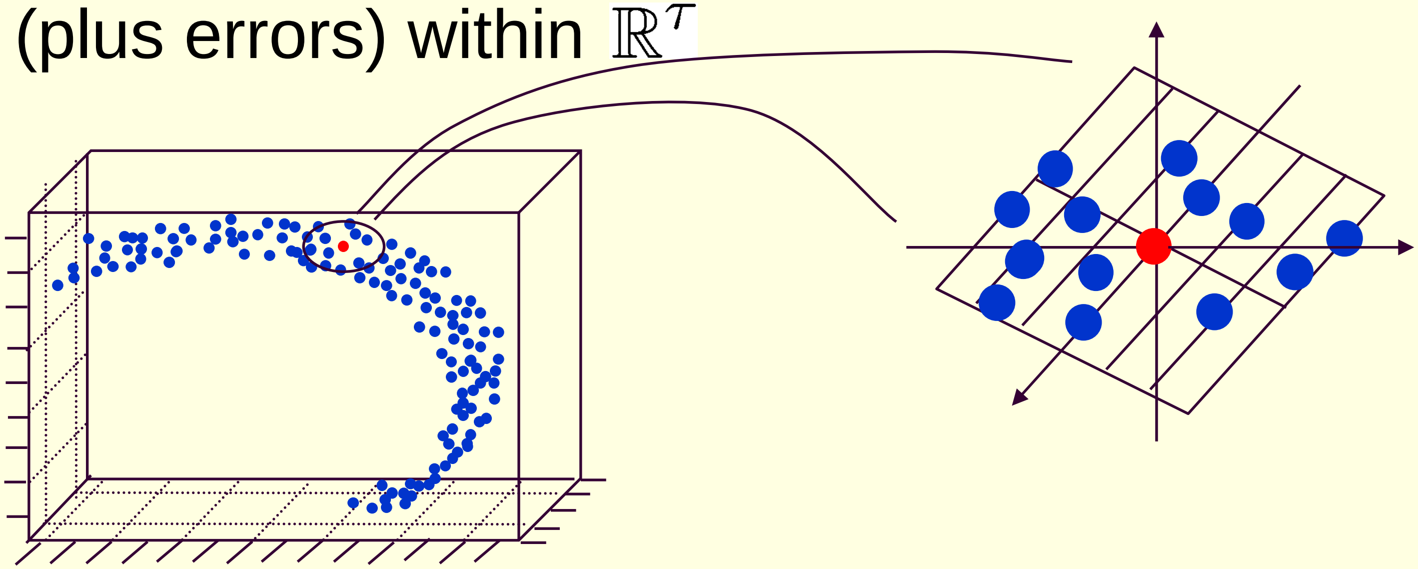
$$W = [\mathbf{w}_1, \ldots, \mathbf{w}_N] \qquad \mathbf{w}_i \in \mathbb{R}^T$$

Sensor Data Assumptions

1) *Dense Deployment* of sensors in space
2) *Neighborhood Preserving*:
   - Neighboring sensor data vectors in $\mathbb{R}^\tau$ correspond to neighboring sensors in $\mathbb{R}^2$
3) *Local Linearity*:
   - Sensor data $\{\mathbf{w}_i\}$ within some ε neighborhood lie approximately in a linear subspace of $\mathbb{R}^\tau$

Summary: Manifold Assumption

- Sensor data is close to a non-linear manifold
  - A twisted, curved, folded sensor location map (plus errors) within $\mathbb{R}^\tau$



  - Equivalently, $\exists$ a smooth function $g : \mathbb{R}^2 \rightarrow \mathbb{R}^\tau$ s.t.

$$\mathbf{w}_i = g(\mathbf{z}_i) + \eta_i$$

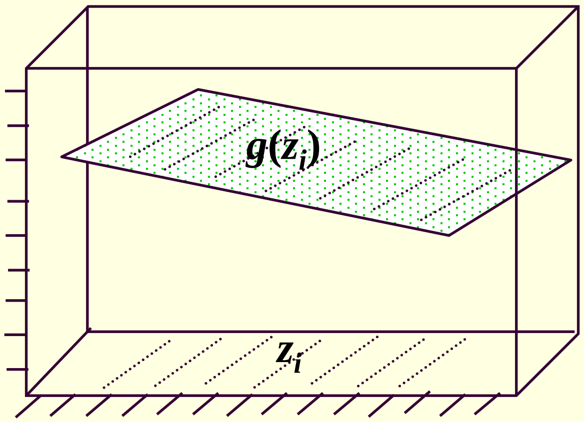
(η_i is additive noise)

