# CSEP 573: Artificial Intelligence

Spring 2014

## Hidden Markov Models & Particle Filtering

Ali Farhadi

Many slides adapted from Dan Weld, Pieter Abbeel, Dan Klein, Stuart Russell, Andrew Moore & Luke Zettlemoyer

# Outline

- Probabilistic sequence models (and inference)
  - Probability and Uncertainty – Preview
  - Markov Chains
  - Hidden Markov Models
  - Exact Inference
  - Particle Filters

# Example

- A robot move in a discrete grid
  - May fail to move in the desired direction with some probability
- Observation from noisy sensor at each time
  - Is a function of robot position
- Goal: Find the robot position (probability that a robot is at a specific position)
- Cannot always compute this probability exactly

➔ Approximation methods
Here: Approximate a distribution by sampling

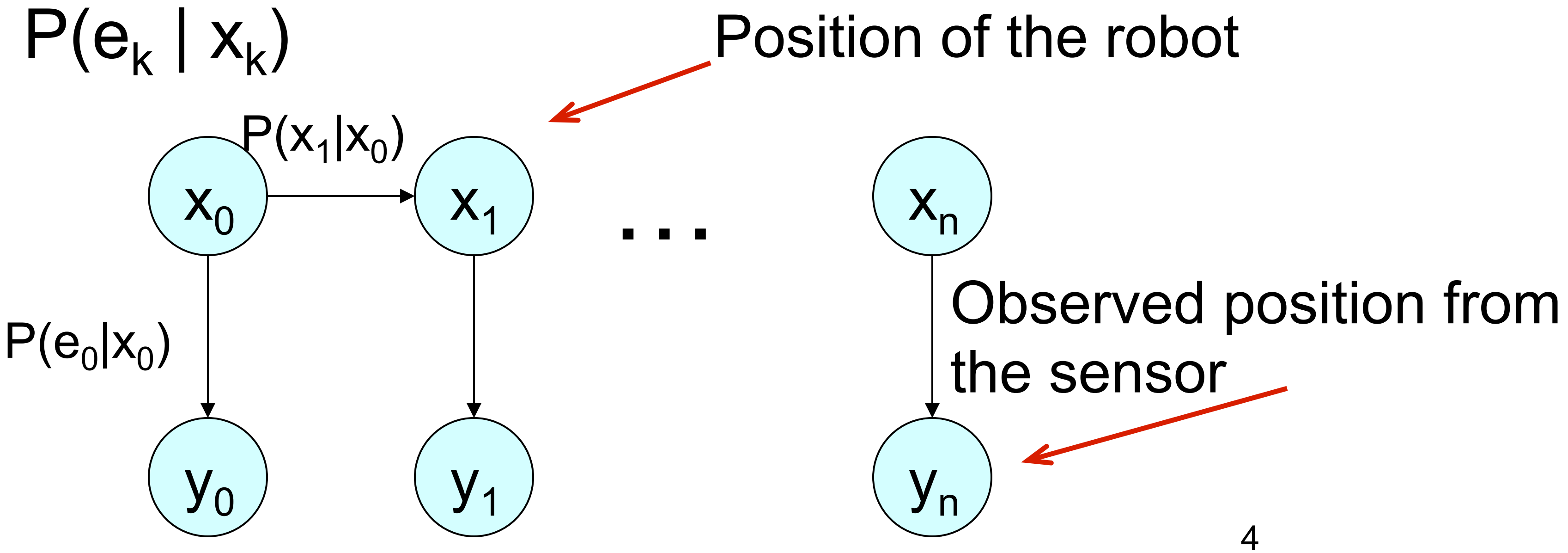
# Hidden Markov Model

- State Space Model
  - Hidden states: Modeled as a Markov Process

$P(x_0)$, $P(x_k \mid x_{k-1})$

  - Observations: $e_k$

$P(e_k \mid x_k)$



# Exact Solution: Forward Algorithm

- Filtering is the inference process of finding a distribution over $X_T$ given $e_1$ through $e_T$ : P( $X_T$ | $e_{1:t}$ )
- We first compute P( $X_1$ | $e_1$ ): $P(x_1|e_1) \propto P(x_1) \cdot P(e_1|x_1)$
- For each t from 2 to T, we have P( $X_{t-1}$ | $e_{1:t-1}$ )
- 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($X_t$ | $e_{1:t-1}$ , $e_t$) = P( $X_t$ | $e_{1:t}$ )

