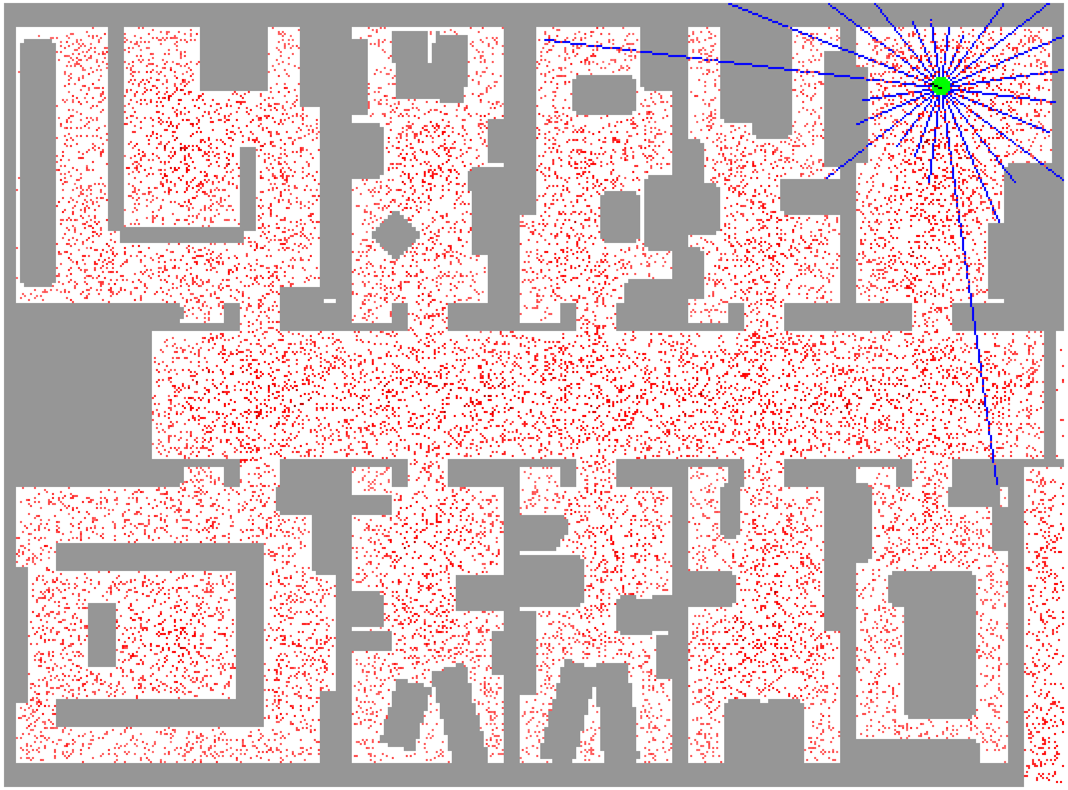
Player/Stage Localization Approach: Monte Carlo Localization

- Based on techniques developed by Fox, Burgard, Dellaert, Thrun (see handout of AAAI'99 article)



(Movie illustrating approach)