Outline of Presentation

- Sensor Data <u>is</u> High-Dimensional Location
- Manifold Learning for Sensor Localization
- Simulation Experiments
  - Random Field Model
  - Results
- Current and Future Work

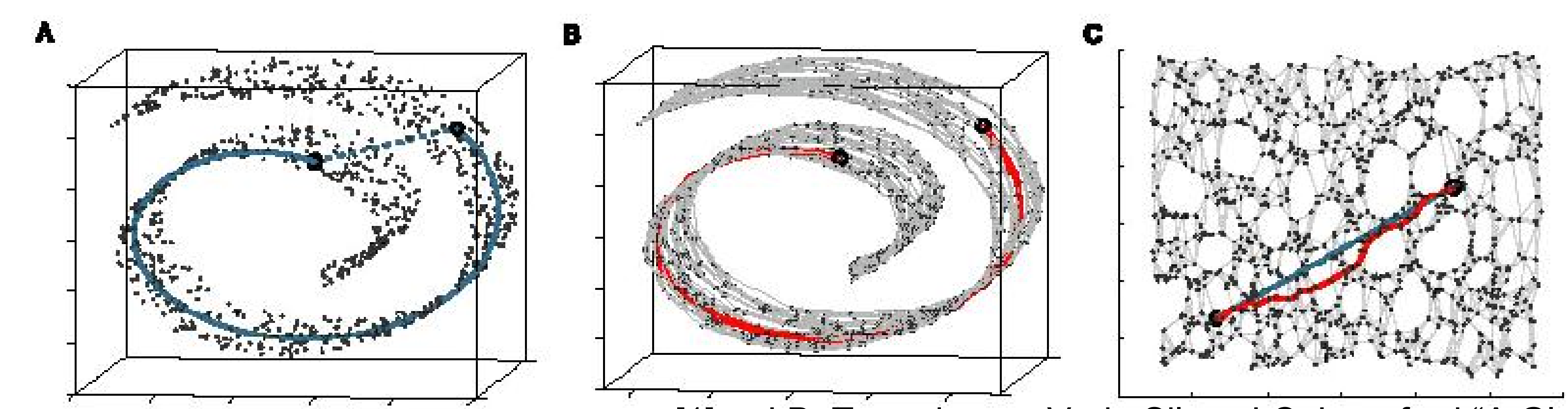
Localization is Functional Analysis

- What if $g(\cdot)$ was linear?
  - Multi-Dimensional Scaling (MDS)
  - Finds least-squares solution
  - Within rotation, mirroring



- Pros:
  - Optimization by eigendecomposition
  - Not prone to local maxima
- Reality:
  - Sensor data vectors aren't linear in the physical coordinates

Isomap Algorithm



Eg: Data points lie in $\mathbb{R}^T$, but on a 'Swiss roll' [1]

[1] J.B. Tenenbaum, V. de Silva, J.C. Langford "A Global Geometric Framework for Nonlinear Dimensionality Reduction" *Science*, 22 Dec 2000.

- Intuition: Don't use long distances in $\mathbb{R}^T$
  - Find K nearest neighbors of each point
- Find shortest path using only neighbors
  - Sum Euclidean distances along shortest path for 'distance between non-neighbors'
- Use MDS on shortest path distances
  - Eigendecomposition of a dense matrix: $O(N^3)$

Other Methods: LLE and Hessian LLE

- Locally Linear Embedding (LLE) [2]
  - Reconstruct local areas using global coords
- Hessian-based LLE (HLLE) [3]
  - Take into account the local curvature
- Intuition: Consider similarity, not difference
- Weight similarity of K nearest neighbors (others are 0)
  - Weight matrices are sparse & symmetric
  - Calculate d+1 eigenvectors w/ smallest eigenvalues

[2] S.T. Roweis and L.K. Saul, "Nonlinear Dimensionality Reduction by Local Linear Embedding" *Science*, 22 Dec 2000.

