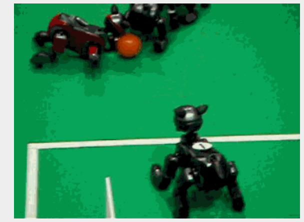
CSE-571
Probabilistic Robotics

Rao-Blackwelized Particle Filters and Applications

Ball Tracking in RoboCup



- Extremely noisy (nonlinear) motion of observer
- Inaccurate sensing, limited processing power
- Interactions between target and

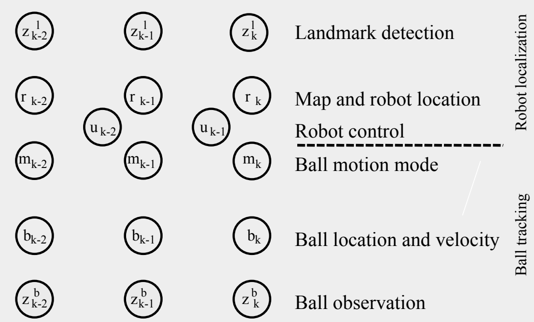
Goal: Unified framework for modeling the ball and its interactions.

t

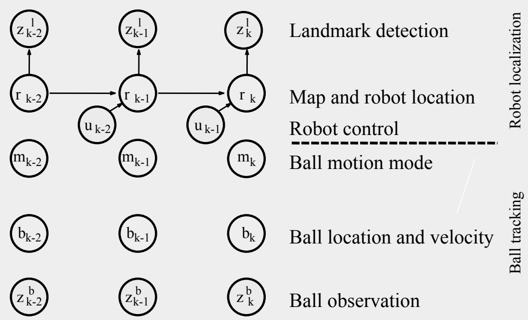
Tracking Techniques

- Kalman Filter
  - Highly efficient, robust (even for nonlinear)
  - Uni-modal, limited handling of nonlinearities
- Particle Filter
  - Less efficient, highly robust
  - Multi-modal, nonlinear, non-Gaussian
- Rao-Blackwellised Particle Filter, MHT
  - Combines PF with KF
  - Multi-modal, highly efficient

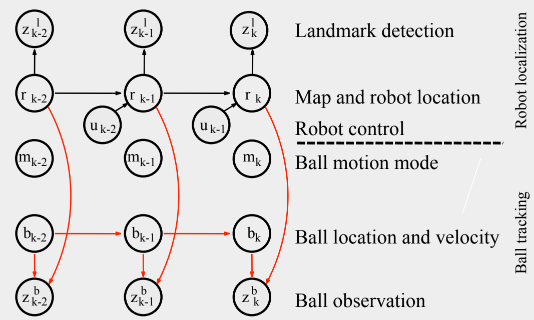
Dynamic Bayes Network for Ball Tracking



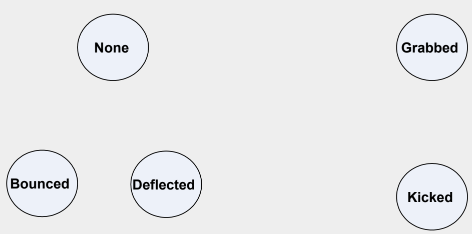
Robot Location



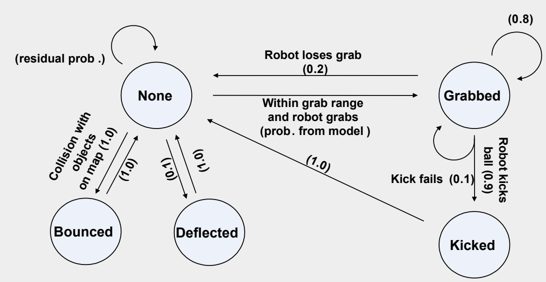
Robot and Ball Location (and velocity)



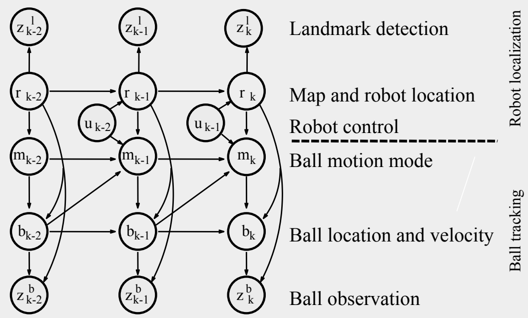
Ball-Environment Interactions



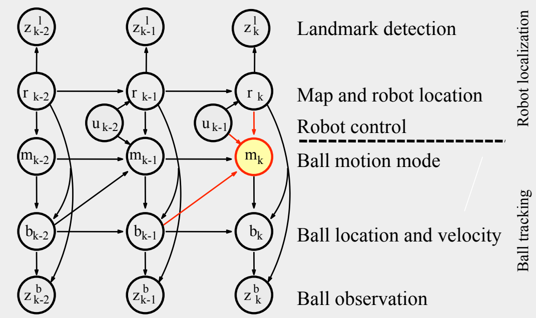
Ball-Environment Interactions



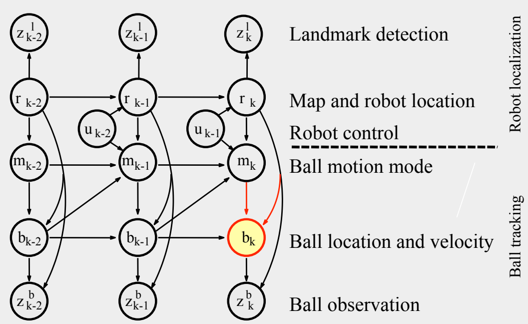
Integrating Discrete Ball Motion Mode



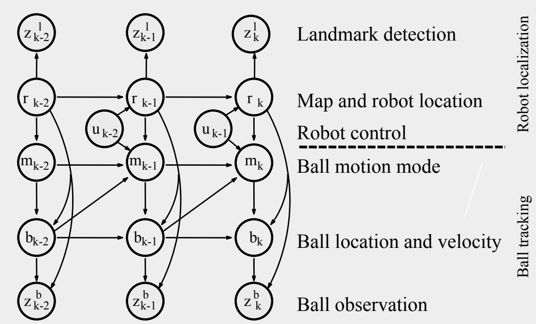
Grab Example (1)



Grab Example (2)



Inference: Posterior Estimation



$$p(b_k, m_k, r_k \mid z^b_{1:k}, z^l_{1:k}, u_{1:k-1})$$

Rao-Blackwellised PF for Inference

- Represent posterior by random samples
- Each sample

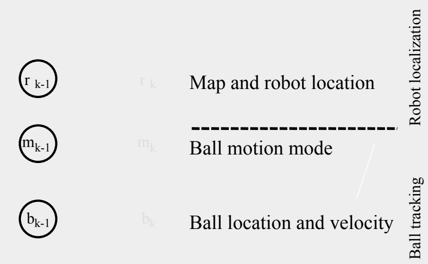
$$s_i = \langle r_i, m_i, b_i \rangle \quad = \quad \langle \langle x, y, \theta \rangle_i, m_i, \langle \mu, \Sigma \rangle_i \rangle$$

contains robot location, ball mode, ball Kalman filter

- Generate individual components of a particle stepwise using the factorization

$$p(b_k, m_{1:k}, r_{1:k} \mid z_{1:k}, u_{1:k-1}) =$$
$$p(b_k \mid m_{1:k}, r_{1:k}, z_{1:k}, u_{1:k-1}) \, p(m_{1:k} \mid r_{1:k}, z_{1:k}, u_{1:k-1}) \cdot p(r_{1:k} \mid z_{1:k}, u_{1:k-1})$$