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

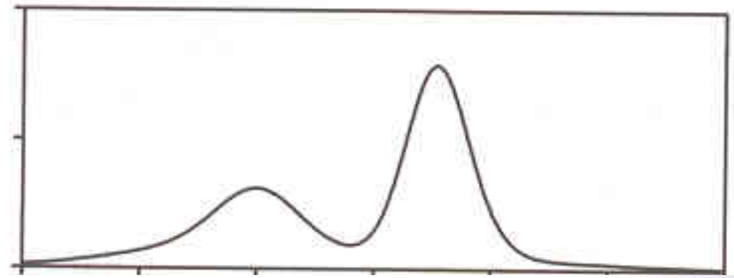
# Approximate Inference:

- Sometimes |X| is too big for exact inference
  - |X| may be too big to even store B(X)
  - E.g. when X is continuous
  - $|X|^2$ may be too big to do updates

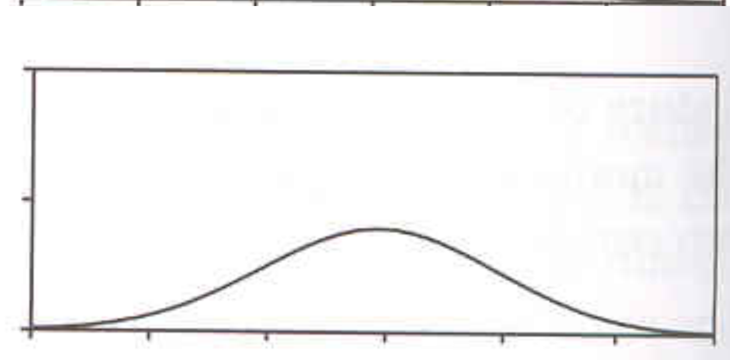
- Solution: approximate inference by sampling
- How robot localization works in practice

# Why Sampling?

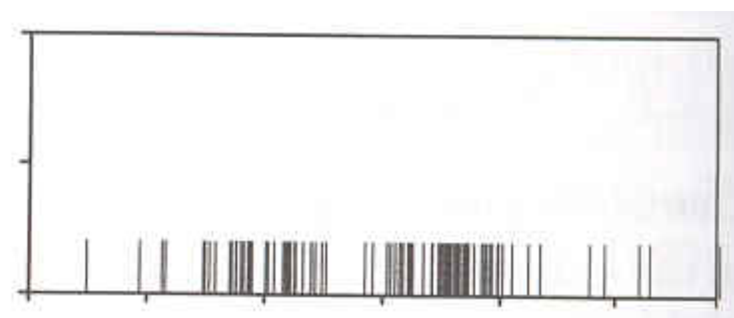
- Goal: Approximate the original distribution:



- Approximate with Gaussian distribution

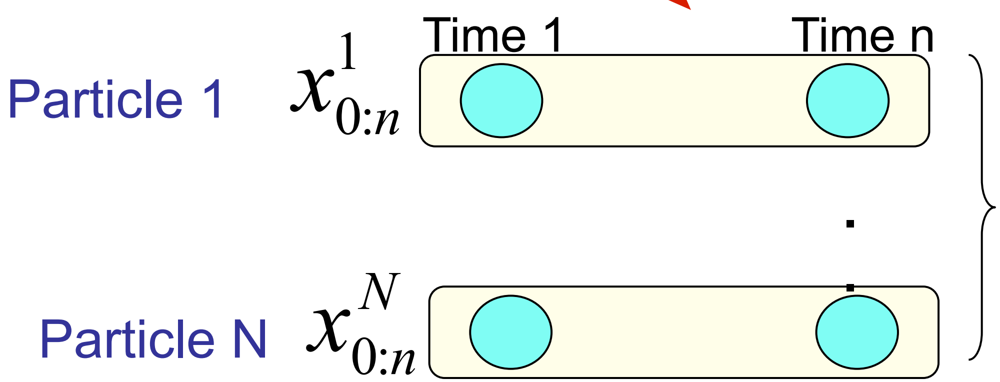


- Draw samples from a distribution close enough to the original distribution
- Here: A general framework for a sampling method