Two types of localization problems

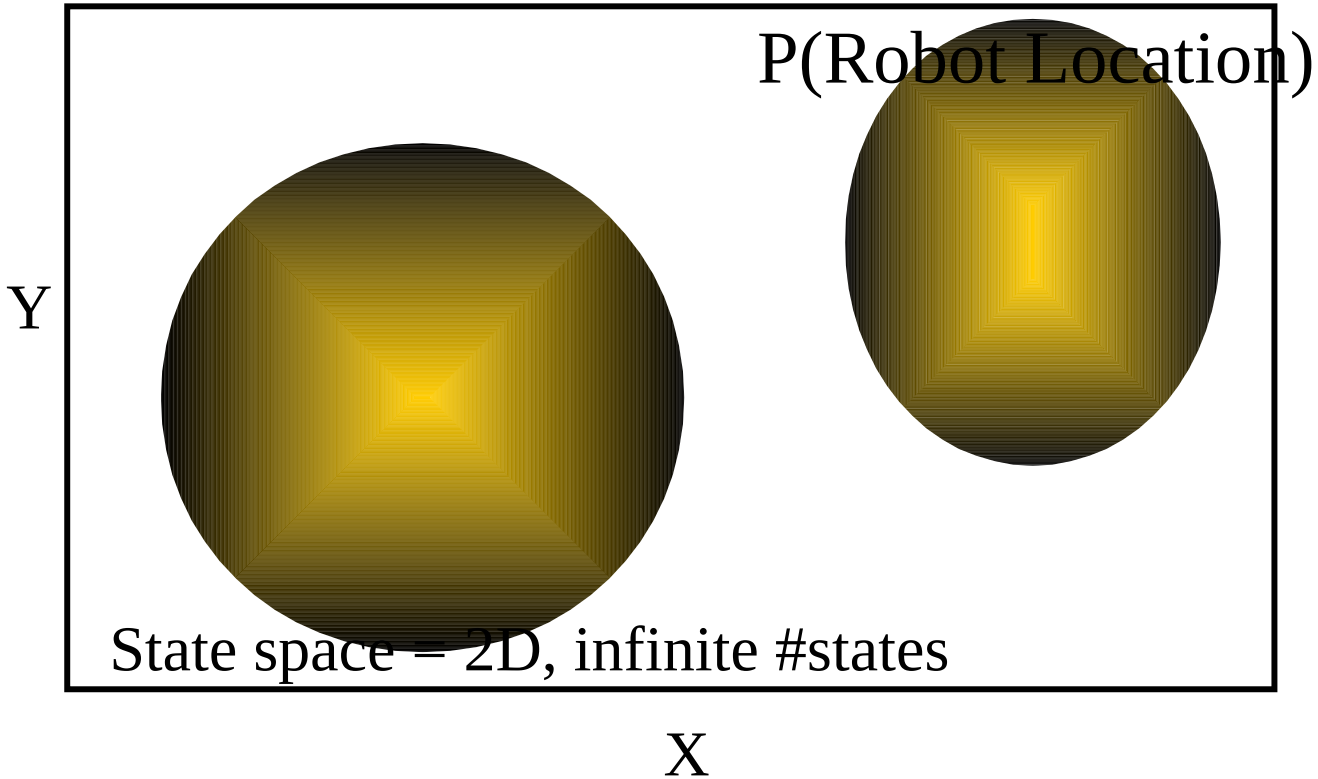
- "Global" localization – figure out where the robot is, but we don't know where the robot started
  - Sometimes called the "hijacked robot problem"

- "Position tracking" – figure out where the robot is, given that we know where the robot started

The Monte Carlo Localization approach of Fox, et al (which is in Player/Stage) can address both problems

Markov Localization

- Key idea: compute a probability distribution over all possible positions in the environment.
  - This probability distribution represents the likelihood that the robot is in a particular location.



Slide adapted from Dellaert presentation “19-Particles.ppt”

Side note: What does "Markov" mean?

- "Markov" means the system obeys the "Markov Property"

- "Markov Property": the conditional probability of the future state is dependent only on the current state. It is independent of the past states.

- For the purposes of robot localization →
  - Future sensor readings are conditionally independent of past readings, given the true current position of the robot.

- Means we don't have to save all the prior sensor data and apply it each time we update beliefs on the robot's location.

Markov Localization (con't.)