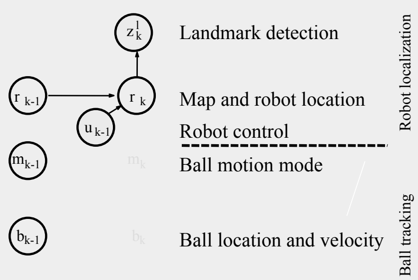
Rao-Blackwellised Particle Filter for Inference



- Draw a sample from the previous sample set:

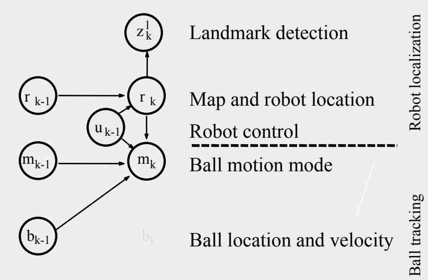
$$\langle r_{k-1}^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)} \rangle$$

Generate Robot Location



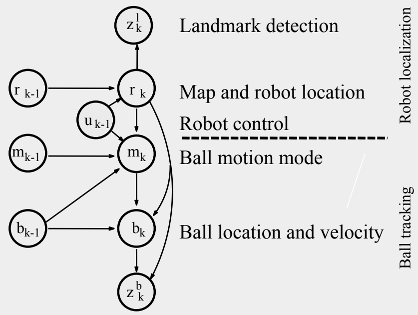
$$r_k^{(i)} \sim p(r_k \mid r_{k-1}^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, z_k, u_{k-1}) \quad \Rightarrow \quad \langle r_k^{(i)}, \_, \_ \rangle$$

Generate Ball Motion Model



$$m_k^{(i)} \sim p(m_k \mid r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, z_k, u_{k-1}) \quad \Rightarrow \quad \langle r_k^{(i)}, m_k^{(i)}, \_ \rangle$$

Update Ball Location and Velocity



$$b_k^{(i)} \sim p(b_k \mid r_k^{(i)}, m_k^{(i)}, b_{k-1}^{(i)}, z_k) \quad \Rightarrow \quad \langle r_k^{(i)}, m_k^{(i)}, b_k^{(i)} \rangle$$

Importance Resampling

- Weight sample by

$$w_k^{(i)} \propto p(z_k^l \mid r_k^{(i)})$$

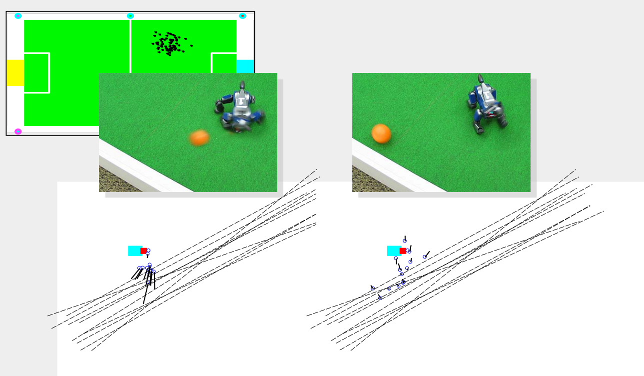
if observation is landmark detection and by

$$w_k^{(i)} \propto p(z_k^b \mid m_k^{(i)}, r_k^{(i)}, b_{k-1}^{(i)})$$
$$= \int p(z_k^b \mid m_k^{(i)}, r_k^{(i)}, b_k^{(i)}) p(b_k^{(i)} \mid m_k^{(i)}, r_k^{(i)}, b_{k-1}^{(i)}) \, db_k$$

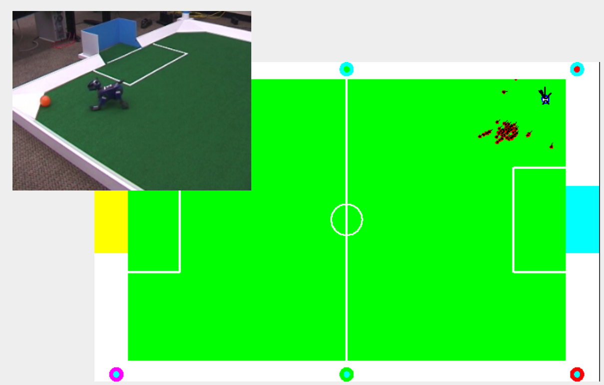
if observation is ball detection.

- Resample

Ball-Environment Interaction

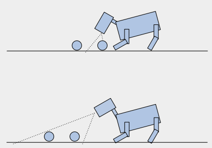


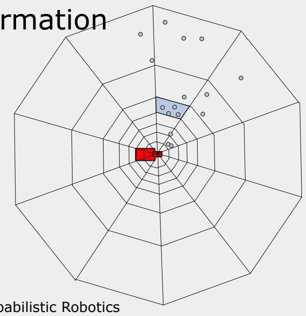
Ball-Environment Interaction



Tracking and Finding the Ball

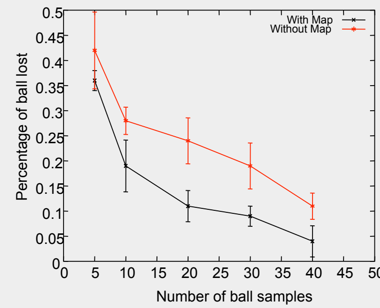
- Cluster ball samples by discretizing pan / tilt angles
- Uses negative information



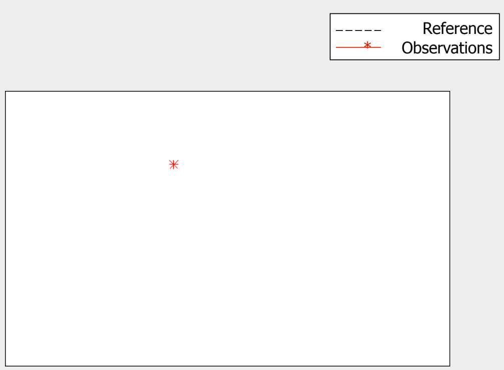


Experiment: Real Robot

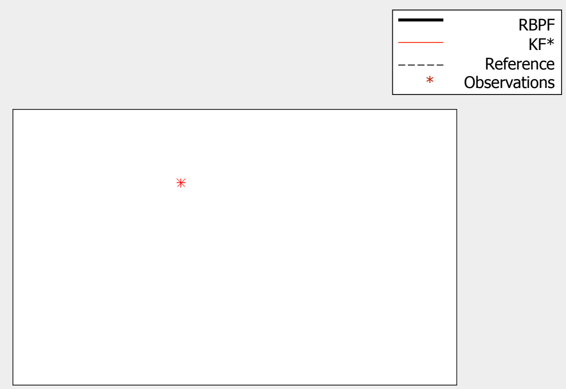
- Robot kicks ball 100 times, tries to find it afterwards
- Finds ball in 1.5 seconds on average



