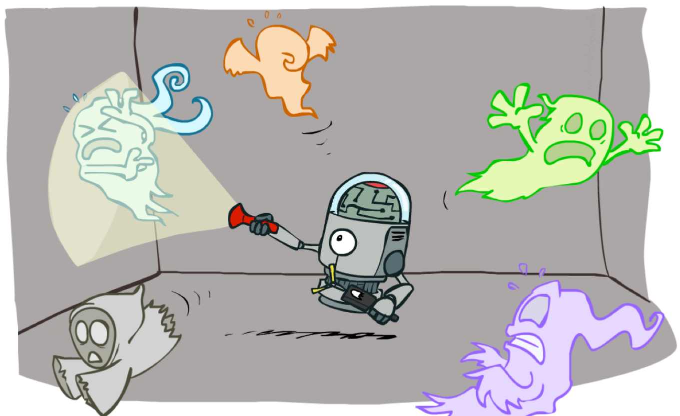
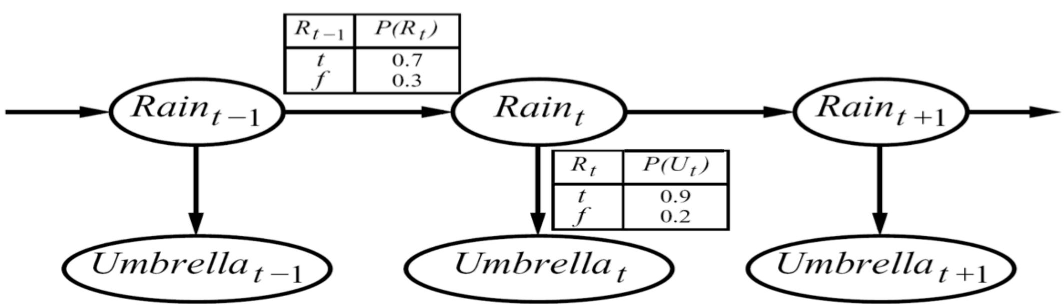
CSE 473: Artificial Intelligence

Particle Filters for HMMs



[Most slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.]

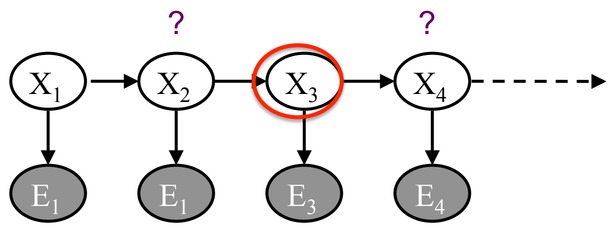
Example



- An HMM is defined by:
  - Initial distribution: $P(X_1)$
  - Transitions: $P(X_t|X_{t-1})$
  - Emissions: $P(E|X)$

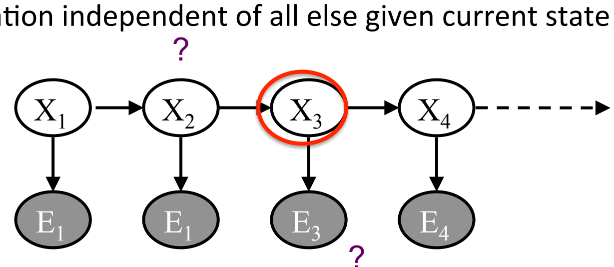
Conditional Independence

- HMMs have two important independence properties:
  - Markov hidden process, future depends on past via the present



Conditional Independence

- HMMs have two important independence properties:
  - Markov hidden process, future depends on past via the present
  - Current observation independent of all else given current state



