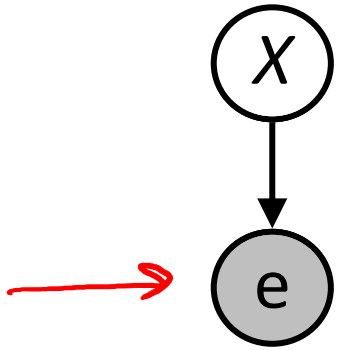
Warm-up as you log in

When sampling with likelihood weighting, what distribution do we have when we multiply fraction of counts times the weight?

$$\frac{N(X=+x)}{N} \cdot \text{weight}(+x)$$



Announcements

Assignments (everything left for the semester):

- HW11 (online) due Tue 4/21
- P5 due Thu 4/30
- HW12 (written) out Tue 4/21, due Tue 4/28

Participation points

- Starting ~~now~~, we're capping the denominator (63 polls) in the participation points calculation

Final Exam:

- 5/4 1-4pm (let us know by today 4/20 if you need it rescheduled)

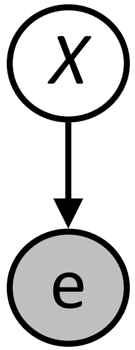
Warm-up as you log in

When sampling with likelihood weighting, what distribution do we have when we multiply fraction of counts times the weight?

$$\hat{P}(X) \times P(e \mid X) \rightarrow P(X, e) \rightarrow P(X \mid e)$$

$$+x \quad N(+x)/N \times weight(+x)$$

$$-x \quad N(-x)/N \times weight(-x)$$

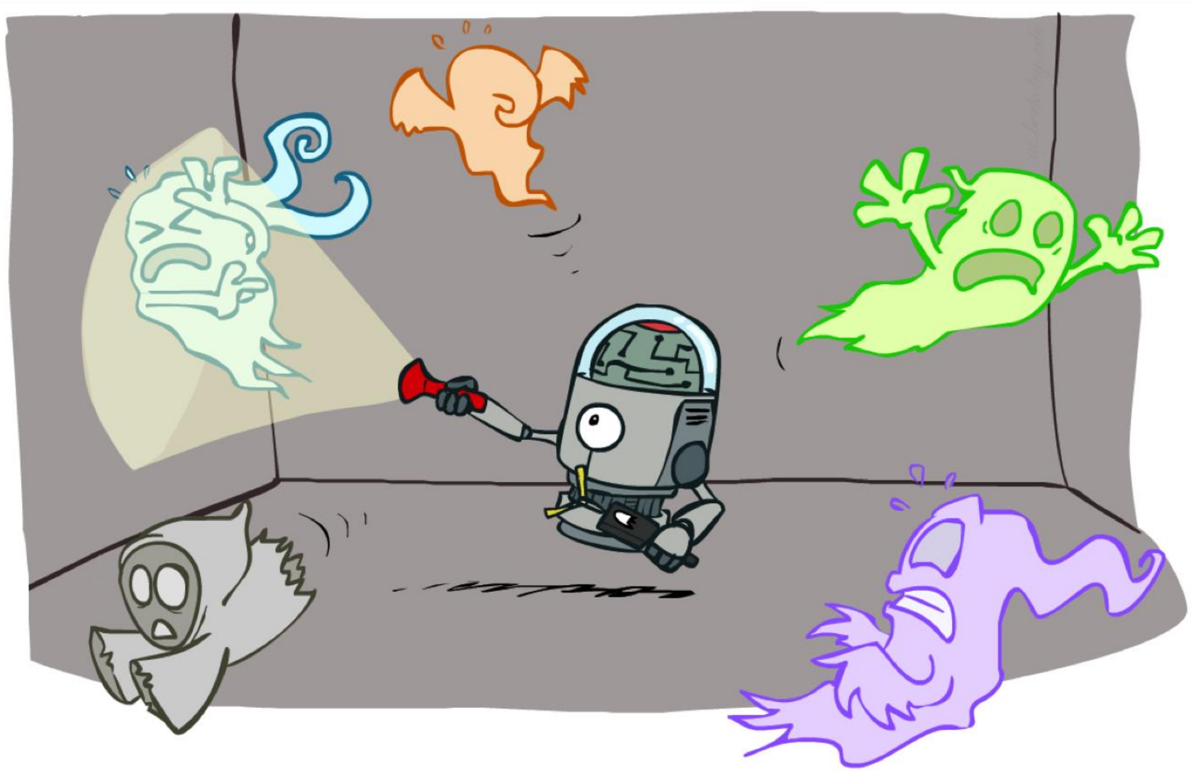


$$weight(+x) = P(e \mid +x)$$

$$w(-x) = P(e \mid -x)$$

AI: Representation and Problem Solving

HMMs and Particle Filtering



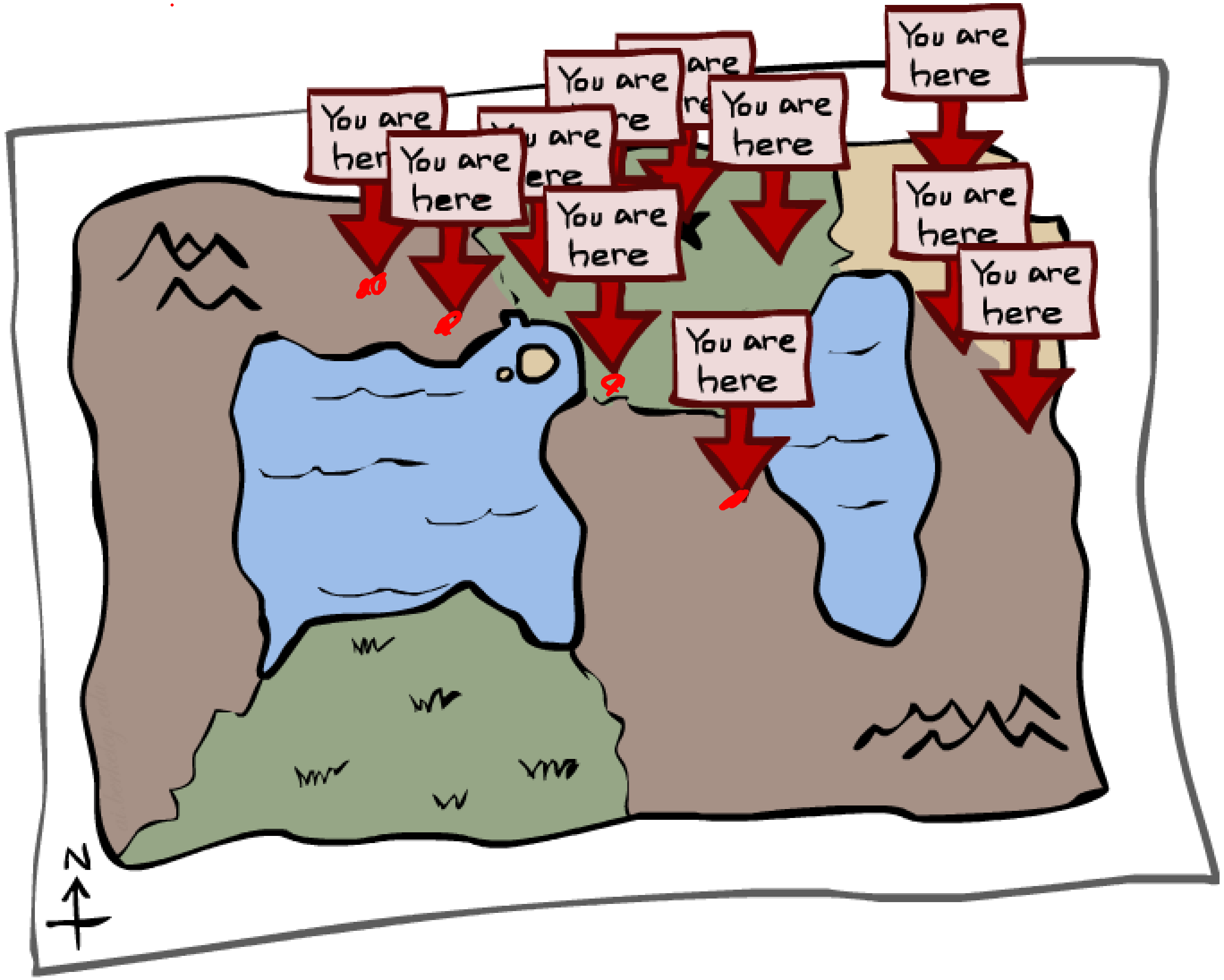
Instructors: Pat Virtue & Stephanie Rosenthal