Simulation Runs

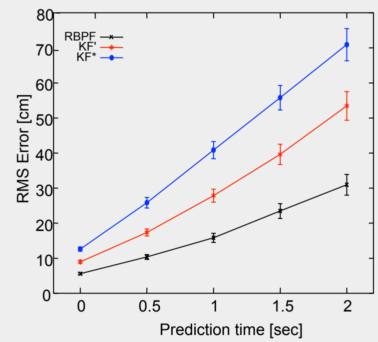


Comparison to KF* (optimized for straight motion)

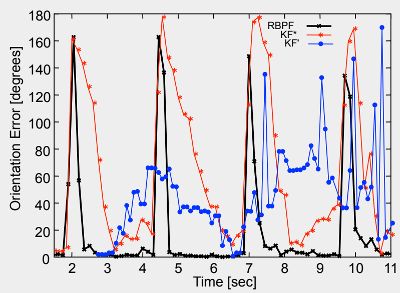


Comparison to KF′ (inflated prediction noise)

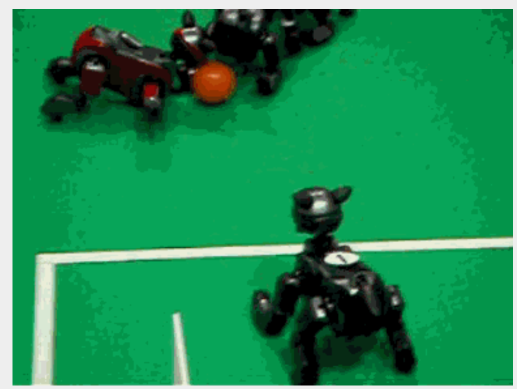
Error vs. Prediction Time



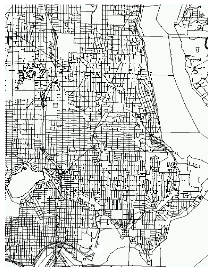
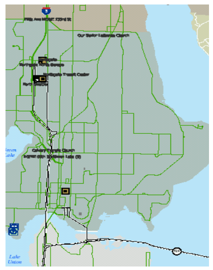
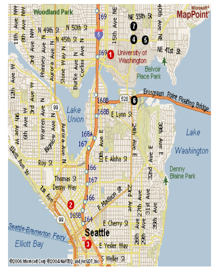
Orientation Errors



Goalie



Geographic Information Systems

STREET MAP Source: Tiger / Line data	BUS ROUTES / STOPS Source: Metro GIS	RESTAURANTS / STORES Source: MS MapPoint
		

Task

[Liao-Fox-Kautz: AAAI-04, AIJ-07]

- Given data stream from a wearable GPS unit
  - Infer the user's location and mode of transportation (foot, car, bus, bike, …)
  - Predict where user will go
  - Detect novel behavior / user errors

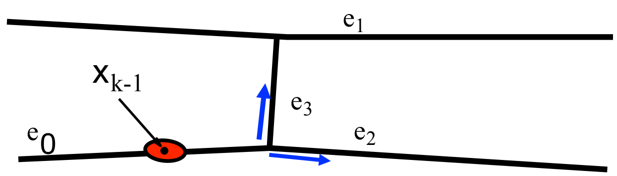
GPS-Tracking Is NOT Trivial

- Dead and semi-dead zones near buildings, trees, *etc*.
- Sparse measurements inside vehicles, especially bus
- Multi-path propagation
- Inaccurate street map
- …

Graph-based Location Estimation

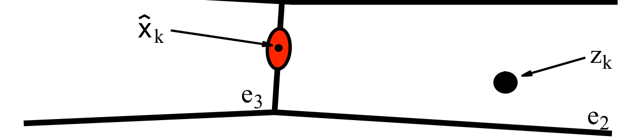
- Map is directed graph
- Location:
  - Edge e
  - Distance d from start of edge
- Prediction:
  - Move along edges according to velocity model
- Correction:
  - Update estimate based on GPS reading

Kalman Filtering on a Graph: Prediction Step



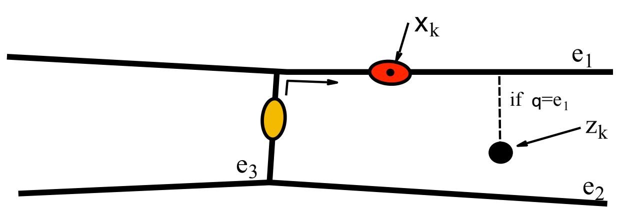
Problem: Predicted location is multi-modal

Kalman Filtering on a Graph: Correction Step



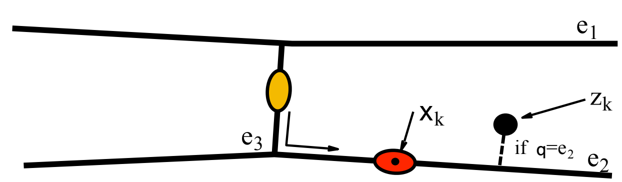
Problem: GPS reading is not on the graph

Kalman Filtering on a Graph: Correction Step



- Probabilistically "snap" GPS reading to the graph
- Perform A* search to compute innovation

Kalman Filtering on a Graph: Correction Step



- Probabilistically "snap" GPS reading to the graph
- Perform A* search to compute innovation

Location Tracking: Inference

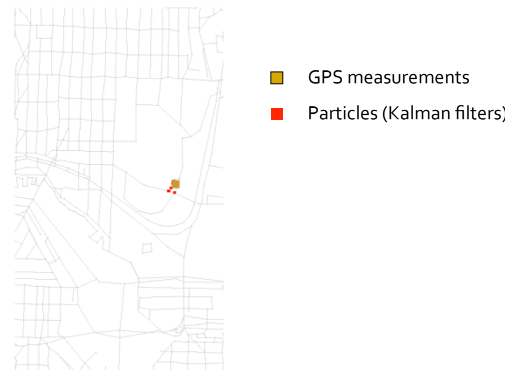
- Rao-Blackwellised particle filter represents posterior by sets of weighted particles:

$$S_k = \{< s^{(i)}, w^{(i)} >, i = 1, \ldots, n\}$$

- Each particle contains Kalman filter for location:

$$s^{(i)} = \left\langle \underbrace{e^{(i)}, v^{(i)}, \theta^{(i)}}_{\text{Edge transitions, velocities, edge associations}}, \underbrace{N^{(i)}(\mu, \sigma^2)}_{\text{Gaussian for position}} \right\rangle$$

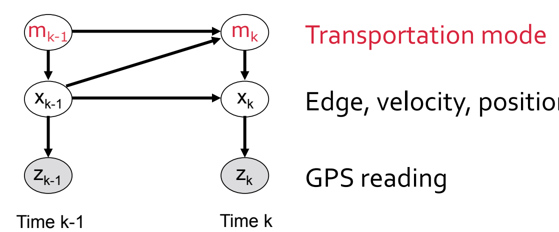
Tracking Example



Infer Mode of Transportation

- Encode prior knowledge into the model
  - Modes have different velocity distributions
  - Buses run on bus routes
  - Get on/off the bus near bus stops
  - Switch to car near car location