[3] D.L. Donoho and C. Grimes, "Hessian eigenmaps: locally linear embedding techniques for high-dimensional data," Publ. Nat. Academy of Science, May 13, 2003

LLE Allows Distributed Algorithms

- Calculation of local linear weights is local
- Distributed algs. exist to calc. extremal eigenvectors
  - Davidson method, extensions [4]
  - Data distribution techniques [5]
  - Block-Jacobi preconditioning [5]
- Adapted for hierarchical networks
- Complexity: $O(KN^2)$

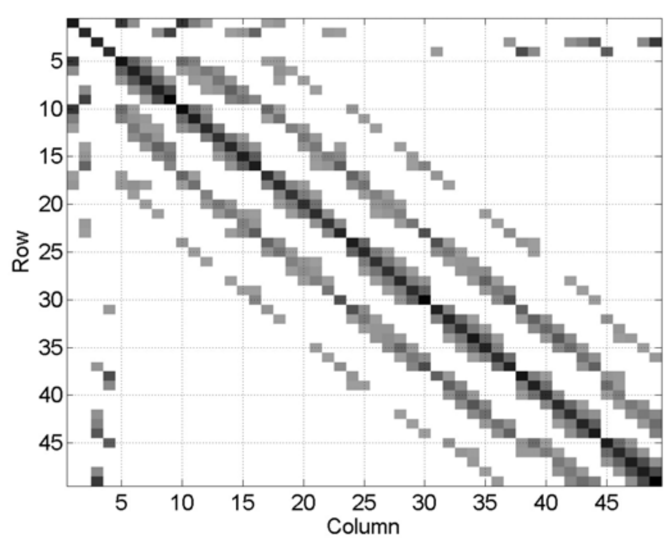


Figure: *Weight matrix for 7 by 7 grid example using LLE algorithm*

[4] E. R. Davidson, "The Iterative Calculation of a Few of the Lowest Eigenvalues and Corresponding Eigenvectors of Large Real-Symmetric Matrices", *J. Comput. Phys.* 14(1), pp 87-94, Jan. 1975

[5] Luca Bergamaschi and Giorgio Pini and Flavio Sartoretto, "Computational experience with sequential and parallel,preconditioned Jacobi–Davidson for large, sparse symmetric matrices", *J. Comput. Phys.*, 188(1), pp. 318-331, June 2003.

Outline of Presentation

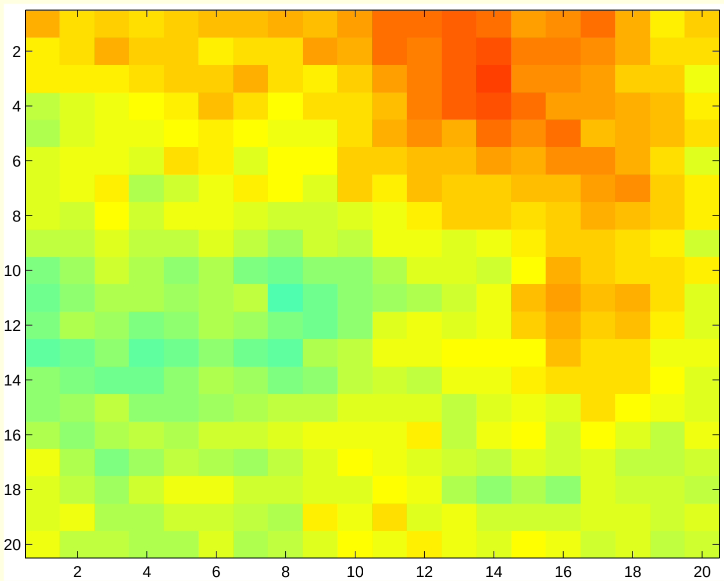
- Sensor Data is High-Dimensional Location
- Manifold Learning for Sensor Localization
- Simulation Experiments
  - Random Field Model
  - Results
- Current and Future Work

Random Field Model for Simulation

- Sense data from a spatially correlated random field
- We use: Gaussian w/ exponential covariance:

$$\mathbf{w}(t) \sim \mathcal{N}(\mu, R(\theta)) \quad \textit{where} \quad \mathbf{w}(t) = [\mathbf{w}_1(t), \ldots, \mathbf{w}_N(t)]^T$$

$$[R(\theta)]_{i,j} = \sigma^2 \exp\left[-\left(\|\mathbf{z}_i - \mathbf{z}_j\|/\delta\right)^{\alpha}\right] \qquad 0 < \alpha \leq 2, \quad \delta > 0$$



- Note: Isotropic Model
  - $R(\theta)$ is a fcn of distance
- $\{\mathbf{w}(t)\}_{t=1\ldots\tau}$ are indep.