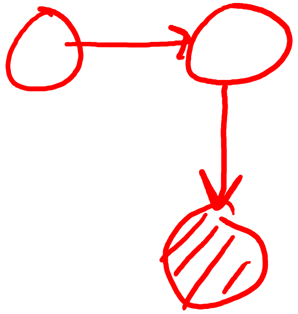
Slide credits: CMU AI and http://ai.berkeley.edu

Demo: Pacman Ghostbusters

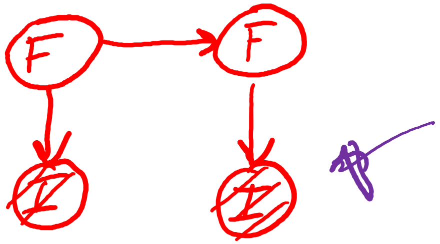
[Demos: ghostbusters (L14D1,3)]

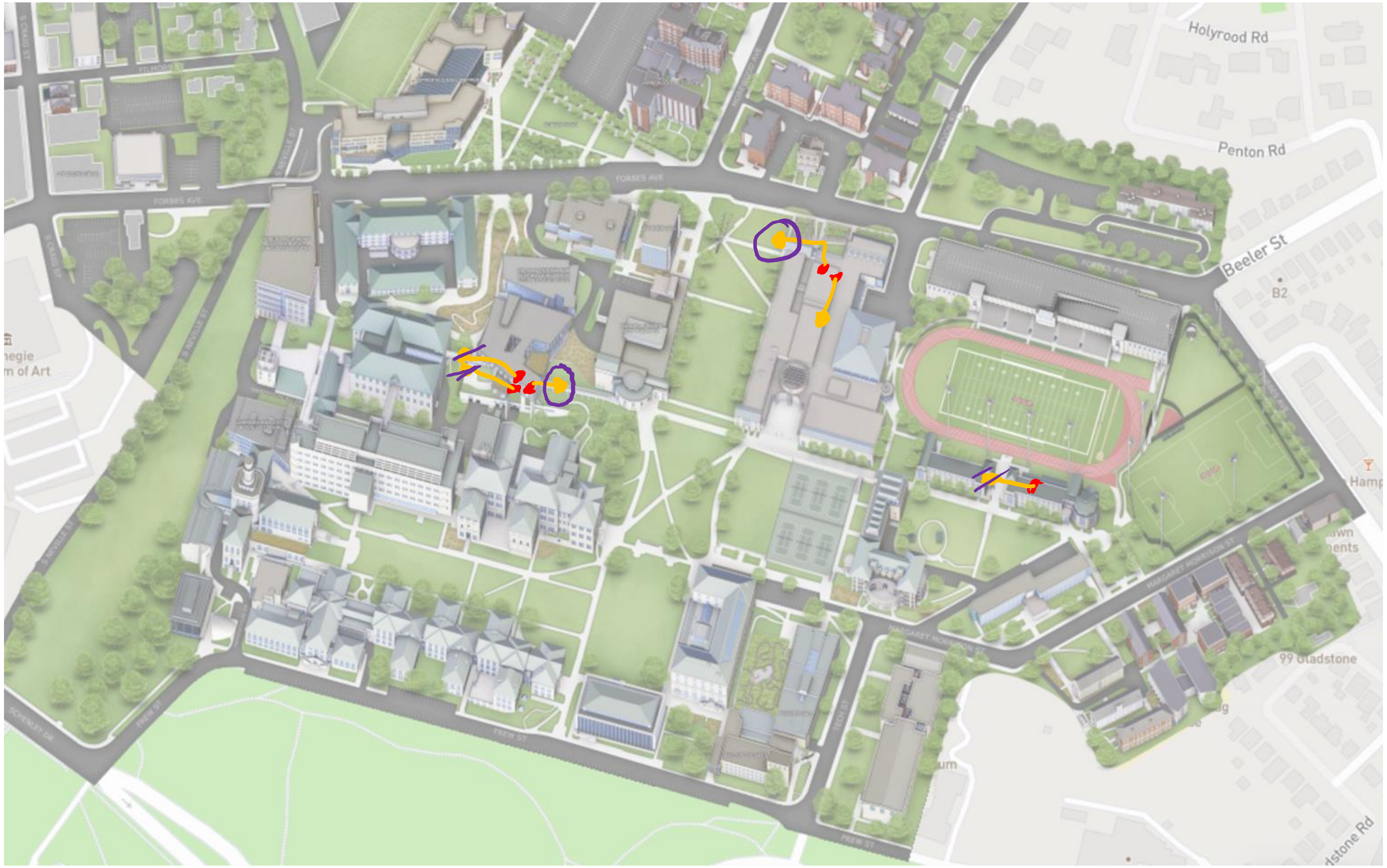
Particle Filtering





Particle Filtering





$P(F_0)$

$\downarrow$ $P(F_{t+1} | F_t)$

$P(F_1)$

$\downarrow$ $P(e_t | F_t)$

$P(F_1 | e_1)$

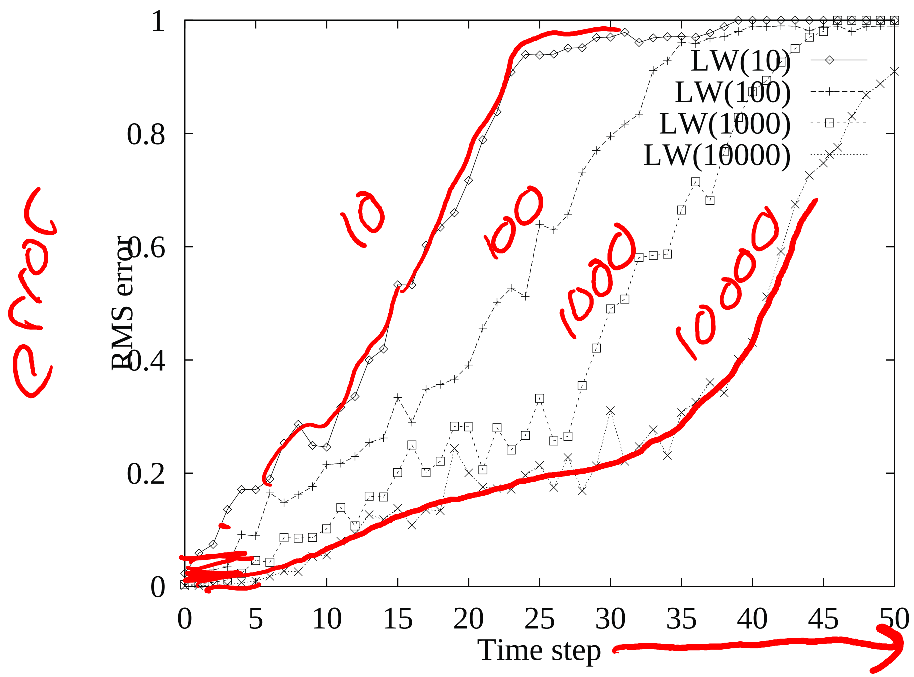
$\downarrow$

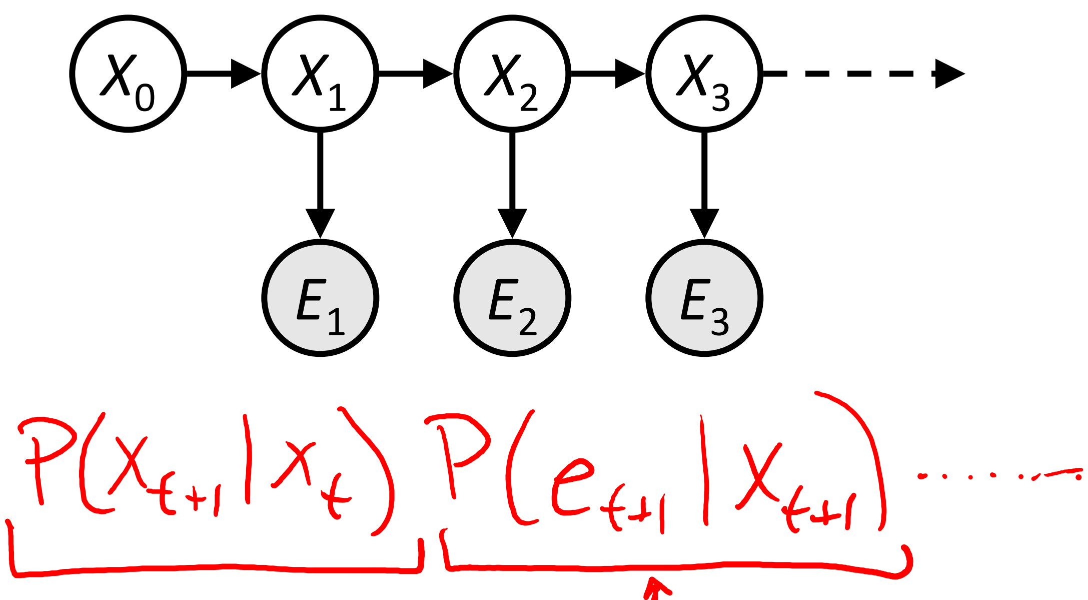
$P(F_2 | e_1)$

We need a new algorithm!

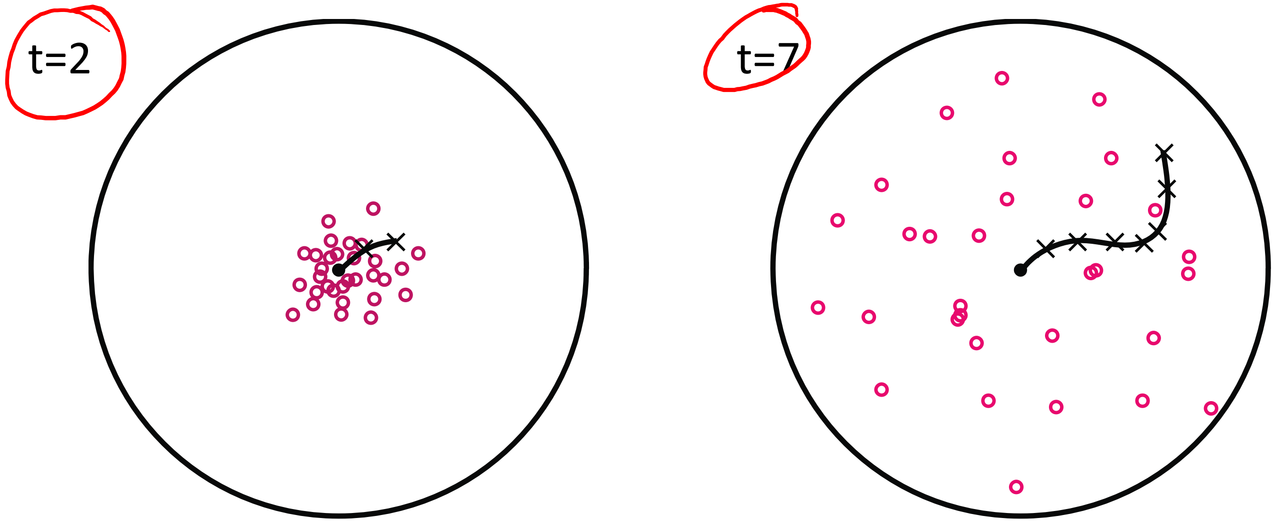
