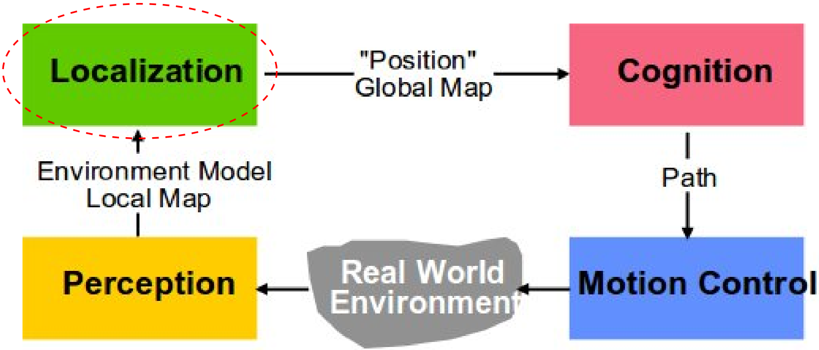
Robotics and Autonomous Systems

Lecture 23: Localization

Terry Payne

Department of Computer Science
University of Liverpool

Today



Localization
"Position"
Global Map
Cognition
Environment Model
Local Map
Path
Perception
Real World
Environment
Motion Control

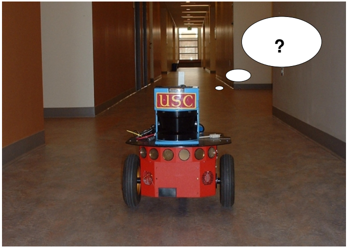
Acknowledgement

- The material on particle filters is heavily based on:

  D. Fox, S. Thrun, F. Dellaert, and W. Burgard, Particle filters for mobile robot localization, in A. Doucet, N. de Freitas and N. Gordon, eds., Sequential Monte Carlo Methods in Practice. Springer Verlag, New York, 2000.

Localization

- We started this course with three questions:



  - Where am I ?
  - Where am I going ?
  - How do I get there ?
- We are now at a point where we can answer the first of these.

Localization

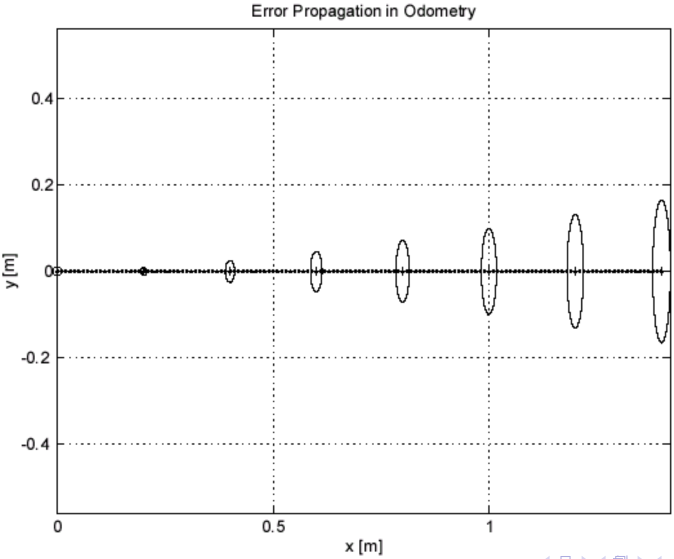
- The basic localization task is to compute current location and orientation (pose) given observations.
  - What constitutes a pose depends on what kind of map we have.
  - But roughly speaking it is (x_I, y_I, θ).
  - The same things we worried about in the motion model
- Do we need to do any more than just use odometry?
  - After all, that sort of worked in the lab.

Why localization?

- In general odometry doesn't hold up well over long distances.
- Range error: integrated path length (distance) of the robots movement
  - Sum of the wheel movements
- Turn error: similar to range error, but for turns
  - Difference of the wheel motions
- Drift error: difference in the error of the wheels leads to an error in the robot's angular orientation.
- Over long periods of time, turn and drift errors far outweigh range errors!