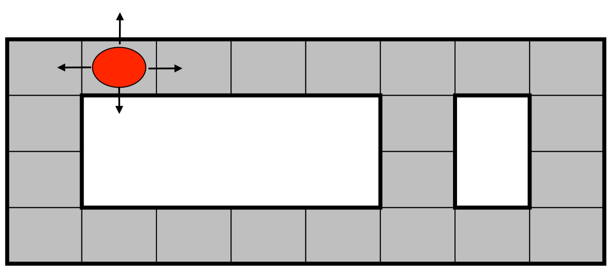
Filtering (aka Monitoring)

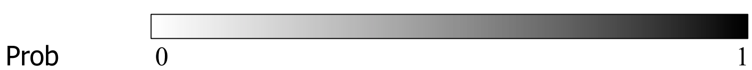
- **The task of tracking the agent's belief state, B(x), over time**
  - B(x) is a distribution over world states – repr agent knowledge
  - We start with B(X) in an initial setting, usually uniform
  - As time passes, or we get observations, we update B(X)

- **Many algorithms for this:**
  - Exact probabilistic inference
  - Particle filter approximation
  - Kalman filter (one method – Real valued values)
    - invented in the 60'for Apollo Program – real-valued state, Gaussian noise

Example: Robot Localization

Example from Michael Pfeiffer



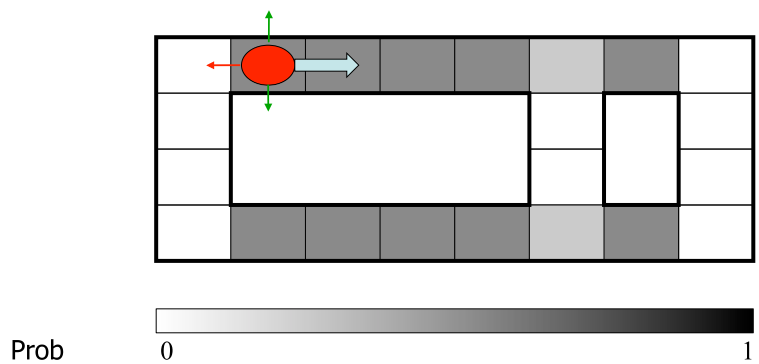


t=0

Sensor model: never more than 1 mistake

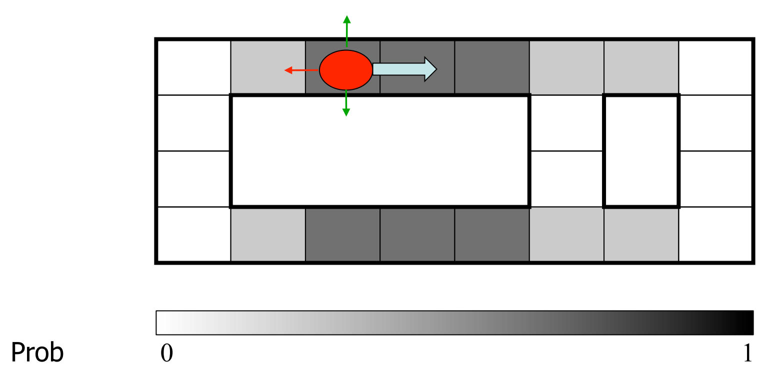
Motion model: may not execute action with small prob.