Example: 7 by 7 Grid of Devices

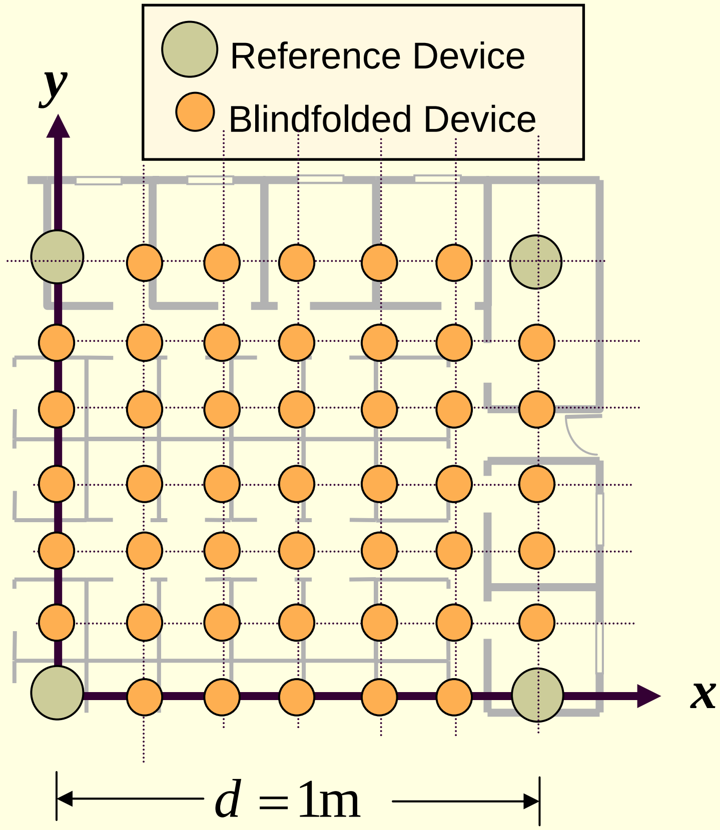
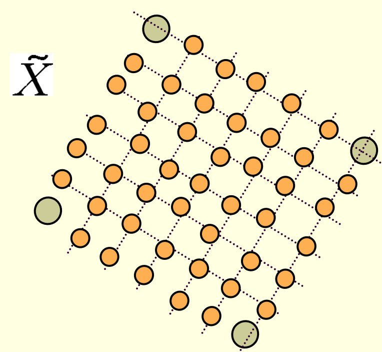
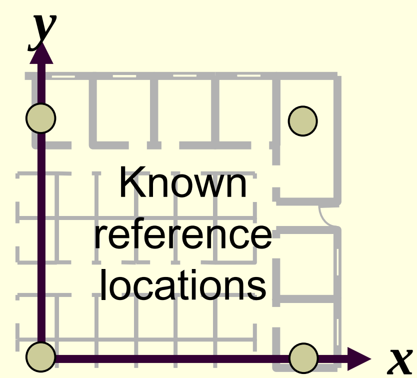


Figure: *Actual device locations in the 7 by 7 grid example*

- 4 *reference* devices
- 45 *blindfolded* devices
- 200 time samples / sensor
- Calculate $\tilde{X}$