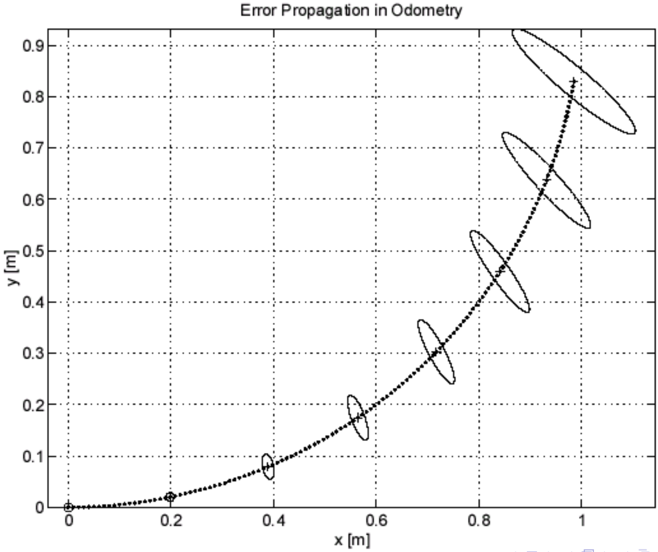
Problems with odometry

- A simple error model based on the kinematics predicts:



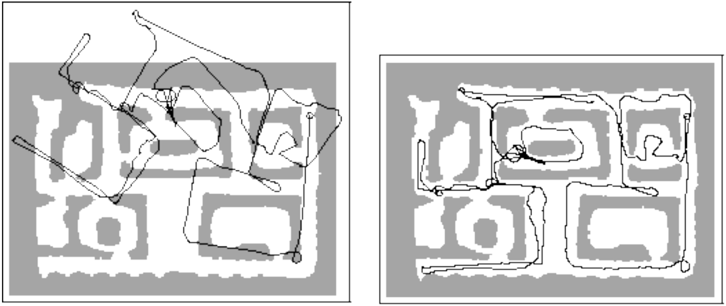
Problems with odometry

- Which is worse when we turn:



Problems with odometry

- Leading to:



- Images from Dieter Fox in his CMU days.

What else?

- If odometry alone doesn't help, what about GPS?
- Non-military GPS is not accurate enough to work on its own.
  - Thrun: Sometimes GPS places you on the wrong side of the road, sometimes off the road completely.

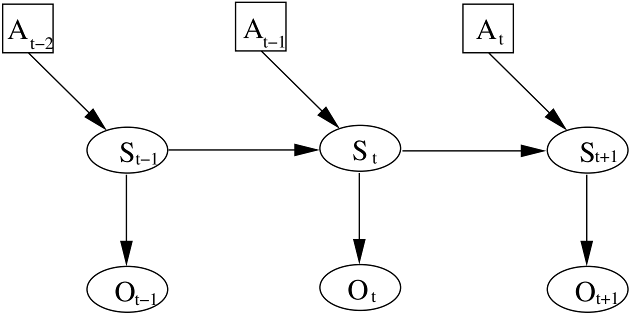
- Doesn't tend to work well indoors.

What else?

- Instead we try to use sensor data to identify where we are on a map.
- It is tempting to try and triangulate.
- But doing this is too prone to error.
  - Sensor noise.
  - Sensor aliasing.
- You get better results if you:
  - Combine data from multiple sensors.
  - Take into account previous estimates of where the robot is.

Bayesian filter

- General schema:



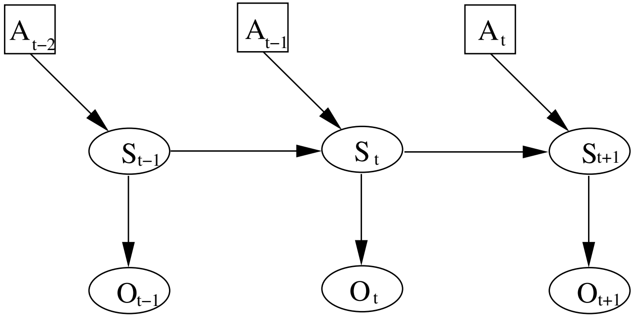
- Here A is action, S is pose and O is observation.
- The point is that position at one time depends on position at the previous time.