Dynamic Bayesian Network



Particles: $s^{(i)} = \left\langle m^{(i)}, e^{(i)}, v^{(i)}, \theta^{(i)}, N^{(i)}(\mu, \sigma^2) \right\rangle$

Infer Location and Transportation

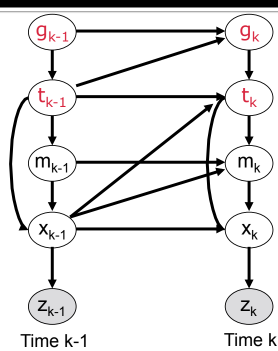
- Measurements
- Projections

Green	Bus mode
Red	Car mode
Blue	Foot mode

Transportation Routines

- Goal (destination):
  - workplace (home, friends, restaurant, ...)
- Trip segments: <start, end, transportation>
  - Home to Bus stop A on Foot
  - Bus stop A to Bus stop B on Bus
  - Bus stop B to workplace on Foot

Hierarchical Model



Goal

Trip segment

Transportation mode

Edge, velocity, position

GPS reading

Particles: $s^{(i)} = \left\langle \langle g,t \rangle^{(i)}, m^{(i)}, e^{(i)}, v^{(i)}, \theta^{(i)}, N^{(i)}(\mu,\sigma^2) \right\rangle$

Model Learning

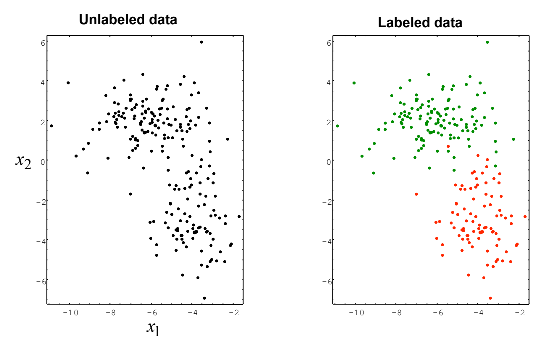
- Key to goal / path prediction and error detection
- Customized model for each user
- Unsupervised model learning
  - Learn variable domains (goals, trip segments)
  - Learn transition parameters (goals, trips, edges)
- Training data
  - 30 days GPS readings of one user, logged every second (when outdoors)

The EM Algorithm

- Iterative method for finding maximum likelihood estimates of parameters in statistical models, where the model depends on unobserved (unlabeled) latent variables.

The problem of finding labels for unlabeled data

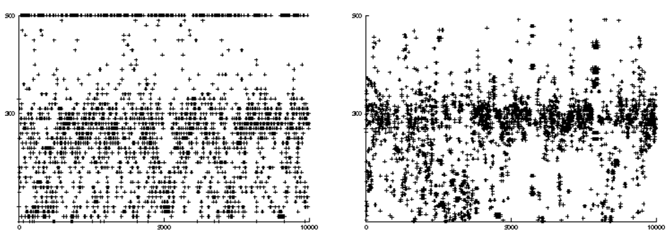
In nature, items often do not come with labels. How can we learn labels without a teacher?



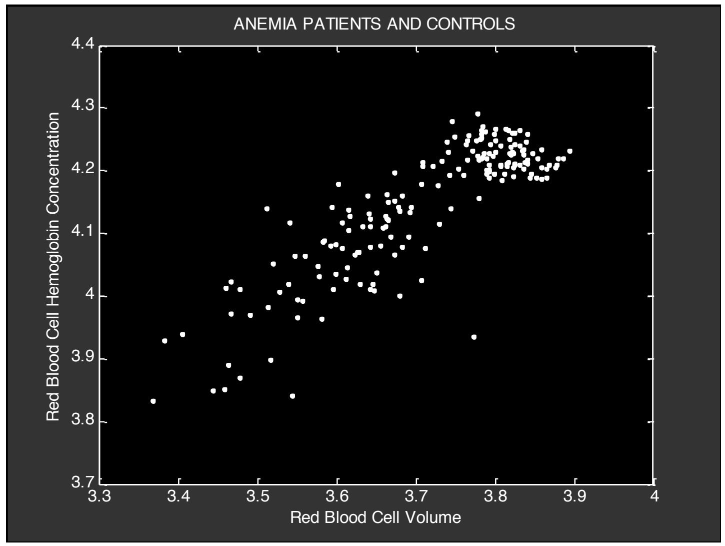
From Shadmehr & Diedrichsen

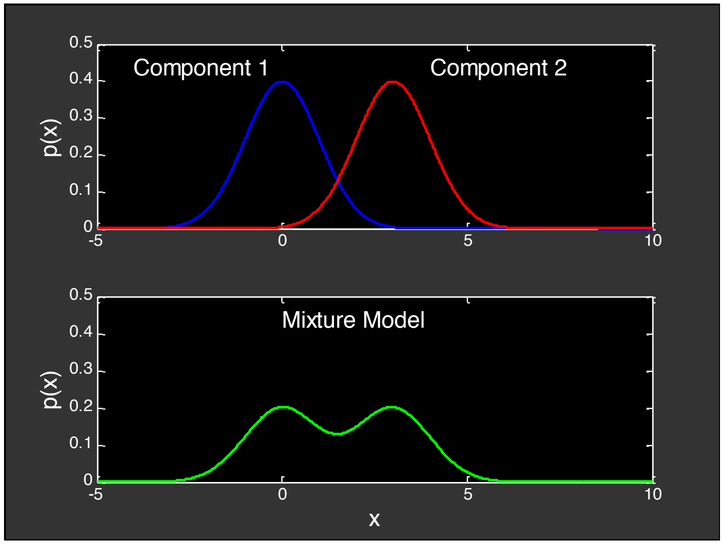
Raw Proximity Sensor Data

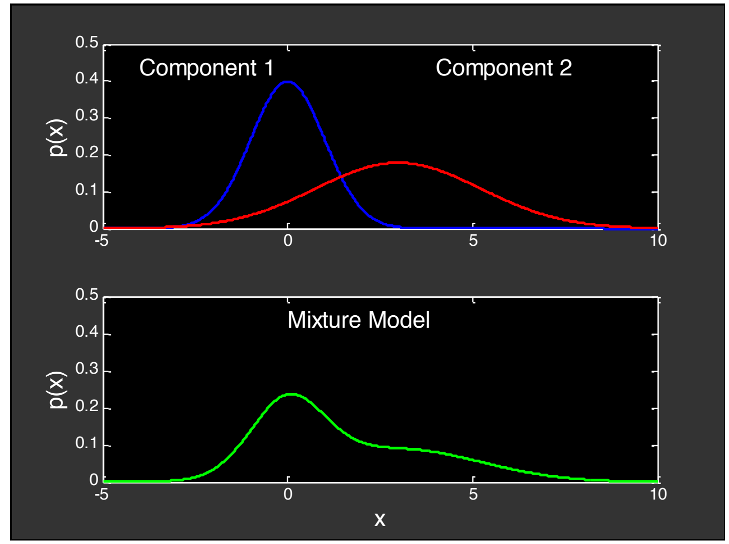
Measured distances for expected distance of 300 cm.