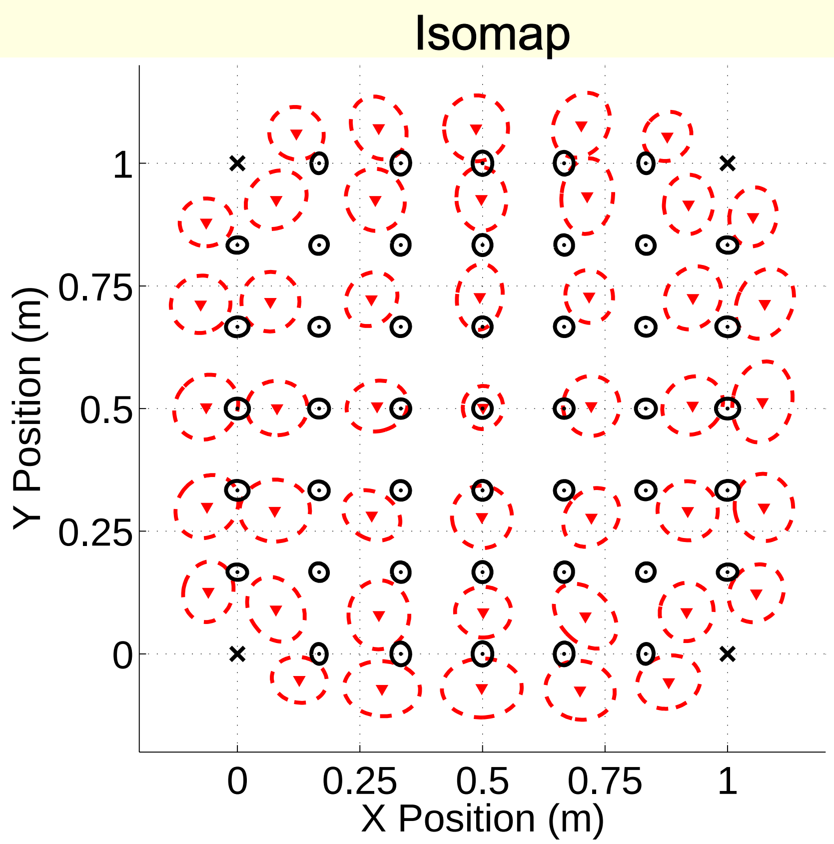
- Rotate (flip) $\tilde{X}$ to match known reference locations
- Run 100 trials per estimator

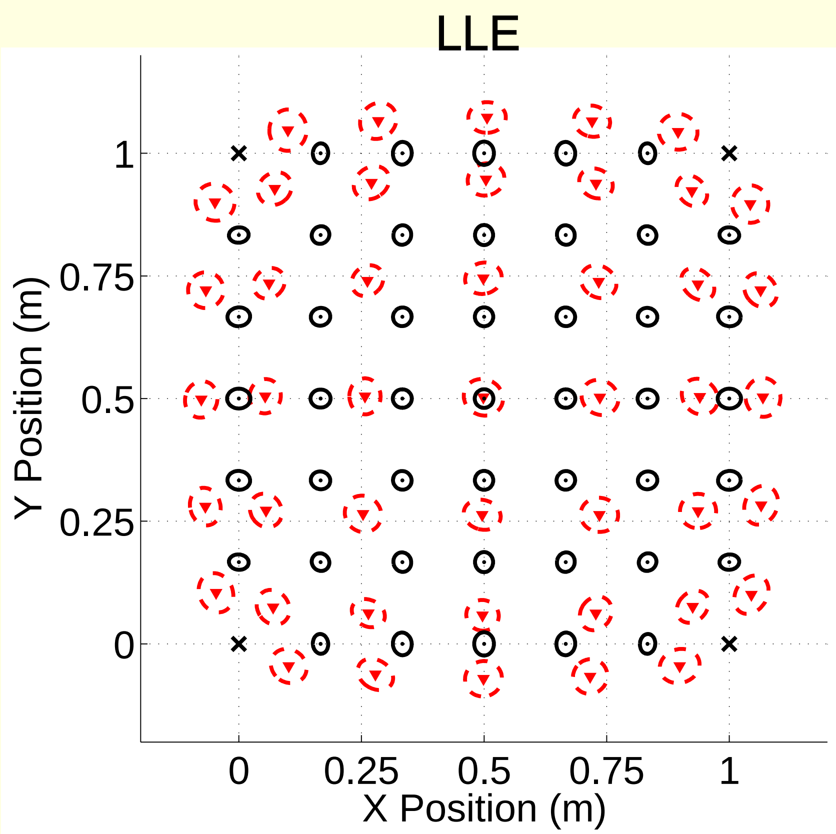
Isomap & LLE Performance in Grid Eg.