- Let $l = \langle x, y, \theta \rangle$ represent a robot position in space
- *Bel*(l) represents the robot's belief that it is at position l
  - *Bel*(l) is a probability distribution, centered on the correct position
  - As the robot moves, *Bel*(l) is updated

- Two probabilistic models used to update *Bel*(l)
  - **Action (or motion) model:** represents movements of robot

$$Bel(l) \neg \int P(l \mid l', a)\; Bel(l')\; dl'$$

"Probability that an action *a* in position l' moves the robot to position l, times the likelihood the robot is in position l', integrated over all possible ways robot could have reached position l"

  - **Perception (or sensing) model:** represents likelihood that robot senses a particular reading at a particular position (related to our discussion last class)

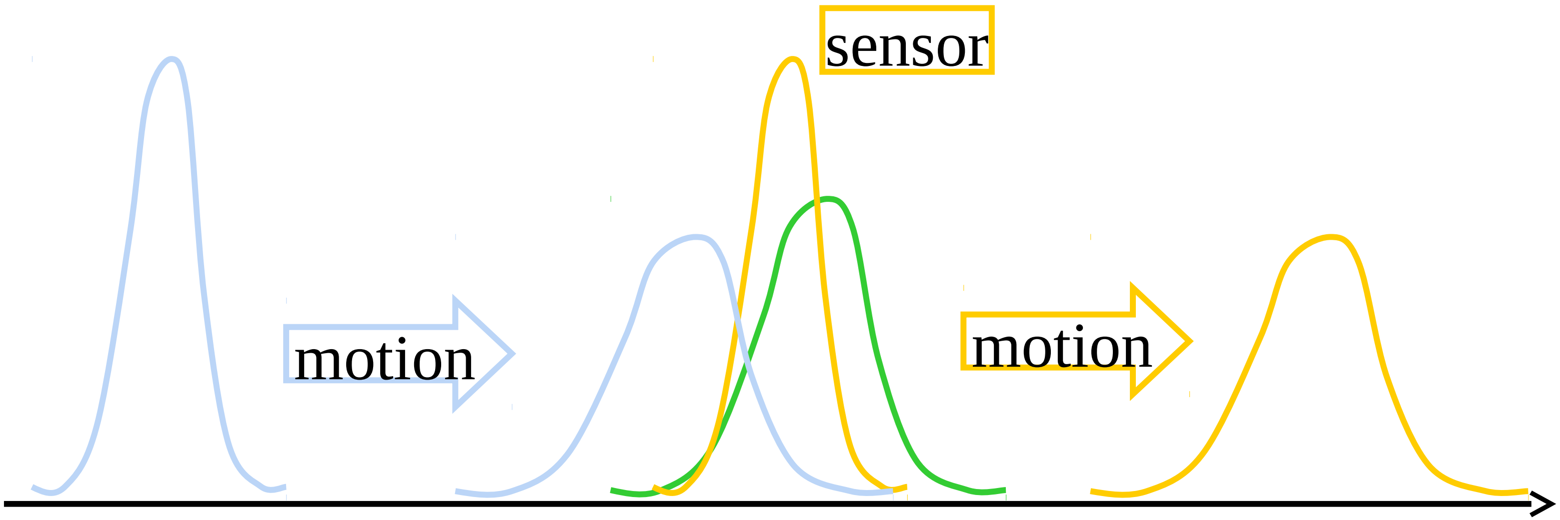
$$Bel(l) \neg \; \alpha \; P(s \mid l)\; Bel(l)$$

"Probability that robot will perceive *s*, given that the robot is in position l, times the likelihood the robot is in position l"

Can implement Markov Localization in different ways

Very commonly used approach:

- Kalman Filter – estimates state from a series of incomplete, noisy measurements (e.g., sensor readings)
  - At each point in time, a new estimate of robot's position is made, using action (sometimes called "motion") model and sensor model
  - Maintains a single estimate of robot's position



(Also, other Markov Localization approaches we won't go into here…)

Slide adapted from Dellaert presentation "19-Particles.ppt"

Different Concept for implementing Markov Localization: Monte Carlo Localization using Particle Filtering

- Maintain multiple estimates of robot's location
- Track possible robot positions, given all previous measurements
- Key idea: represent the belief that a robot is at a particular location by a set of "samples", or "particles"

Represent *Bel*(*l*) by set of N weighted, random samples, called *particles*:

$$S = \{s_i \mid i = 1..N\}$$

where a sample, s_i, is of the form: $\langle\langle x, y, \theta\rangle, p\rangle$

Here, $\langle x, y, \theta\rangle$ represents robot's position (just like before)

p represents a weight, where sum of all p's is 1 (analogous to discrete probability)

Side Note: What does “Monte Carlo” mean?

