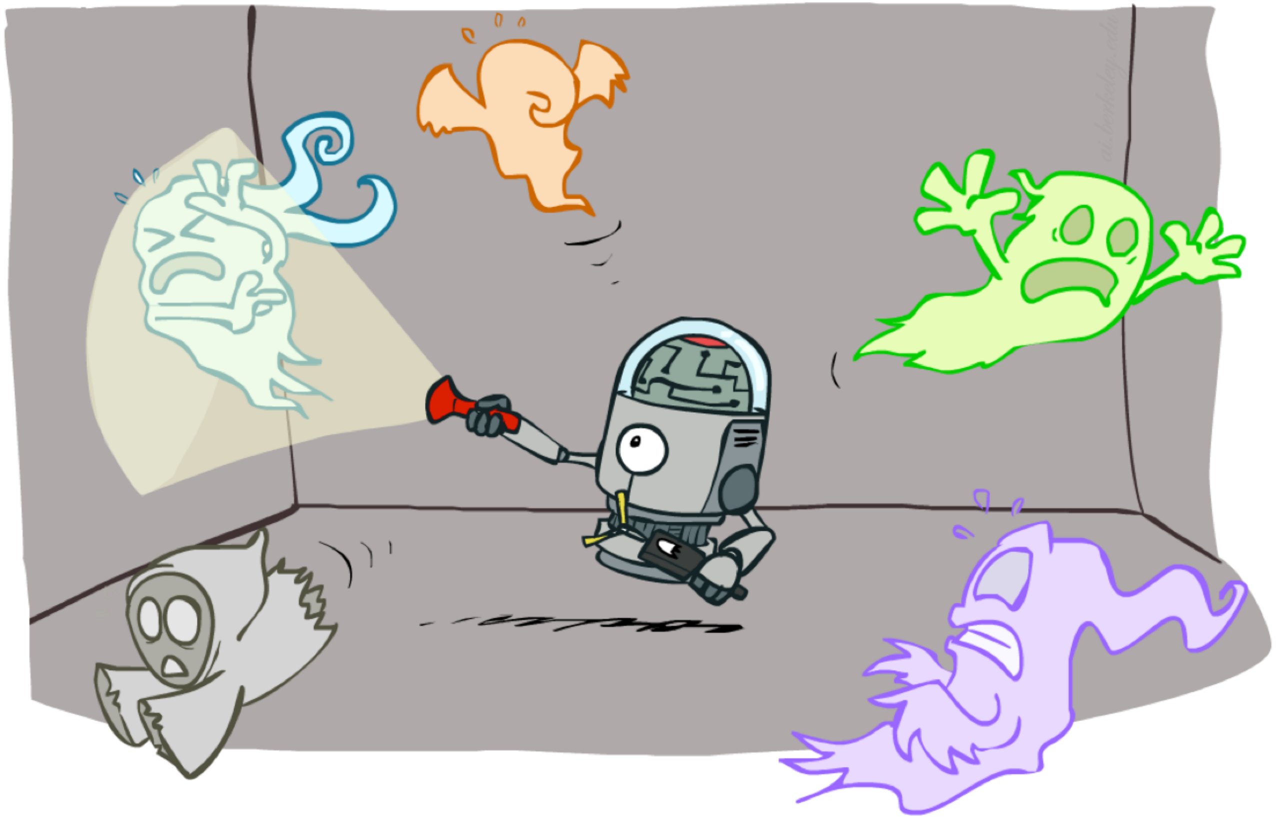
CS 3317: Artificial Intelligence

Particle Filters



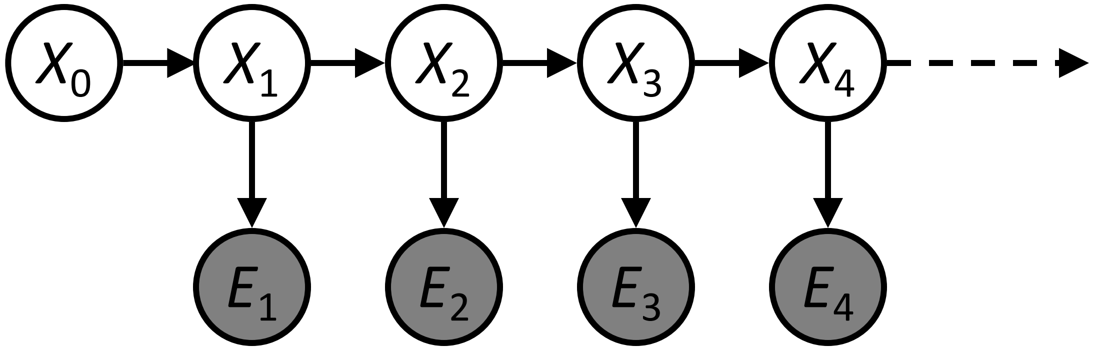
Instructor: Panpan Cai

[Slides adapted from UC Berkeley CS188]

Hidden Markov Models

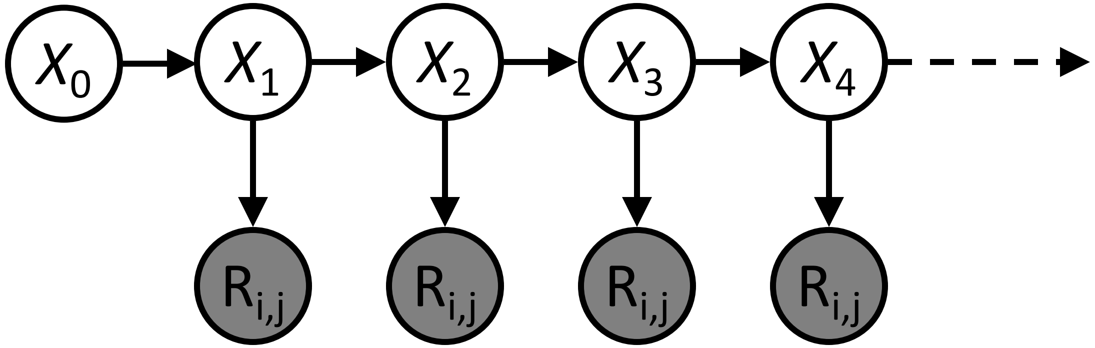
- Markov chains not so useful for most agents
  - Real-problems are often partially observable
  - Use observations to update your beliefs

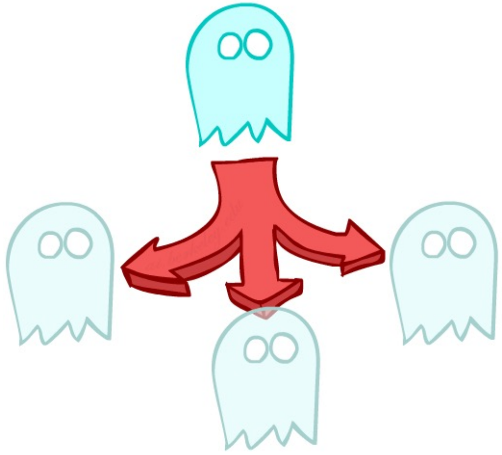
- Hidden Markov models (HMMs)
  - Underlying Markov chain over states X_i
  - You observe evidences at each time step

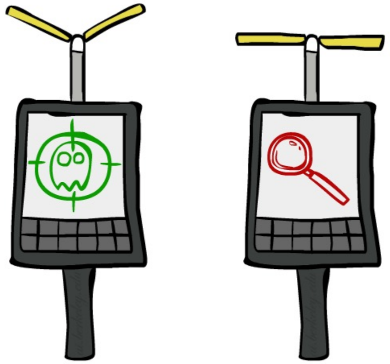


Example: Ghostbusters HMM

- $P(X_0)$ = uniform
- $P(X'|X)$ = usually move clockwise, but sometimes move in a random direction or stay in place
- $P(R_{ij}|X')$ = same sensor model as before: red means close, green means far away.