Example: Robot Localization



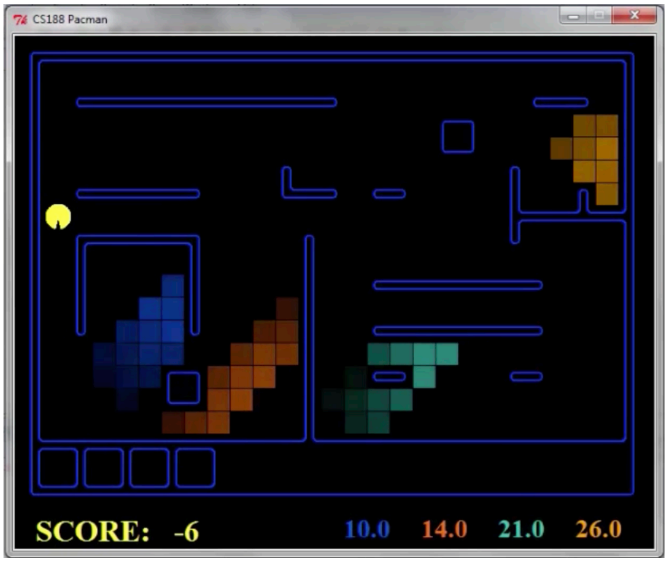
t=1

Example: Robot Localization



t=2

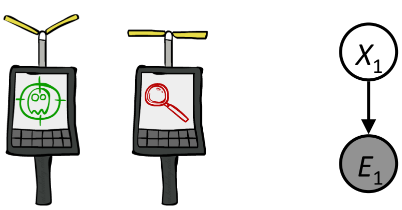
Pacman – Sonar (P4)



[Demo: Pacman – Sonar – No Beliefs(L14D1)]

Inference: Base Cases

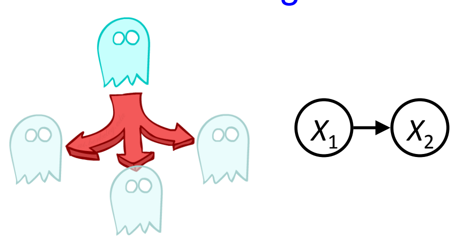
"Observation"



$$P(X_1|e_1)$$

$$P(x_1|e_1) = P(x_1, e_1)/P(e_1)$$
$$\propto_{X_1} P(x_1, e_1)$$
$$= P(x_1)P(e_1|x_1)$$

"Passage of Time"



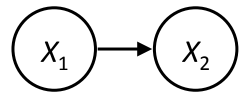
$$P(X_2)$$

$$P(x_2) = \sum_{x_1} P(x_1, x_2)$$
$$= \sum_{x_1} P(x_1)P(x_2|x_1)$$

Summary: Online Belief Updates

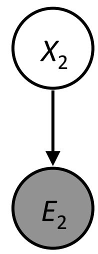
- Every time step, we start with current P(X | evidence)
- We update for time:

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



- We update for evidence:

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



- The forward algorithm does both at once (and doesn't normalize)

The Forward Algorithm

- We are given evidence at each time and want to know

$$B_t(X) = P(X_t|e_{1:t})$$

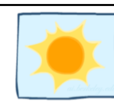
- We use the single (time-passage+observation) updates:

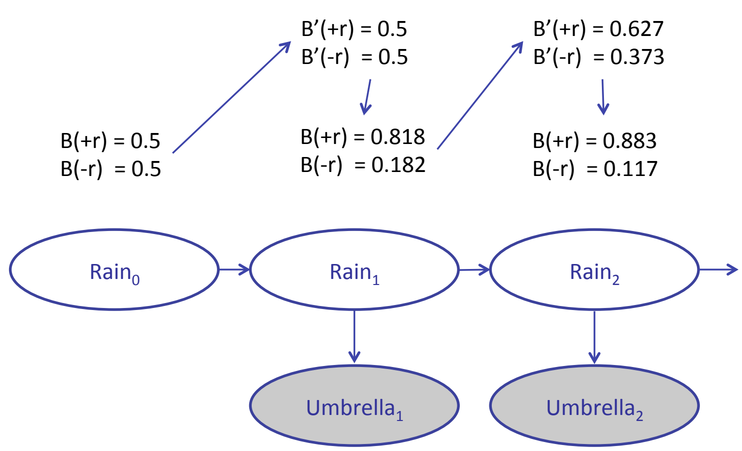
We can normalize as we go if we want to have P(x|e) at each time step, or just once at the end…

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

Video of Demo Pacman – Sonar (with beliefs)

Example: Weather HMM





R_t	R_{t+1}	$P(R_{t+1}\|R_t)$
+r	+r	0.7
+r	-r	0.3
-r	+r	0.3
-r	-r	0.7

R_t	U_t	$P(U_t\|R_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

Complexity of the Forward Algorithm?