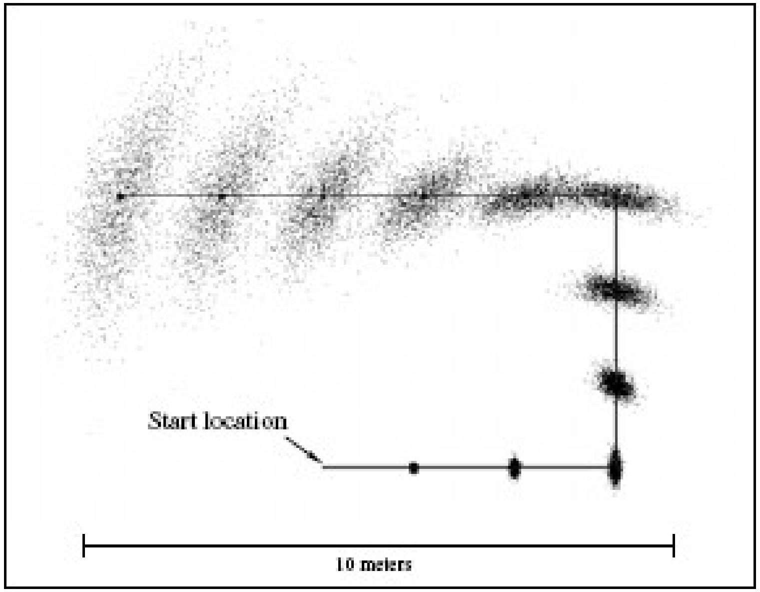
- Refers to techniques that are stochastic / random / non-deterministic

- Used in lots of modeling and simulation approaches
  - Particularly useful when the system has significant uncertainty in the inputs (e.g., robot localization!)

Updating beliefs using Monte Carlo Localization (MCL)

- As before, 2 models: Action (Motion) Model, Perception (Sensing) Model
- Robot Motion Model:
  - When robot moves, MCL generates N new samples that approximate robot's position after motion command.
  - Each sample is generated by randomly drawing from previous sample set, with likelihood determined by *p* values.
  - For sample drawn with position l', new sample l is generated from $P(l \mid l', a)$
  - *p* value of new sample is 1/N

Sampling-based approximation of position belief for non-sensing robot



(From Fox, et al, AAAI-99)

Updating beliefs using Monte Carlo Localization (MCL) (con't.)

- Robot Sensing Model:
  - Re-weight sample set, according to the likelihood that robot's current sensors match what would be seen at a given location
  - Let < *l*, *p*> be a sample.
  - Then, $p \leftarrow \alpha\, P(s \mid l)$

Here, *s* is the sensor measurement;

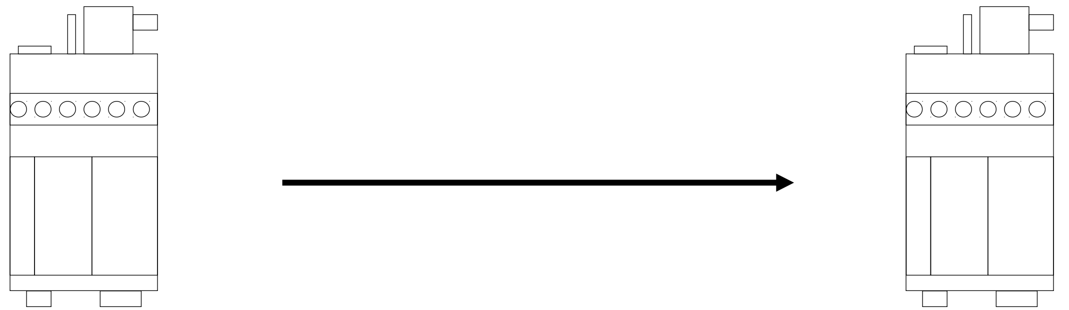
α a normalization constant to enforce

the sum of *p's* equaling 1

- After applying Motion model and Sensing model:
  - Resample, according to latest weights
  - Add a few uniformly distributed, random samples
    - Very helpful in case robot completely loses track of its location