When $|X|$ is more than 10^6 or so (e.g., 3 ghosts in a 10x20 world), exact inference becomes infeasible

Likelihood weighting fails completely – number of samples needed grows ***exponentially*** with T





We need a new idea!



The problem: sample state trajectories go off into low-probability regions, ignoring the evidence; too few "reasonable" samples

Solution: kill the bad ones, make more of the good ones

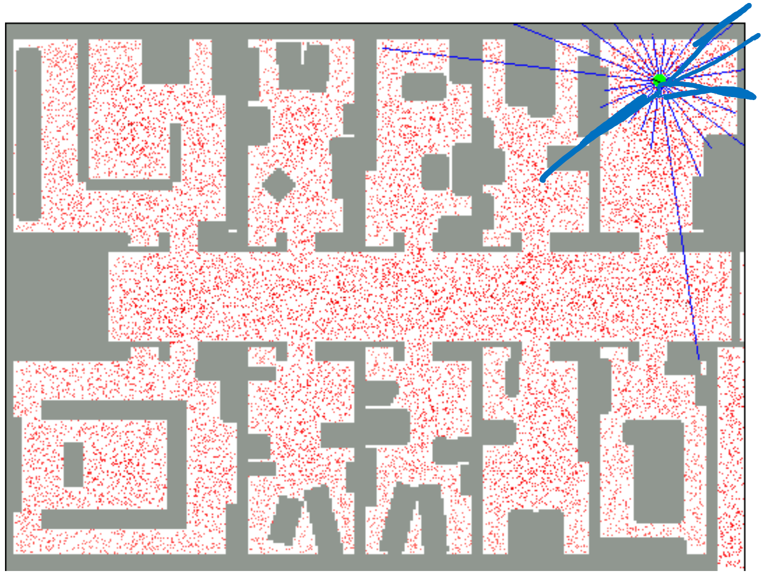
This way the population of samples stays in the high-probability region