Key:

1-σ uncertainty ellipses ○ CRB ◌ Estimator × Reference Device	• Actual Location ▼ Estimator Mean

- Both show bias
- LLE variance near CRB

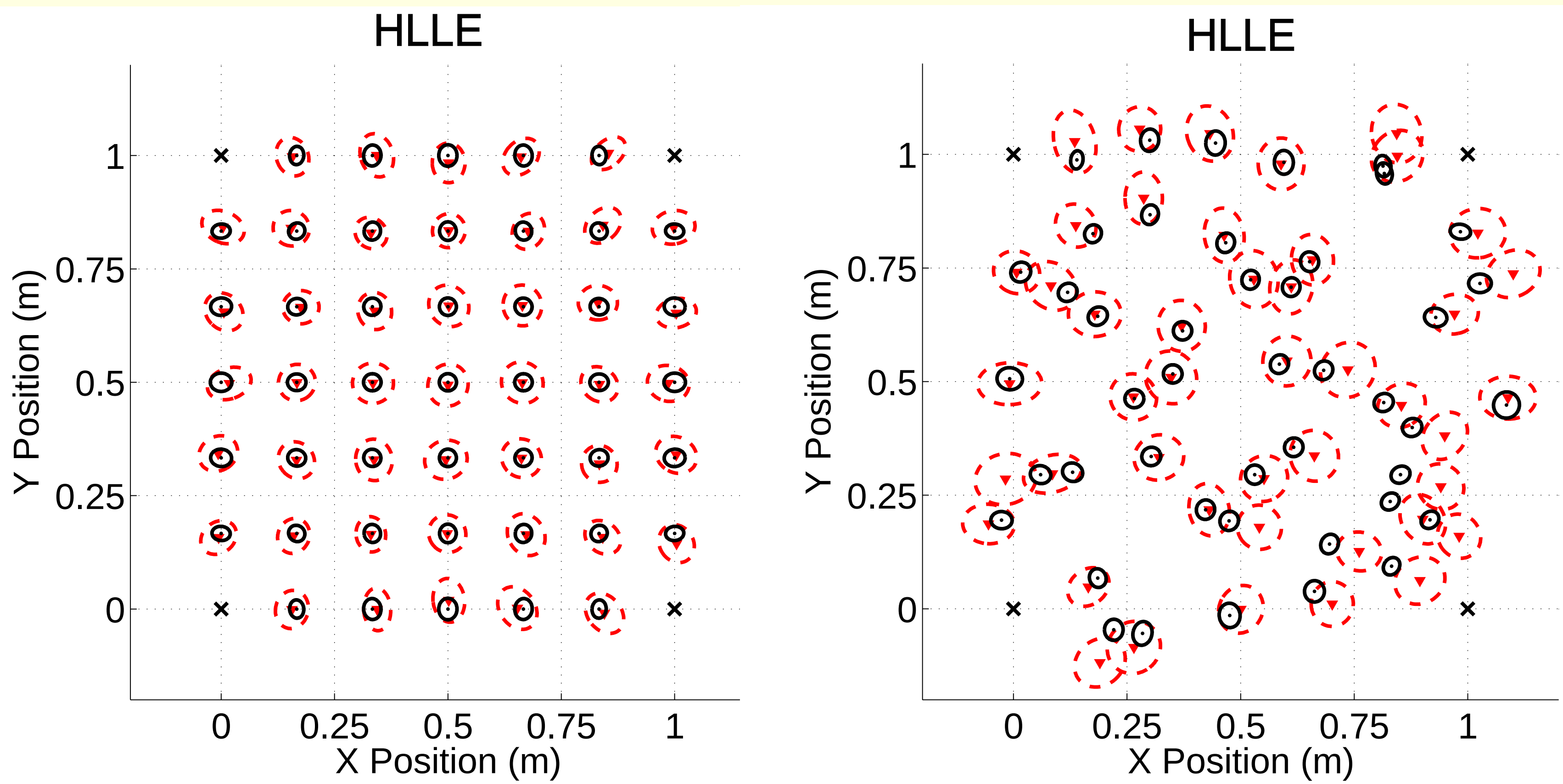




HLLE Performance in Grid, Grid+Noise

