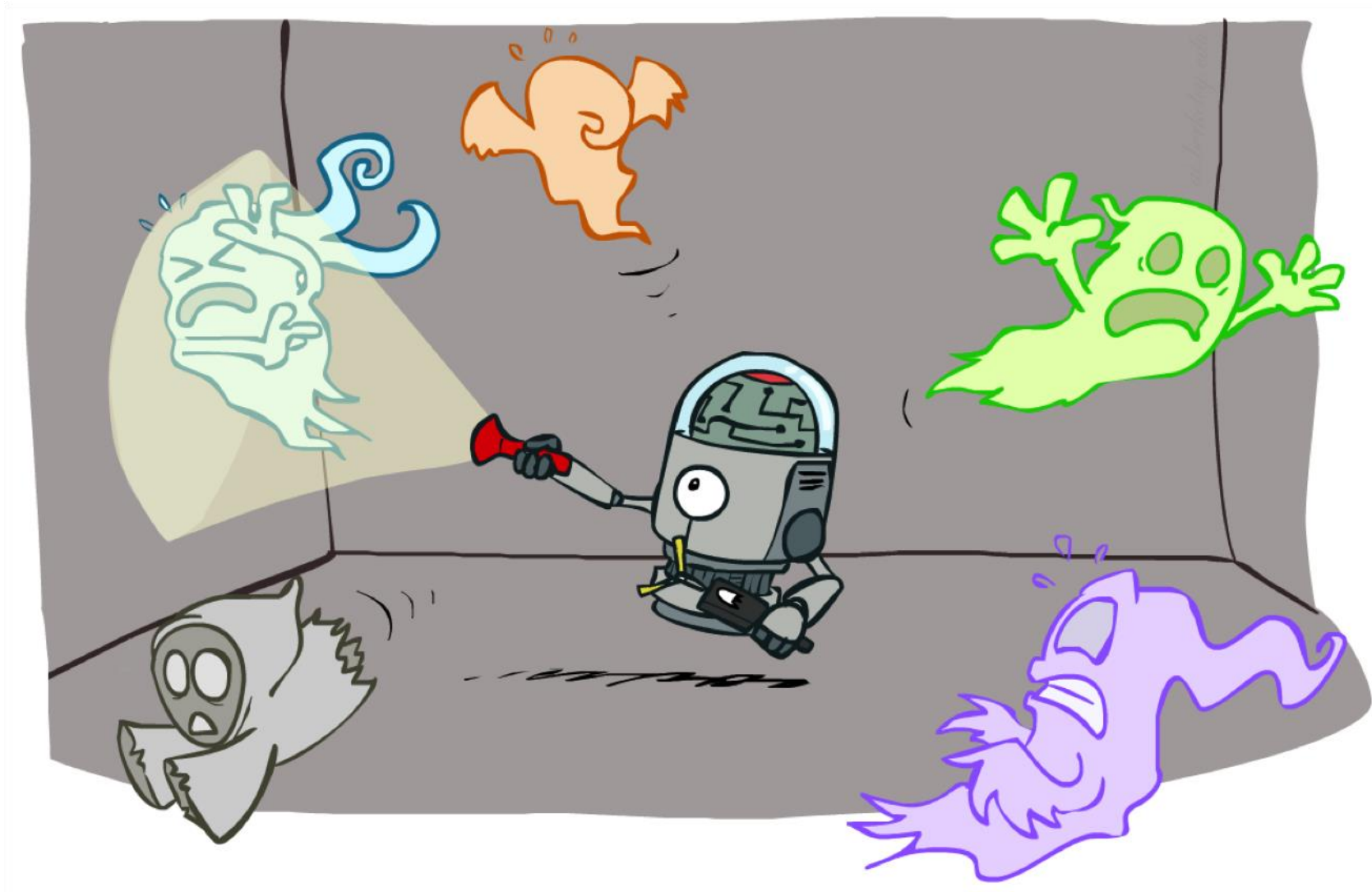


Particle Filters

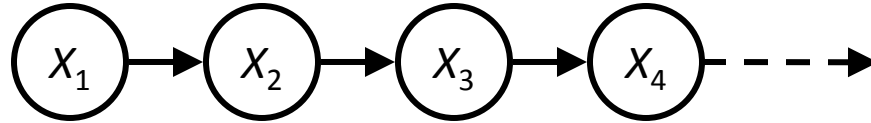


Today

- HMMs
 - Particle filters
 - Demo bonanza!
 - Most-likely-explanation queries

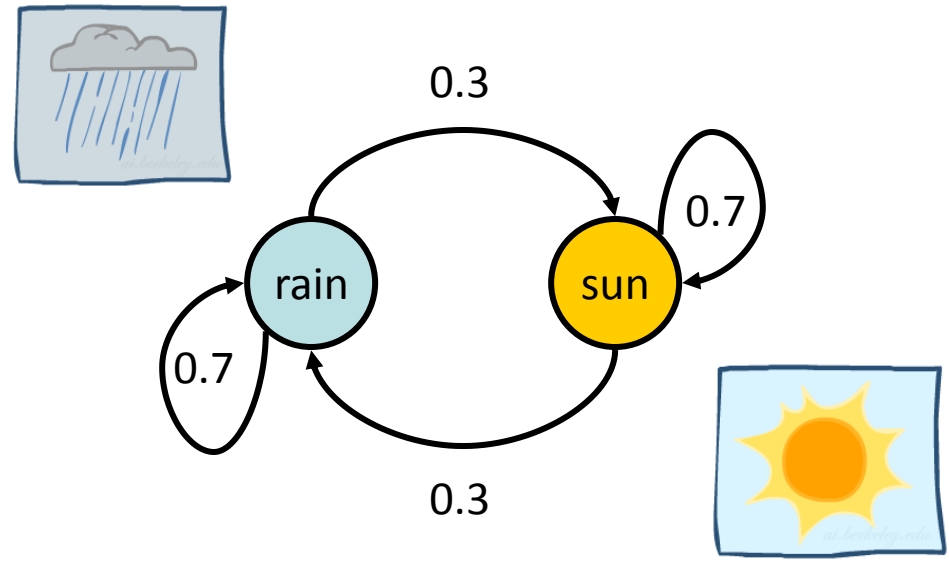
Recap: Reasoning Over Time

- Markov models



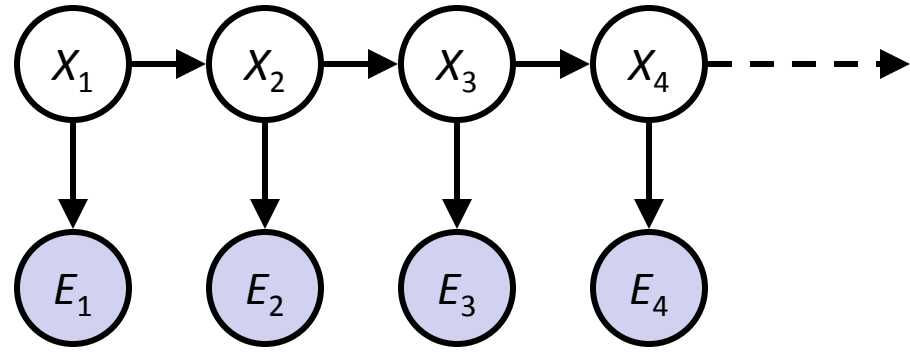
$P(X_1)$

$P(X|X_{-1})$



$P(E|X)$