The localization problem(s)

- There are a number of flavors of localization:
  - Position tracking
  - Global localization
  - Kidnapped robot problem
  - Multi-robot localization
- All are hard, but variations of the technique we will look at helps to solve all of them.

Bayesian filter

- General schema (again):



- Here *A* is action, *S* is pose and *O* is observation.
- The point is that position at one time depends on position at the previous time.

Bayesian filter

- The pose at time t depends upon:
  - The pose at time $t-1$, and
  - The action at time $t-1$.
- The pose at time t determines the observation at time t.
- So, if we know the pose we can say what the observation is.

Bayesian filter

- The pose at time t depends upon:
  - The pose at time $t-1$, and
  - The action at time $t-1$.
- The pose at time t determines the observation at time t.
- So, if we know the pose we can say what the observation is.

- But this is backwards...
- To help us out of this bind we need to bring in probabilities (as mentioned before they are also helpful because sensor data is noisy).

Bayesian filter

- The technique we will use for localization is a form of Bayesian filter.
- The key idea is that we calculate a probability distribution over the set of possible poses.
- That is we compute the probability of each pose that is in the set of all possible poses.
- We do this informed by all the data that we have.

Bayesian filter

- We call the probability that we calulate the belief.
- We denote the belief by:

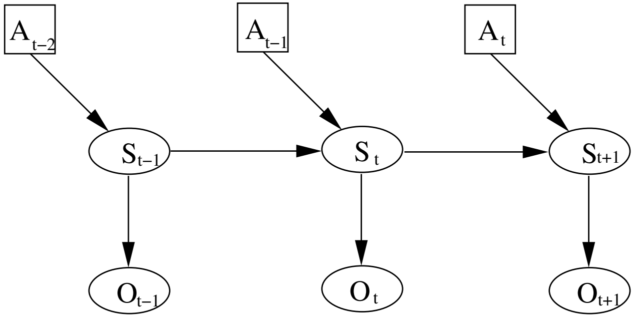
$$Bel(s_t) = \Pr(x_t \mid d_{0,\ldots,t})$$

  where $d_{0,\ldots,t}$ is all the data from time 0 to t.
- Two kinds of data are important:
  - Observations o_t
  - Actions a_t

  just as in the general scheme.

Bayesian filter

- Now, the basic principle behind using Bayes filters is that if we know the current state, then future states do not depend on past states.
- The Markov assumption.



Bayesian filter

- So, we can calculate the belief recursively based on:
  - The next state density or motion model

$$\Pr(s_t \mid s_{t-1}, a_{t-1})$$

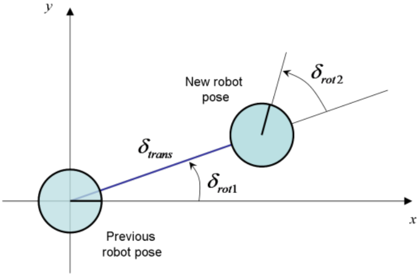
  - The sensor model

$$\Pr(o_t \mid s_t)$$

- In other words, belief about the current location is a function of belief about the previous location, what the robot did, and what the robot can see.

Bayesian filter

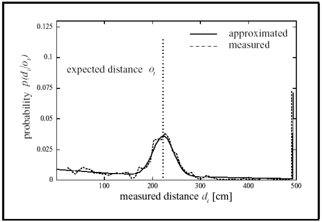
- The motion model, obviously enough, predicts how the robot moves.



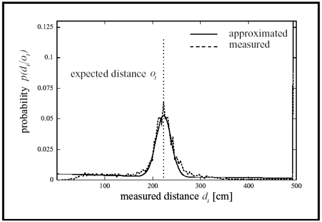
- The model should take into account the fact that the motion is uncertain.

Bayesian filter

- The sensor model captures both the landmarks the robot can see, and the lack of precise knowledge in where the robot must be to see them.



Ultrasound.

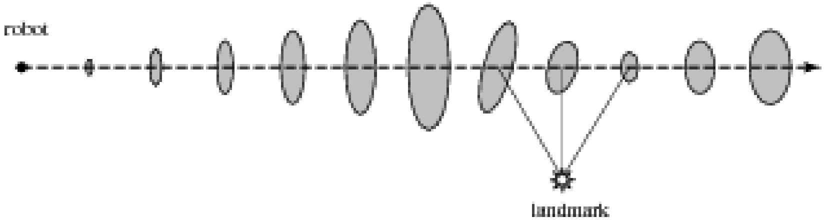


Laser range-finder.