# Approximate Solution: Perfect Sampling

Robot path till time n




Assume we can sample from the original distribution

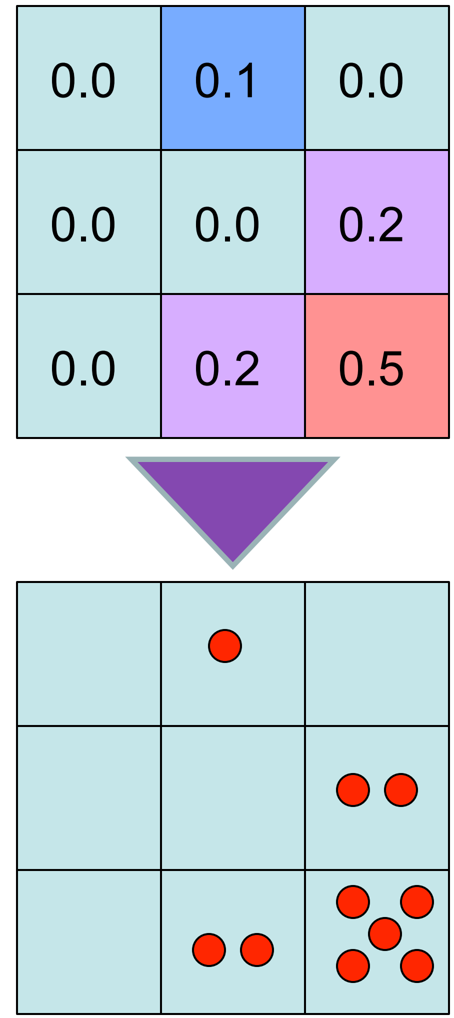
$$p(x_{0:n} \mid y_{0:n})$$

$$P(x_{0:n} \mid y_{0:n}) = \frac{1}{N}\left( \text{Number of samples that match with query} \right)$$

Converges to the exact value for large N

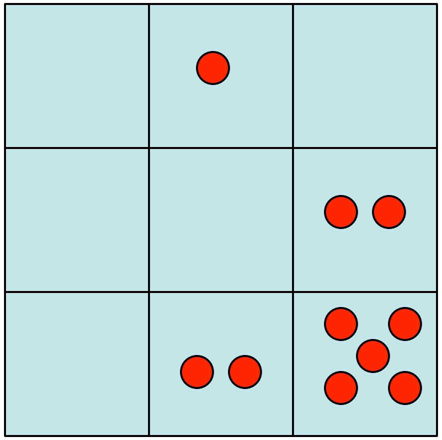
# Approximate Inference: Particle Filtering

- 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
- 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 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)

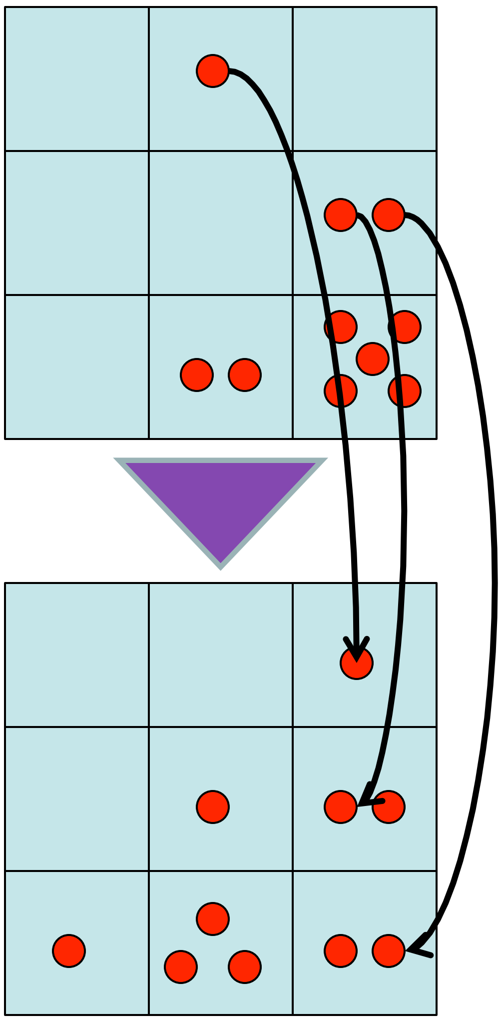
# Particle Filtering: Elapse Time

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

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

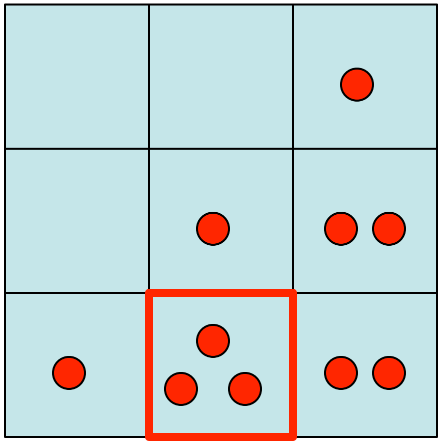
  - This is like prior sampling – samples' frequencies reflect the transition probs
  - Here, most samples move clockwise, but some move in another direction or stay in place