- Hidden Markov models



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

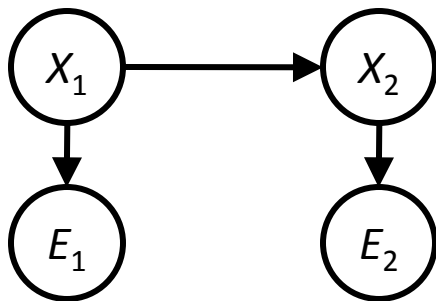
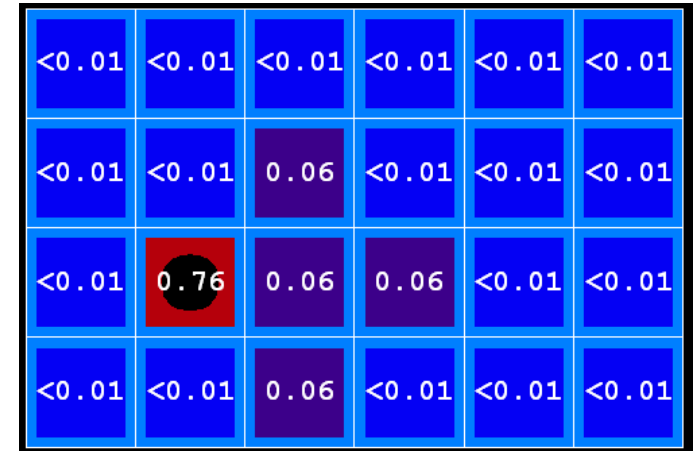
Recap: Filtering

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: $\langle P(\text{rain}), P(\text{sun}) \rangle$

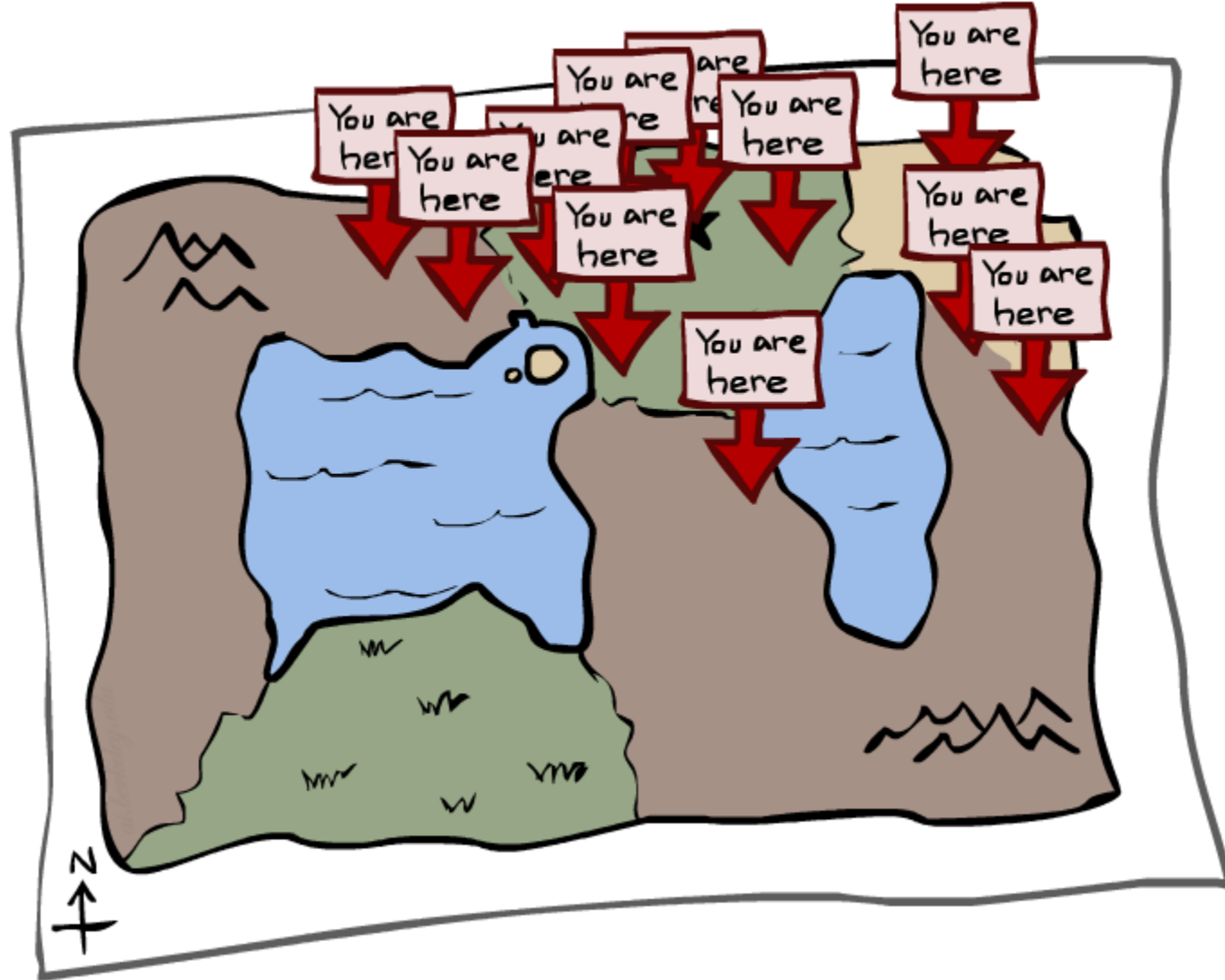
$P(X_1)$ $\langle 0.5, 0.5 \rangle$ *Prior on X_1*