- o_t in the above is the distance the sensor says the object is away from the robot, d_t is the real distance.
- The map tells us how far the object is, d_t, and the graph tells us how likely this is.

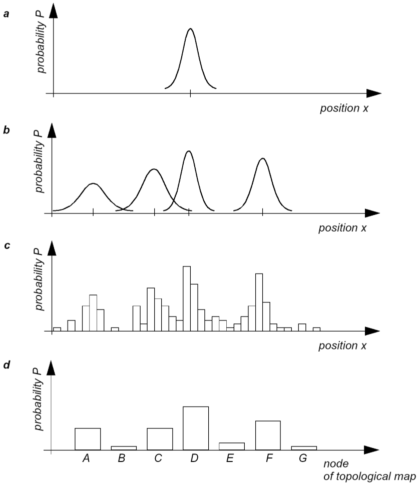
Bayesian filter

- Overall, the filtering procedure works to reduce uncertainty of location when landmarks are observed.



- Diagram assumes that landmarks are identifiable—otherwise, *Bel* is multimodal

Models of belief



- Single hypothesis, continuous distribution
- Multiple hypothesis, continuous distribution
- Multiple hypothesis, discrete distribution
- Topological map, discrete distribution

More filtering

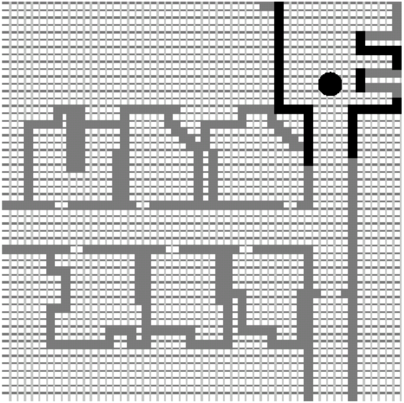
- Handling the kind of probability distributions that the Bayes filter requires is a bit tricky.
- So we improvise.
- Three different approaches:
  - Assume everything is Gaussian.
  - Make the environment discrete.
  - Take a sampling approach.
- All are used with differing degrees of success.

More filtering

- Assuming Gaussian distributions gives us Kalman filters.
  - Fast and accurate.
  - Only really work for position tracking.
- A discrete environment gives us Markov localization.
  - Simple.
  - Accuracy requires huge memory.
- We'll start by looking at Markov localization.

Markov Localization

- We start with a map that breaks the world into a grid:



- There are many ways to do this, as we saw last lecture.

Markov Localization

- Initally we have a uniform distribution over the possible locations.
- For every observation, for every location, we check what we observe against the map.
    - Apply the sensor model to find out how likely the observation is from that location.
    - Update the probability of the location.
- Then we normalize the probabilities — make sure they all add up to 1.

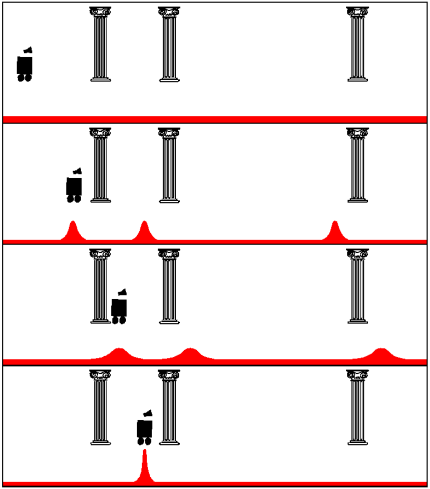
Markov Localization

- For every motion, for every location
  - Apply the sensor model to find out what new locations are how likely.
  - Update the probability of those locations.
- Then we normalize the probabilities — make sure they all add up to 1.