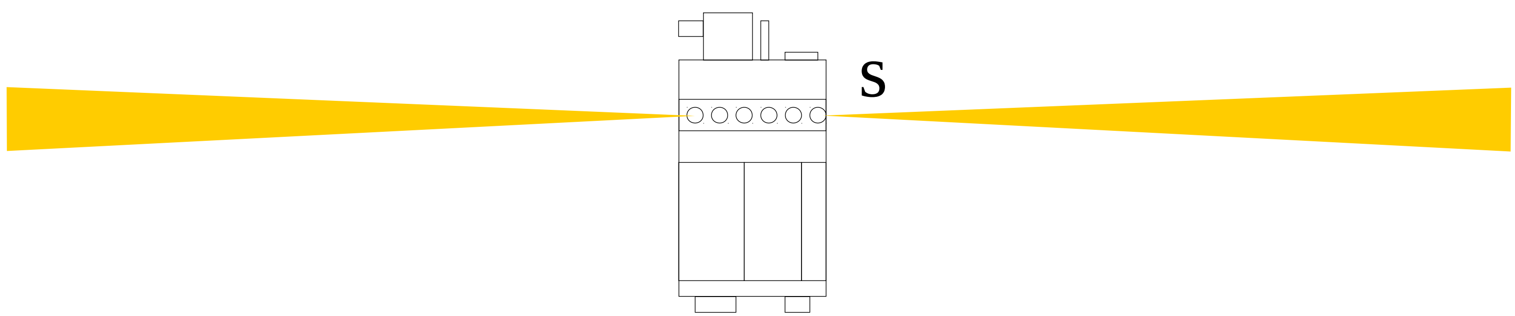
Side Note: Common Terminology

- Prediction Phase: Applying motion model



$<x,y,\theta>_{t-1}$ $<x,y,\theta>_{t}$

- Measurement Phase: Applying sensor model



$<x,y,\theta>_{t}$

Slide adapted from Dellaert presentation “19-Particles.ppt”

Adapting the Size of the Sample Set

- Number of samples needed to achieve a desired level of accuracy varies dramatically depending on the situation
  - During global localization: robot is ignorant of where it is → need lots of samples
  - During position tracking: robot's uncertainty is small → don't need as many samples

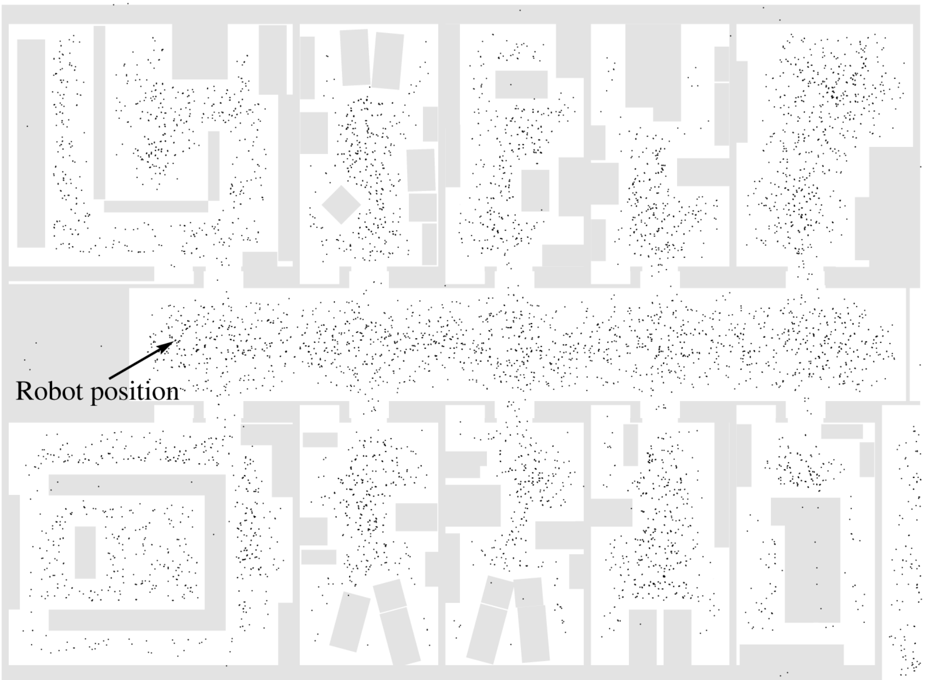
- MCL determines sample size "on the fly"
  - Compare P(l) and P(l | s) (I.e., belief before and after sensing) to determine sample size
  - The more divergence, the more samples that are kept

