CSE-473
Introduction to Artificial Intelligence

Kalman Filters and
Rao-Blackwelized Particle Filters

Bayes Filter Reminder

- Prediction

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$$

- Correction

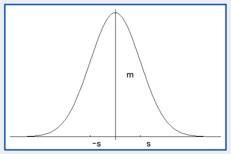
$$bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$$

Gaussians

$$p(x) \sim N(\mu, \sigma^2):$$

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

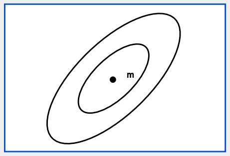
Univariate



$$p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}):$$

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

Multivariate



Properties of Gaussians

$$\left.\begin{array}{l} X \sim N(\mu, \sigma^2) \\ Y = aX + b \end{array}\right\} \Rightarrow Y \sim N(a\mu + b, a^2\sigma^2)$$

$$\left.\begin{array}{l} X_1 \sim N(\mu_1, \sigma_1^2) \\ X_2 \sim N(\mu_2, \sigma_2^2) \end{array}\right\} \Rightarrow p(X_1)\cdot p(X_2) \sim N\left(\frac{\sigma_2^2}{\sigma_1^2+\sigma_2^2}\mu_1 + \frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2}\mu_2,\ \frac{1}{\sigma_1^{-2}+\sigma_2^{-2}}\right)$$

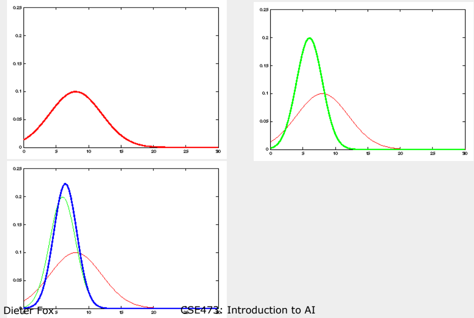
Multivariate Gaussians

$$\left.\begin{array}{l} X \sim N(\mu, \Sigma) \\ Y = AX + B \end{array}\right\} \Rightarrow Y \sim N(A\mu + B, A\Sigma A^T)$$

$$\left.\begin{array}{l} X_1 \sim N(\mu_1, \Sigma_1) \\ X_2 \sim N(\mu_2, \Sigma_2) \end{array}\right\} \Rightarrow p(X_1)\cdot p(X_2) \sim N\left(\frac{\Sigma_2}{\Sigma_1+\Sigma_2}\mu_1 + \frac{\Sigma_1}{\Sigma_1+\Sigma_2}\mu_2,\ \frac{1}{\Sigma_1^{-1}+\Sigma_2^{-1}}\right)$$

- We stay in the “Gaussian world” as long as we start with Gaussians and perform only linear transformations.

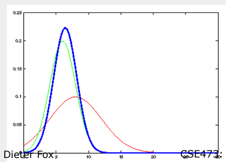
Kalman Filter Updates in 1D



Kalman Filter Updates in 1D

$$bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - \bar{\mu}_t) \\ \sigma_t^2 = (1 - K_t)\bar{\sigma}_t^2 \end{cases} \quad \text{with} \quad K_t = \frac{\bar{\sigma}_t^2}{\bar{\sigma}_t^2 + \sigma_{obs,t}^2}$$

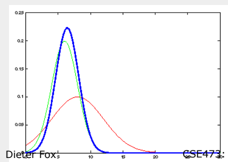
$$bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - C_t\bar{\mu}_t) \\ \Sigma_t = (I - K_t C_t)\bar{\Sigma}_t \end{cases} \quad \text{with} \quad K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

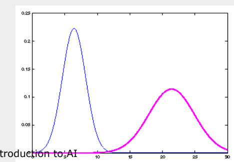


Kalman Filter Updates in 1D

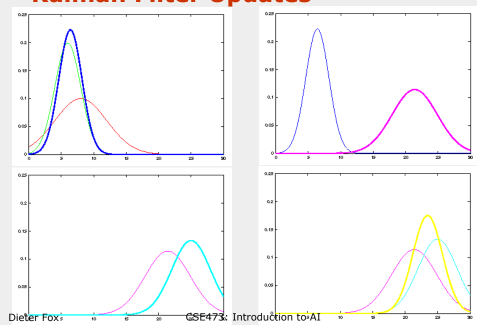
$$\overline{bel}(x_t) = \begin{cases} \bar{\mu}_t = a_t \mu_{t-1} + b_t u_t \\ \bar{\sigma}_t^2 = a_t^2 \sigma_t^2 + \sigma_{act,t}^2 \end{cases}$$

$$\overline{bel}(x_t) = \begin{cases} \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t \\ \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t \end{cases}$$