Markov Localization

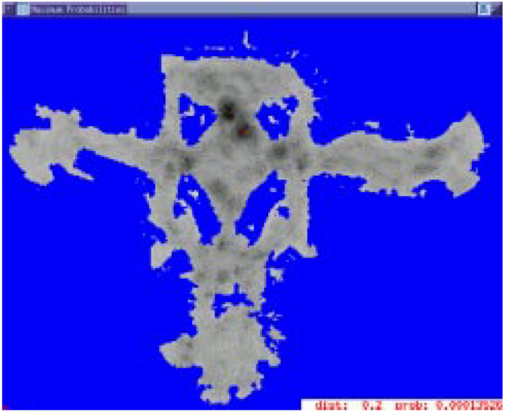
- We repeat this process for every item of sensor data and every motion.

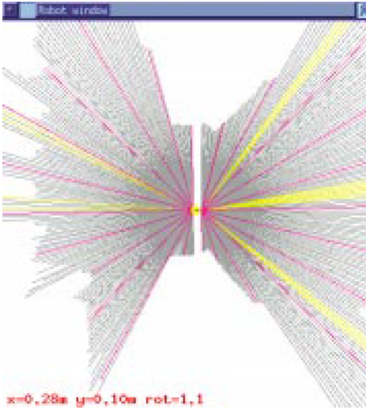
- Crudely what happens:



Markov Localization

- After 1 scan.

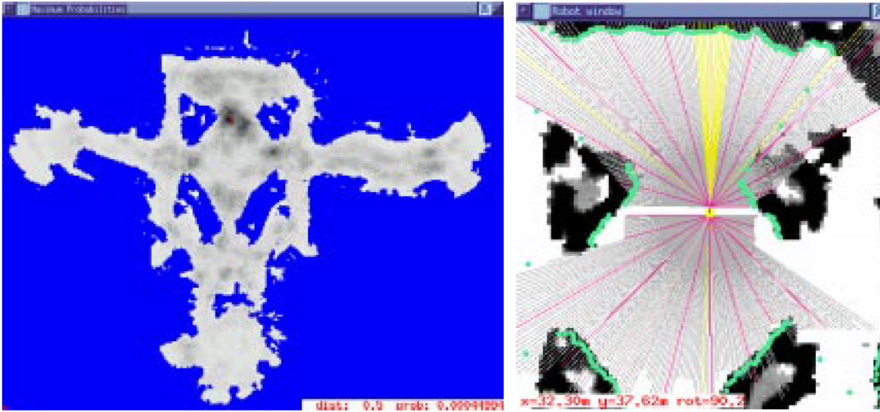




- W. Burgard

Markov Localization

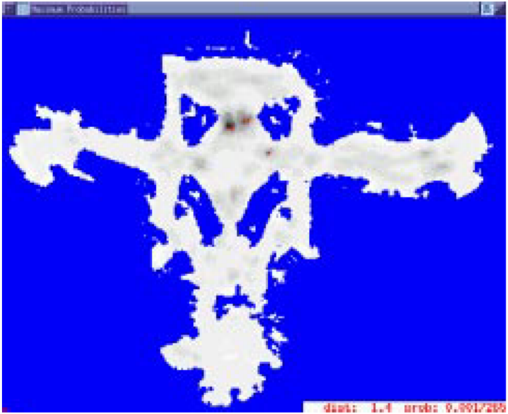
- After 2 scans.

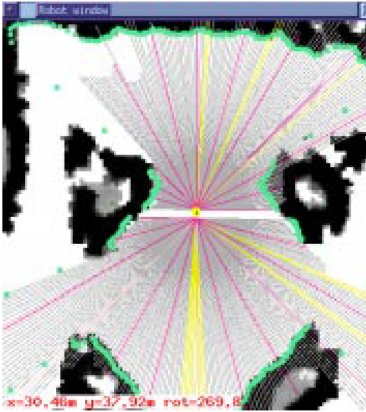


- W. Burgard

Markov Localization

- After 3 scans.

