Introduction to Artificial Intelligence

V22.0472-001 Fall 2009

Lecture 18: Particle &
Kalman Filtering

Rob Fergus – Dept of Computer Science, Courant Institute, NYU

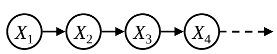
Slides from John DeNero, Dan Klein, Haris Baltzakis, Dieter Fox

Announcements

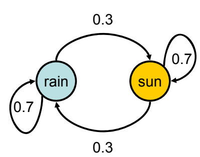
- Final exam will be at 7pm on Wednesday December 14th
  - Date of last class
  - 1.5 hrs long
- I won't ask anything about the last few classes.

Recap: Reasoning Over Time

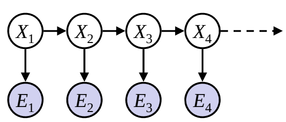
- Stationary Markov models



$P(X_1)$ $\quad P(X|X_{-1})$



- Hidden Markov models



$P(E|X)$

X	E	P
rain	umbrella	0.9
rain	no umbrella	0.1
sun	umbrella	0.2
sun	no umbrella	0.8

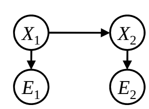
Recap: Filtering

Elapse time: compute P(X_t | $e_{1:t-1}$)

$$P(x_t|e_{1:t-1}) = \sum_{x_{t-1}} P(x_{t-1}|e_{1:t-1}) \cdot P(x_t|x_{t-1})$$

Observe: compute P(X_t | $e_{1:t}$)

$$P(x_t|e_{1:t}) \propto P(x_t|e_{1:t-1}) \cdot P(e_t|x_t)$$

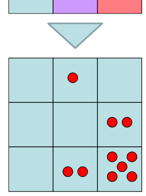


	Belief: <P(rain), P(sun)>	
$P(X_1)$	<0.5, 0.5>	*Prior on X_1*
$P(X_1 \mid E_1 = umbrella)$	<0.82, 0.18>	*Observe*
$P(X_2 \mid E_1 = umbrella)$	<0.63, 0.37>	*Elapse time*
$P(X_2 \mid E_1 = umb, E_2 = umb)$	<0.88, 0.12>	*Observe*

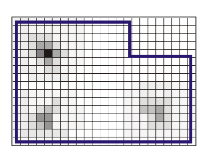
Particle Filtering

- Sometimes |X| is too big to use exact inference
  - |X| may be too big to even store B(X)
  - E.g. X is continuous
  - $|X|^2$ may be too big to do updates
- Solution: approximate inference
  - Track samples of X, not all values
  - Samples are called particles
  - Time per step is linear in the number of samples
  - But: number needed may be large
  - In memory: list of particles, not states
- This is how robot localization works in practice

0.0	0.1	0.0
0.0	0.0	0.2
0.0	0.2	0.5



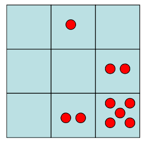
Example: State Representations for Robot Localization



Grid Based approaches (Markov localization)

Representation: Particles

- Our representation of P(X) is now a list of N particles (samples)
  - Generally, N << |X|
  - Storing map from X to counts would defeat the point
- P(x) approximated by number of particles with value x
  - So, many x will have P(x) = 0!
  - More particles, more accuracy
- For now, all particles have a weight of 1



Particles:
(3,3)
(2,3)
(3,3)
(3,2)
(3,3)
(3,2)
(2,1)
(3,3)
(3,3)
(2,1)

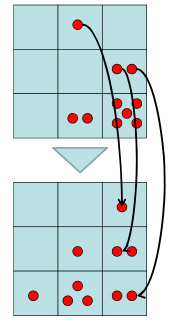
7

Particle Filtering: Elapse Time

- Each particle is moved by sampling its next position from the transition model

$$x' = \text{sample}(P(X'|x))$$

  - This is like prior sampling – samples' frequencies reflect the transition probs
  - Here, most samples move clockwise, but some move in another direction or stay in place
- This captures the passage of time
  - If we have enough samples, close to the exact values before and after (consistent)