Kalman Filter Updates

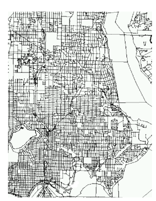


Kalman Filter Summary

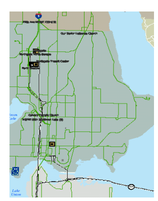
- Highly efficient: Polynomial in measurement dimensionality k and state dimensionality n:
  $O(k^{2.376} + n^2)$
- Optimal for linear Gaussian systems!
- Most systems are nonlinear → EKF

Geographic Information Systems

STREET MAP
Source: Tiger / Line data



BUS ROUTES / STOPS
Source: Metro GIS



RESTAURANTS / STORES
Source: MS MapPoint

Task

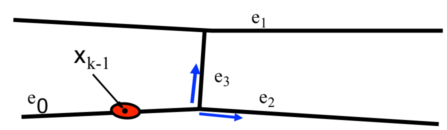
[Liao-Fox-Kautz: AAAI-04, AIJ-07]

- 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

Graph-based Location Estimation

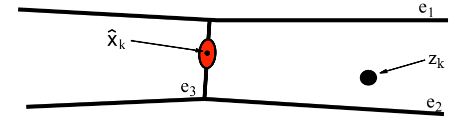
- Map is directed graph
- Location:
  - Edge e
  - Distance d from start of edge
- Prediction:
  - Move along edges according to velocity model
- Correction:
  - Update estimate based on GPS reading

Kalman Filtering on a Graph: Prediction Step



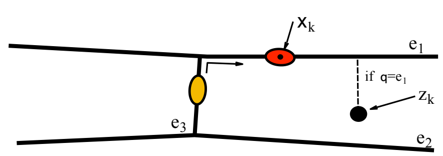
Problem: Predicted location is multi-modal

Kalman Filtering on a Graph: Correction Step



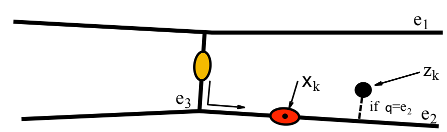
Problem: GPS reading is not on the graph

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:

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

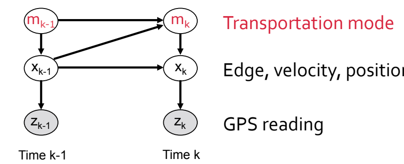
Tracking Example

GPS measurements
Particles (Kalman filters)

Infer Mode of Transportation

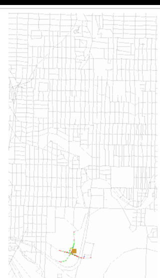
- Encode prior knowledge into the model
  - Modes have different velocity distributions
  - Buses run on bus routes
  - Get on/off the bus near bus stops
  - Switch to car near car location

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



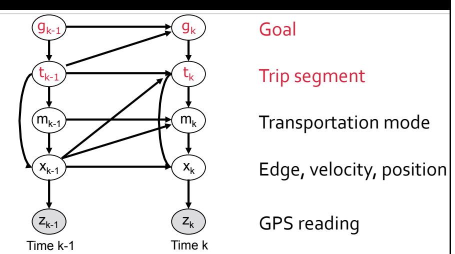
Measurements
Projections

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

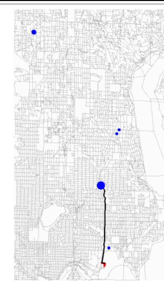


Particles: $s^{(i)} = \left\langle \langle g,t \rangle^{(i)}, m^{(i)}, e^{(i)}, v^{(i)}, \theta^{(i)}, N^{(i)}(\mu,\sigma^2) \right\rangle$

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)