1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

$P(X_0)$

1/6	1/6	1/2
0	1/6	0
0	0	0

$P(X'|X=<1,2>)$

Filtering / Belief Tracking

- Filtering is the task of tracking the *belief state*

  $B_t(X) = P(X_t \mid e_1, \ldots, e_t)$

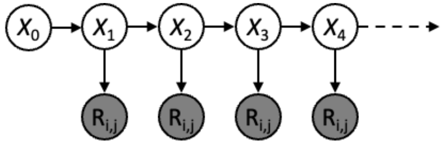
  over time

- We start with $B_0(X)$ in an initial setting, usually *uniform*

- As time passes, or when we get observations, we update $B(X)$

Ghostbusters Basic Dynamics

Ghostbusters – Circular Dynamics -- HMM



Ghostbusters Circular Dynamics (No-observation)

Ghostbusters Whirlpool Dynamics

Recursive Filtering

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

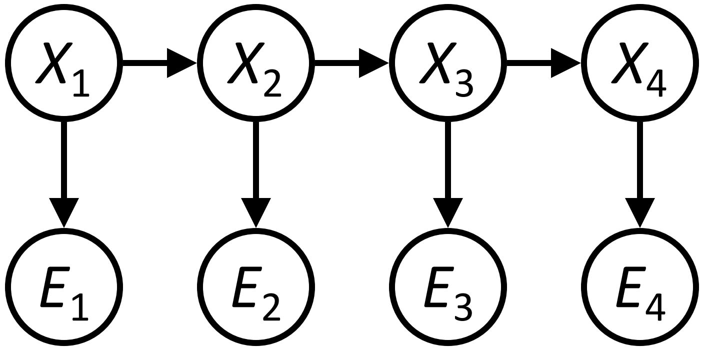
$$B_t(X) = P(X_t|e_{1:t})$$

- Idea of recursive filtering:
  - Start with $P(X_0)$
  - Derive B_t in terms of B_{t-1} given e_t
  - Equivalently, derive B_{t+1} in terms of B_t given e_{t+1}

Two Steps

1. Passage of time: predict possible transitions
2. Observation: update by incorporating the observation

$B(X_t) = P(X_t|e_{1:t})$ $B'(X_{t+1})$

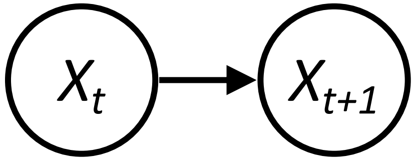


$B(X_{t+1})$

Step 1: Passage of Time

- Assume we have current belief P(X | evidence to date)

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



$$P(X_{t+1}|X_t)$$

- After one time step passes:

$$P(X_{t+1}|e_{1:t}) = \sum_{x_t} P(X_{t+1}, x_t|e_{1:t})$$

$$= \sum_{x_t} P(X_{t+1}|x_t, e_{1:t}) P(x_t|e_{1:t})$$

$$= \sum_{x_t} P(X_{t+1}|x_t) P(x_t|e_{1:t})$$

- Or compactly:

$$B'(X_{t+1}) = \sum_{x_t} P(X'|x_t) B(x_t)$$

- Basic idea: beliefs get "pushed" through the transitions
  - With the "B" notation, we have to be careful about what time step t the belief is about, and what evidence it includes

Example: Passage of Time

- As time passes, uncertainty "accumulates"

(Transition model: ghosts usually go clockwise)

<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
<0.01	<0.01	1.00	<0.01	<0.01	<0.01
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01

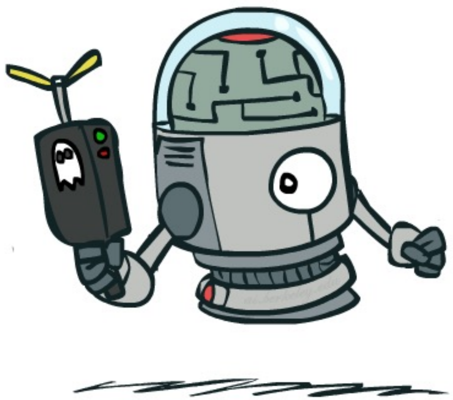
T = 0

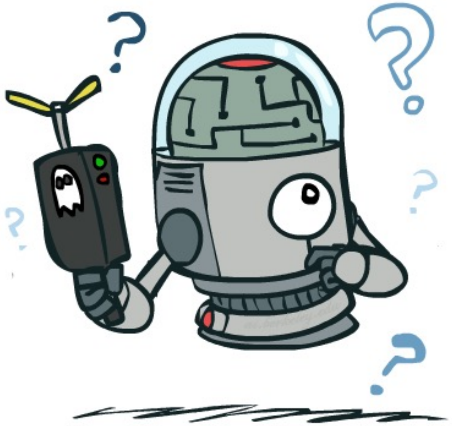
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
<0.01	<0.01	0.06	<0.01	<0.01	<0.01
<0.01	0.76	0.06	0.06	<0.01	<0.01
<0.01	<0.01	0.06	<0.01	<0.01	<0.01

T = 1

0.05	0.01	0.05	<0.01	<0.01	<0.01
0.02	0.14	0.11	0.35	<0.01	<0.01
0.07	0.03	0.05	<0.01	0.03	<0.01
0.03	0.03	<0.01	<0.01	<0.01	<0.01

T = 4





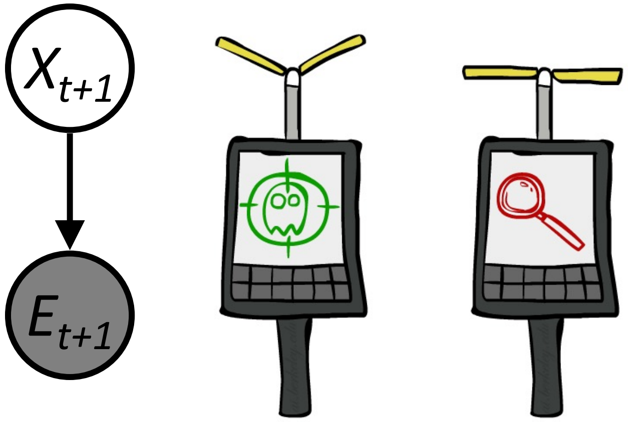
Step 2: Observation

- Assume we have the belief after passage of time:

$$B'(X_{t+1}) = P(X_{t+1}|e_{1:t})$$

- After evidence comes in:

$$\begin{aligned} P(X_{t+1}|e_{1:t+1}) &= P(X_{t+1}, e_{t+1}|e_{1:t})/P(e_{t+1}|e_{1:t}) \\ &\propto_{X_{t+1}} P(X_{t+1}, e_{t+1}|e_{1:t}) \\ &= P(e_{t+1}|e_{1:t}, X_{t+1})P(X_{t+1}|e_{1:t}) \\ &= P(e_{t+1}|X_{t+1})P(X_{t+1}|e_{1:t}) \end{aligned}$$



$P(e_{t+1}|X_{t+1})$

- Or, compactly:

$$B(X_{t+1}) \propto_{X_{t+1}} P(e_{t+1}|X_{t+1})B'(X_{t+1})$$

- Basic idea: beliefs “reweighted” by likelihood of evidence
- Unlike passage of time, we have to renormalize