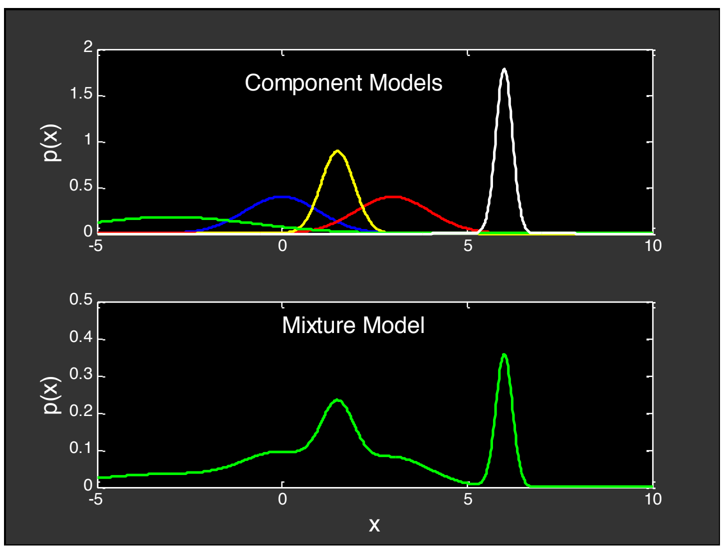


Sonar Laser









Mixtures

If our data is not labeled, we can hypothesize that:

1. There are exactly m classes in the data: $y \in \{1,2,\ldots,m\}$
2. Each class y occurs with a specific frequency: $P(y)$
3. Examples of class y are governed by a specific distribution: $p(\mathbf{x}|y)$

According to our hypothesis, each example $x^{(i)}$ must have been generated from a specific "mixture" distribution:

$$p(\mathbf{x}) = \sum_{j=1}^{m} P(y=j)\, p(\mathbf{x}|y=j)$$

We might hypothesize that the distributions are Gaussian:

Parameters of the distributions $\theta = \{P(y=1), \mu_1, \Sigma_1, \cdots, P(y=m), \mu_m, \Sigma_m\}$

$$p(\mathbf{x}|\theta) = \sum_{j=1}^{m} P(y=j)\, N(\mathbf{x}|\mu_j, \Sigma_j)$$

Mixing proportions **Normal distribution**

Learning Mixtures from Data

Consider fixed K = 2

e.g., Unknown parameters Q = $\{m_1, s_1, m_2, s_2, a_1\}$

Given data D = $\{x_1,\ldots\ldots x_N\}$, we want to find the parameters Q that "best fit" the data

1977: The EM Algorithm

- Dempster, Laird, and Rubin
  - General framework for likelihood-based parameter estimation with missing data
    - start with initial guesses of parameters
    - E-step: estimate memberships given params
    - M-step: estimate params given memberships
    - Repeat until convergence
  - Converges to a (local) maximum of likelihood
  - E-step and M-step are often computationally simple
  - Generalizes to maximum a posteriori (with priors)

EM for Mixture of Gaussians

- E-step: Compute probability that point x_j was generated by component i:

$$p_{ij} = \alpha\, P(x_j \mid C = i)\, P(C = i)$$

$$p_i = \sum_j p_{ij}$$

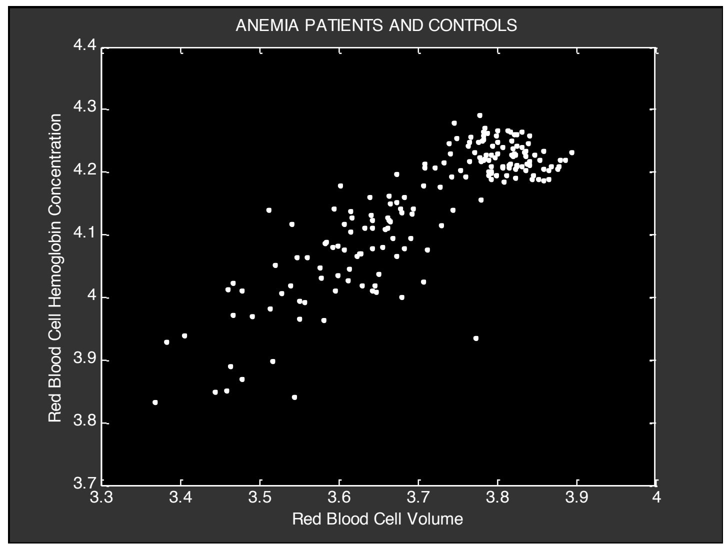
- M-step: Compute new mean, covariance, and component weights:

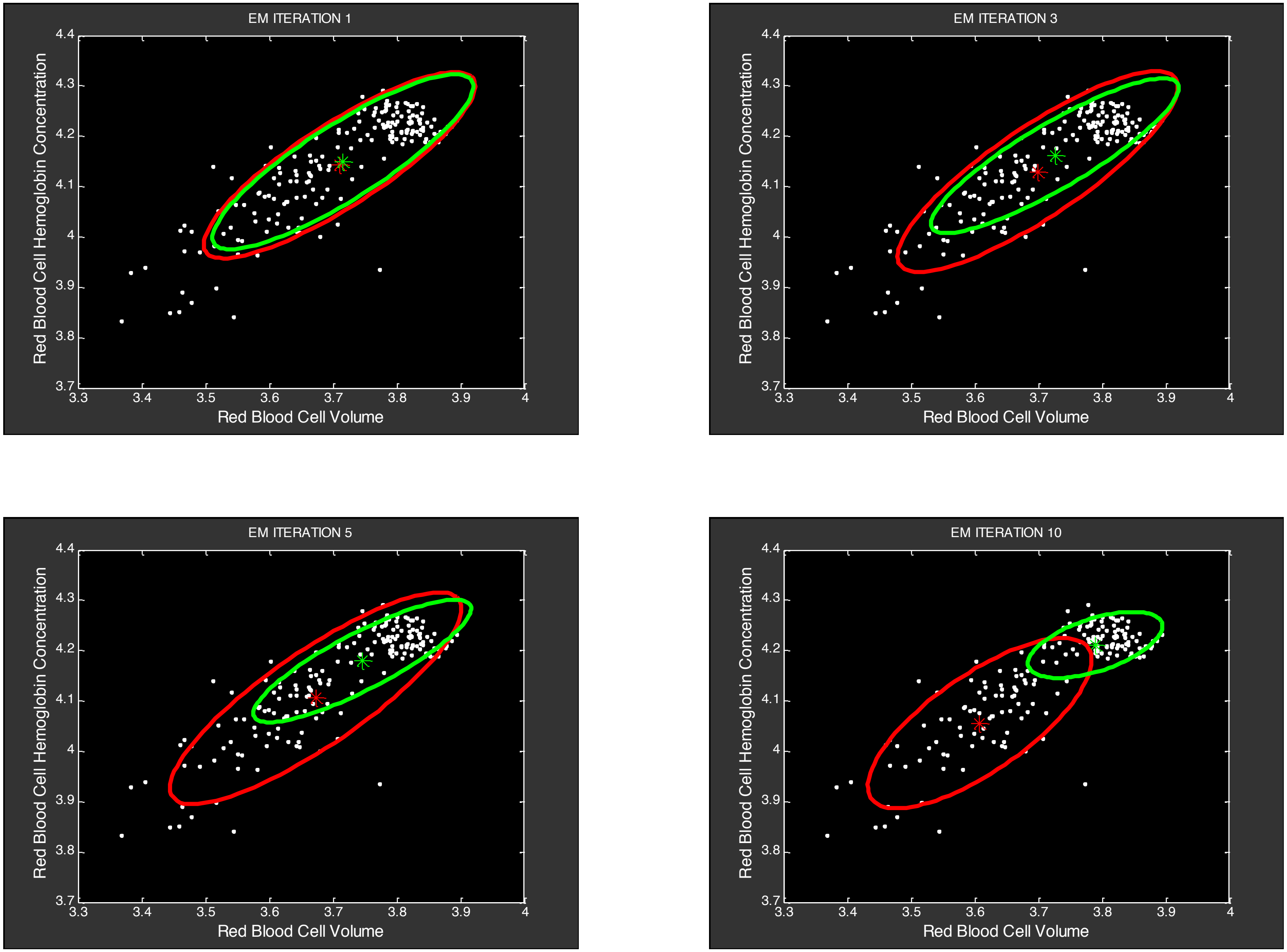
$$\mu_i \leftarrow \sum_j p_{ij} x_j / p_i$$