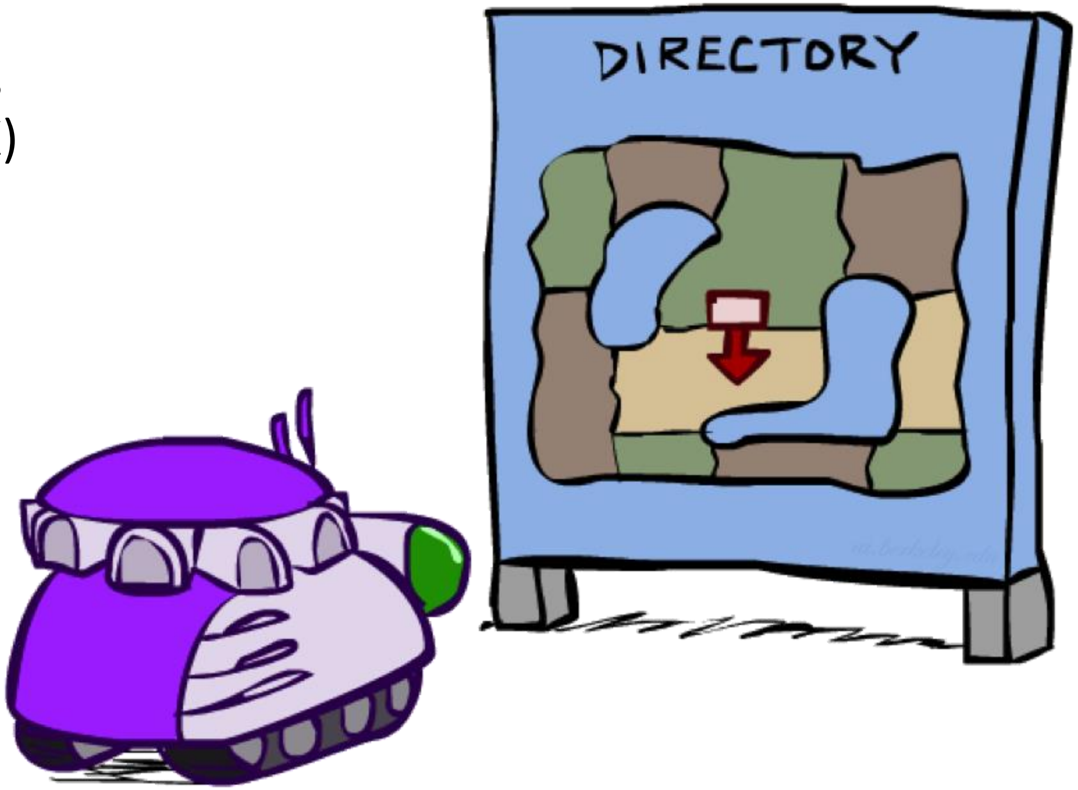
This is called resampling or survival of the fittest

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)

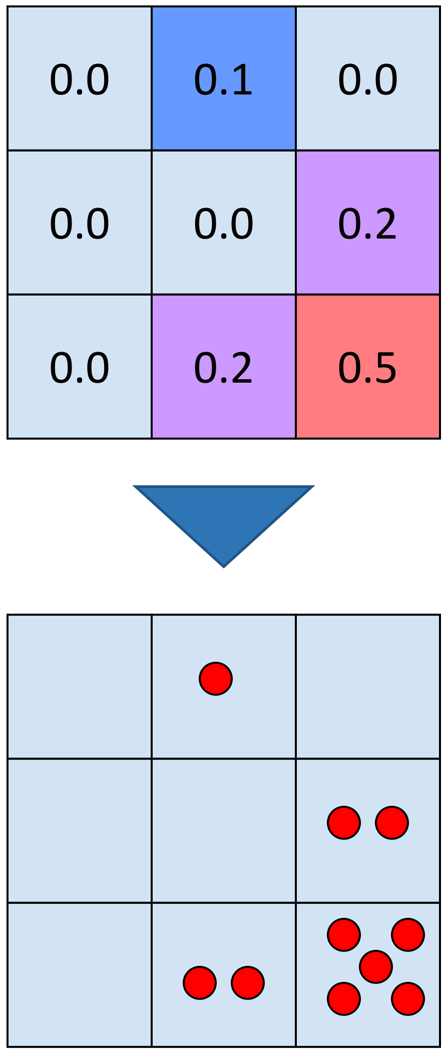
[Dieter Fox, et al.]

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

Particle Filtering

$\hat{P}(X_t | e_{1:t})$ $B(X)$

- Represent belief state by a set of samples
  - Samples are called ***particles***
  - Time per step is linear in the number of samples
  - But: number needed may be large
- This is how robot localization works in practice



Representation: Particles

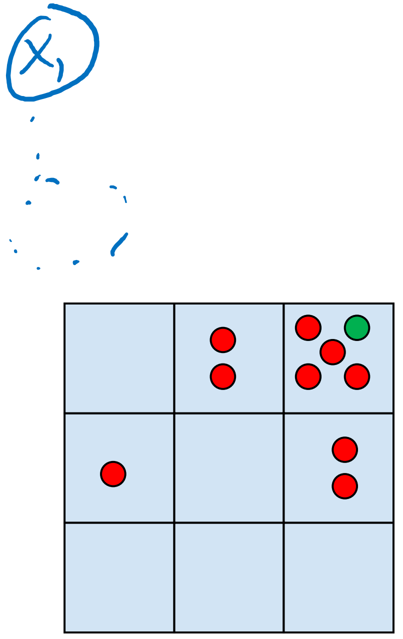
Our representation of P(X) is now a list of N particles (samples)

- 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 may have P(x) = 0!
- More particles, more accuracy
- Usually we want a low-dimensional marginal
  - E.g., “Where is ghost 1?” rather than “Are ghosts 1,2,3 in {2,6], [5,6], and [8,11]?”

For now, all particles have a weight of 1



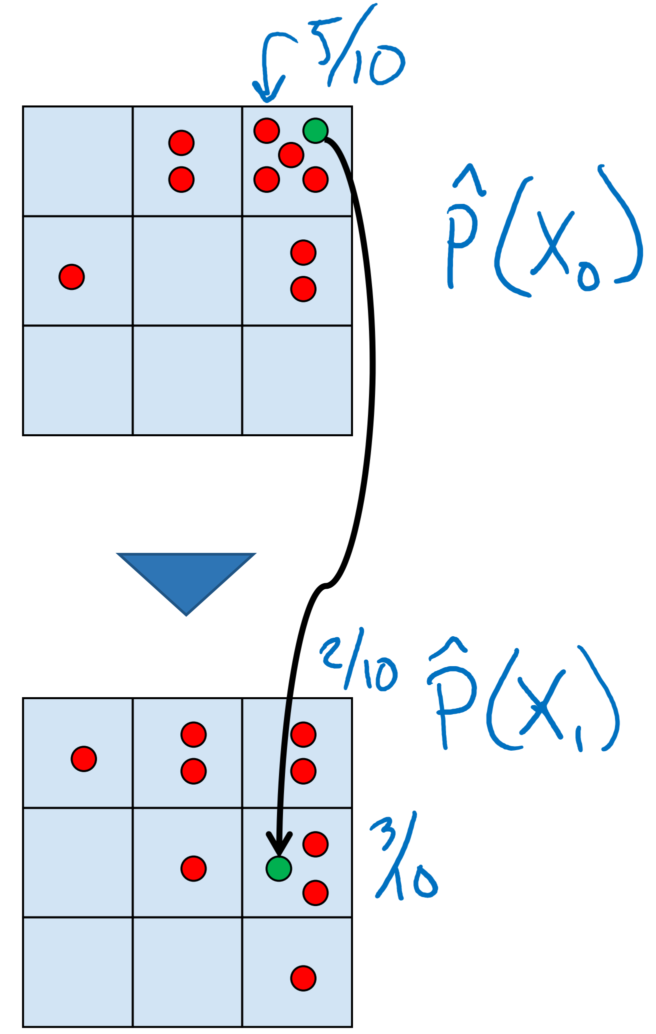
Particles:
(3,3)
(2,3)
(3,3)
(3,2)
(3,3)
(3,2)
(1,2)
(3,3)
(3,3)
(2,3)

Particle Filtering: Propagate forward

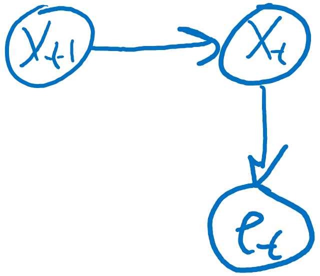
- A particle in state x_t is moved by sampling its next position directly from the transition model:
  - $x_{t+1} \sim P(X_{t+1} \mid x_t)$
  - Here, most samples move clockwise, but some move in another direction or stay in place
- This captures the passage of time
  - If enough samples, close to exact 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)