Example: Observation

- As we get observations, beliefs get reweighted, uncertainty “decreases”

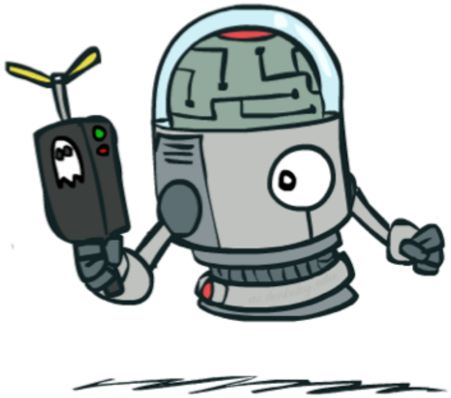
0.05	0.01	0.05	<0.01	<0.01	<0.01
0.02	0.14	0.11	0.35	<0.01	<0.01
0.07	0.03	0.05	<0.01	0.03	<0.01
0.03	0.03	<0.01	<0.01	<0.01	<0.01

Before observation

<0.01	<0.01	<0.01	<0.01	0.02	<0.01
<0.01	<0.01	<0.01	0.83	0.02	<0.01
<0.01	<0.01	0.11	<0.01	<0.01	<0.01
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01

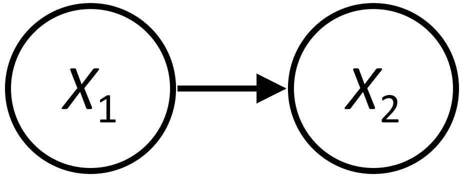
After observation

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



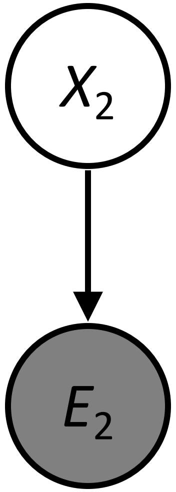
Online Belief Updates

- 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)

Forward Algorithm

- Transition matrix T, observation matrix O_t
  - Observation matrix contains likelihoods for E_t along its diagonal
  - E.g., for U_1 = true, $O_1 = \begin{pmatrix} 0.2 & 0 \\ 0 & 0.9 \end{pmatrix}$
- Forward algorithm becomes:
  - $B_{t+1} = \alpha\, O_{t+1} T^{\top} B_t$
  - easy to implement in Python or MATLAB
  - *lazy* normalization

X_{t-1}	$P(X_t \mid X_{t-1})$	
	sun	rain
sun	0.9	0.1
rain	0.3	0.7

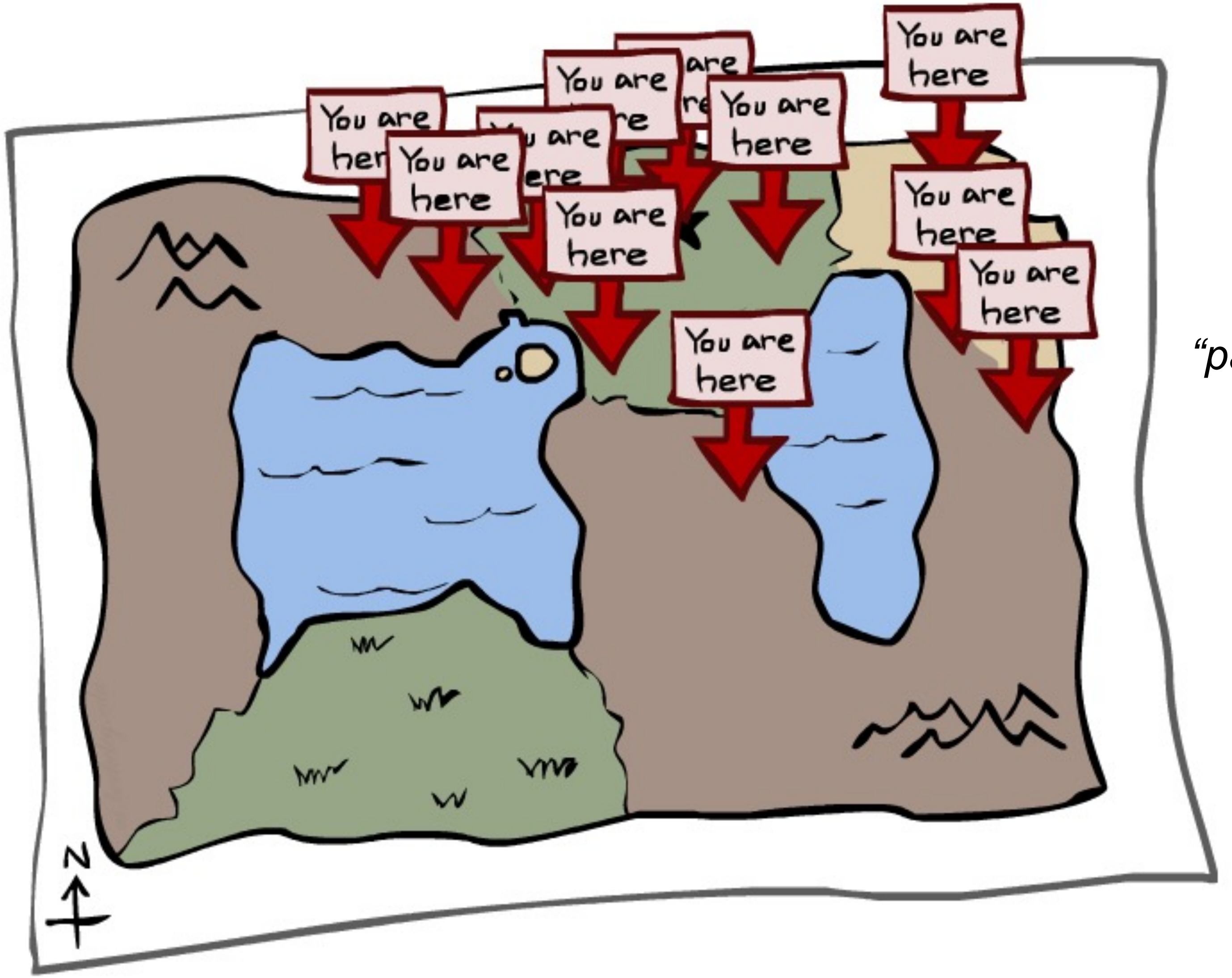
W_t	$P(U_t \mid W_t)$	
	true	false
sun	0.2	0.8
rain	0.9	0.1

Forward Algorithm Complexity

- $$P(X_{t+1} \mid e_{1:t+1}) = \underbrace{\alpha}_{\text{Normalize}} \underbrace{P(e_{t+1} \mid X_{t+1})}_{\text{Update}} \underbrace{\sum_{x_t} P(X_{t+1} \mid x_t)\, P(x_t \mid e_{1:t})}_{\text{Predict}}$$

- $B_{t+1} = \text{FORWARD}(B_t\,,\, e_{t+1})$
  - Cost per time step: $O(|X|^2)$ where $|X|$ is the number of states.
  - $O(|X|^2)$ is infeasible for models with large state spaces. (T_T)
  - *Approximate* filtering algorithms !!! (^_^)