- We are given evidence at each time and want to know

$$B_t(X) = P(X_t|e_{1:t})$$

- We use the single (time-passage+observation) updates:

$$P(x_t|e_{1:t}) \propto_X P(x_t, e_{1:t})$$

We can normalize as we go if we want to have P(x|e) at each time step, or just once at the end…

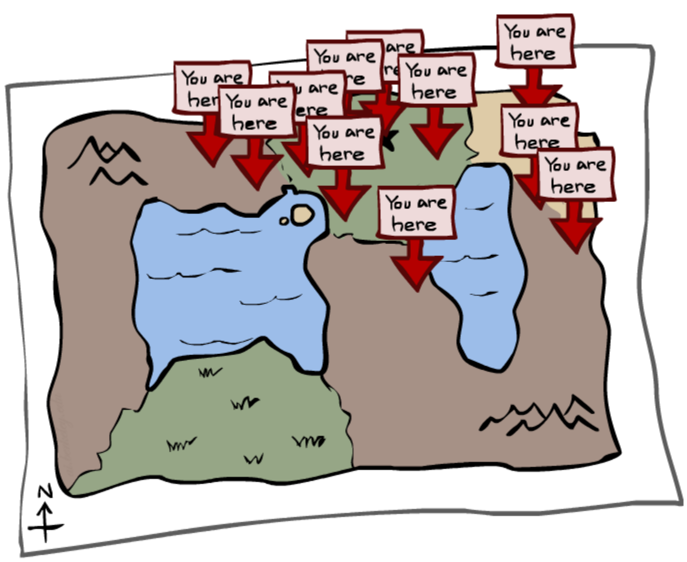
$$= \sum_{x_{t-1}} P(x_{t-1}, x_t, e_{1:t})$$

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

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

- Complexity? $O(|X|^2)$ time & O(X) space

Particle Filtering

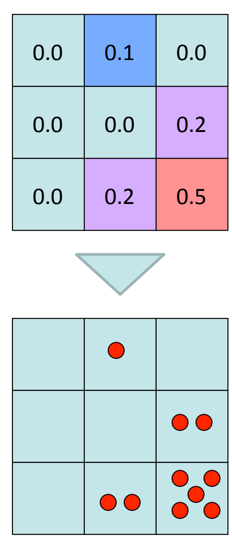


Particle Filtering Overview

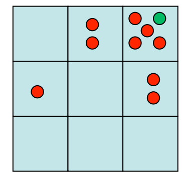
- Approximation technique to solve filtering problem
- Represents P distribution with samples
- Still operates in two steps
  - Elapse time
  - Incorporate observations

Particle Filtering

- Filtering: approximate solution
- Sometimes |X| is too big to use exact inference
  - |X| may be too big to even store B(X)
  - E.g. X is continuous
- 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
- Particle is just new name for sample



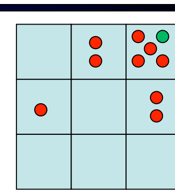
Representation: Particles



- 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 purpose
- P(x) approximated by **(number of particles with value x) / N**
  - More particles, more accuracy

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

Representation: Particles

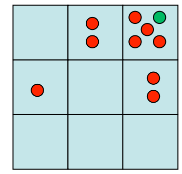


- Our representation of P(X) is now a list of N particles (samples)
  - Generally, N << |X|
  - Storing map from X to counts would defeat the purpose
- P(x) approximated by **(number of particles with value x) / N**
  - More particles, more accuracy
- What is P((3,3))? 5/10 = 50%

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

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 purpose



- P(x) approximated by **(number of particles with value x) / N**
  - More particles, more accuracy