Predict Goal and Path

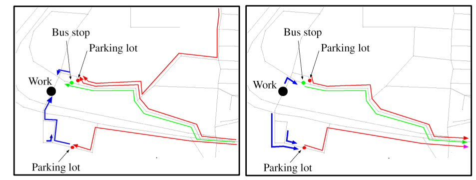


Predicted goal

Predicted path

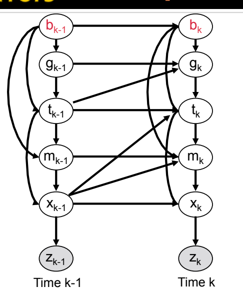
Learned Transition Parameters

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

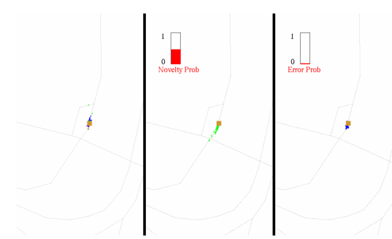
Transportation mode

Edge, velocity, position

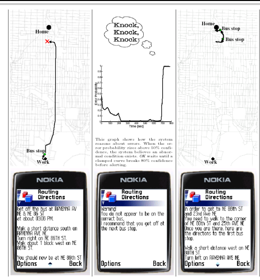
GPS reading

Application: Opportunity Knocks

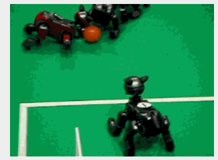
Detect User Errors



Application: Opportunity Knocks

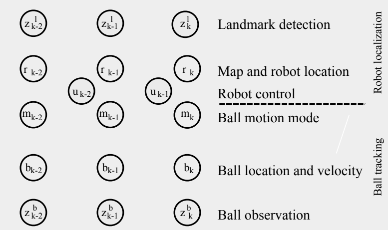


Ball Tracking in RoboCup

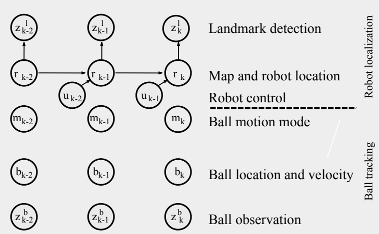


- Extremely noisy (nonlinear) motion of observer
- Inaccurate sensing, limited processing power
- Interactions between target and environment
- Interactions between robot(s) and target

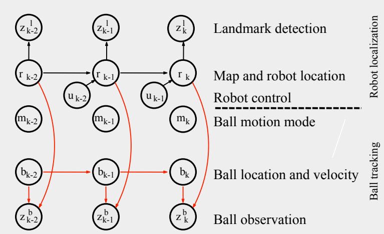
Dynamic Bayes Network for Ball Tracking



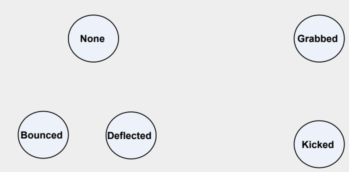
Robot Location



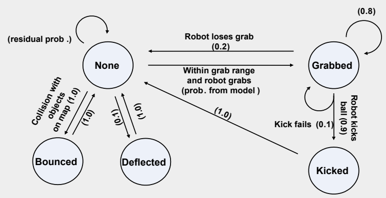
Robot and Ball Location (and velocity)



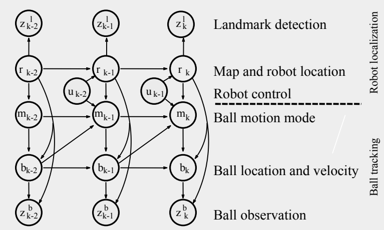
Ball-Environment Interactions



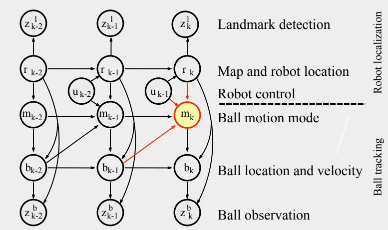
Ball-Environment Interactions



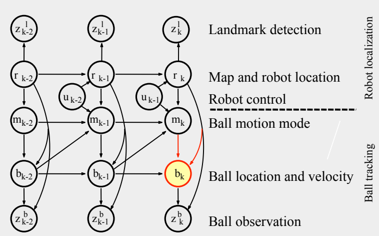
Integrating Discrete Ball Motion Mode



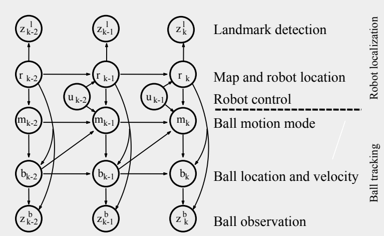
Grab Example (1)