What sensor to use for localization?

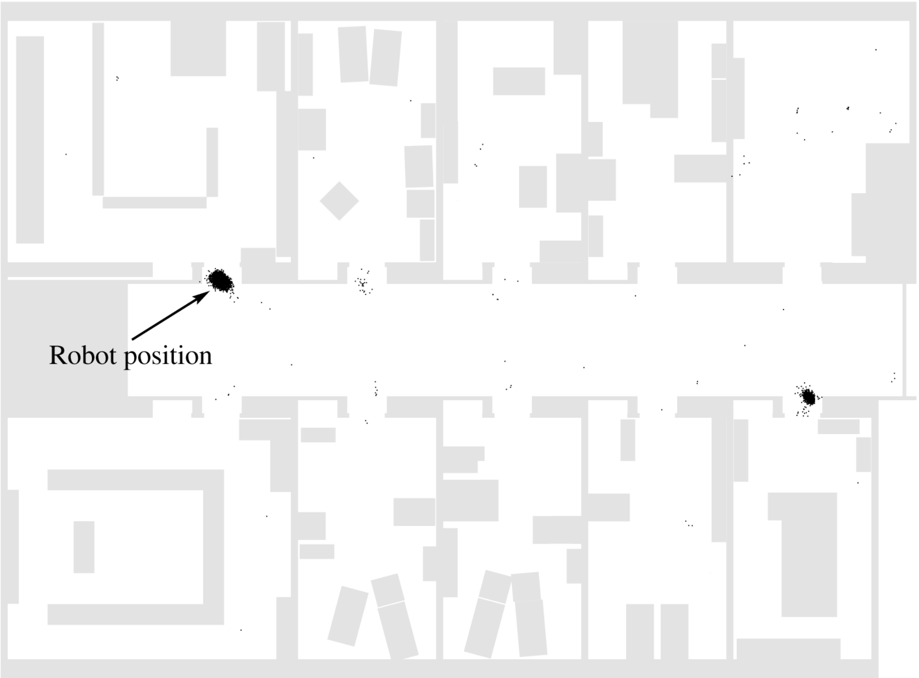
- Can work with:
  - Sonar
  - Laser
  - Vision
  - Radio signal strength