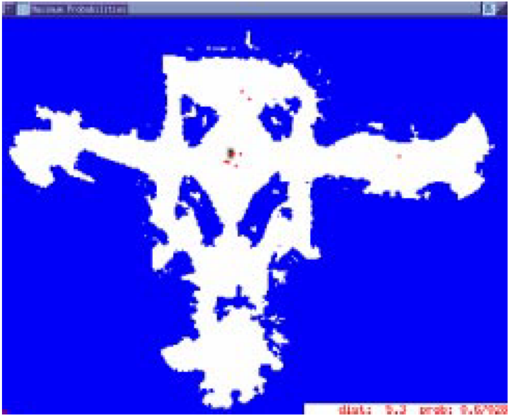


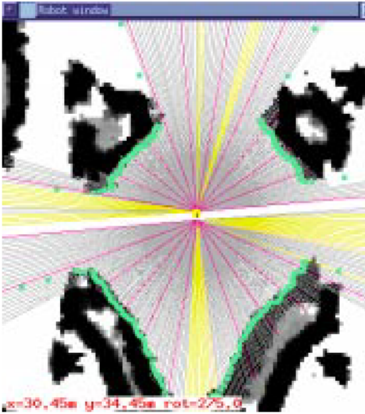


- W. Burgard

Markov Localization

- After 13 scans.

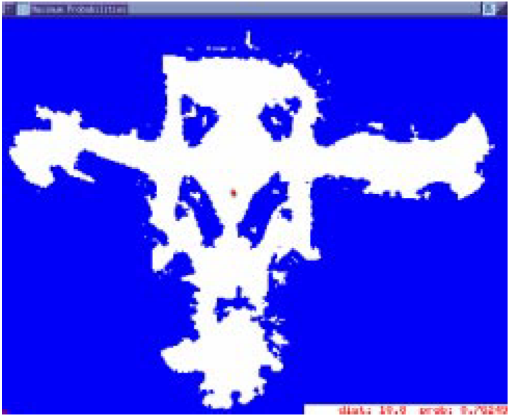


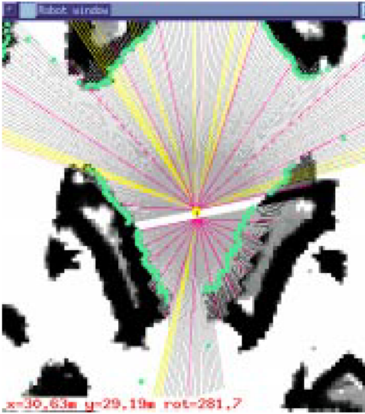


- W. Burgard

Markov Localization

- After 21 scans.

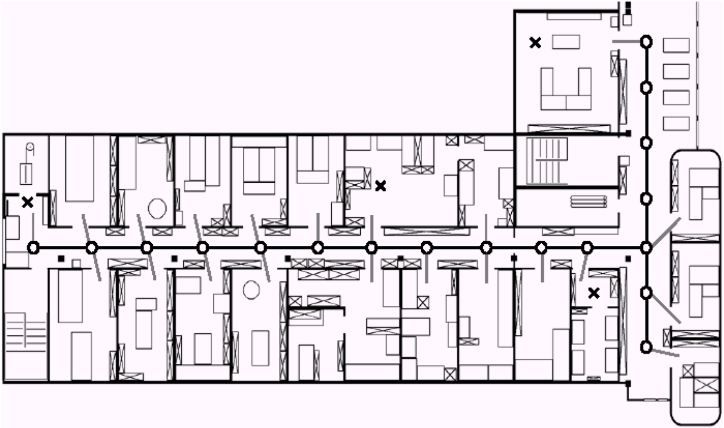




- W. Burgard

Topological maps

- Another way to make the map discrete is to use a topological map.



- Treat it the same way as the grid map.
- Fewer locations is good and bad.

Improving on Markov Localization

- The problem with Markov localization is that if the area is big, we need to consider a lot of possible locations.
  - Memory and processor intensive
- Particle filters use sampling techniques to reduce the number of possible positions, and hence the number of calculations.
- The sampling approach is what we will consider next.
- Rather than compute the whole distribution, we pick possible locations (samples) and do the calculations for them.
- This can work with surprisingly few samples (or particles).