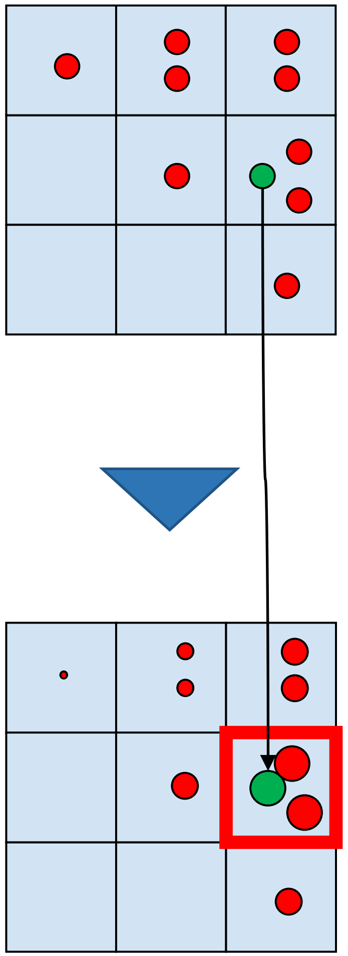


Particle Filtering: Observe



- Slightly trickier:
  - Don’t sample observation, fix it
  - Similar to likelihood weighting, weight samples based on the evidence
    - $W = P(e_t \mid x_t)$
  - Normalize the weights: particles that fit the data better get higher weights, others get lower weights

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***

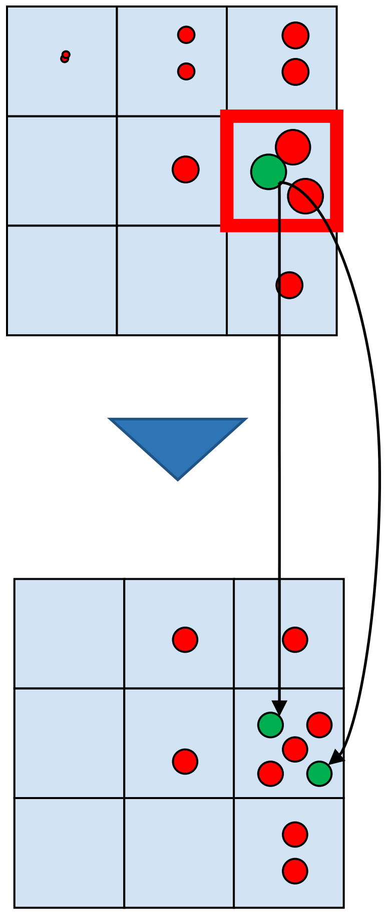
We have an updated belief distribution based on the weighted particles

We sample N new particles from the weighted belief distributions

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

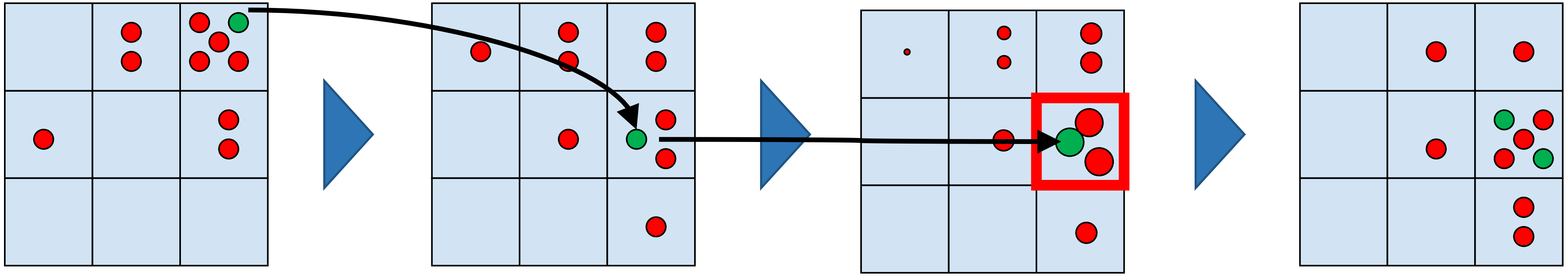
(New) Particles:
(3,2)
(2,2)
(3,2)
(2,3)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(3,2)



Summary: Particle Filtering

Particles: track samples of states rather than an explicit distribution

Propagate forward   Weight   Resample



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)

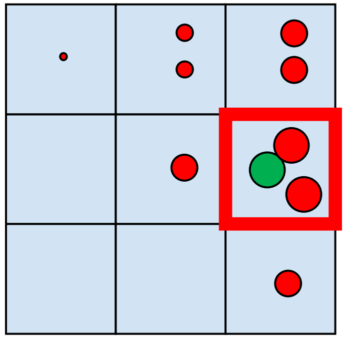
Consistency: see proof in AIMA Ch. 15

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

Weighting and Resampling

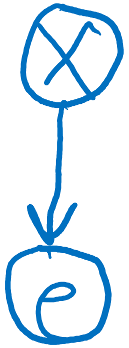
How to compute a belief distribution given weighted particles

Weight



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



$\hat{P}(X) \times P(e|X) = P(X,e)$

$\hat{P}(X)$:

1/10	2/10	2/10
	1/10	3/10
		1/10

$P(e|X)$:

1/10	2/10	4/10
	4/10	9/10
		4/10

$P(X,e)$:

1/100	4/100	8/100
0/100	4/100	27/100
0/100	0/100	4/100

$1+4+8+4+27+4$

$\frac{48}{100}$

$\frac{\frac{1}{100}}{\frac{48}{100}}$

$\frac{P(X,e)}{\sum_x P(x,e)}$

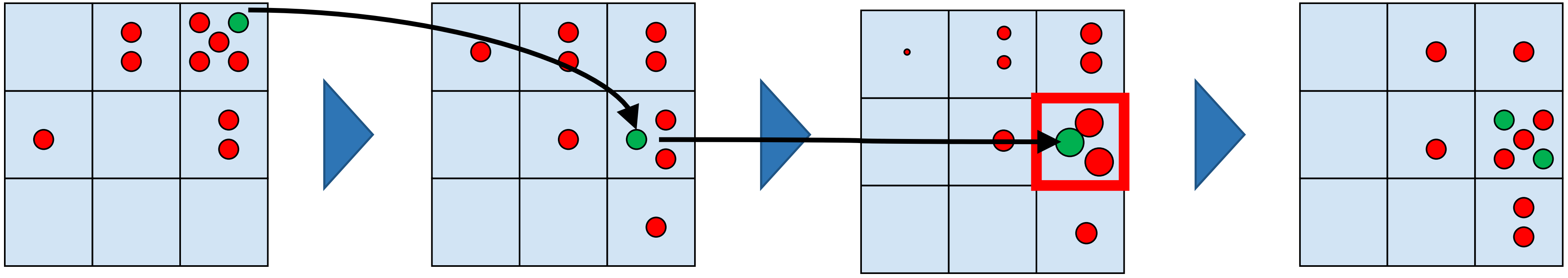
$P(X|e)$:

1/48	4/48	8/48
0	4/48	27/48
0	0	4/48

Summary: Particle Filtering

Particles: track samples of states rather than an explicit distribution

Propagate forward　　Weight　　Resample



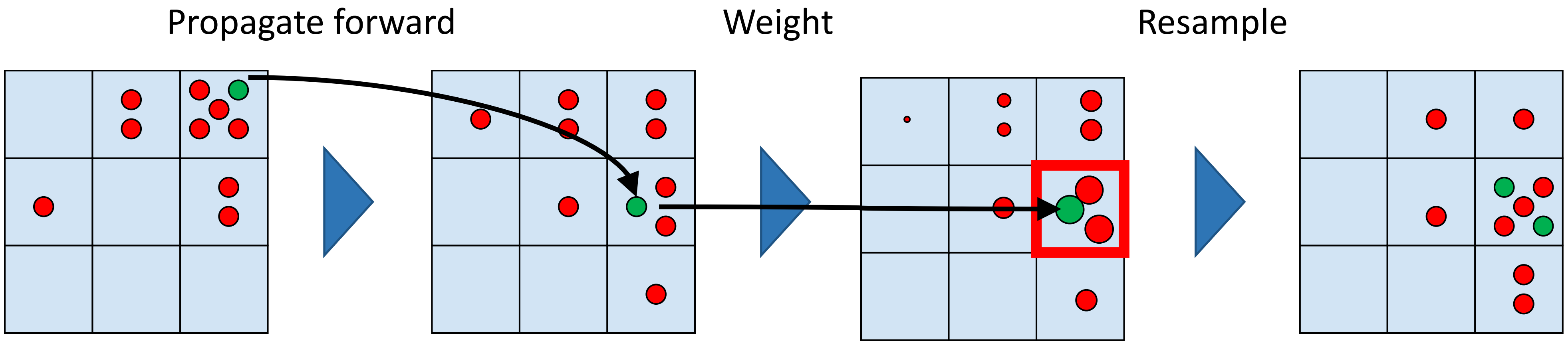
Particles:	Particles:	Particles:	(New) Particles:
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(2,3)	(2,3)	(2,3) w=.2	(2,2)
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(3,2)	(3,1)	(3,1) w=.4	(2,3)
(3,3)	(3,3)	(3,3) w=.4	(3,3)
(3,2)	(3,2)	(3,2) w=.9	(3,2)
(1,2)	(1,3)	(1,3) w=.1	(1,3)
(3,3)	(2,3)	(2,3) w=.2	(2,3)
(3,3)	(3,2)	(3,2) w=.9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

Consistency: see proof in AIMA Ch. 15

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

Piazza Poll 1

If we only have one particle which of these steps are unnecessary?



$.9 \downarrow \frac{.9}{.9} = 1$

0

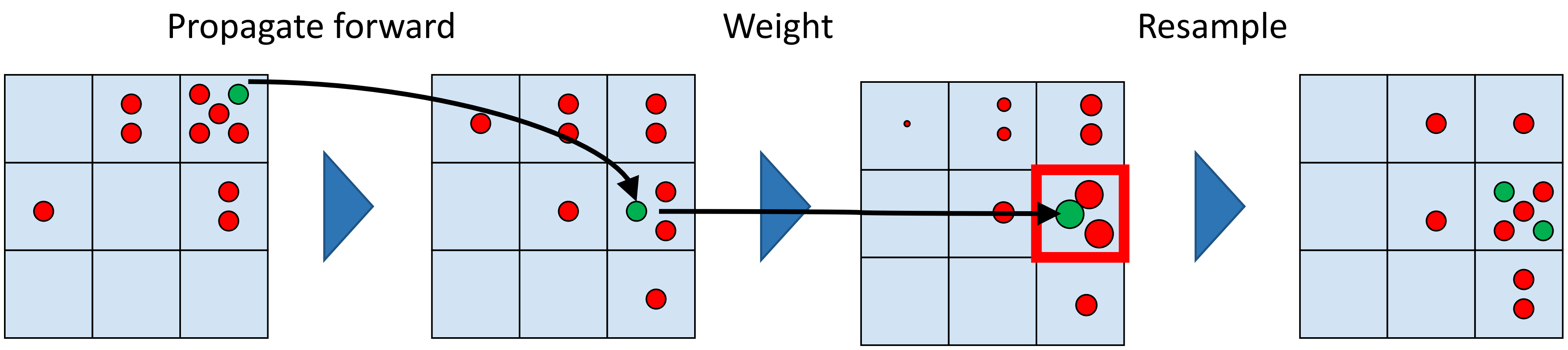
Select all that are unnecessary.

A. Propagate forward
B. Weight
C. Resample
D. None of the above

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

Piazza Poll 1

If we only have one particle which of these steps are unnecessary?



Select all that are unnecessary.

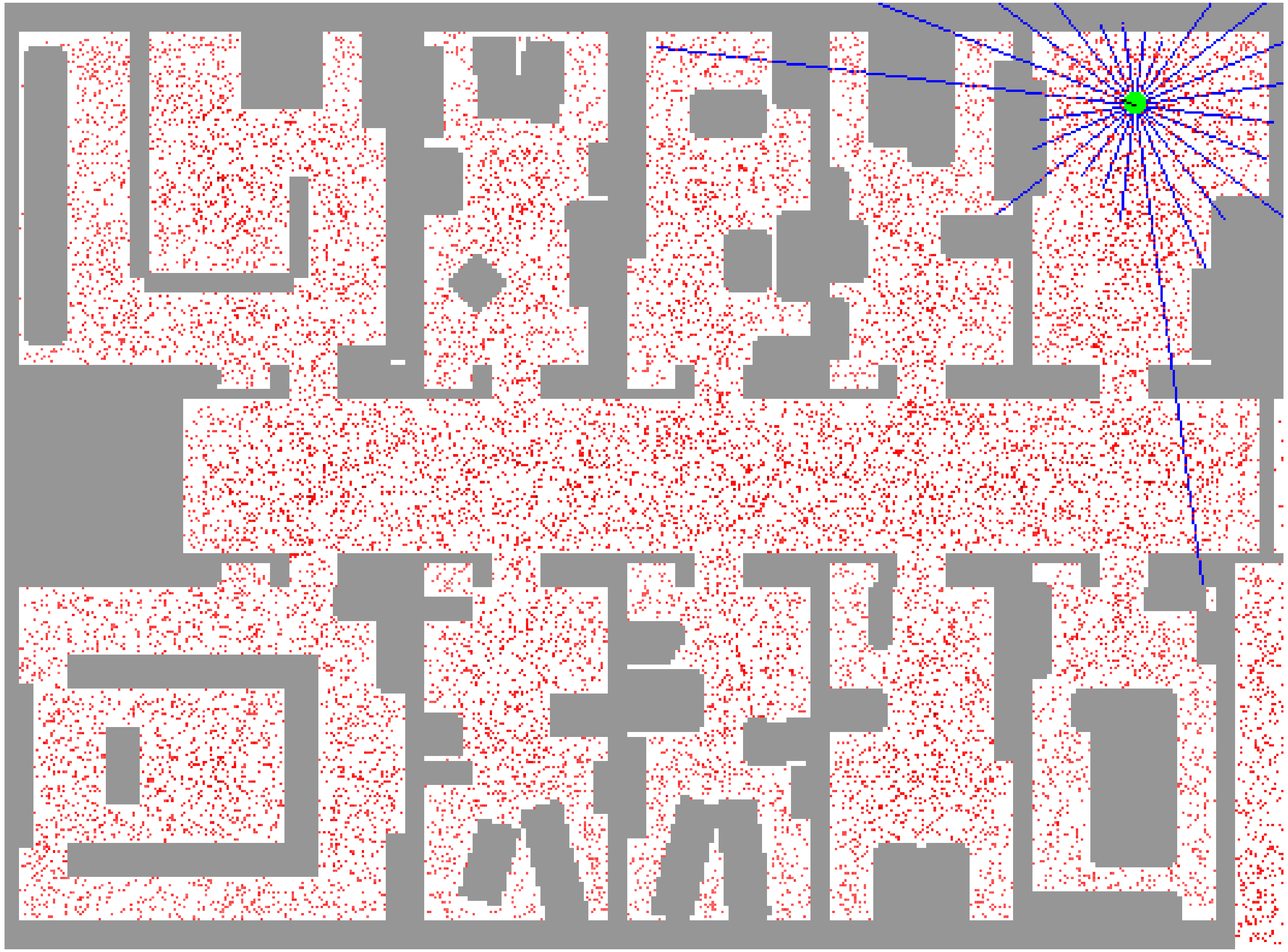
A. Propagate forward
B. Weight
C. Resample
D. None of the above

Unless the weight is zero, in which case, you'll want to resample from the beginning ☹

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

Demo: Pacman Particle Filtering

Particle Filter Localization (Laser)



[Dieter Fox, et al.]