$$\sigma^2 \leftarrow \sum_j p_{ij} (x_j - \mu_i)^2 / p_i$$

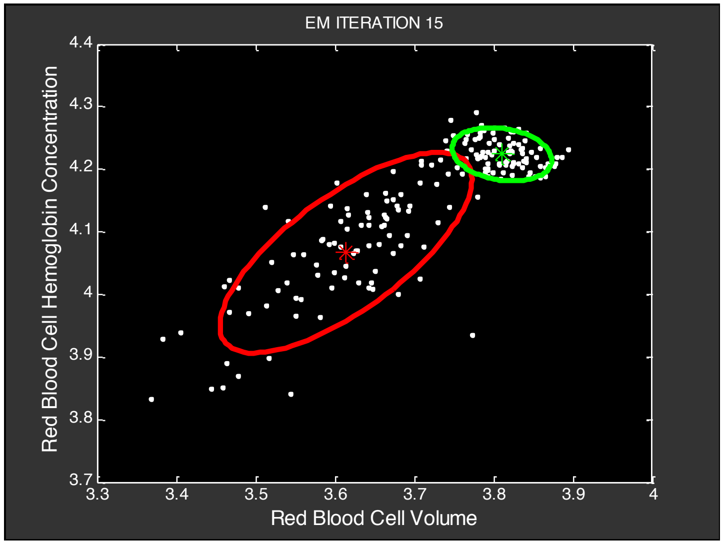
$$w_i \leftarrow p_i$$

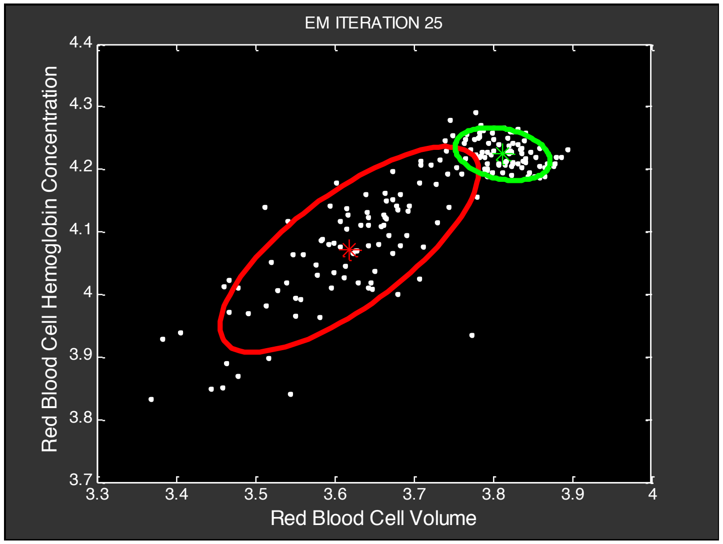
55

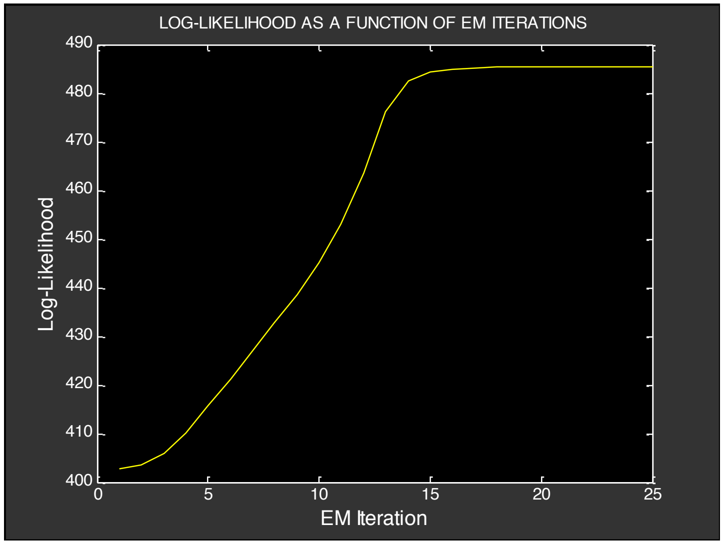


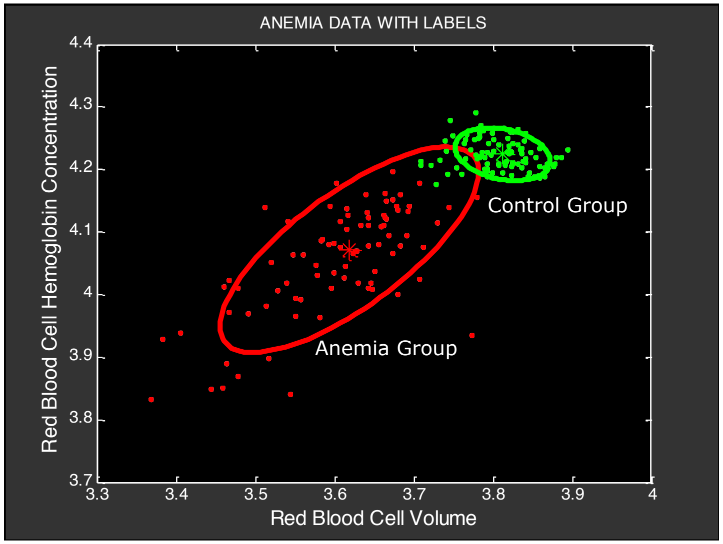


EM ITERATION 1
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
EM ITERATION 3
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
EM ITERATION 5
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
EM ITERATION 10
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume

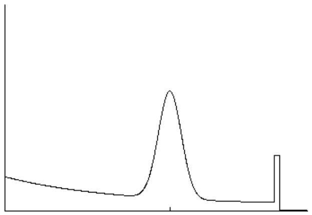

EM ITERATION 15
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume


EM ITERATION 25
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume


LOG-LIKELIHOOD AS A FUNCTION OF EM ITERATIONS
Log-Likelihood
EM Iteration


ANEMIA DATA WITH LABELS
Control Group
Anemia Group
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume