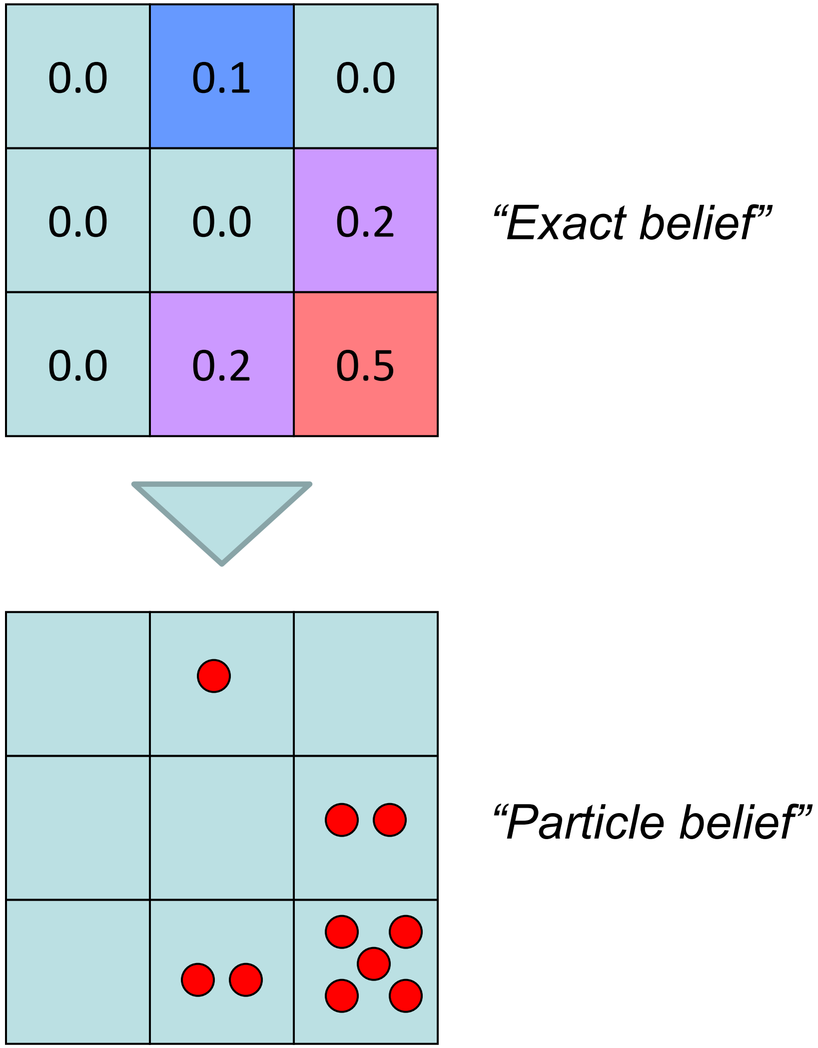
Particle Filtering



“particles”

Particle Filtering

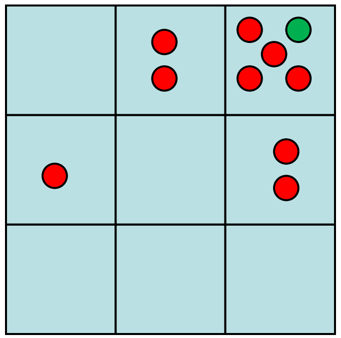
- Filtering: approximate solution
- Sometimes |X| is too big to use exact inference
  - |X| may be too big to even store B(X)
  - E.g. X is continuous
- Solution: approximate inference
  - Track *samples* of X, not all values
  - Samples are called *particles*
  - Time per step is linear in the *number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



Representation: Particles

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

"Particle belief"



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

Particle Filtering: Predict

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

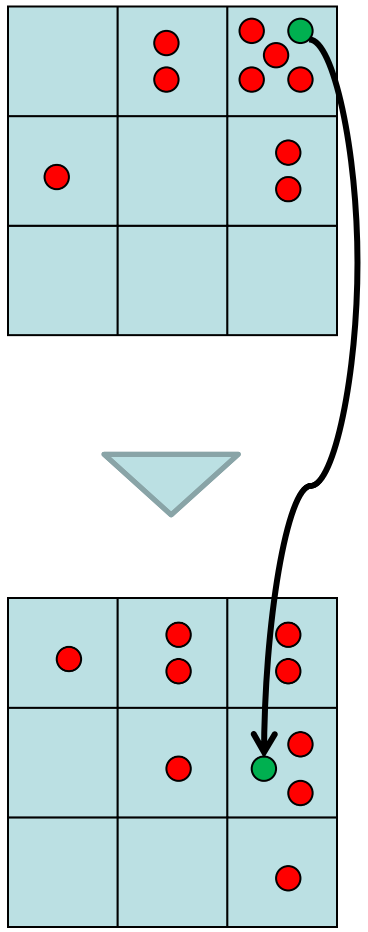
$$x' = \text{sample}(P(X'|x))$$

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

- Predict approximates the passage of time
  - If enough samples, close to exact values (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: Update

- Slightly trickier:
  - Don’t sample observation, *fix* it
  - Similar to *likelihood weighting*, weight samples based on the probability of evidence

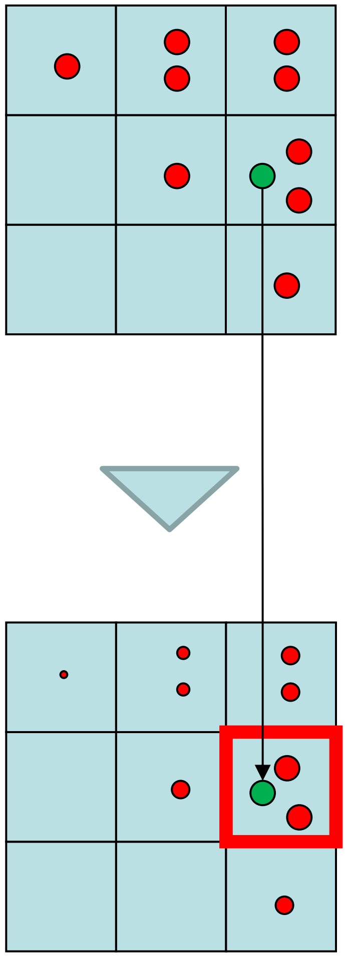
$$w(x) = P(e|x)$$

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

  - The weights don’t sum to N, since all have been downweighted
    - In fact, they now sum to an approximation of N*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

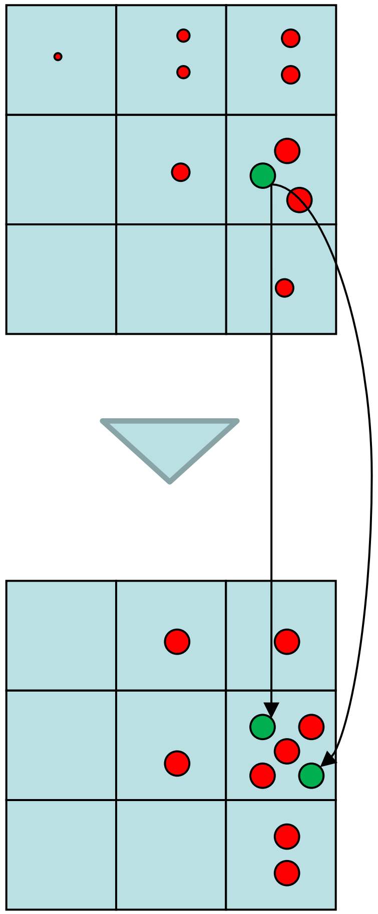


Particle Filtering: Resample

- Rather than tracking weighted samples, we *resample*
- We sample N times from the *weighted sample distribution*
- Weights of particles become 1 again
- Finishes the update for this time step, continue with the next one