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



# Particle Filtering: Observe

- How handle noisy observations?
- Suppose sensor gives red reading?



# Particle Filtering: Observe

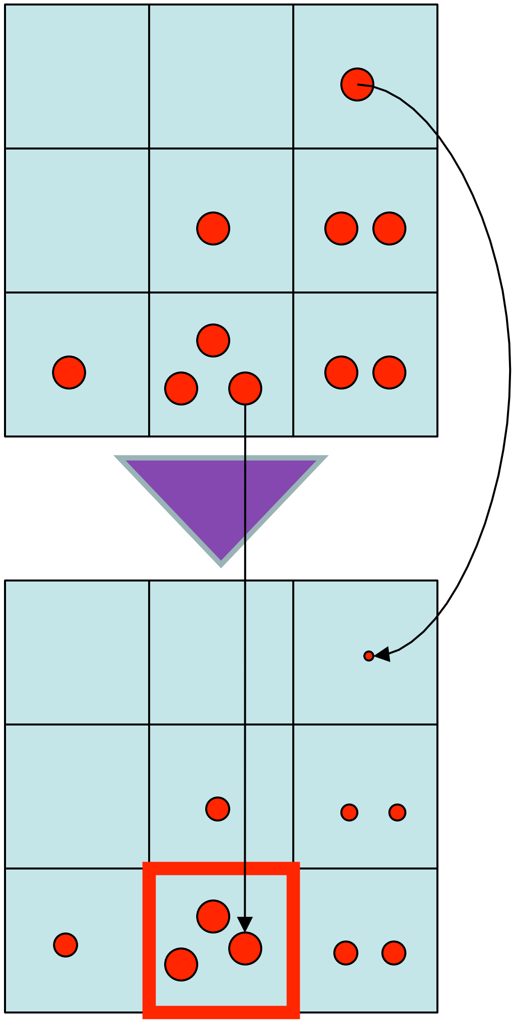
Slightly trickier:

- We don't sample the observation, we fix it
- Instead: ***downweight samples*** based on the evidence (form of likelihood weighting)

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

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

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

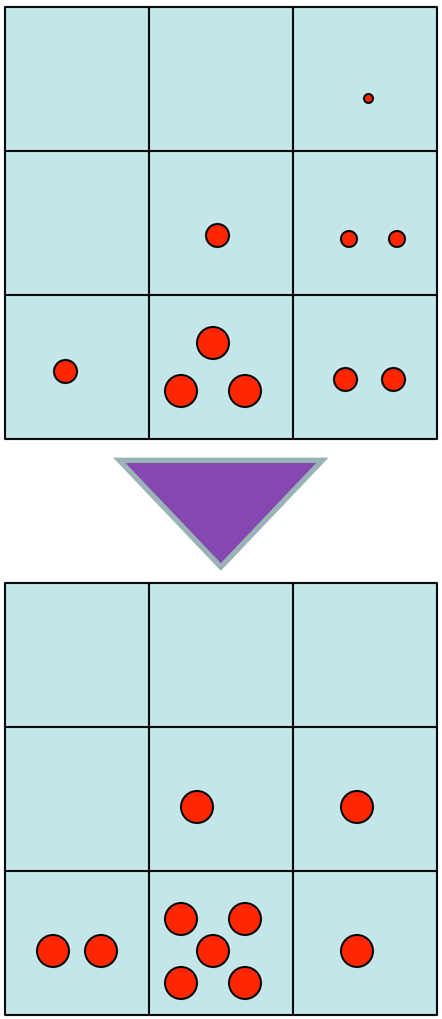


# Particle Filtering: Resample

- 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

New 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 Filtering Summary

- Represent current belief P(X | evidence to date) as set of n samples (actual assignments X=x)
- For each new observation e:
  1. Sample transition, once for each current particle x