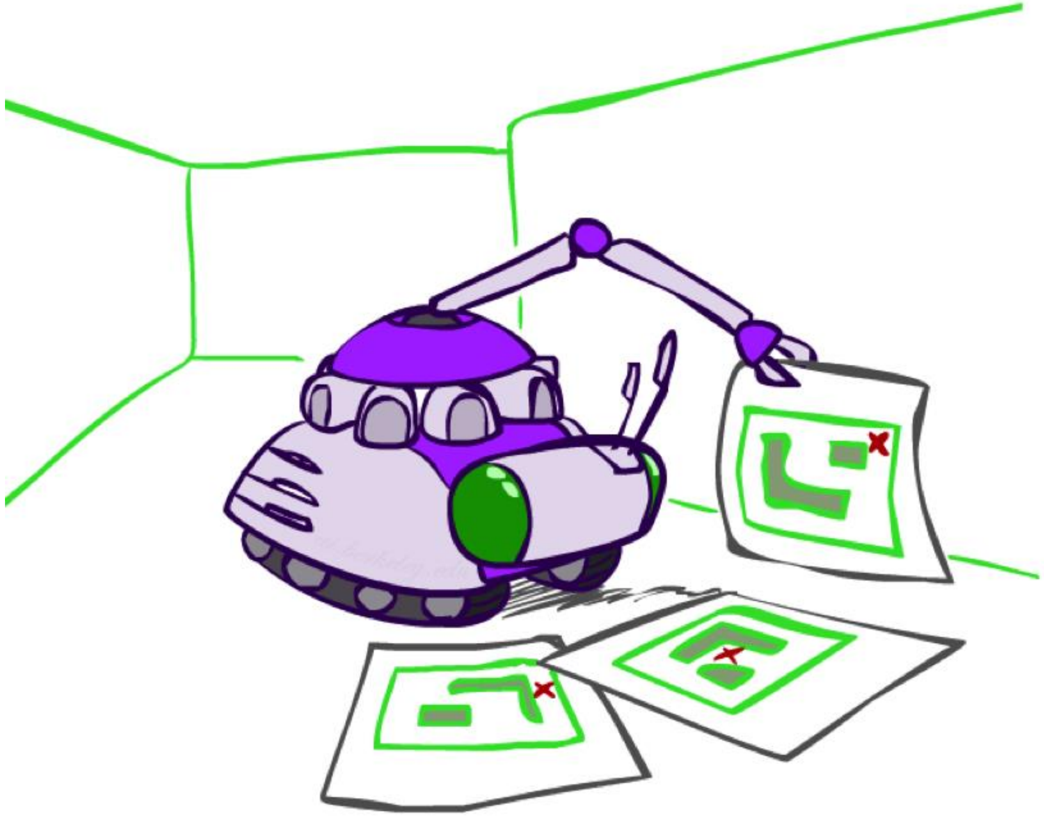
[Video: global-floor.gif]

Robot Mapping

SLAM: Simultaneous Localization And Mapping

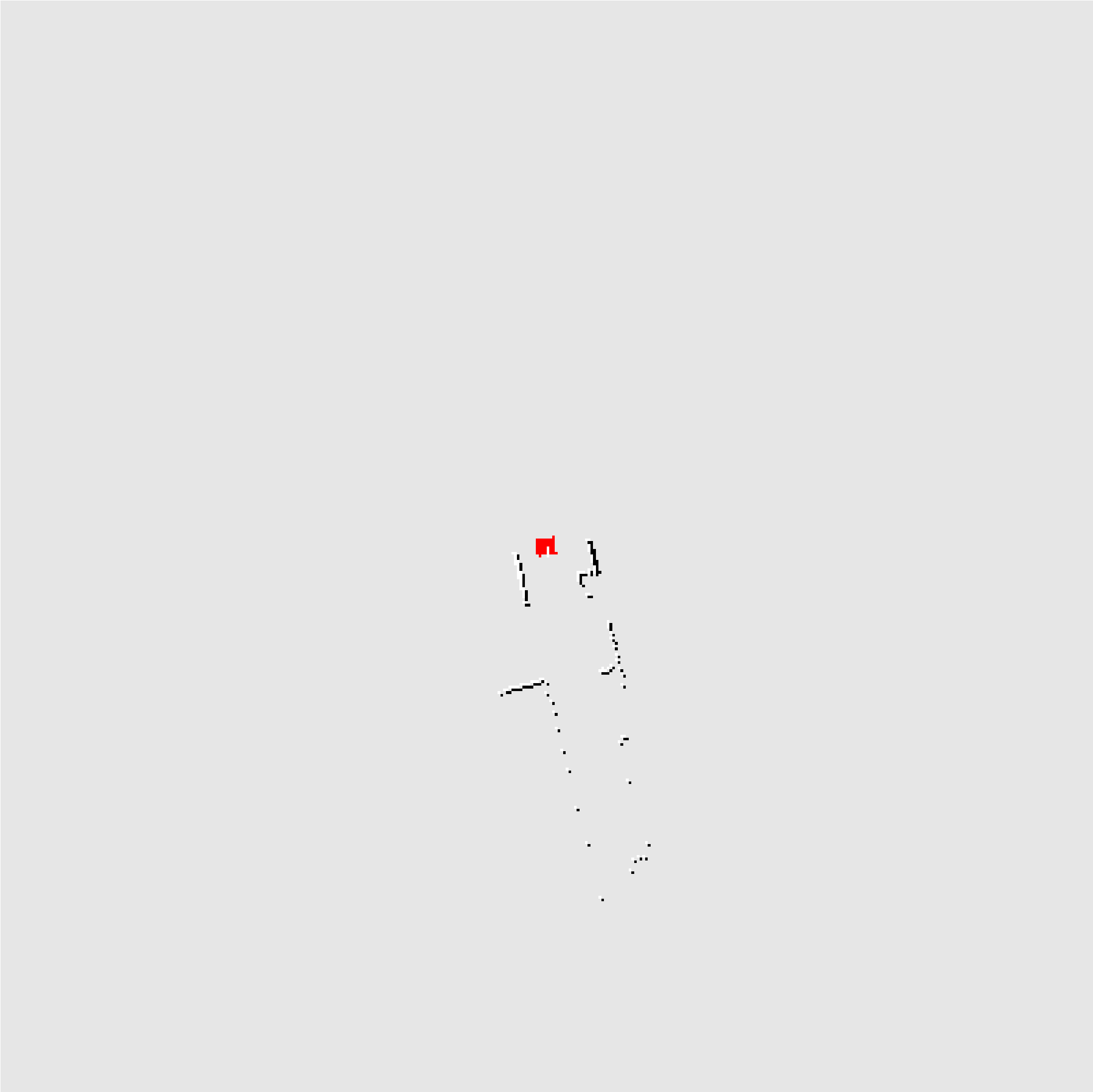
- We do not know the map or our location
- State consists of position AND map!
- Main techniques: Kalman filtering (Gaussian HMMs) and particle methods