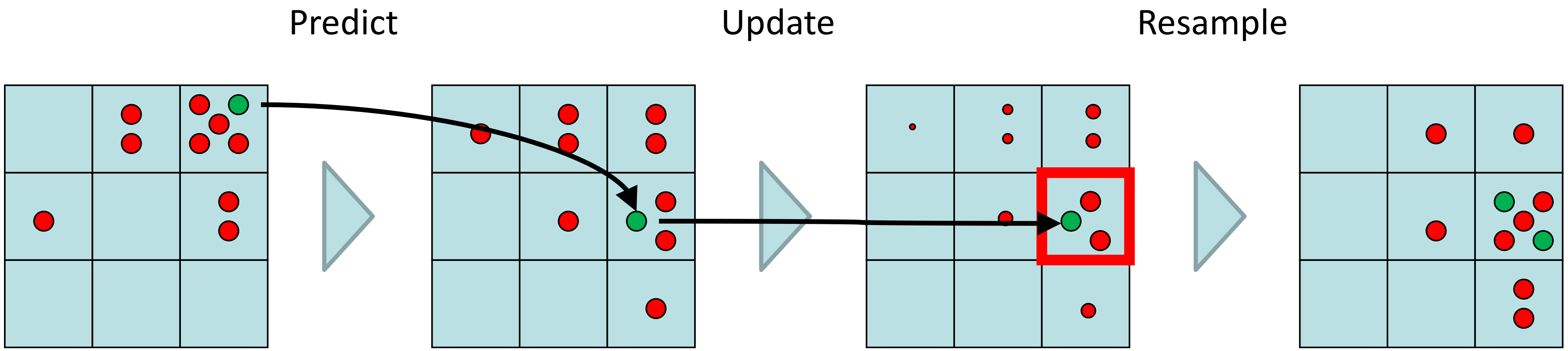
Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4

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



Putting Together: 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)]

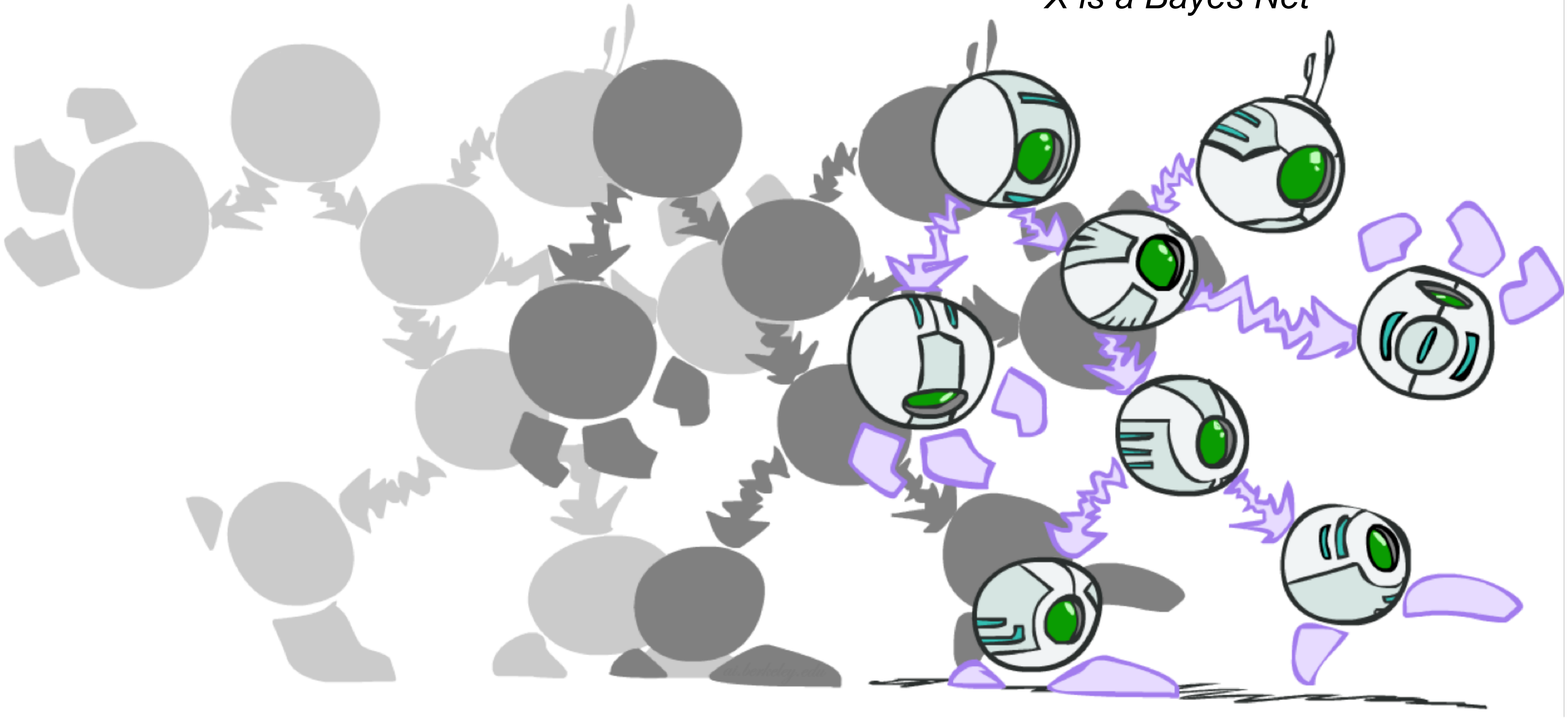
Video of Demo – Moderate Number of Particles