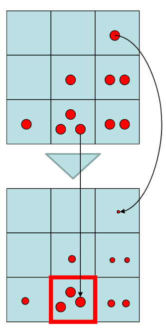
Particle Filtering: Observe

- Slightly trickier:
  - Don't do rejection sampling (why not?)
  - We don't sample the observation, we fix it
  - This is similar to likelihood weighting, so we downweight our samples based on the evidence

$$w(x) = P(e|x)$$

$$B(X) \propto P(e|X)B'(X)$$

  - Note that, as before, the probabilities don't sum to one, since most have been downweighted (in fact they sum to an approximation of P(e))

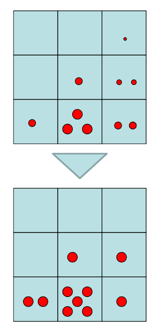


Particle Filtering: Resample

- Rather than tracking weighted samples, we resample
- N times, we choose from our weighted sample distribution (i.e. draw with replacement)
- This is equivalent to renormalizing the distribution
- Now the update is complete for this time step, continue with the next one

Old Particles:
(3,3) w=0.1
(2,1) w=0.9
(2,1) w=0.9
(3,1) w=0.4
(3,2) w=0.3
(2,2) w=0.4
(1,1) w=0.4
(3,1) w=0.4
(2,1) w=0.9
(3,2) w=0.3

Old Particles:
(2,1) w=1
(2,1) w=1
(2,1) w=1
(3,2) w=1
(2,2) w=1
(2,1) w=1
(1,1) w=1
(3,1) w=1
(2,1) w=1
(1,1) w=1



Particle Filter Algorithm

$$Bel(x_t) = \eta\, p(z_t | x_t) \int p(x_t | x_{t-1}, u_{t-1})\, Bel(x_{t-1})\, dx_{t-1}$$

draw x^i_{t-1} from $Bel(x_{t-1})$

draw x^i_t from $p(x_t | x^i_{t-1}, u_{t-1})$

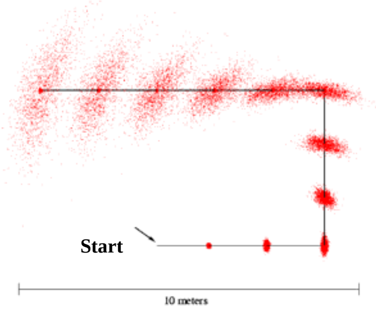
Importance factor for x^i_t:

$$w^i_t = \frac{\text{target distribution}}{\text{proposal distribution}} = \frac{\eta\, p(z_t | x_t)\, p(x_t | x_{t-1}, u_{t-1})\, Bel\,(x_{t-1})}{p(x_t | x_{t-1}, u_{t-1})\, Bel\,(x_{t-1})} \propto p(z_t | x_t)$$

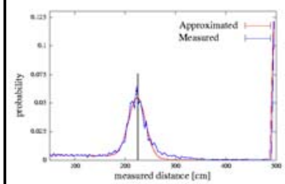
Robot Localization

- In robot localization:
  - We know the map, but not the robot's position
  - Observations may be vectors of range finder readings
  - State space and readings are typically continuous (works basically like a very fine grid) and so we cannot store B(X)
  - Particle filtering is a main technique