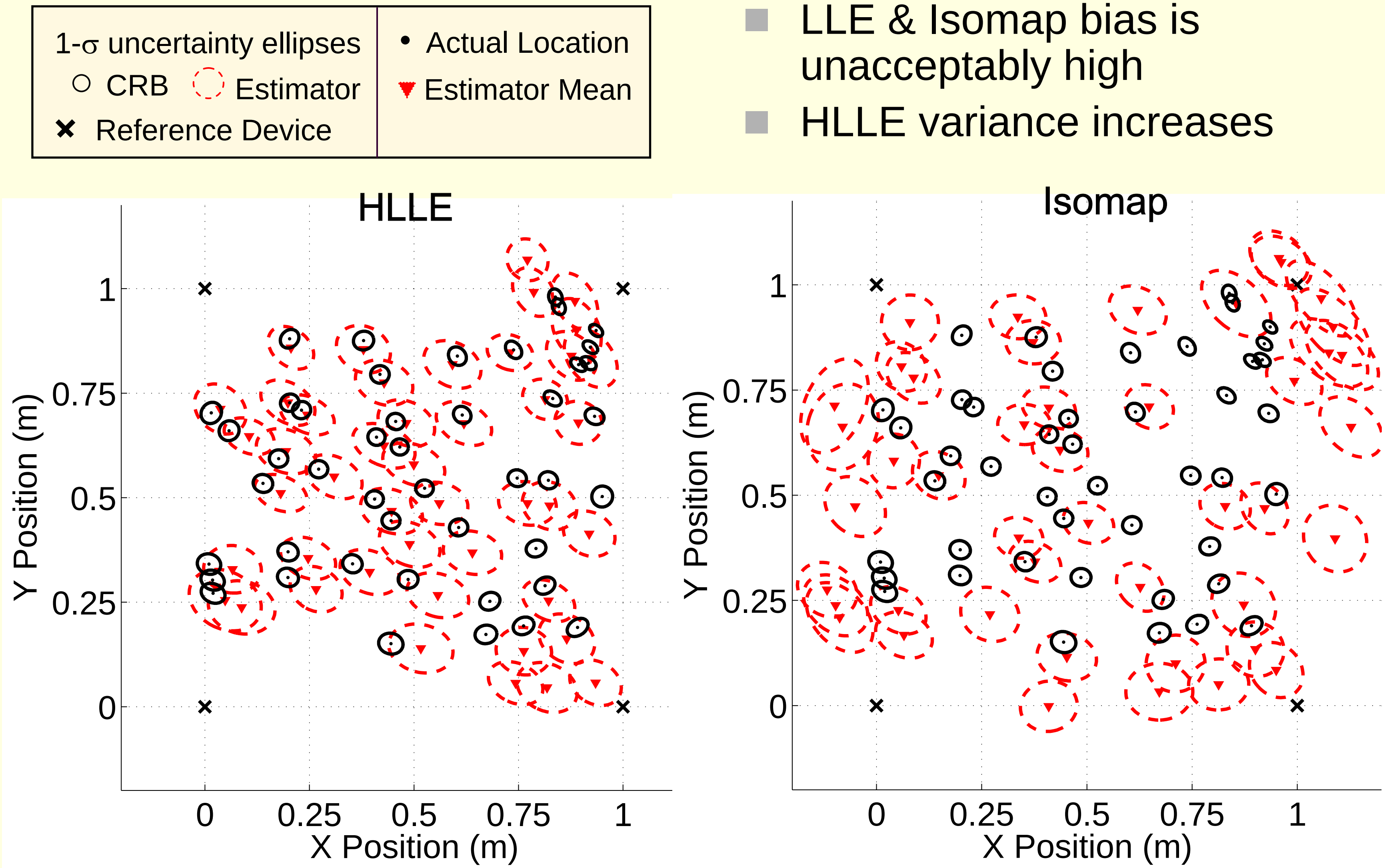
1-σ uncertainty ellipses	• Actual Location
○ CRB ◌ Estimator	▼ Estimator Mean
× Reference Device	

- Removes bias in grid case
  - Same variance vs. LLE
- Small bias in grid+error case