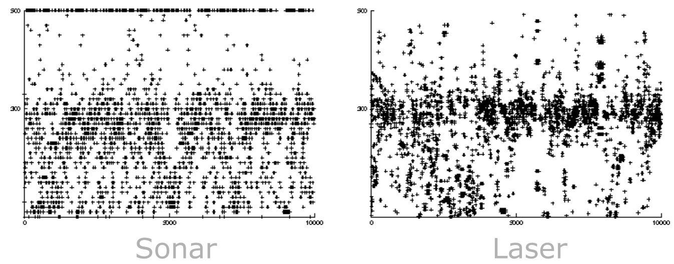
Mixture Density



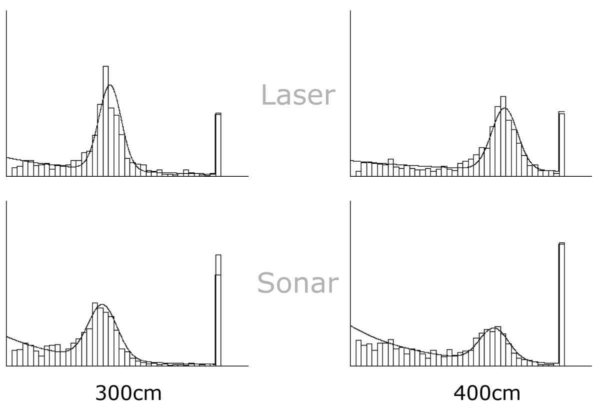
$$P(z \mid x, m) = \begin{pmatrix} \alpha_{\text{hit}} \\ \alpha_{\text{unexp}} \\ \alpha_{\max} \\ \alpha_{\text{rand}} \end{pmatrix}^T \cdot \begin{pmatrix} P_{\text{hit}}(z \mid x, m) \\ P_{\text{unexp}}(z \mid x, m) \\ P_{\max}(z \mid x, m) \\ P_{\text{rand}}(z \mid x, m) \end{pmatrix}$$

Raw Sensor Data

Measured distances for expected distance of 300 cm.



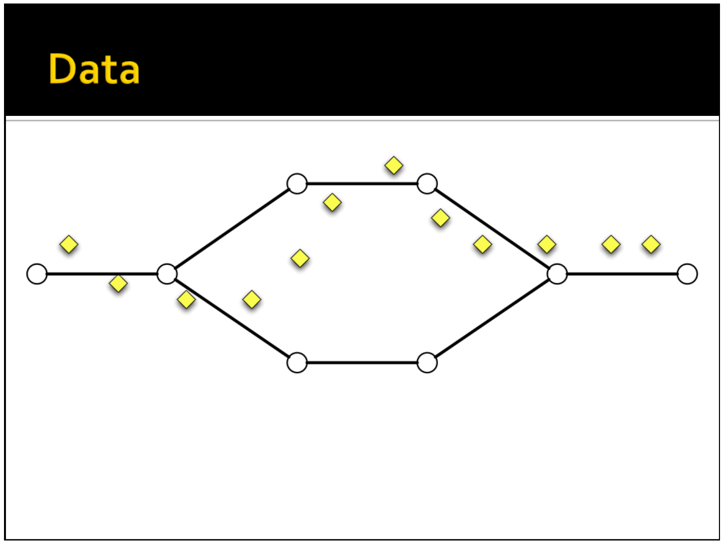
Approximation Results

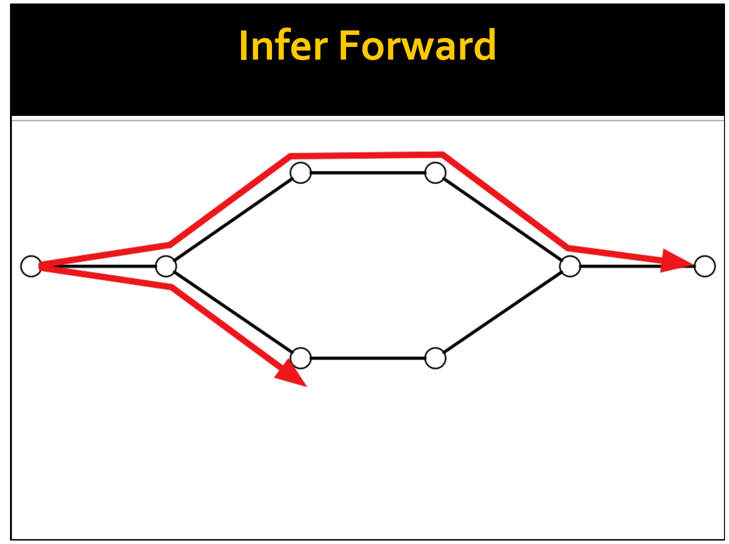


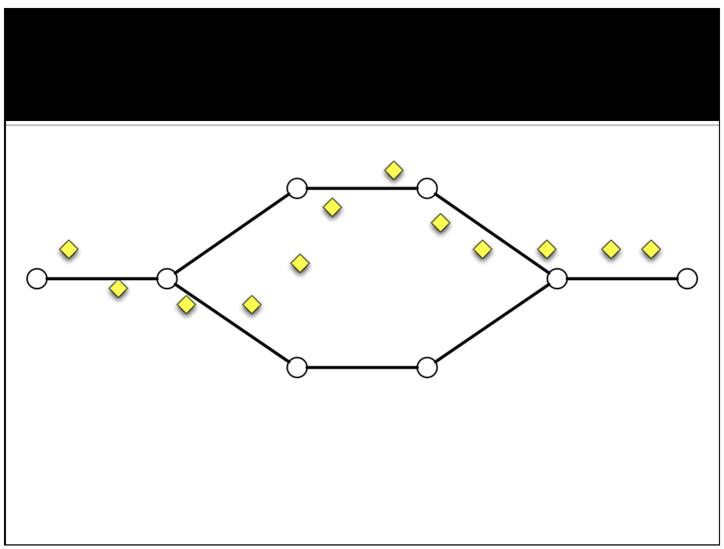
Learning

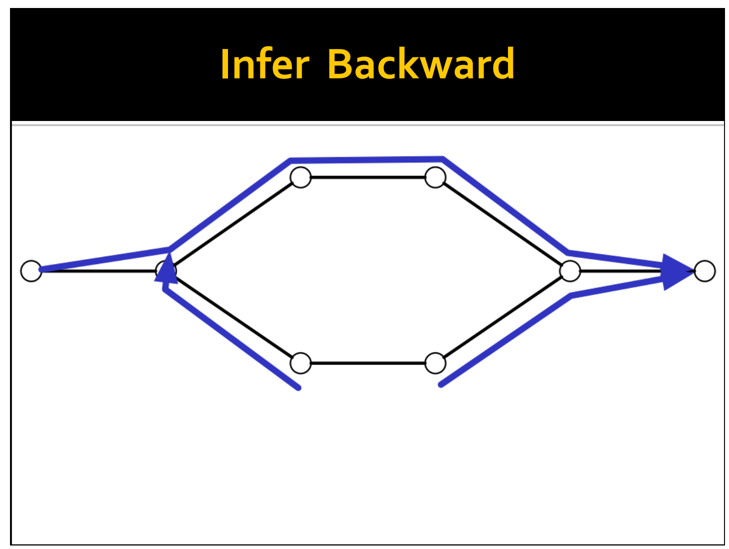
- E-Step
  - Infer the transportation behavior given the model
  - Smooth our inference
    - Infer the data forward in time
    - Infer the data backward in time