Video of Demo – One Particle

Video of Demo – Huge Number of Particles

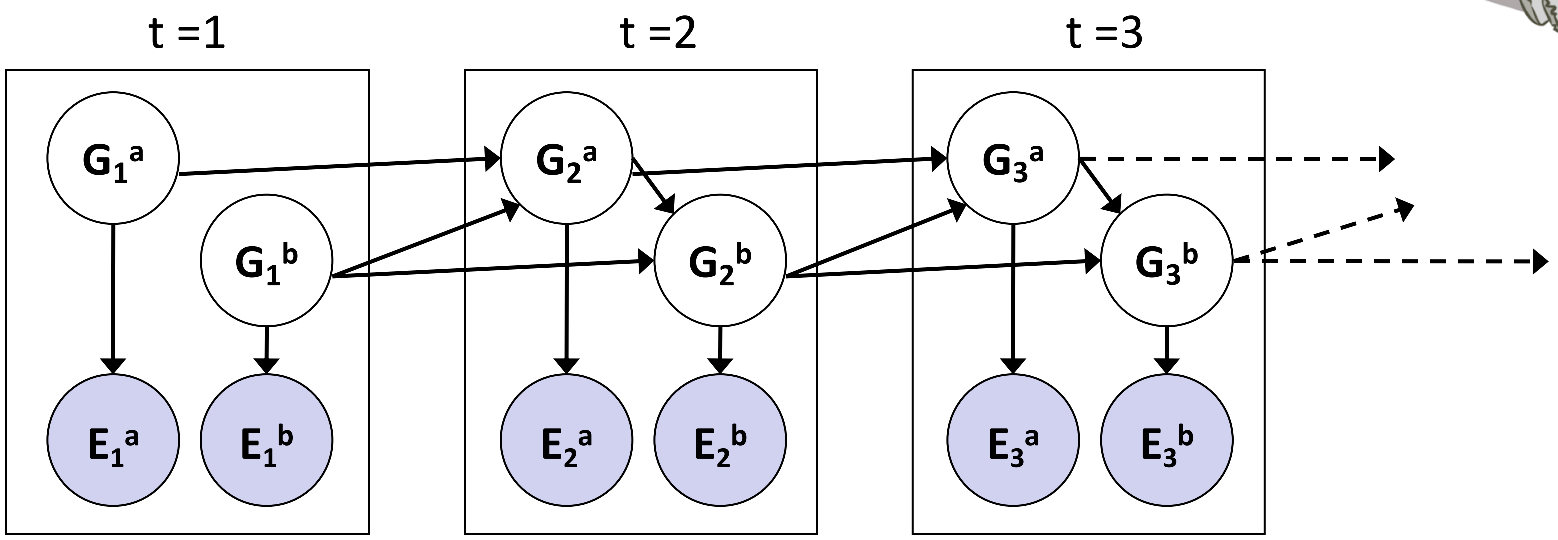
Dynamic Bayes Nets

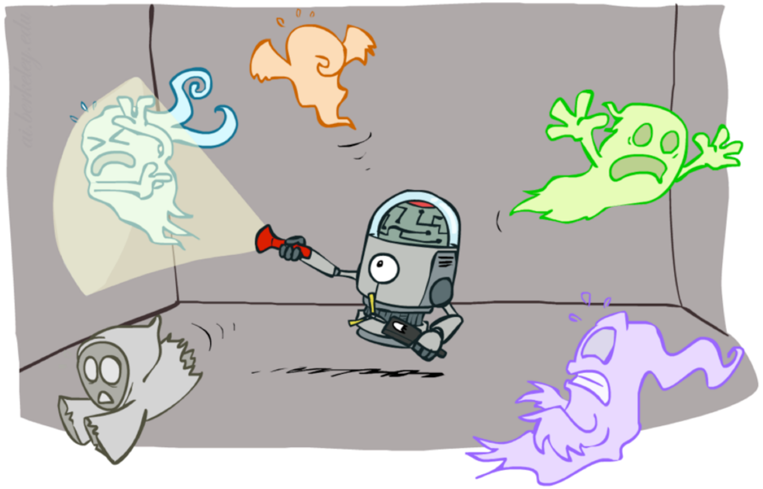
"X is a Bayes Net"

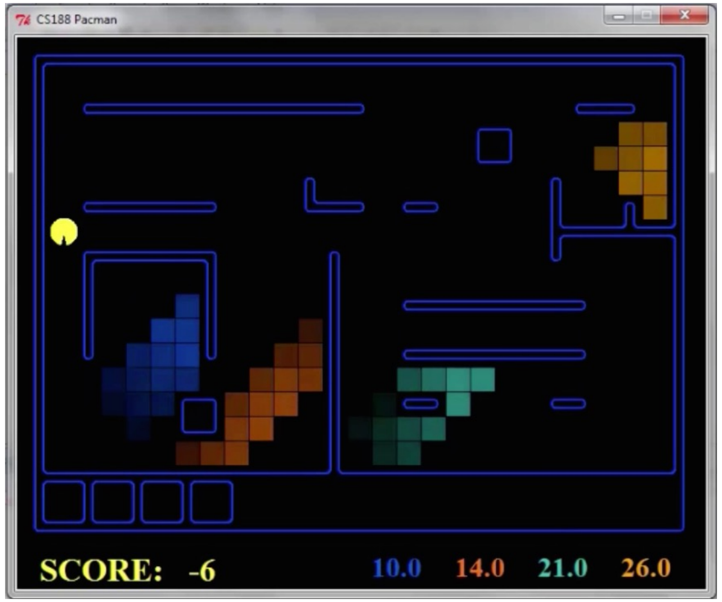


Dynamic Bayes Nets (DBNs)

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

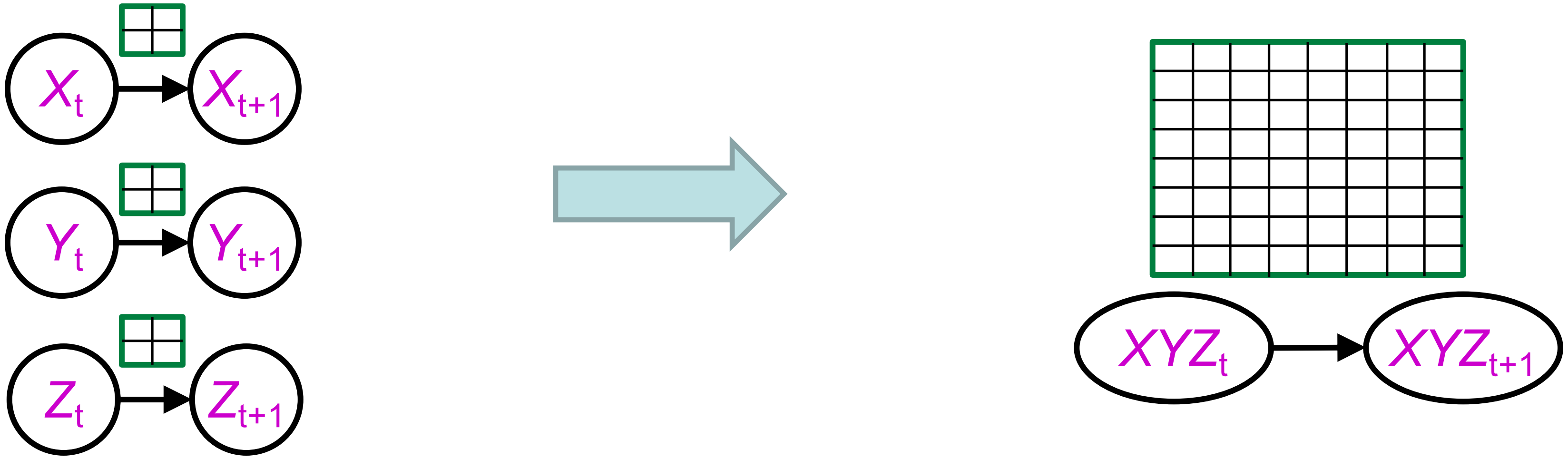






DBNs and HMMs

- Every HMM is a single-variable DBN
- Every discrete DBN is an HMM
  - HMM state is Cartesian product of DBN state variables

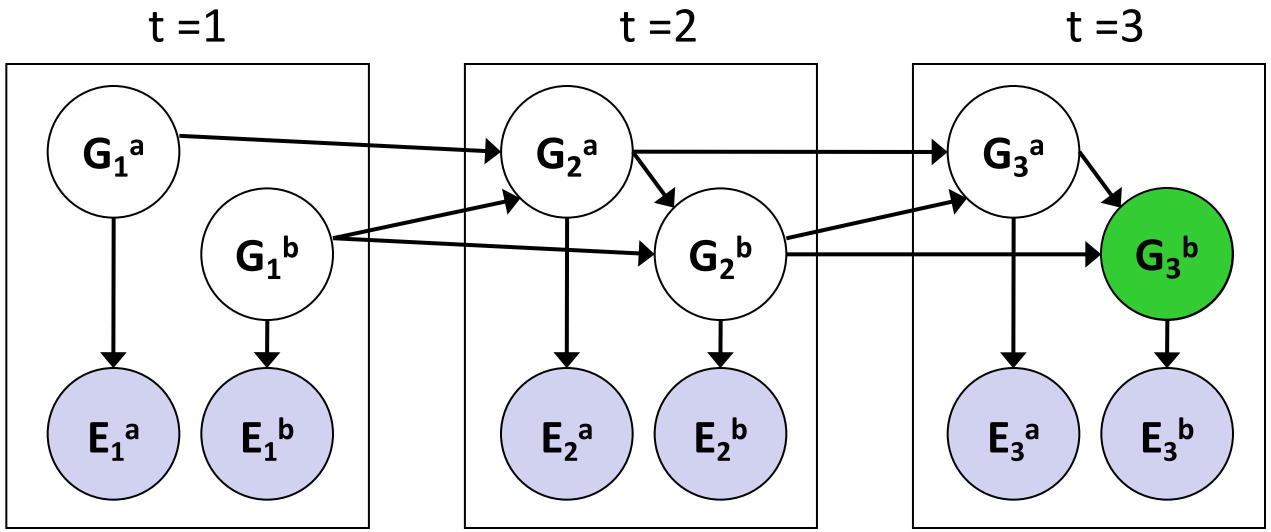


- Sparse dependencies => exponentially fewer parameters in DBN
  - E.g., 20 Boolean state variables, 3 parents each;