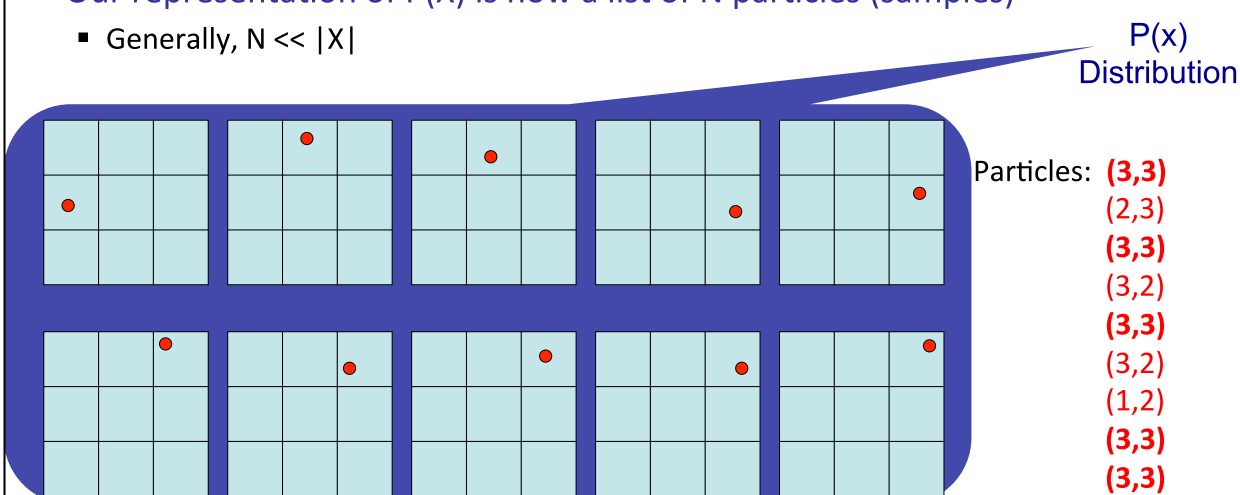
- What is P((2,2))? 0/10 = 0%

- In fact, many x may have P(x) = 0!

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

Particles: Better Illustration

- Our representation of P(X) is now a list of N particles (samples)
  - Generally, N << |X|



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

Particle Filtering: Elapse Time

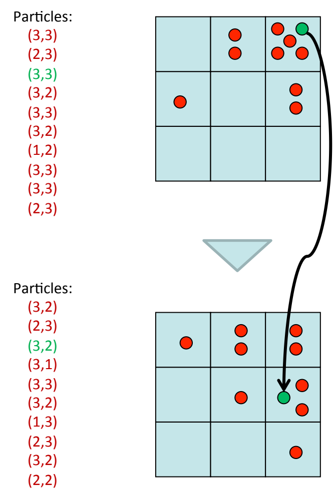
- Each particle is moved by sampling its next position from the transition model

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

  Aka: sample($P(x_{t+1} \mid x_t)$)

  - This is like ***prior sampling*** – samples' frequencies reflect the transition probabilities
  - Here, most samples move clockwise, but some move in another direction or stay in place

- This captures the passage of time
  - If enough samples, close to exact values before and after (consistent)



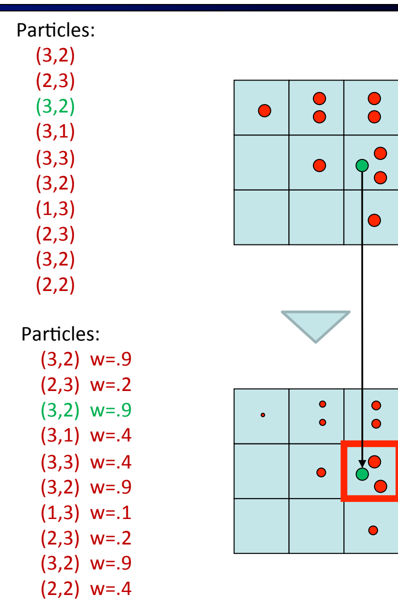
Particle Filtering: Observe

- Slightly trickier:
  - Don't sample observation, fix it
  - Similar to likelihood weighting, ***downweight samples based on the evidence***