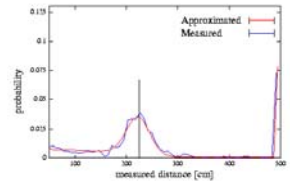
Robot Motion Model



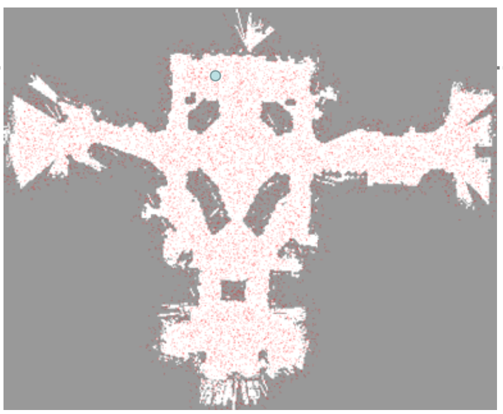
Proximity Sensor Model

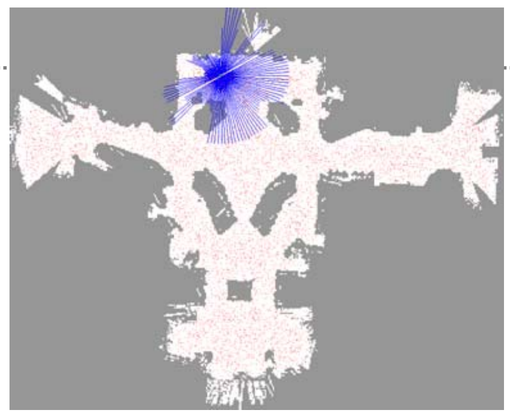


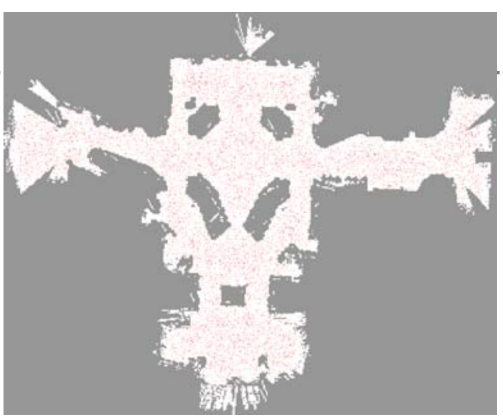
Laser sensor

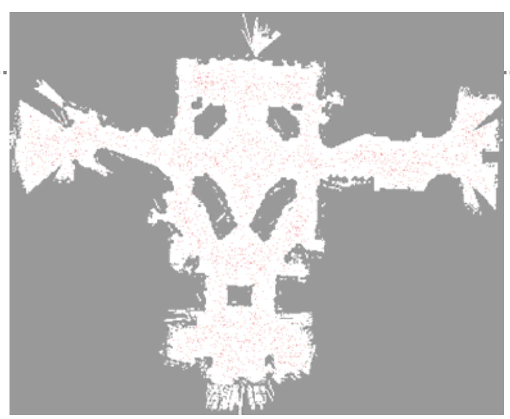


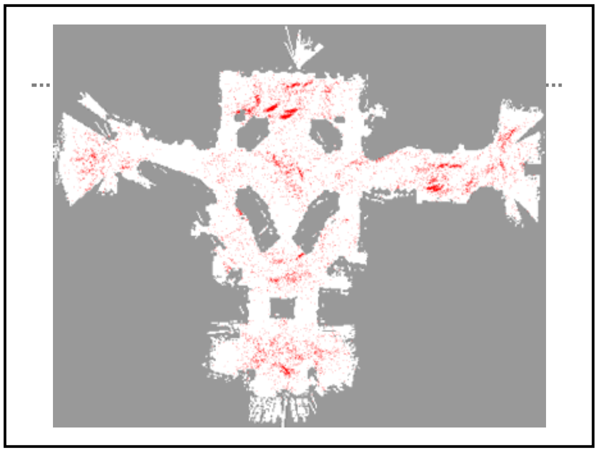
Sonar sensor

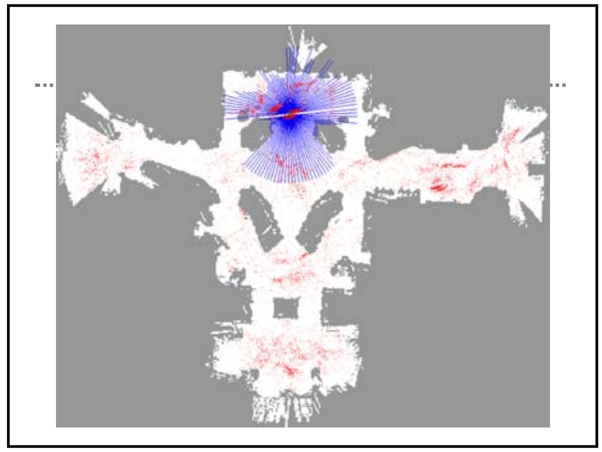


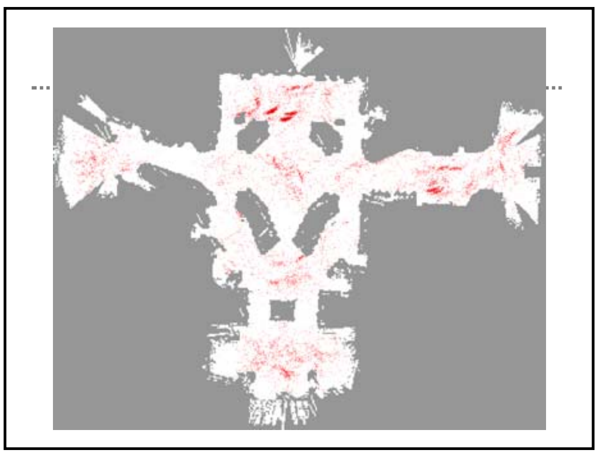


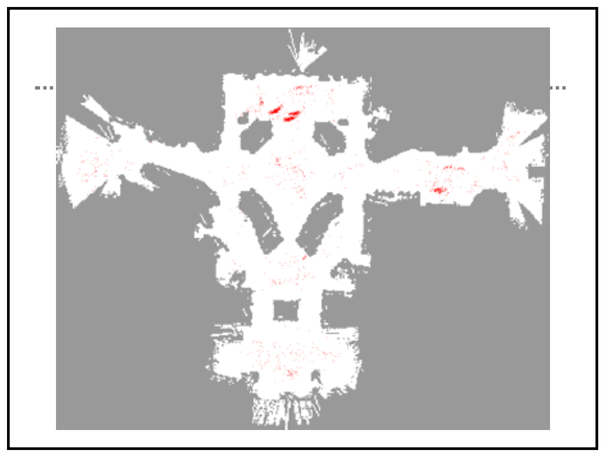


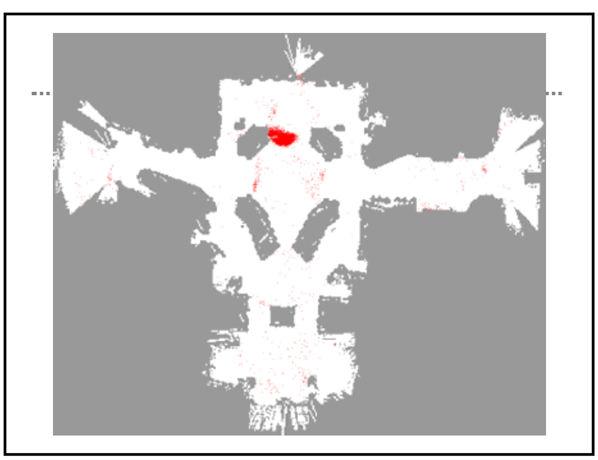


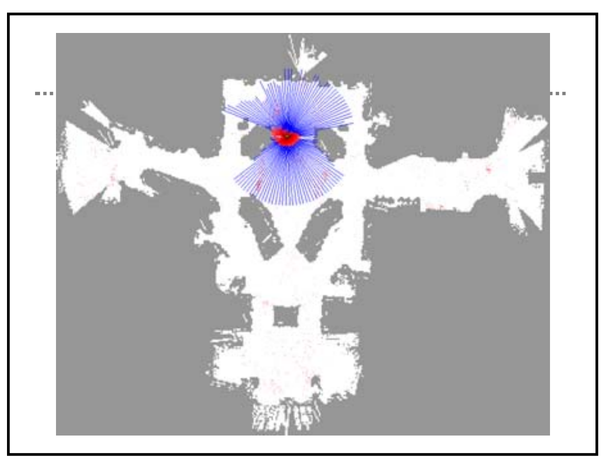