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

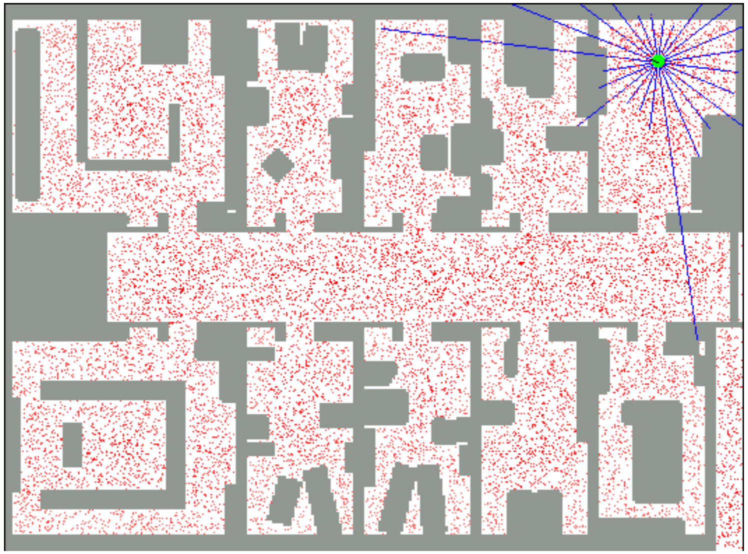
  2. For each new sample x', compute importance weights for the new evidence e:

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

  3. Finally, normalize by resampling the importance weights to create N new 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