$P(X_1 | E_1 = \text{umbrella})$ $\langle 0.82, 0.18 \rangle$ *Observe*

$P(X_2 | E_1 = \text{umbrella})$ $\langle 0.63, 0.37 \rangle$ *Elapse time*

$P(X_2 | E_1 = \text{umb}, E_2 = \text{umb})$ $\langle 0.88, 0.12 \rangle$ *Observe*

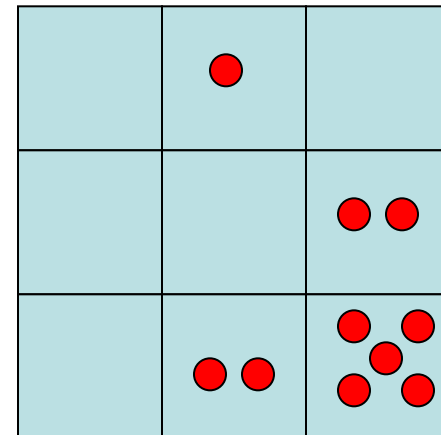
Particle Filtering



Particle Filtering

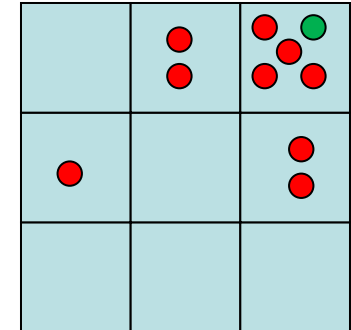
- Filtering: approximate solution
- Sometimes $|X|$ is too big to use exact inference
 - $|X|$ may be too big to even store $B(X)$
 - E.g. X is continuous
- Solution: approximate inference
 - Track samples of X , not all values
 - Samples are called particles
 - Time per step is linear in the number 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

0.0	0.1	0.0
0.0	0.0	0.2
0.0	0.2	0.5



Representation: Particles

- Our representation of $P(X)$ is now a list of N particles (samples)
 - Generally, $N \ll |X|$
 - Storing map from X to counts would defeat the point
- $P(x)$ approximated by number of particles with value x
 - So, many x may 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)
(1,2)
(3,3)
(3,3)
(2,3)

Particle Filtering: Elapse Time

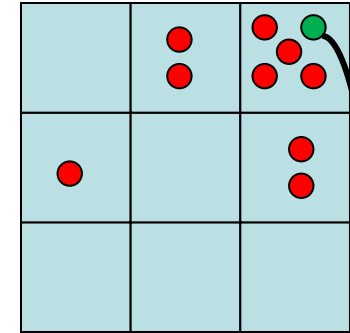
- Each particle is moved by sampling its next position from the transition model

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

- 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)

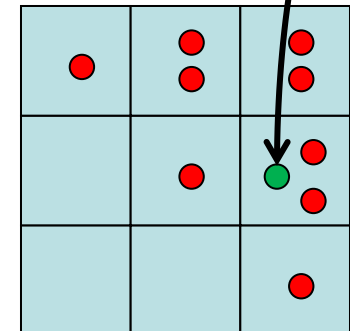
Particles:

(3,3)
(2,3)
(3,3)
(3,2)
(3,3)
(3,2)
(1,2)
(3,3)
(3,3)
(2,3)



Particles:

(3,2)
(2,3)
(3,2)
(3,1)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(2,2)



Particle Filtering: Observe

- Slightly trickier:

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

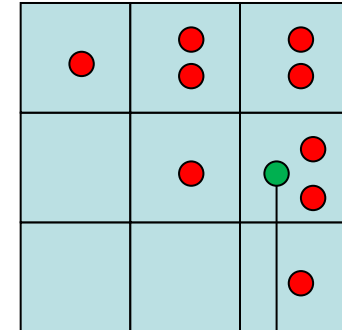
$$w(x) = P(e|x)$$

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

- 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 $P(e)$)

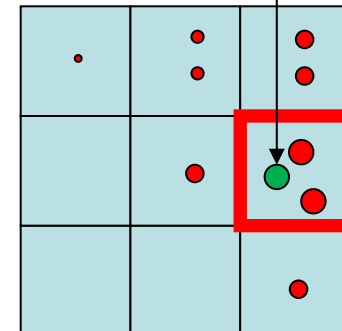
Particles:

(3,2)
(2,3)
(3,2)
(3,1)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(2,2)



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



Particle Filtering: Resample

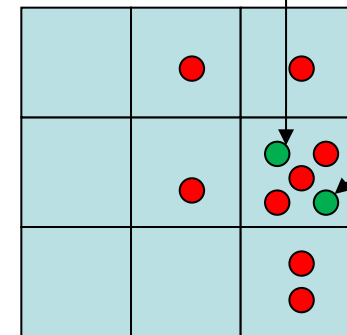
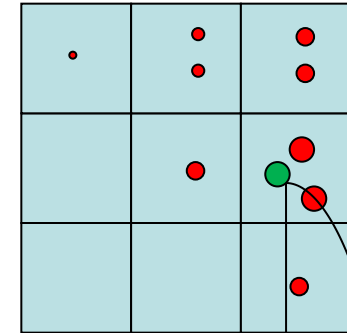
- Rather than tracking weighted samples, we resample
- N times, we choose from our weighted sample distribution (i.e. draw with replacement)
- This is equivalent to renormalizing the distribution
- Now the update is complete for this time step, continue with the next one