Particle filter

- Also known as “Monte-Carlo Localization”.
- We approximate $Bel(s_t)$by a set of samples:

$$Bel(s_t) \approx \{s_t^{(i)}, w_t^{(i)}\}_{i=1,\ldots,m}$$

- Each $s_t^{(i)}$ is a possible pose, and each $w_t^{(i)}$ is the probability of that pose (also called an importance factor).
- Initially we have a set of samples (typically uniform) that give us $Bel(s_o)$.
- Then we update with the following algorithm.

Particle filter

$s_{t+1} = \varnothing$

for $j = 1$ to m
  // `apply the motion model`
  generate a new sample $s_{t+1}^{(j)}$ from $s_t^{(j)}$, a_t and $\Pr(s_{t+1} \mid s_t, a_t)$
  // `apply the sensor model`
  compute the weight $w_{t+1}^{(j)} = \Pr(o_{t+1} \mid s_{t+1})$

// `pick points randomly but biased by their weight`
for $j = 1$ to m
  pick a random $s_{t+1}^{(i)}$ from s_{t+1} according to $w_{t+1}^{(1)}, \ldots, w_{t+1}^{(m)}$

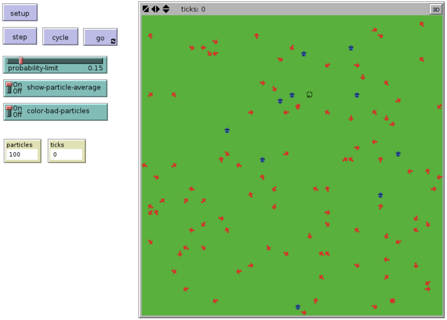
normalize w_{t+1} in s_{t+1}
return s_{t+1}

Particle filter

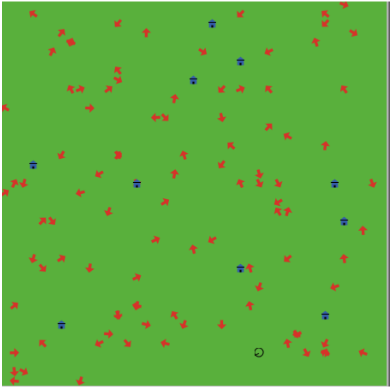
- And that is all it takes.

Particle filter

- How does this work?

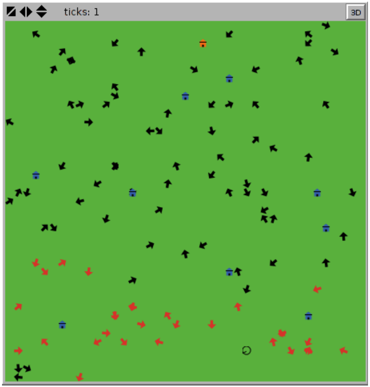


Particle filter



- After intialisation. Robot is the small circle, arrows are particles. The direction of the arrow shows the angle component of the pose. Blue houses are beacons.

Particle filter

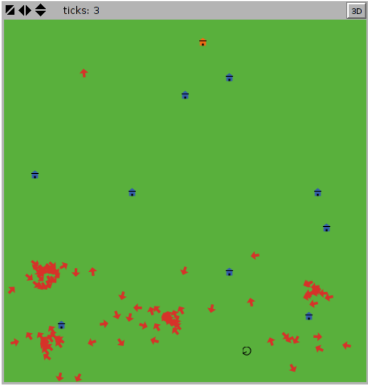


- Orange house is the beacon observed by the robot. Black particles are those whose weight, determined by the sensor model, is below a threshold.

Particle filter

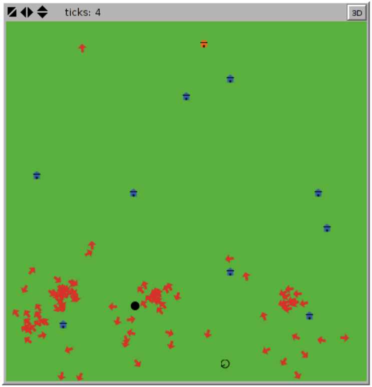
- Notice how the remaining red particles are on approximately the same radius around the observed beacon as the robot.