Example Results

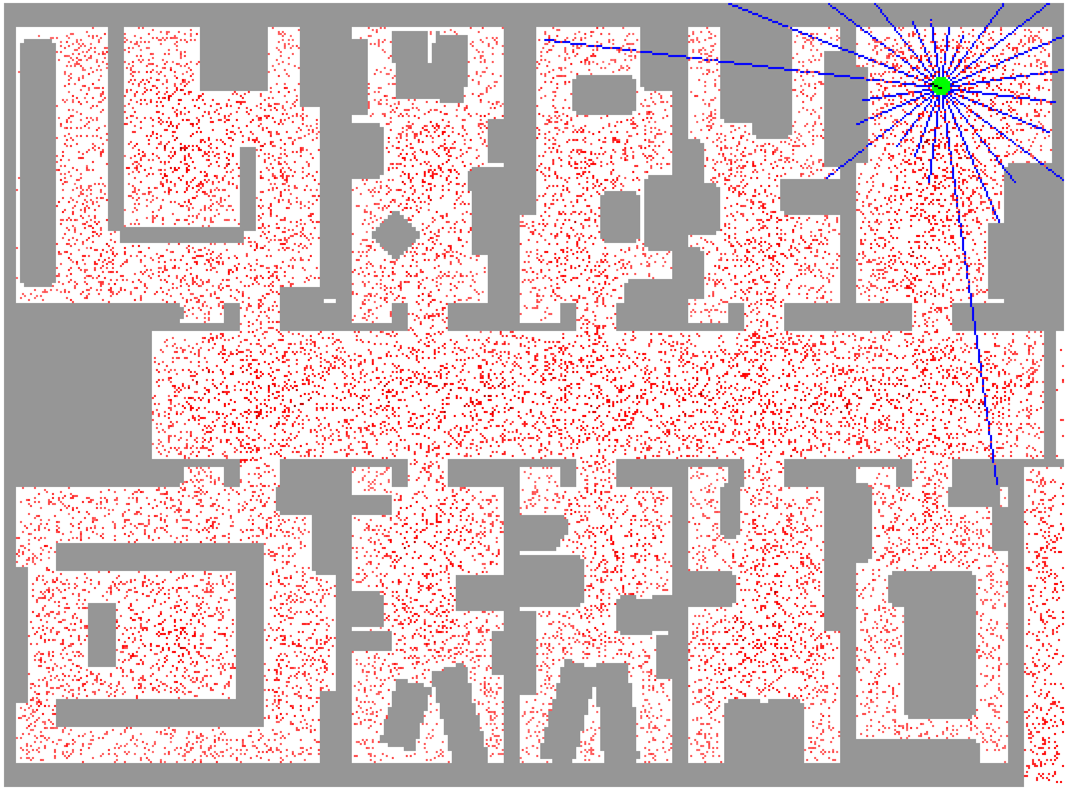
Initially, robot doesn't know where it is (see particles representing possible robot locations distributed throughout the environment)



After robot moves some, it gets better estimate (see particles clustered an a few areas, with a few random particles also distributed around for robustness)