    DBN has $20 \times 2^3 = 160$ parameters, HMM has $2^{20} \times 2^{20} =\sim 10^{12}$ parameters

Exact Inference in DBNs

- Variable elimination applies to dynamic Bayes nets
- Offline: "unroll" the network for T time steps, then eliminate variables to find $P(X_T | e_{1:T})$



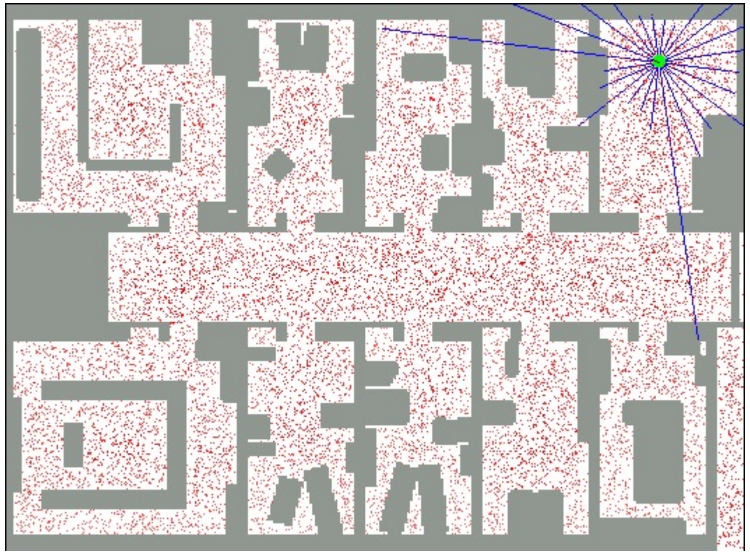
- Online: eliminate all variables from the previous time step; store factors for current time only
- Problem: largest factor contains all variables for current time (plus a few more)

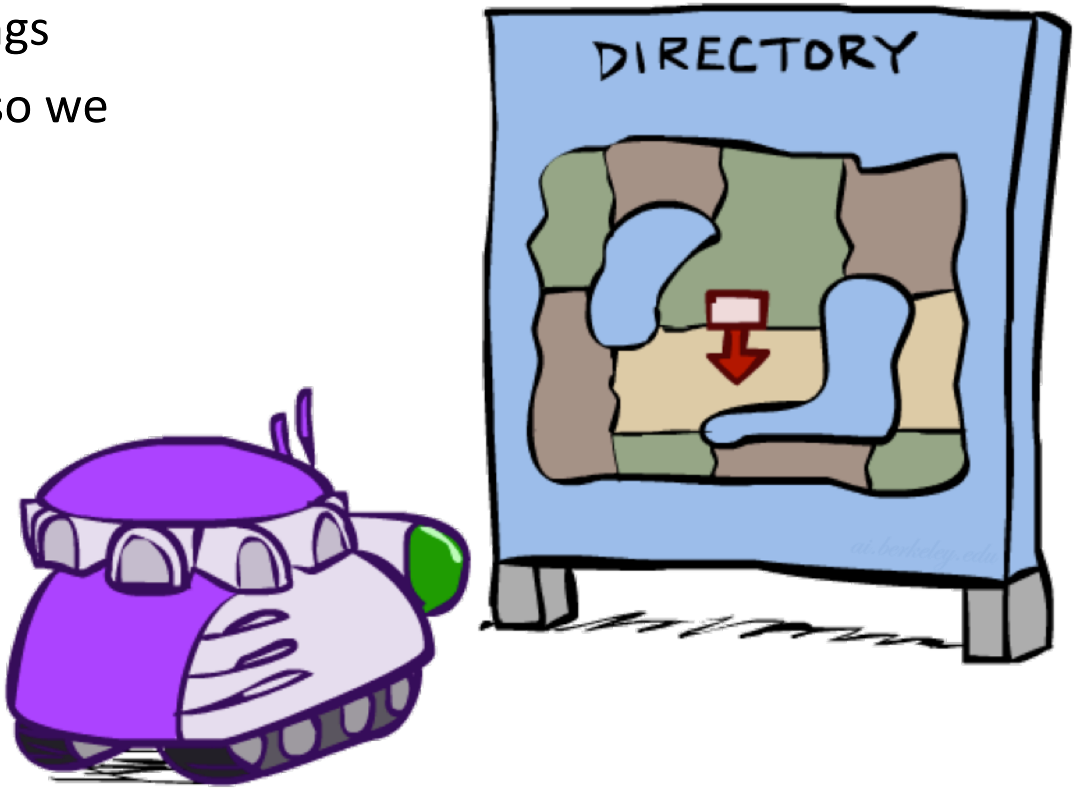
DBN Particle Filters

- A particle is a *complete* sample for a time step
- **Initialize**: Generate prior samples for the t=0 Bayes net
  - Example particle: $\mathbf{G_0^a} = (3,3)$ $\mathbf{G_0^b} = (5,3)$
- **Predict**: Sample a successor for each particle
  - Example successor: $\mathbf{G_1^a} = (2,3)$ $\mathbf{G_1^b} = (6,3)$
- **Update**: Weight each *entire* sample by the likelihood of the evidence conditioned on the sample
  - Likelihood: $P(\mathbf{E_1^a} \mid \mathbf{G_1^a}) * P(\mathbf{E_1^b} \mid \mathbf{G_1^b})$
- **Resample:** Select prior samples (tuples of values) in proportion to their weights

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, so we cannot represent an exact belief
  - Particle filtering is a main technique



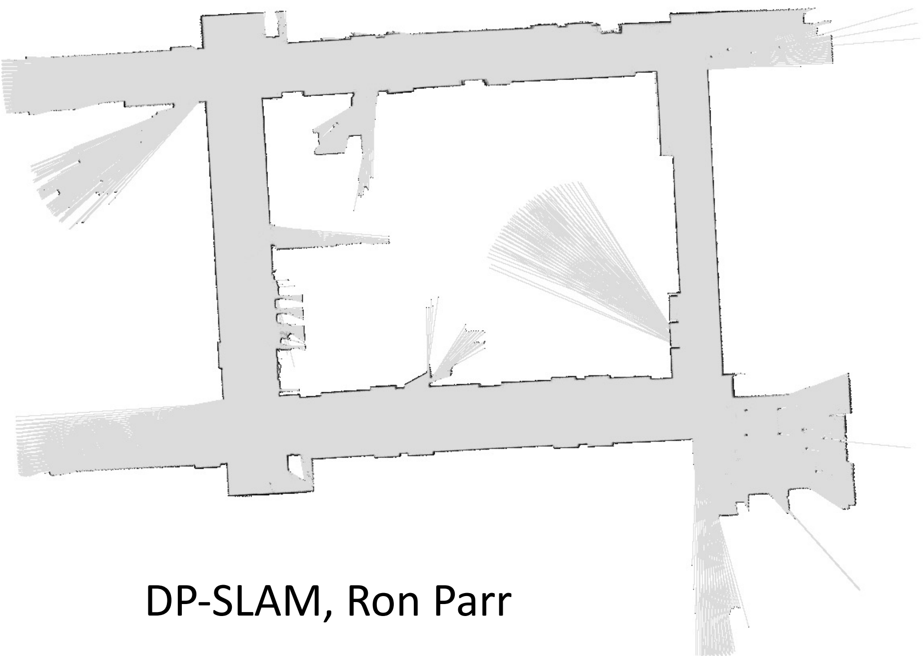


Particle Filter Localization (Sonar)