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

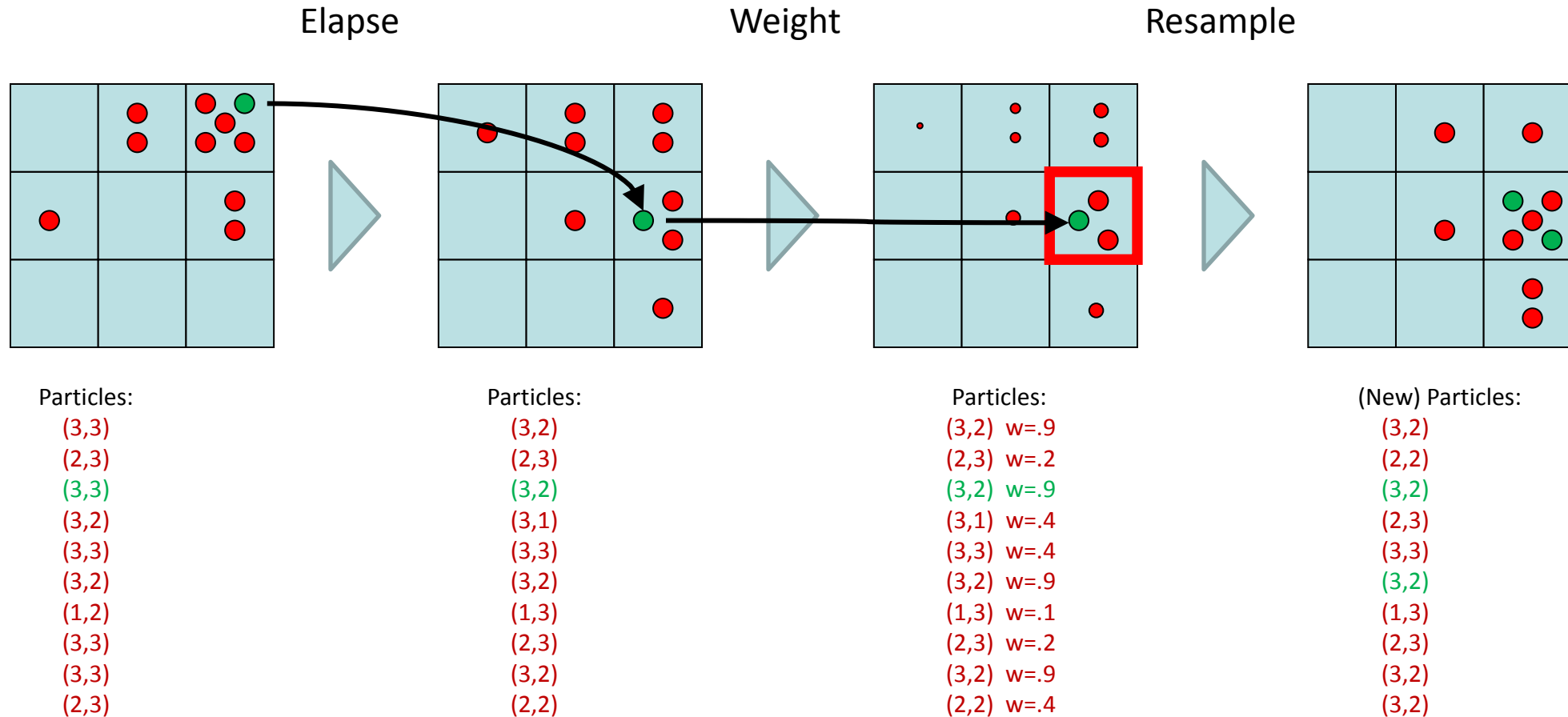
(New) Particles:

(3,2)
(2,2)
(3,2)
(2,3)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(3,2)



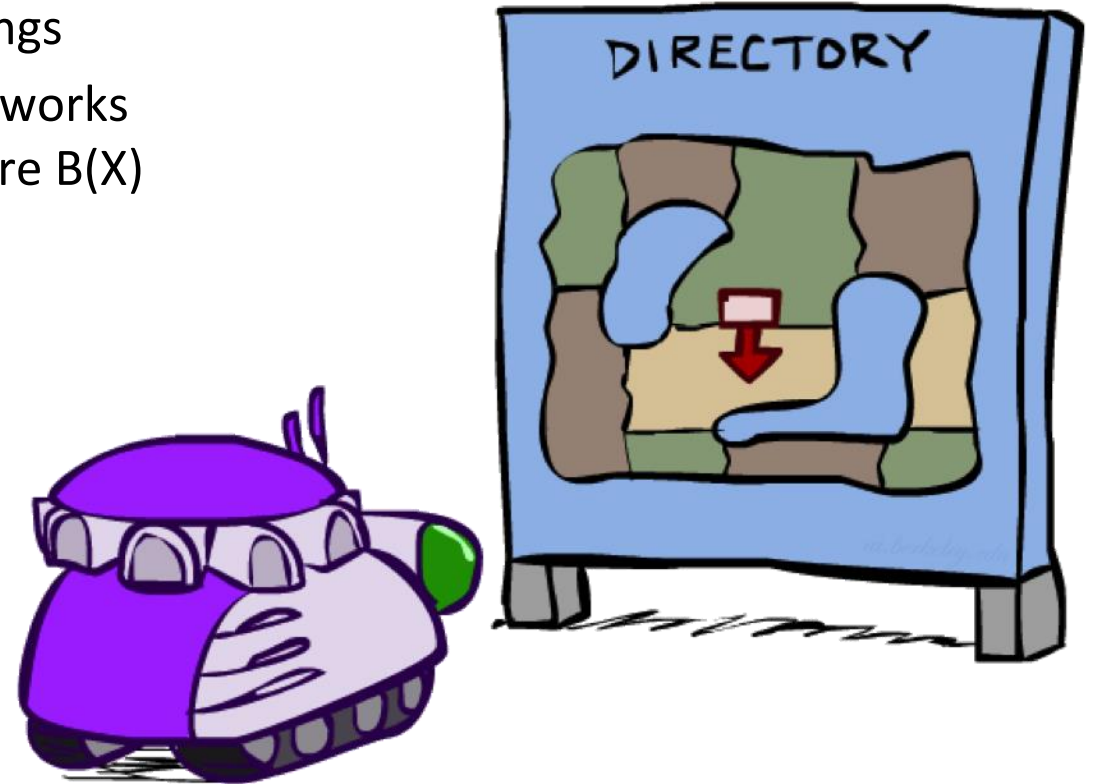
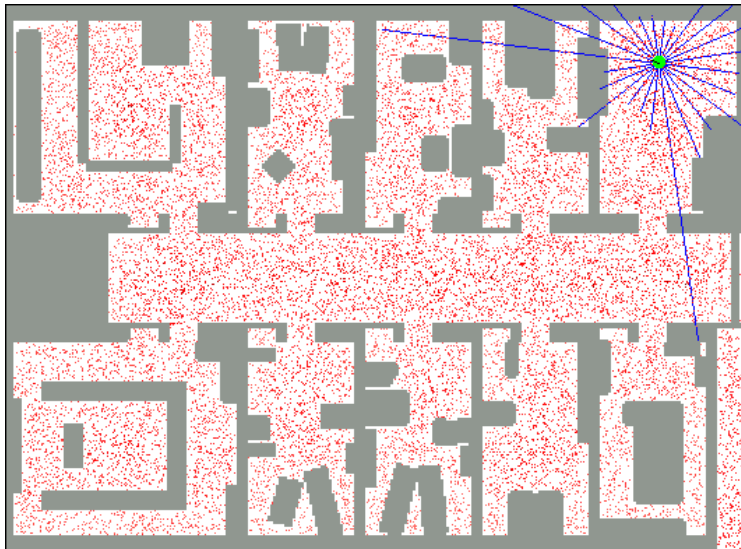
Recap: Particle Filtering