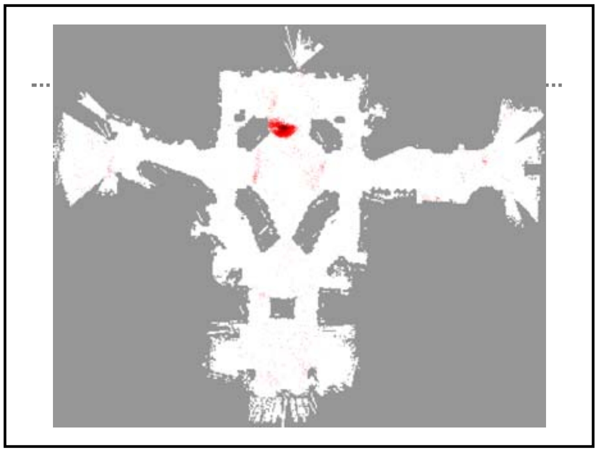


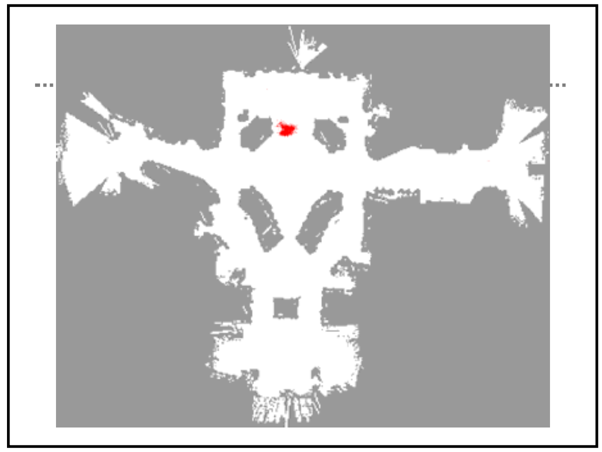


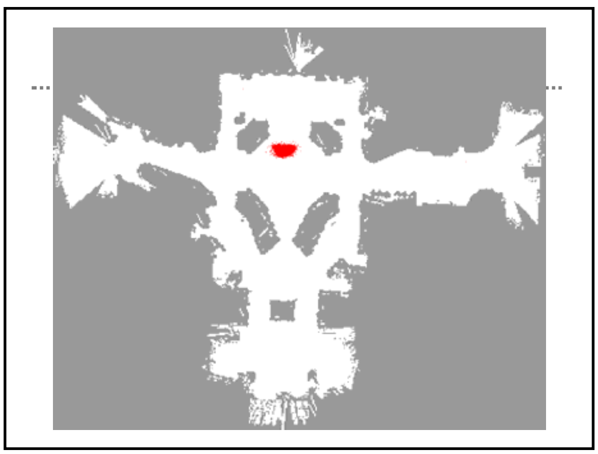


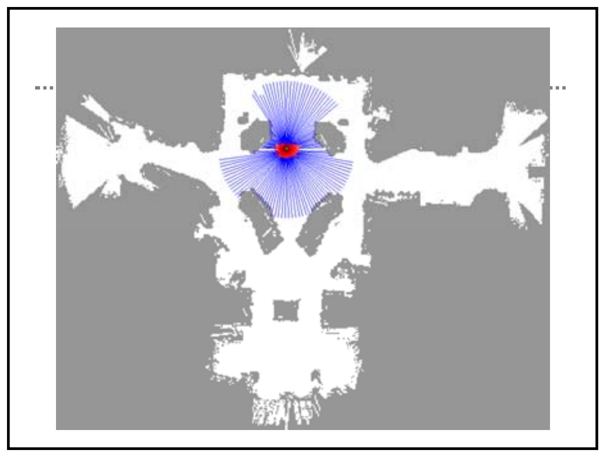


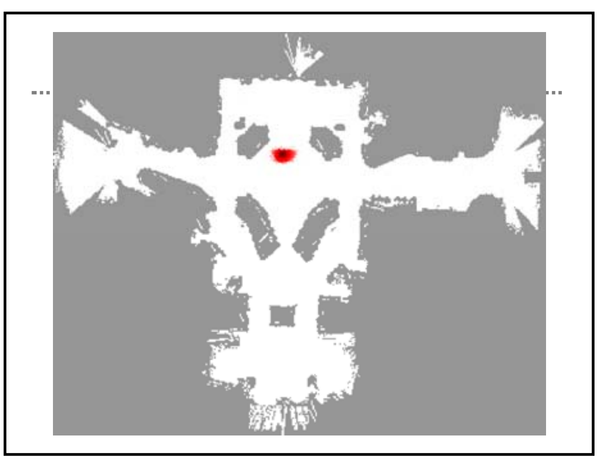


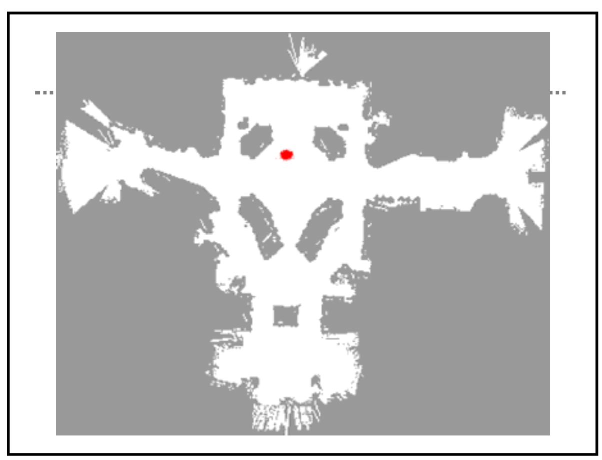


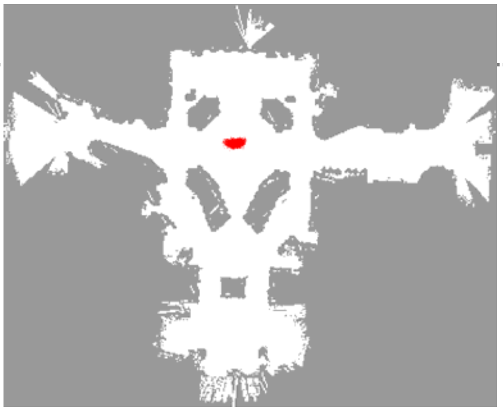


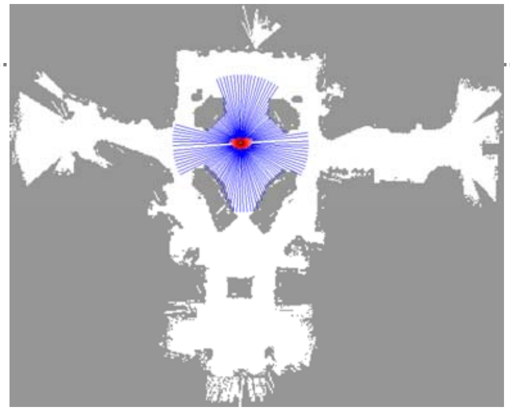


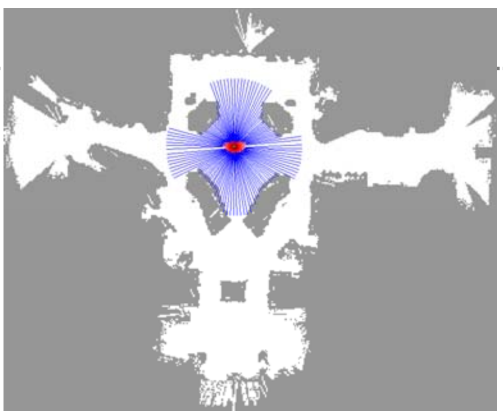












Robotic Cars

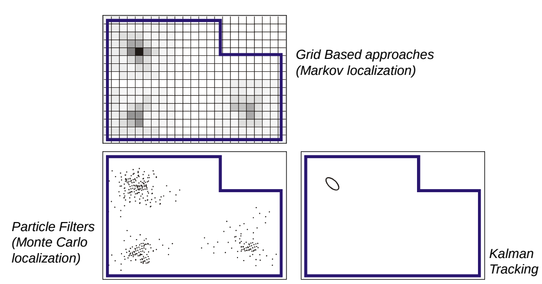
- DARPA Grand Challenge
- DARPA Urban Challenge