Particle filter



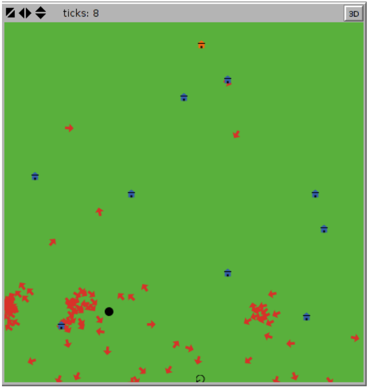
- Black particles are removed, and we resample from the ones that are left. A few random particles are added.

Particle filter



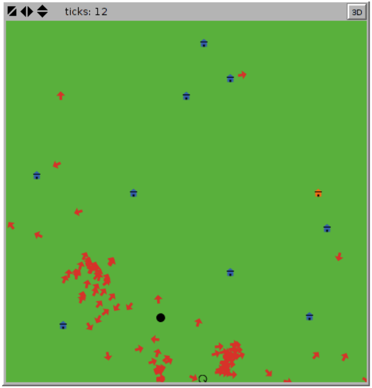
- Robot moves, particles are updated with the motion model. Black dot is the average of the particles — where the particle filter thinks the robot is.

Particle filter



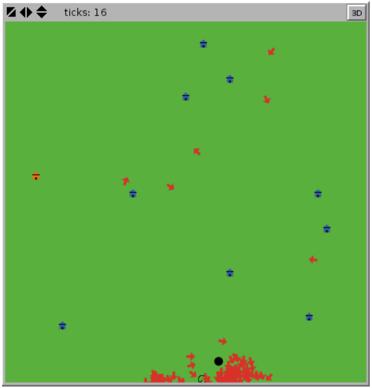
- After another cycle of observation, resampling, and moving.

Particle filter



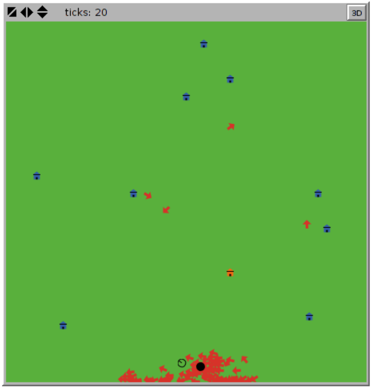
- After a third cycle.

Particle filter



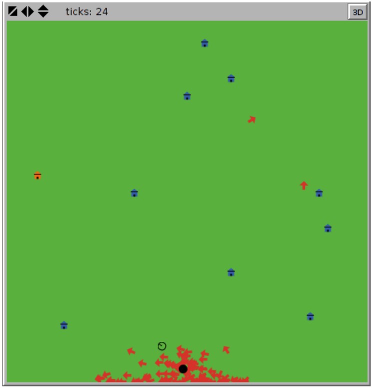
- After a fourth cycle.

Particle filter



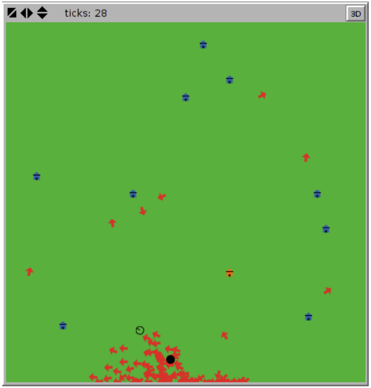
- After a fifth cycle.

Particle filter



- After a sixth cycle.

Particle filter



- After a seventh cycle.

Particle filter

- Thus after several repetitions, the particle “home in” on the correct location of the robot.
- And then they track it as it moves.

Effectiveness

- All localization is limited by the noise in sensors:
  - There are techniques for reducing noise by modelling spurious measurements.
  - Cannot remove all uncertainty.
- Discrete, grid-based approaches can reduce average error below 5cm.
  - However this is hard to do in real-time.
  - Requires huge amounts of memory.
- Particle filters with feasible sample sizes ($\approx$ 1000) have comparable error rates.

Effectiveness

- With much smaller numbers of particles ($\approx$ 100) we have average errors of around 10cm.
- This is sufficient for many tasks.

Summary

- This lecture looked at the problem of localization
    - How we have the robot figure out where it is.
- We discussed why odometry is not sufficient.
- We then described probabilistic localization techniques, concentrating on:
    - Markov localization
    - Particle filters