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



Robot Localization

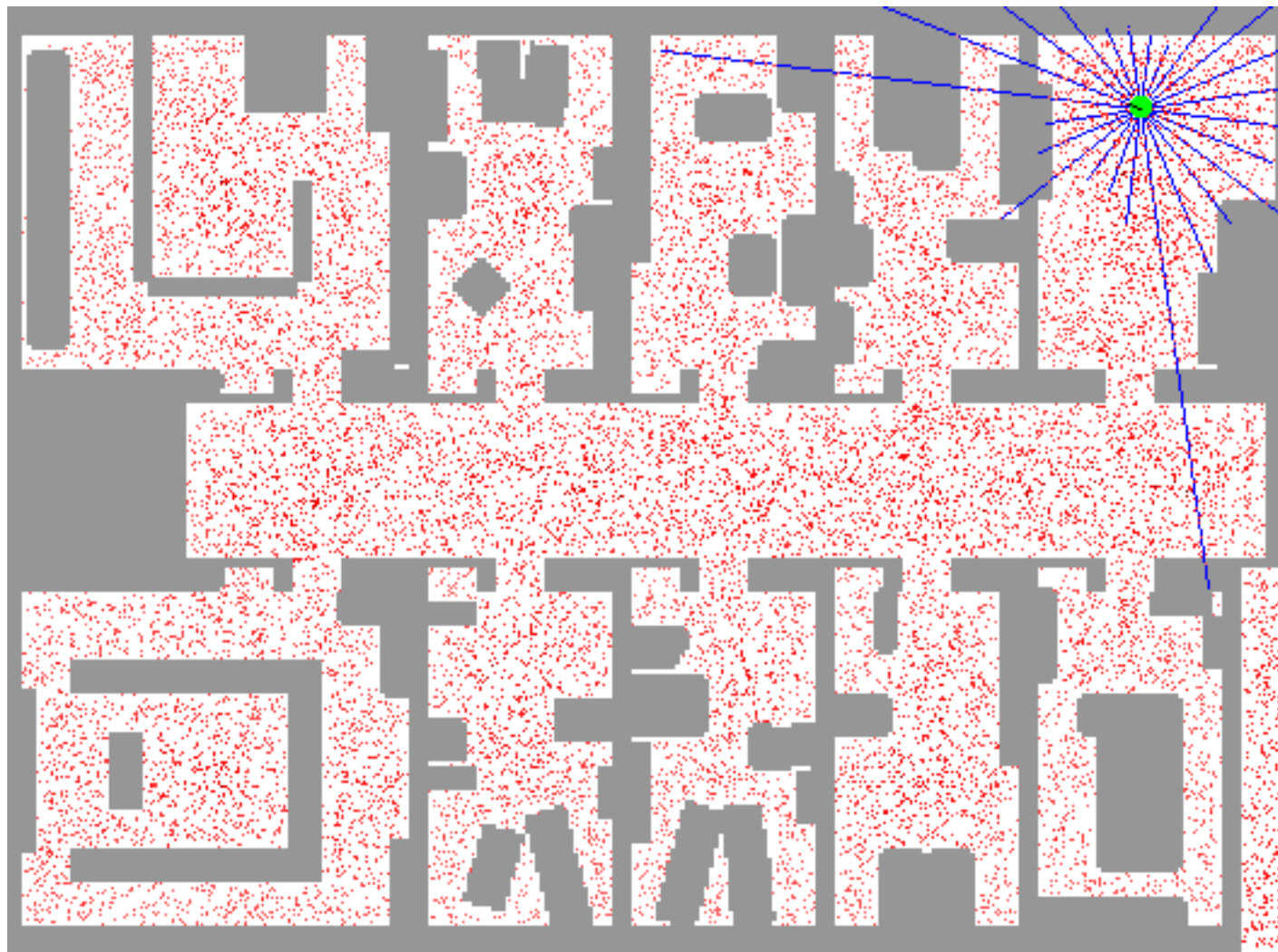
- In robot localization:
 - We know the map, but not the robot's position
 - Observations may be vectors of range finder readings
 - State space and readings are typically continuous (works basically like a very fine grid) and so we cannot store $B(X)$
 - Particle filtering is a main technique



Particle Filter Localization (Sonar)