- http://www.youtube.com/watch?v=SQFEmR50HAk

SLAM

- SLAM = Simultaneous Localization And Mapping
  - We do not know the map or our location
  - Our belief state is over maps and positions!
  - Main techniques: Kalman filtering (Gaussian HMMs) and particle methods

DP-SLAM, Ron Parr

Example: State Representations for Robot Localization



36

Kalman Filters - Equations

Recursive filter for estimating state of linear dynamical system from noisy measurements

$$\left.\begin{array}{l} P(x_t|x_{t-1}) \approx N(Ax_{t-1}, \Gamma) \\ P(y_t|x_t) \approx N(Cx_t, \Sigma) \end{array}\right\}$$

A: State transition matrix (n x n)

C: Measurement matrix (m x n)

w: Process noise (є R^n),

v: Measurement noise(є R^m)

$$\left.\begin{array}{l} x_t = A_t x_{t-1} + w_t \\ y_t = C_t x_t + v_t \\ w_t \approx N(0, \Gamma) \\ v_t \approx N(0, \Sigma) \end{array}\right\}$$

Process dynamics (motion model)

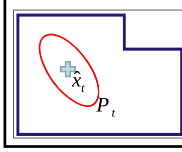
measurements (observation model)

Where : $N(x; m, V) = \frac{1}{|2\pi V|^{1/2}} \exp\left(-\frac{1}{2}(x-m)^T V^{-1}(x-m)\right)$