DP-SLAM, Ron Parr



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

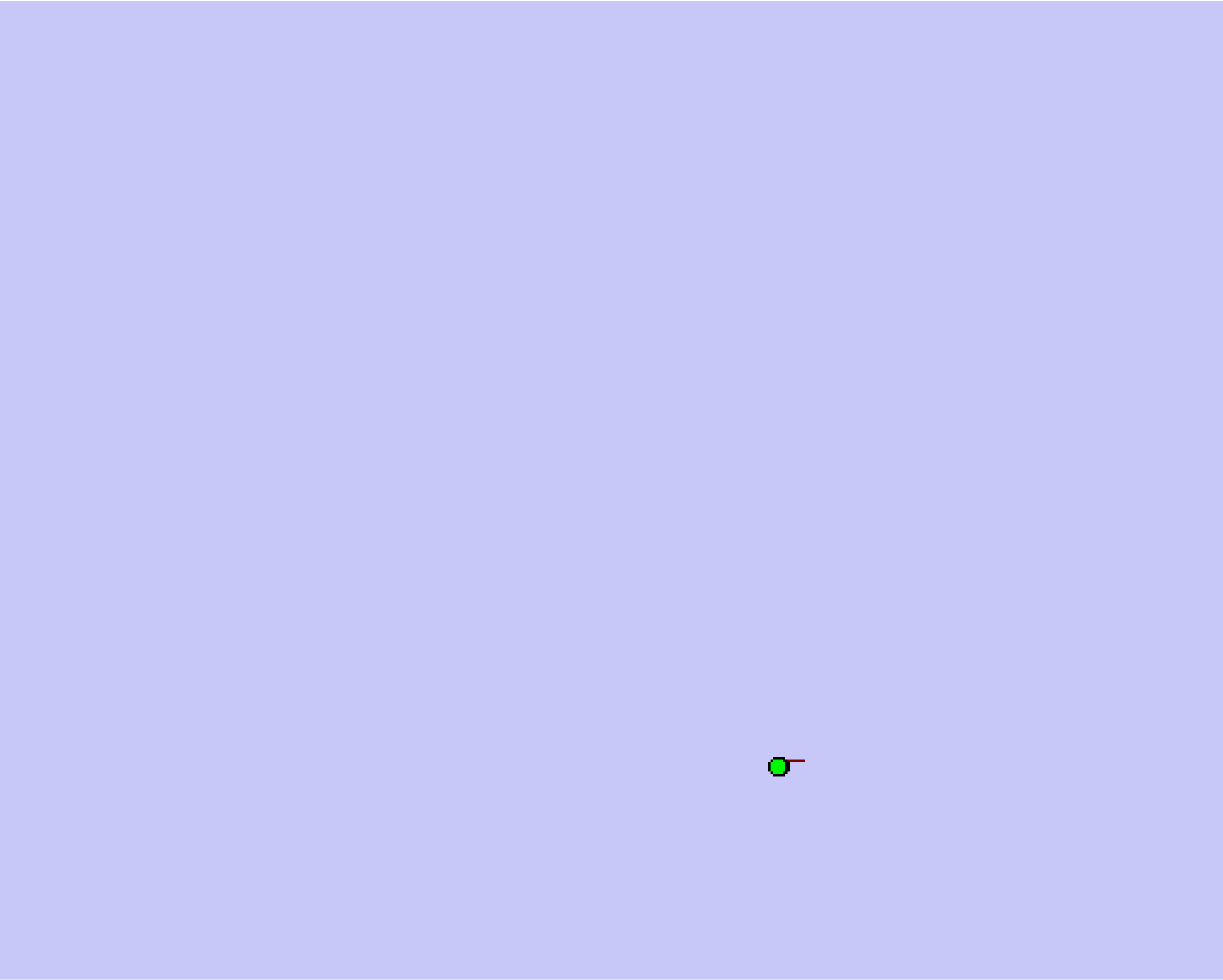
Particle Filter SLAM – Video 1



[Sebastian Thrun, et al.]

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

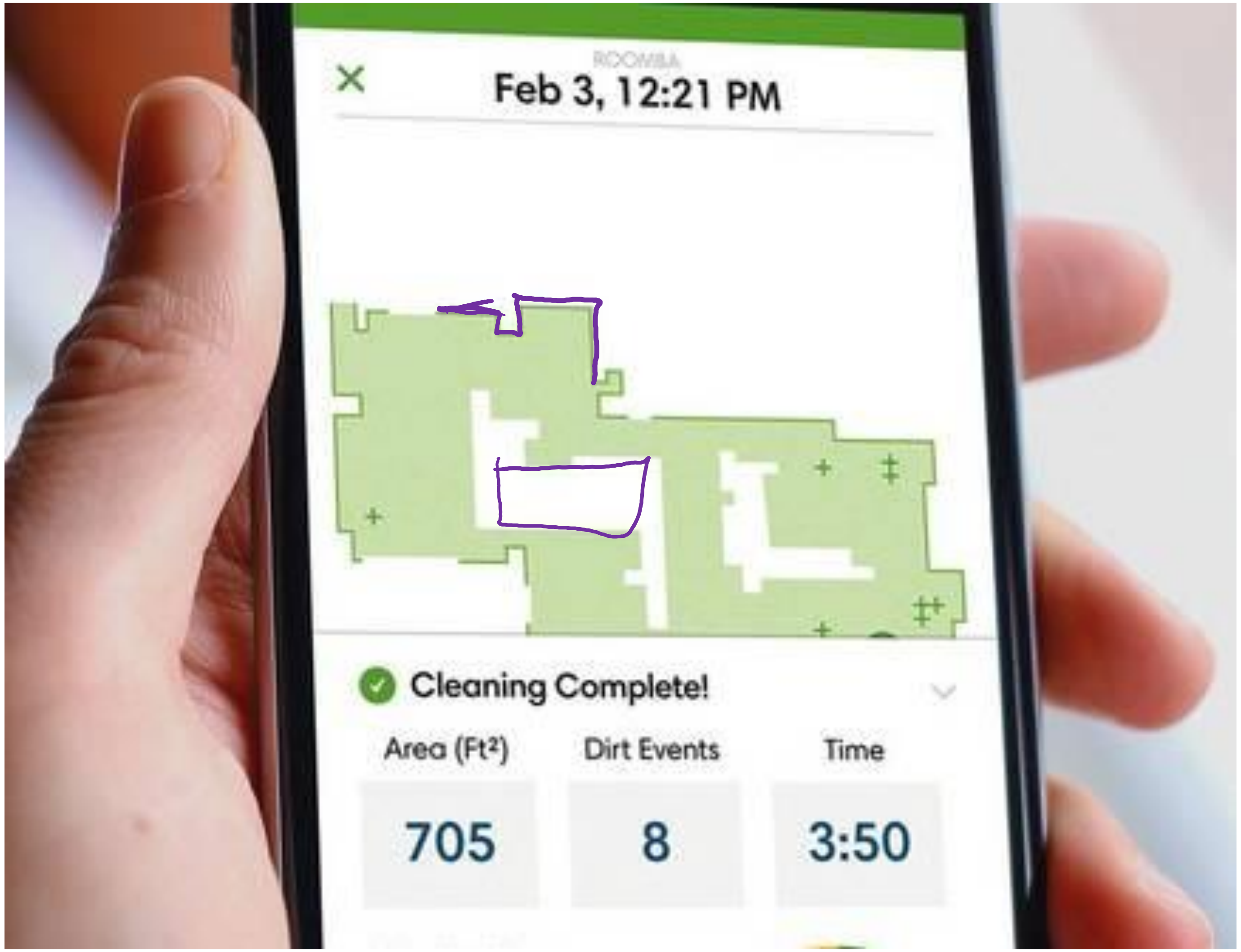
Particle Filter SLAM – Video 2



[Dirk Haehnel, et al.]

[Demo: PARTICLES-SLAM-fastslam.avi]

SLAM



https://www.irobot.com/