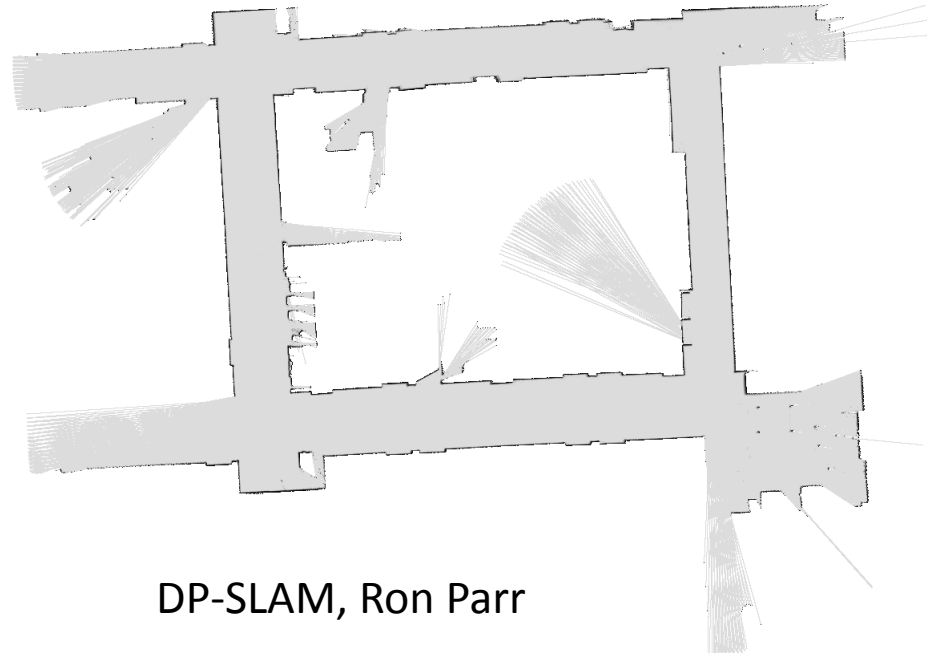


Particle Filter Localization (Laser)