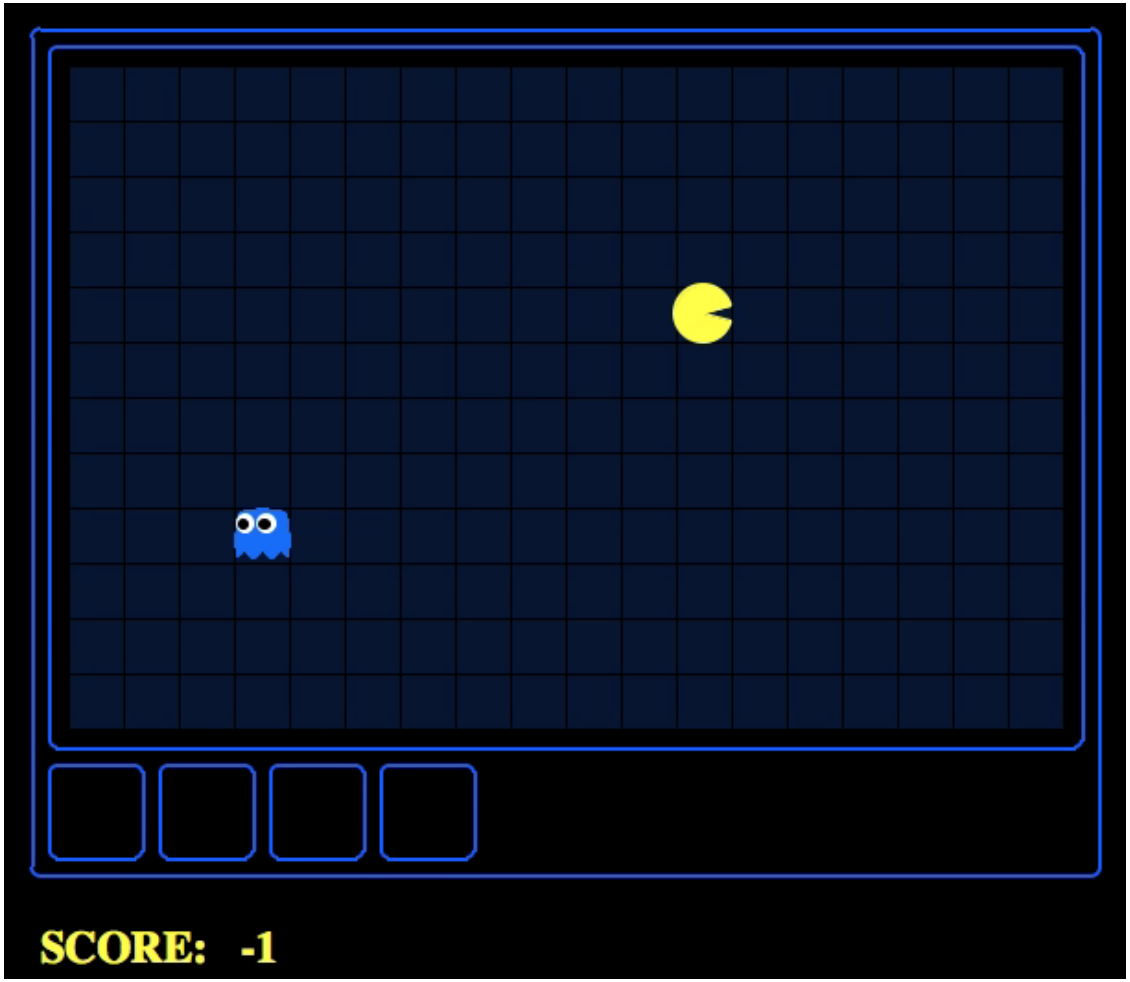


# Robot Localization

QuickTime™ and a
GIF decompressor
are needed to see this picture.

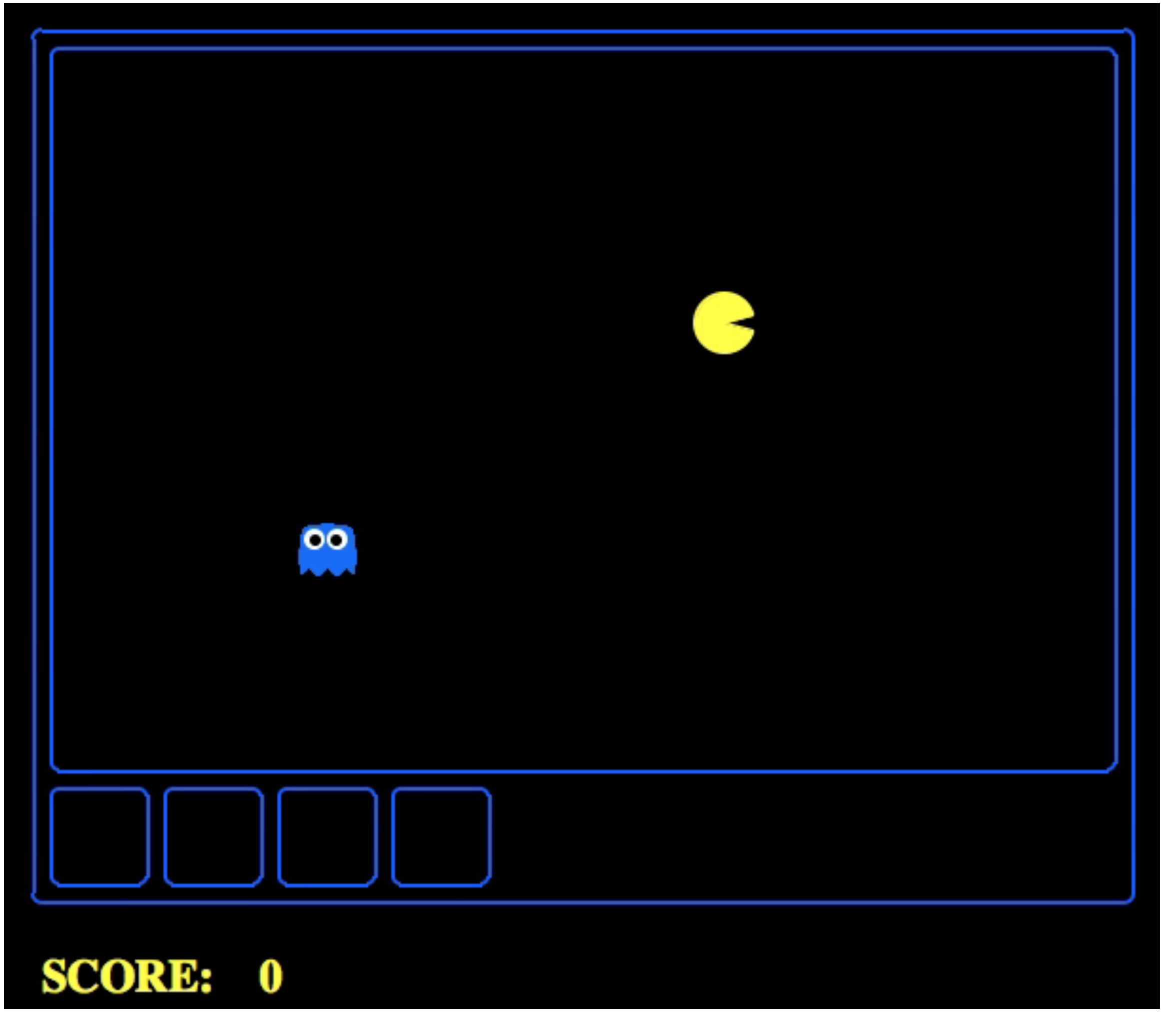
# Which Algorithm?