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

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

  - As before, the probabilities don't sum to one, since all have been downweighted (in fact they now sum to (N times) an approximation of P(e))

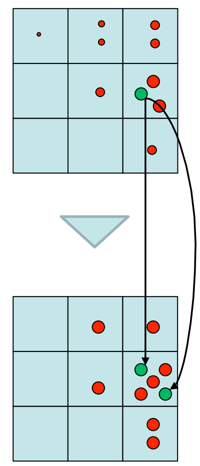


Particle Filtering Observe Part II: Resample

- 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

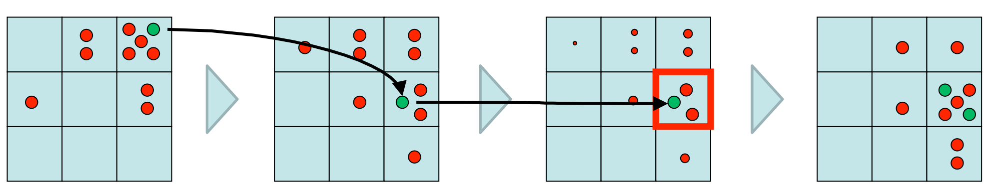
Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4

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



Recap: Particle Filtering

- Particles: track samples of states rather than an explicit distribution



Particles:	Particles:	Particles:	(New) Particles:
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(2,3)	(2,3)	(2,3) w=.2	(2,2)
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(3,2)	(3,1)	(3,1) w=.4	(2,3)
(3,3)	(3,3)	(3,3) w=.4	(3,3)
(3,2)	(3,2)	(3,2) w=.9	(3,2)
(1,2)	(1,3)	(1,3) w=.1	(1,3)
(3,3)	(2,3)	(2,3) w=.2	(2,3)
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(2,3)	(2,2)	(2,2) w=.4	(3,2)

[Demos: ghostbusters particle filtering (L15D3,4,5)]

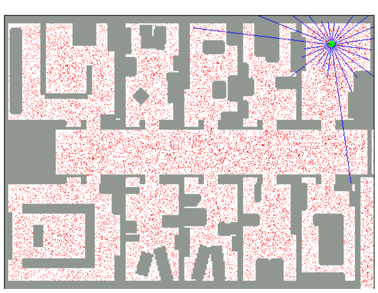
Video of Demo – Moderate Number of Particles