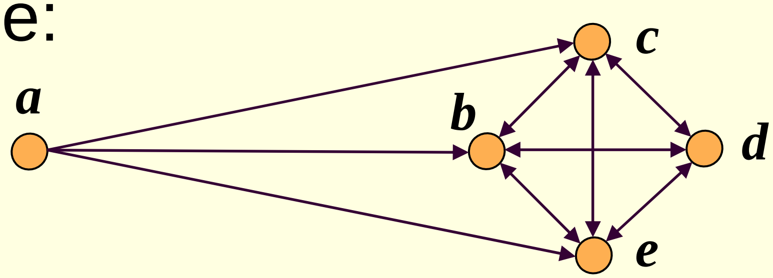
Performance in Random Deployment

- LLE & Isomap bias is unacceptably high
- HLLE variance increases



Recent Developments

- Cause of robustness issue:
  - Asymmetry of *k*-nearest-neighbors relation
  - Example:



    - Assign 3 n.n. to devices *a*-*e*. Although '*a*' has 8 neighbors, it is no one else's neighbor!
  - Having no devices consider you a nearest neighbor causes 0 eigenvalue in HLLE

K-Nearest-Neighbors Adjustment

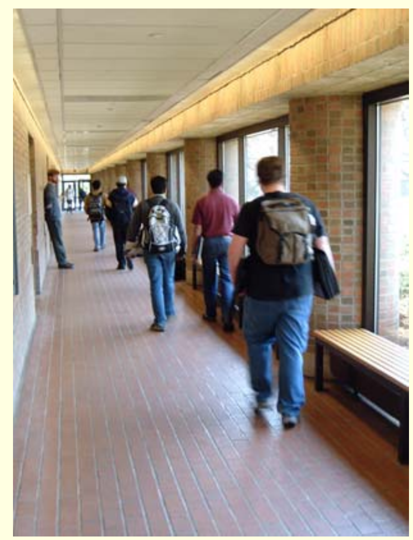
- Robust approaches for neighbor selection:
  1) Enforce symmetry: Include another device if it includes you.
     - Tends to include distant neighbors
     - Negative influence in accuracy (even when avg. # neighbors is kept constant)
  2) Take pity: Include another device if less than k_{min} others do & you are the next-closest.
     - Choice of k_{min} can be << k (we use $k_{min}=3$)
     - Negligible impact on accuracy, since it rarely changes the connectivity

Outline of Presentation

- Sensor Data is High-Dimensional Location
- Manifold Learning for Sensor Localization
- Simulation Experiments
  - Random Field Model
  - Results
- Current and Future Work

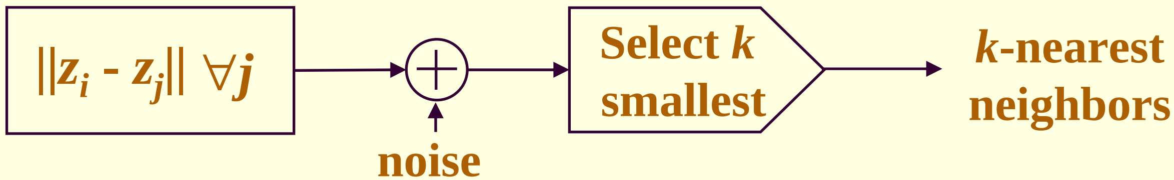
Current and Future Research

- Acoustic sensor network measurements
  - Measurements of background noises over time
  - Future: To what extent are sensor fields isotropic?



Current and Future Research

- Biasing Effect of Neighborhood Selection



  - When distances are r.v.'s, selecting the k-nearest neighbors produces a biased sample
  - Future: Strategies for bias removal
  - Future: Analysis of manifold learning in noise

Current and Future Research

- Applying Weighted Least Squares
  - Isomap, MDS currently solve an identically-weighted LS problem
  - Shorter distances tend to be more accurate

Conclusions

- Use sensor data to estimate sensor location
  - (Instead of / In addition to) Measured ranges
- Benefits of Manifold Learning Algorithms
  - Can be distributed
  - Not model-based
  - Optimization is non-iterative (finds a global optimum)
  - $O(KN^2)$, or $O(KN)$ at each sensor