Return to Movie



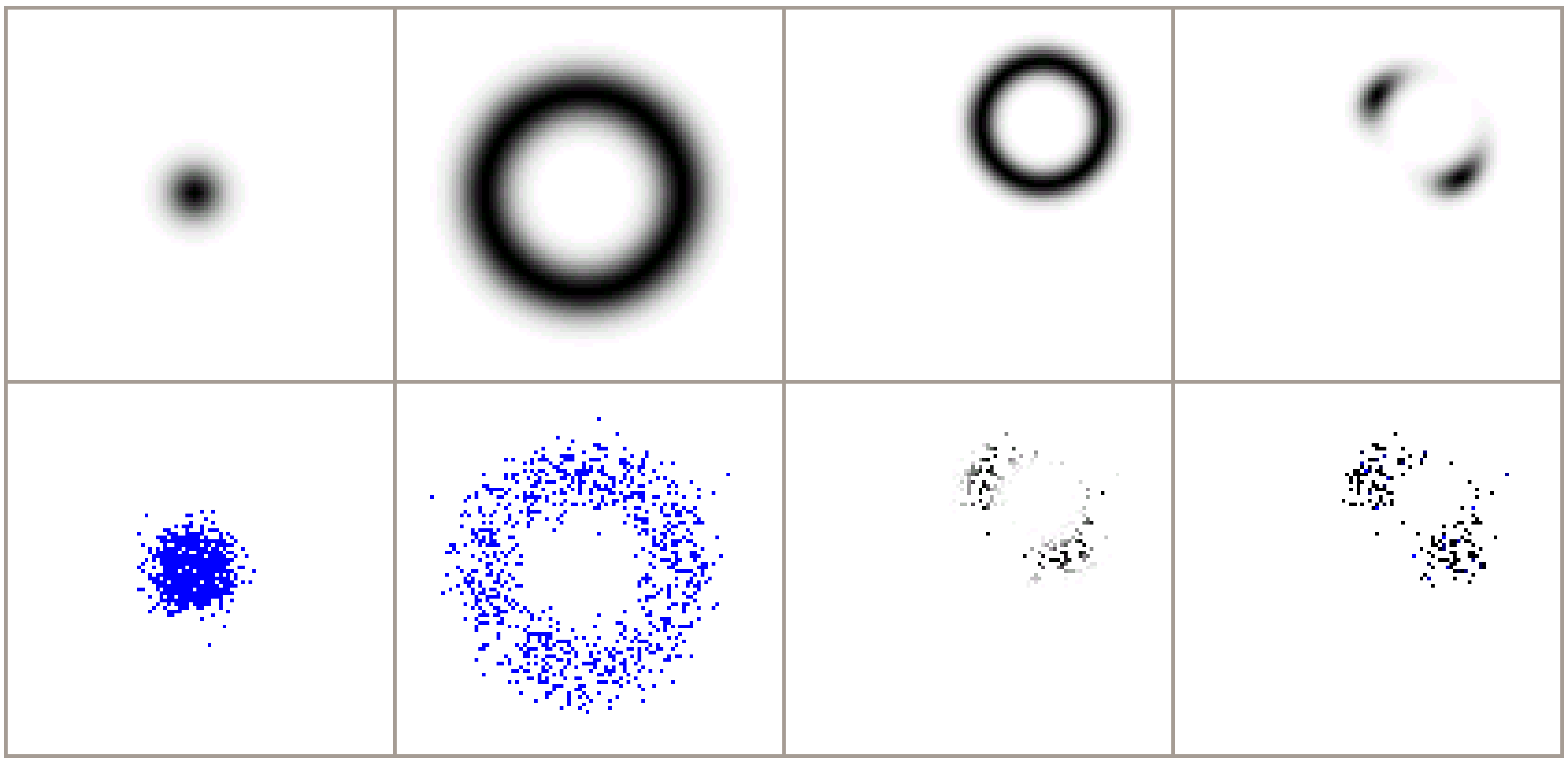
More movies

- Dieter Fox movie: MCL using Sonar

- Dieter Fox movie: MCL using Laser

Summarizing the process: Particle Filtering

S_{t-1} S'_t weighted S'_t S_t



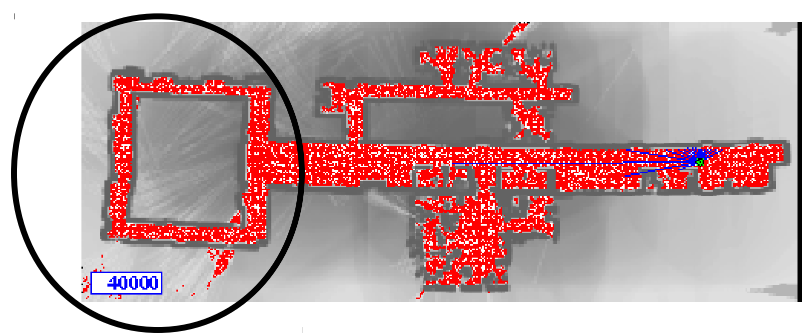
Predict ReWeight Resample

Slide adapted from Dellaert presentation “19-Particles.ppt”

Mapping is much easier if robot can localize

- SLAM: Simultaneous Localization And Mapping
- If robot knows where it is, then it can merge its sensor measurements as it moves, effectively building a map
- But, lots of details, like "closing the loop" in maps that are very important, and challenging
  - "Closing the loop": splicing together pieces of the map that represent the same part of the environment, but which are explored by robot at different times

Example:



- We won't go into details here…