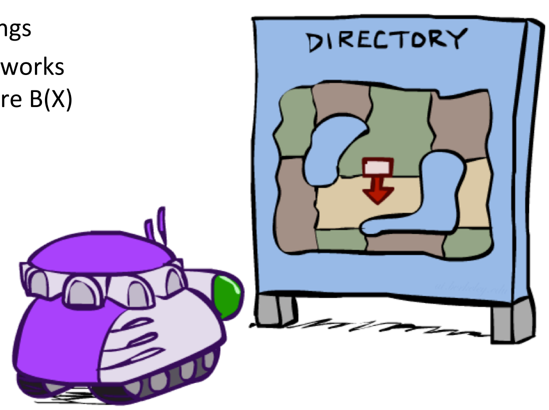
Video of Demo – One Particle

Video of Demo – Huge Number of Particles

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

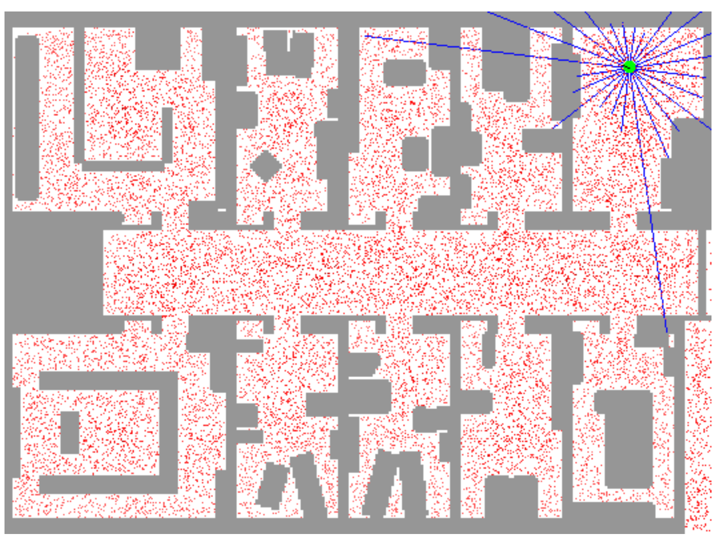




Particle Filter Localization (Sonar)

[Video: global-sonar-uw-annotated.avi]

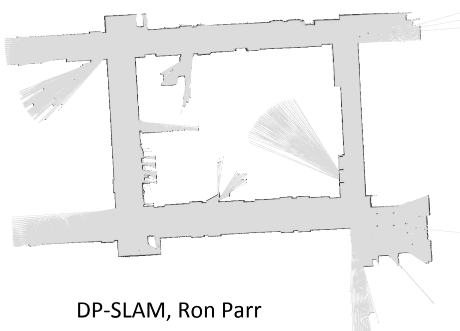
Particle Filter Localization (Laser)



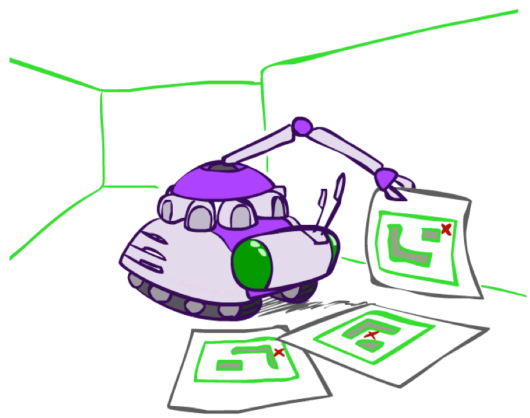
[Video: global-floor.gif]

Robot Mapping

- SLAM: Simultaneous Localization And Mapping
  - We do not know the map or our location
  - State consists of position AND map!
  - Main techniques: Kalman filtering (Gaussian HMMs) and particle methods

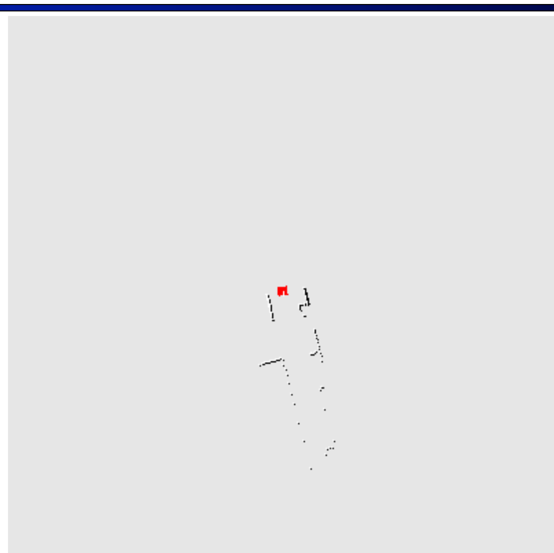


DP-SLAM, Ron Parr



[Demo: PARTICLES-SLAM-mapping1-new.avi]

Particle Filter SLAM – Video 1



[Demo: PARTICLES-SLAM-mapping1-new.avi]

Particle Filter SLAM – Video 2

[Demo: PARTICLES-SLAM-fastslam.avi]