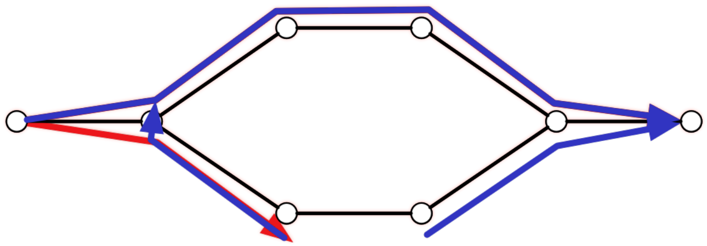
68


Data

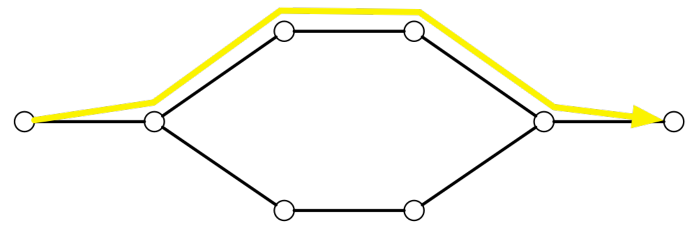

Infer Forward




Infer Backward



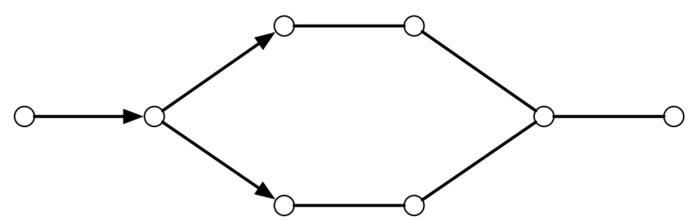
Smoothed Inference

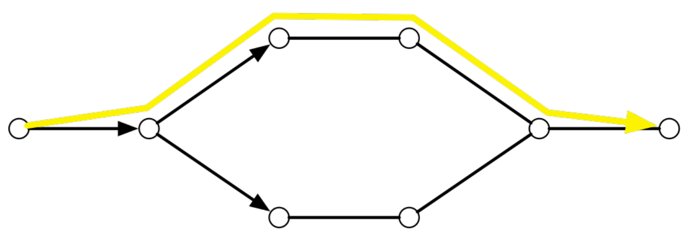


Learning

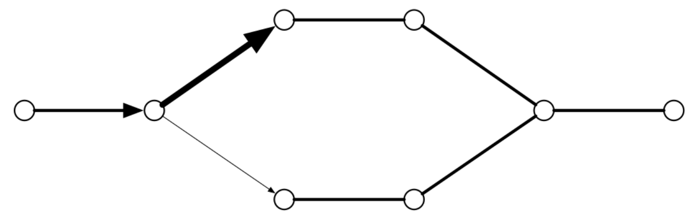
- M-Step
  - Update the model parameters to better explain the smoothed inference
  - Stochastic version of the Baum-Welch Algorithm
    - Count how many particles move from one edge to the next
    - Update the transition probabilities to reflect the counts

75

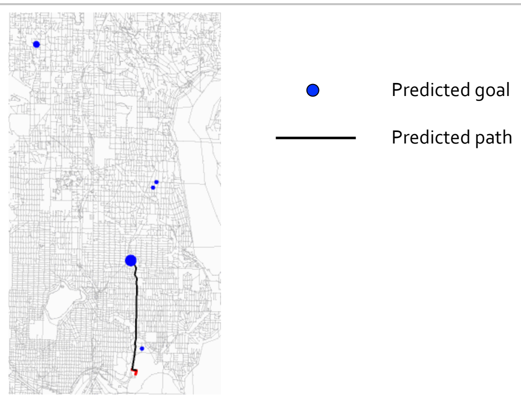




Re-estimate Model

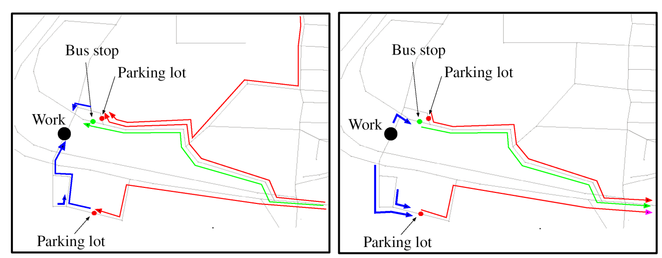


Predict Goal and Path



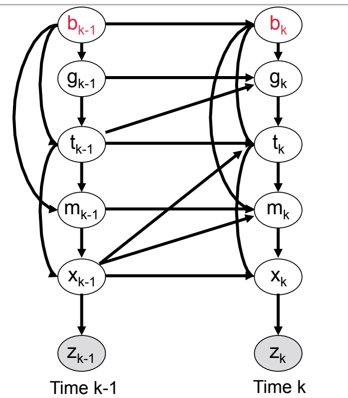
Learned Transition Parameters

GOING TO THE WORKPLACE GOING HOME



High probability transitions: **bus** **car** **foot**

Detect Atypical Behavior and User Errors [Patterson-Liao-etAl: Ubicomp-04]



Behavior mode
normal / unknown / error

Goal

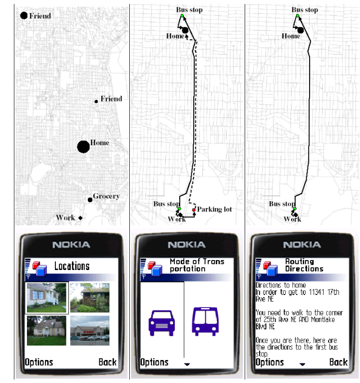
Trip segment

Transportation mode

Edge, velocity, position

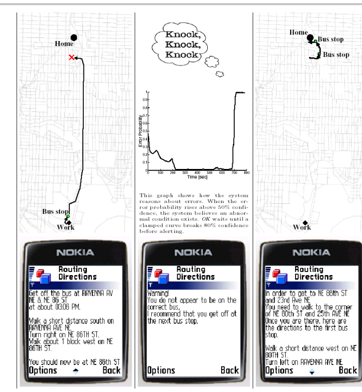
GPS reading

Application: Opportunity Knocks



Detect User Errors

Application: Opportunity Knocks



Discussion

- Particle filters are intuitive and simple
  - Support point-wise thinking (reduced uncertainty)
  - It's an art to make them work
  - Good for test implementation if system behavior is not well known
- Inefficient compared to Kalman filter
- Rao-Blackwellization
  - Only sample discrete / highly non-linear parts of state space
  - Solve remaining part analytically (KF,discrete)