37

Kalman Filters - Update

$$\left.\begin{array}{l} x_t = A_t x_{t-1} + w_t \\ y_t = C_t x_t + v_t \\ w_t \approx N(0, \Gamma) \\ v_t \approx N(0, \Sigma) \end{array}\right\}$$



Predict state, covariance $\hat{x}_t^- = A\hat{x}_{t-1}$, $P_t^- = AP_{t-1}A^T + \Gamma$

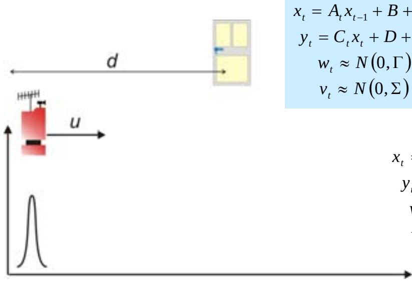
Compute Gain $K_t = P_t^- C^T (CP_t^- C^T + \Sigma)^{-1}$

Compute Innovation $J_t = \hat{y}_t - C\hat{x}_t^-$

Update $\hat{x}_t = \hat{x}_t - K_t J_t$, $P_t = (I - K_t C)P_k^-$

38

Kalman Filter - Example

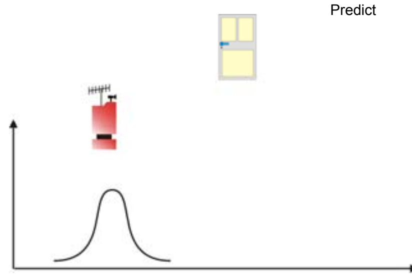


$$\begin{array}{ll} x_t = A_t x_{t-1} + B + w_t & A_t = [1] \\ y_t = C_t x_t + D + v_t & B_t = [u_t] \\ w_t \approx N(0, \Gamma) & C_t = [-1] \\ v_t \approx N(0, \Sigma) & D_t = [1] \end{array}$$

$$\left.\begin{array}{l} x_t = x_{t-1} + u_t + w_t \\ y_t = d - x_t + v_t \\ w_t \approx N(0, \Gamma) \\ v_t \approx N(0, \Sigma) \end{array}\right\}$$

39

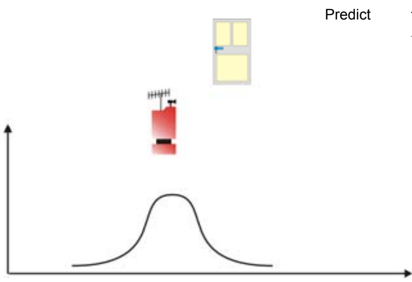
Kalman Filter - Example



Predict $\hat{x}_t^- = A\hat{x}_{t-1} + B$, $P_t^- = AP_{t-1}A^T + \Gamma$

40

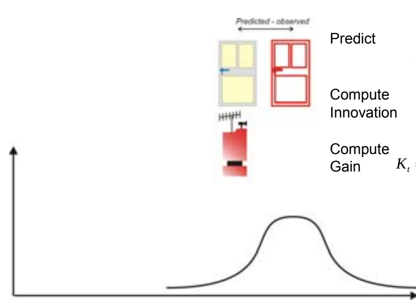
Kalman Filter - Example



Predict $\hat{x}_t^- = A\hat{x}_{t-1} + B$, $P_t^- = AP_{t-1}A^T + \Gamma$

41

Kalman Filter - Example



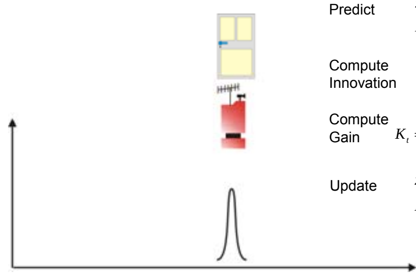
Predict $\hat{x}_t^- = A\hat{x}_{t-1} + B$, $P_t^- = AP_{t-1}A^T + \Gamma$