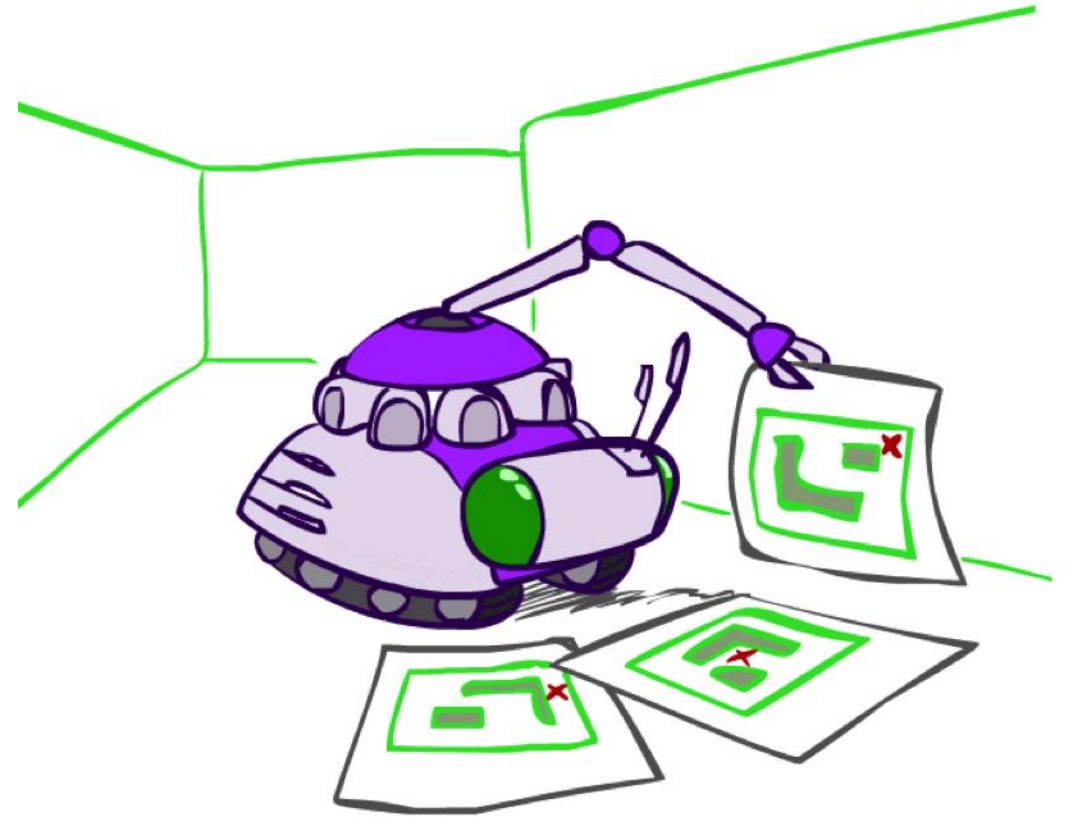


Robot Mapping

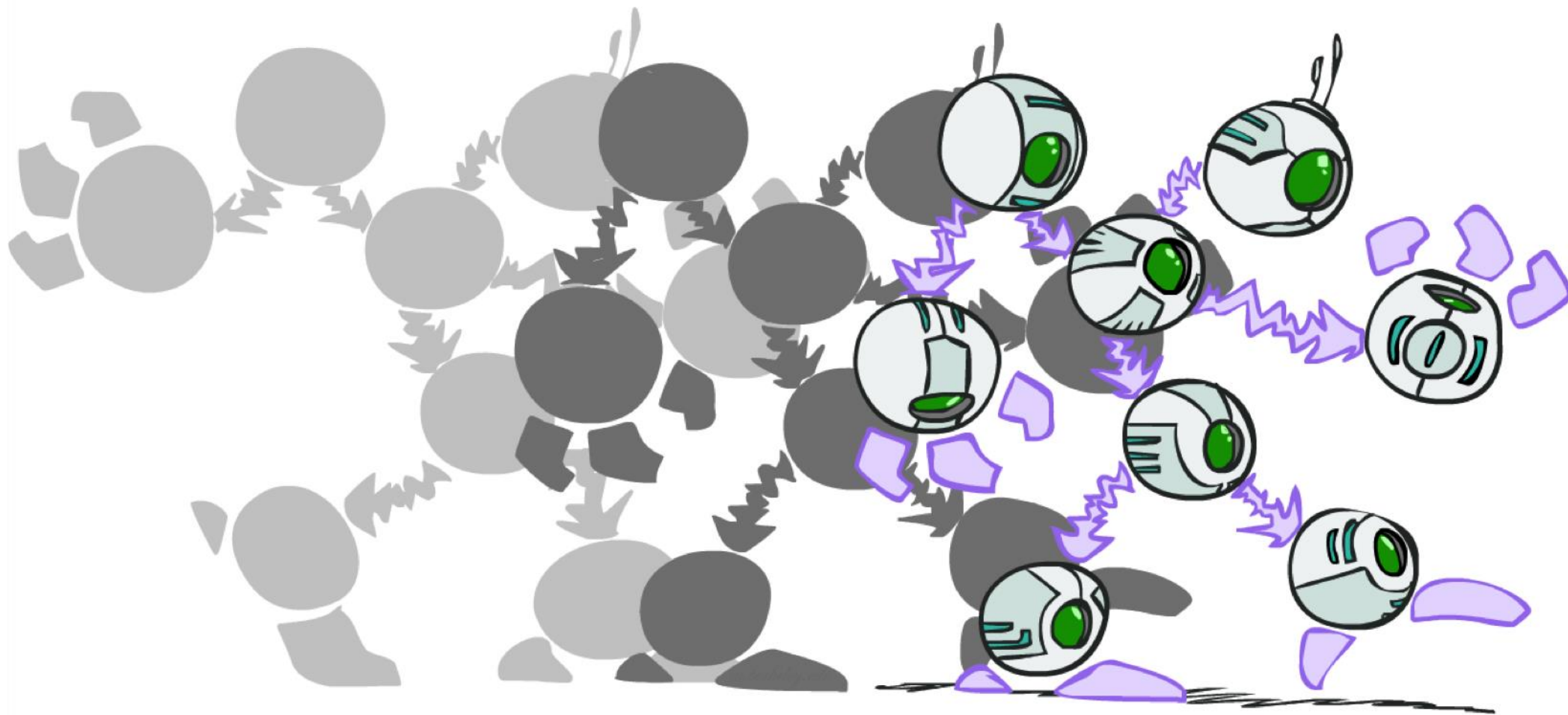
- SLAM: Simultaneous Localization And Mapping
 - We do not know the map or our location
 - State consists of position AND map!
 - Main techniques: Kalman filtering (Gaussian HMMs) and particle methods



DP-SLAM, Ron Parr

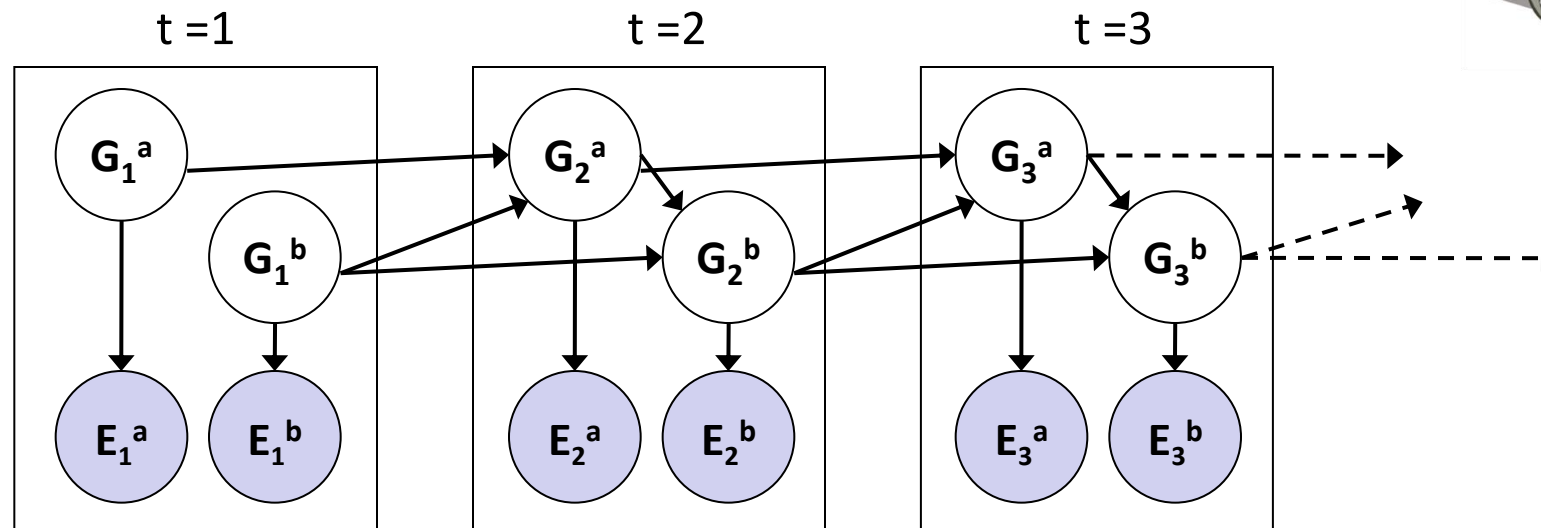
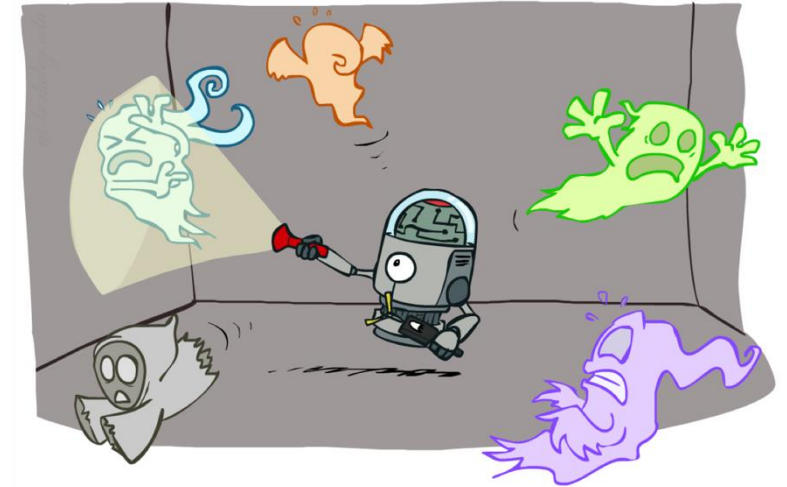