Exact filter, uniform initial beliefs



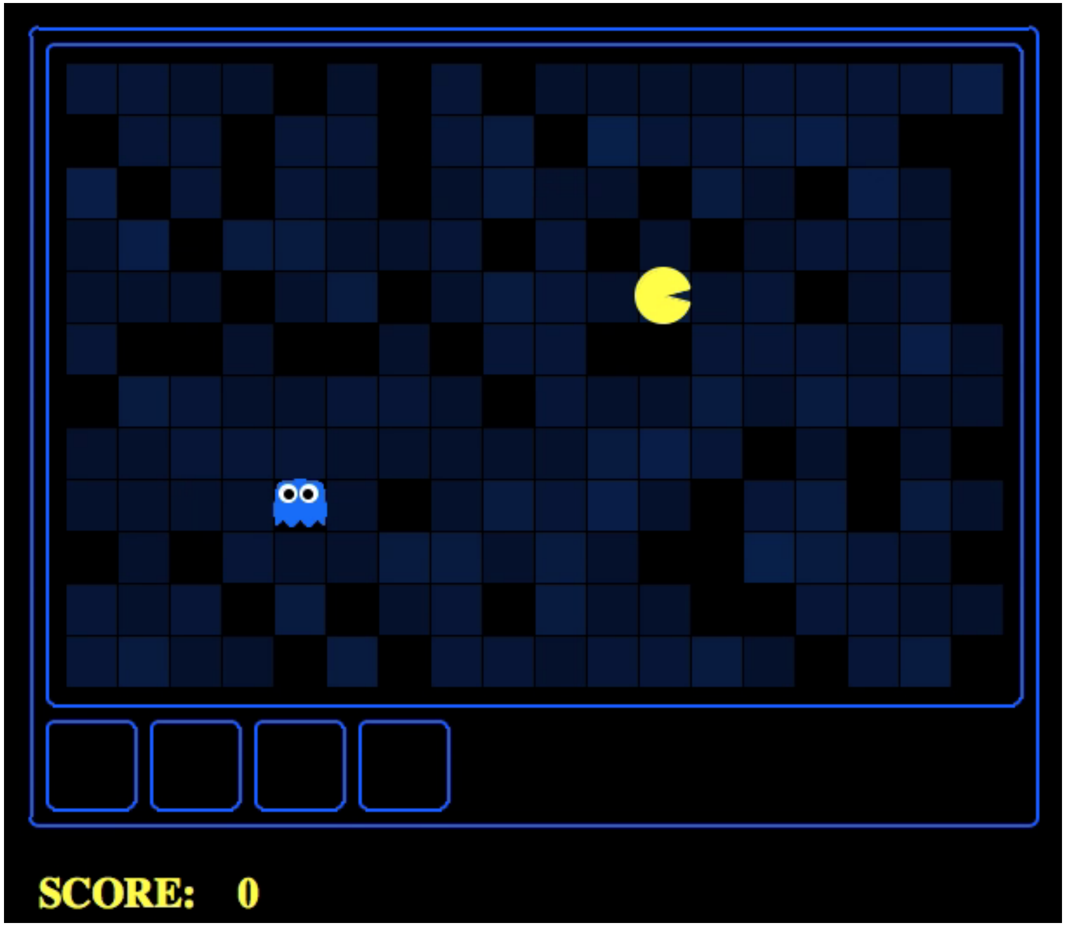
# Which Algorithm?

Particle filter, uniform initial beliefs, 25 particles



# Which Algorithm?

Particle filter, uniform initial beliefs, 300 particles



# P4: Ghostbusters

- Plot: Pacman's grandfather, Grandpac, learned to hunt ghosts for sport.
- He was blinded by his power, but could hear the ghosts' banging and clanging.
- Transition Model: All ghosts move randomly, but are sometimes biased
- Emission Model: Pacman knows a "noisy" distance to each ghost