Compute Innovation $J_t = \hat{y}_t - C\hat{x}_t^-$

Compute Gain $K_t = P_t^- C^T (CP_t^- C^T + \Sigma)^{-1}$

42

Kalman Filter – Example



Predict $\hat{x}_t^- = A\hat{x}_{t-1} + B$

$P_t^- = AP_{t-1}A^T + \Gamma$

Compute Innovation $J_t = \hat{y}_t - C\hat{x}_t^-$

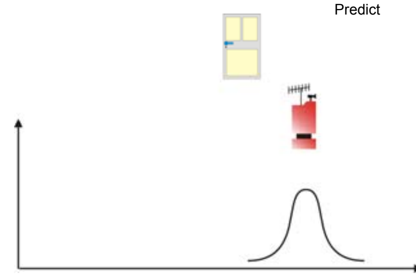
Compute Gain $K_t = P_t^- C^T (CP_t^- C^T + \Sigma)^{-1}$

Update $\hat{x}_t = \hat{x}_t - K_t J_t$

$P_t = (I - K_t C)P_k^-$

43

Kalman Filter – Example



Predict $\hat{x}_t^- = A\hat{x}_{t-1} + B$

$P_t^- = AP_{t-1}A^T + \Gamma$

44

Kalman Filter Applications

Apollo guidance computer

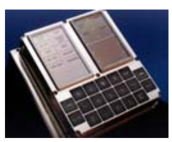
Cruise missiles

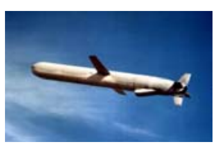
Airplane autopilot

Robotics

Finance

…





Continuous State Approaches

- Perform very accurately if the inputs are precise (performance is optimal with respect to any criterion in the linear case).
- Computational efficiency.

- Requirement that the initial state is known.
- Inability to recover from catastrophic failures
- Inability to track Multiple Hypotheses the state (Gaussians have only one mode)

46

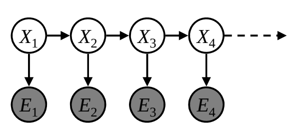
Discrete State Approaches

- Ability (to some degree) to operate even when its initial pose is unknown (start from uniform distribution).
- Ability to deal with noisy measurements.
- Ability to represent ambiguities (multi modal distributions).

- Computational time scales heavily with the number of possible states (dimensionality of the grid, number of samples, size of the map).
- Accuracy is limited by the size of the grid cells/number of particles-sampling method.
- Required number of particles is unknown

47

Best Explanation Queries



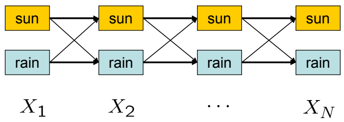
- Query: most likely seq:

$$\arg\max_{x_{1:t}} P(x_{1:t}|e_{1:t})$$

48

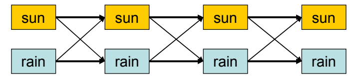
State Path Trellis

- State trellis: graph of states and transitions over time



- Each arc represents some transition $x_{t-1} \rightarrow x_t$
- Each arc has weight $P(x_t|x_{t-1})P(e_t|x_t)$
- Each path is a sequence of states
- The product of weights on a path is the seq's probability
- Can think of the Forward (and now Viterbi) algorithms as computing sums of all paths (best paths) in this graph

Viterbi Algorithm



$$x^*_{1:T} = \arg\max_{x_{1:T}} P(x_{1:T}|e_{1:T}) = \arg\max_{x_{1:T}} P(x_{1:T}, e_{1:T})$$

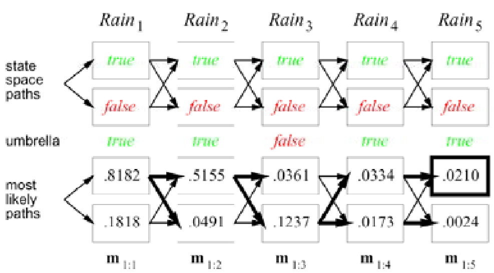
$$m_t[x_t] = \max_{x_{1:t-1}} P(x_{1:t-1}, x_t, e_{1:t})$$

$$= \max_{x_{1:t-1}} P(x_{1:t-1}, e_{1:t-1})P(x_t|x_{t-1})P(e_t|x_t)$$

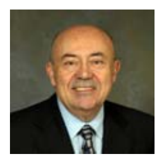
$$= P(e_t|x_t) \max_{x_{t-1}} P(x_t|x_{t-1}) \max_{x_{1:t-2}} P(x_{1:t-1}, e_{1:t-1})$$

$$= P(e_t|x_t) \max_{x_{t-1}} P(x_t|x_{t-1}) m_{t-1}[x_{t-1}]$$

Example



Andrew Viterbi



QUALCOMM