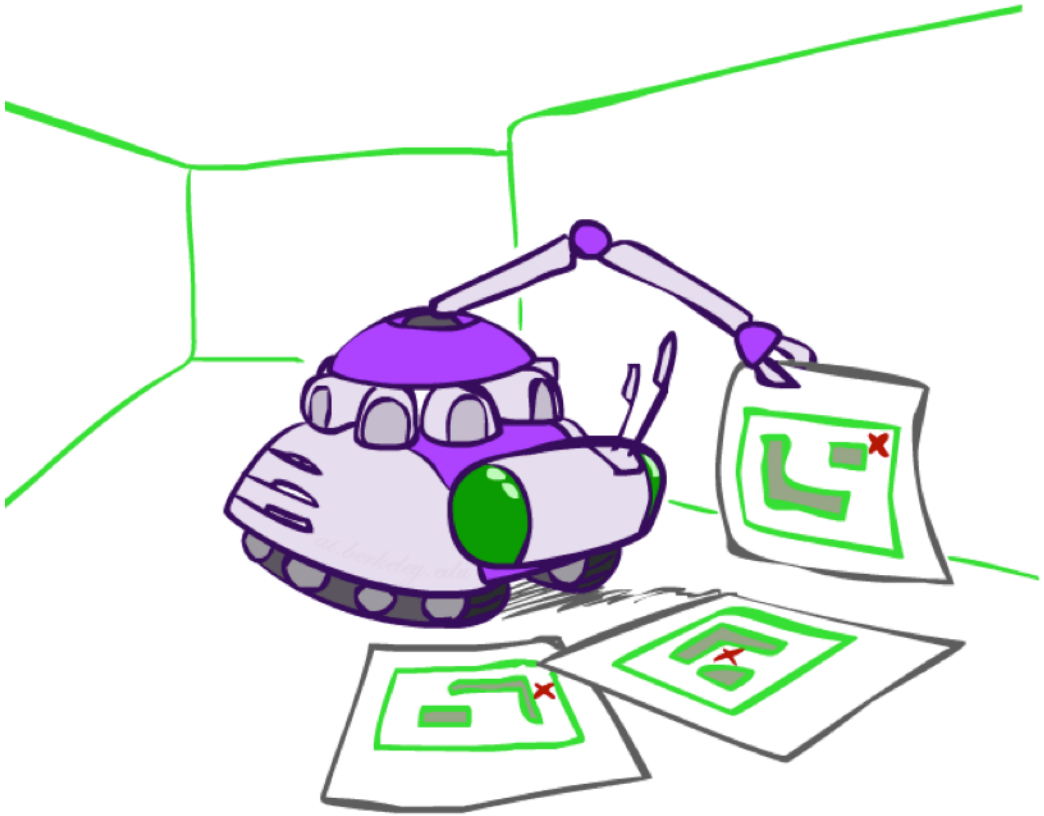
[Video: global-sonar-uw-annotated.avi]

Robot Mapping

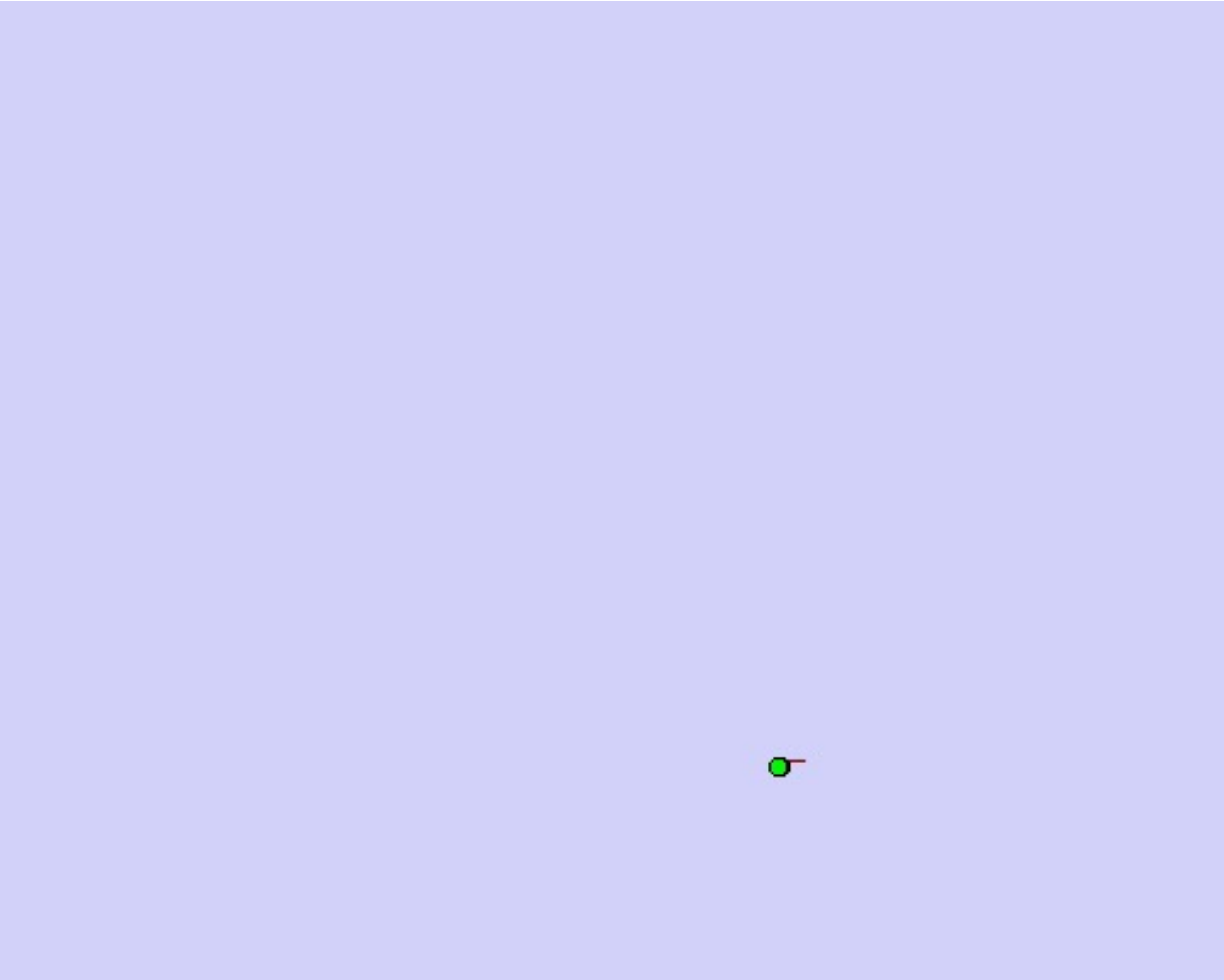
- SLAM: Simultaneous Localization And Mapping
  - Robot does not know map or location
  - State $x_t^{(j)}$ consists of position+orientation, map!
  - (Each map usually inferred exactly given sampled position+orientation sequence: RBPF)



DP-SLAM, Ron Parr



Particle Filter SLAM – Video 2



[Demo: PARTICLES-SLAM-fastslam.avi]