Dynamic Bayes Nets



Dynamic Bayes Nets (DBNs)

- We want to track multiple variables over time, using multiple sources of evidence
- Idea: Repeat a fixed Bayes net structure at each time
- Variables from time t can condition on those from $t-1$



- Dynamic Bayes nets are a generalization of HMMs

DBN Particle Filters

- A particle is a complete sample for a time step
- **Initialize:** Generate prior samples for the $t=1$ Bayes net
 - Example particle: $\mathbf{G}_1^a = (3,3)$ $\mathbf{G}_1^b = (5,3)$
- **Elapse time:** Sample a successor for each particle
 - Example successor: $\mathbf{G}_2^a = (2,3)$ $\mathbf{G}_2^b = (6,3)$
- **Observe:** Weight each entire sample by the likelihood of the evidence conditioned on the sample
 - Likelihood: $P(\mathbf{E}_1^a | \mathbf{G}_1^a) * P(\mathbf{E}_1^b | \mathbf{G}_1^b)$
- **Resample:** Select prior samples (tuples of values) in proportion to their likelihood

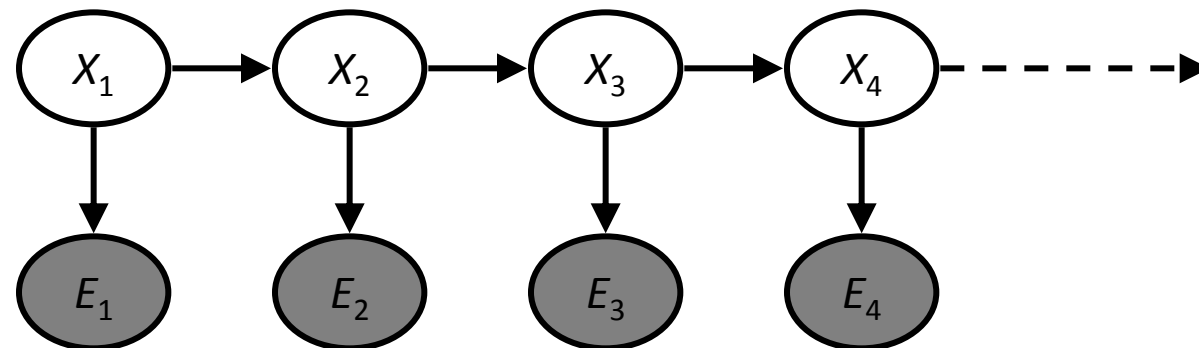
Most Likely Explanation



HMMs: MLE Queries

- HMMs defined by

- States X
- Observations E
- Initial distribution: $P(X_1)$
- Transitions: $P(X|X_{-1})$
- Emissions: $P(E|X)$



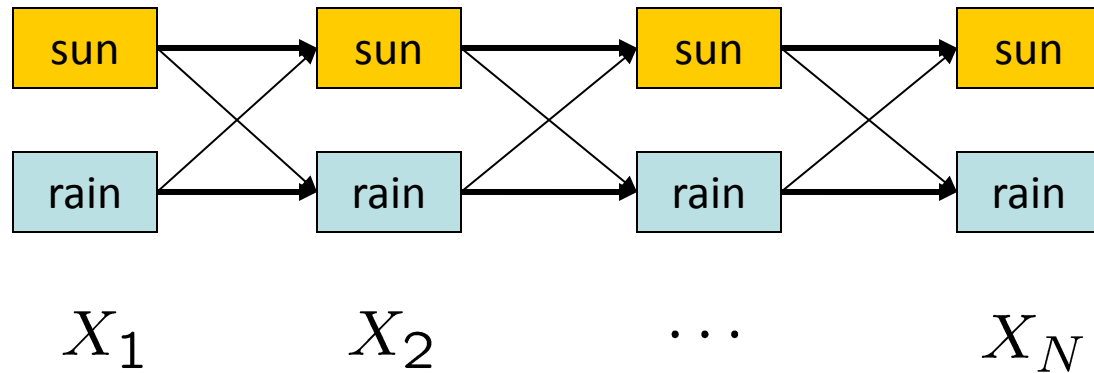
- New query: most likely explanation:

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

- New method: the Viterbi algorithm

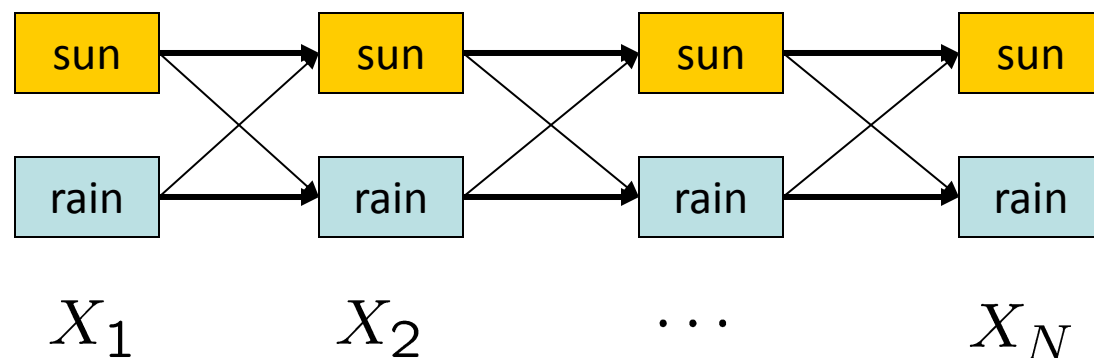
State Trellis

- 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 that sequence's probability along with the evidence
- Forward algorithm computes sums of paths, Viterbi computes best paths

Forward / Viterbi Algorithms



Forward Algorithm (Sum)

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

Viterbi Algorithm (Max)

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