Grab Example (2)

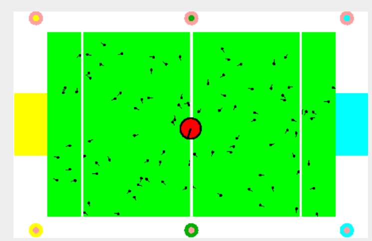


Inference: Posterior Estimation



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

Particle Filter for Robot Localization



Rao-Blackwellised PF for Inference

- Represent posterior by random samples
- Each sample

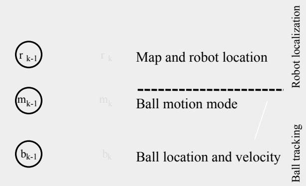
$$s_i = \langle r_i, m_i, b_i \rangle = \langle \langle x, y, \theta \rangle_i, m_i, \langle \mu, \Sigma \rangle_i \rangle$$

contains robot location, ball mode, ball Kalman filter

- Generate individual components of a particle stepwise using the factorization

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

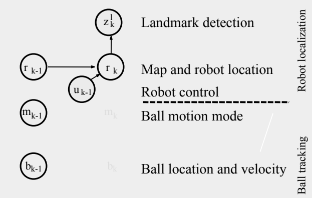
Rao-Blackwellised Particle Filter for Inference



- Draw a sample from the previous sample set:

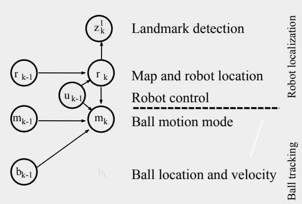
$$\langle r_{k-1}^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)} \rangle$$

Generate Robot Location



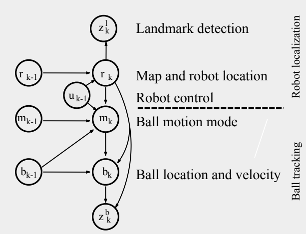
$$r_k^{(i)} \sim p(r_k \mid r_{k-1}^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, z_k, u_{k-1}) \quad \Rightarrow \quad \langle r_k^{(i)}, \_, \_ \rangle$$

Generate Ball Motion Model



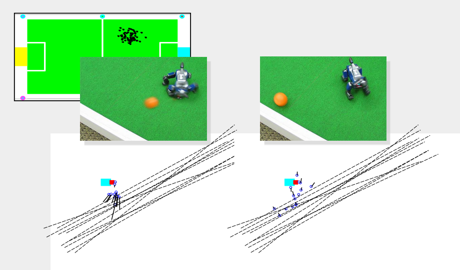
$$m_k^{(i)} \sim p(m_k \mid r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, z_k, u_{k-1}) \quad \Rightarrow \quad \langle r_k^{(i)}, m_k^{(i)}, \_ \rangle$$

Update Ball Location and Velocity

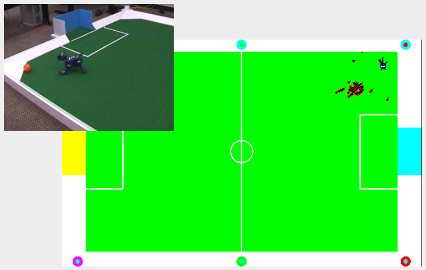


$$b_k^{(i)} \sim p(b_k \mid r_k^{(i)}, m_k^{(i)}, b_{k-1}^{(i)}, z_k) \quad \Rightarrow \quad \langle r_k^{(i)}, m_k^{(i)}, b_k^{(i)} \rangle$$

Ball-Environment Interaction



Ball-Environment Interaction



Dieter Fox CSE473: Introduction to AI 49

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)

Dieter Fox CSE473: Introduction to AI 50