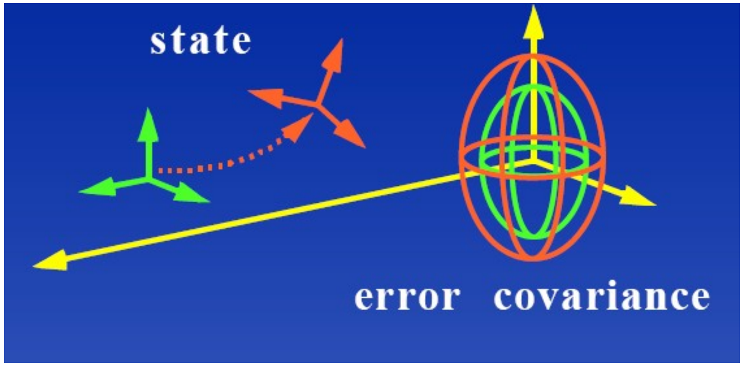
Particle Filters; Simultaneous Localization and Mapping (Intelligent Autonomous Robotics)

Subramanian Ramamoorthy
School of Informatics

Recap: State Estimation using Kalman Filter

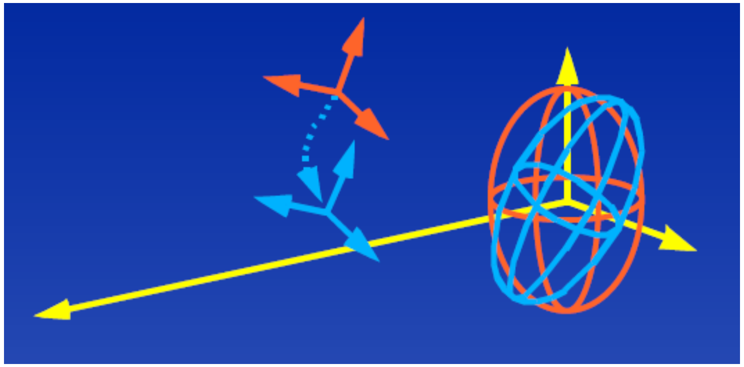
- Project state and error covariance forward in time:

e.g., during a time interval, you expect the ball to go from 1m to 1.5 m, with some uncertainty increase



- Update estimate after measurement:

In fact, vision sees ball going to 1.7 m so you update your estimates to Conclude ball must be at 1.6 m with some new level of uncertainty

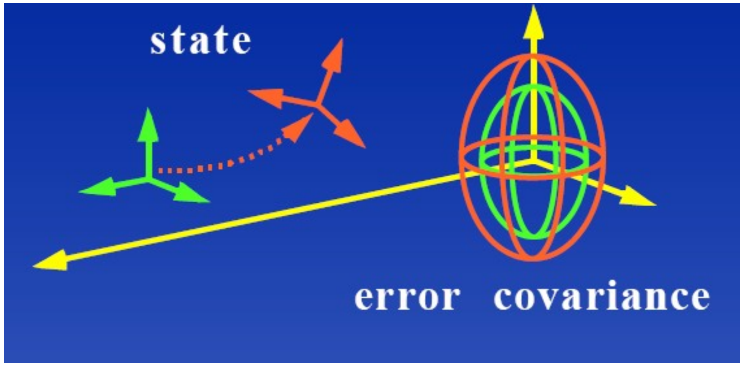


Recap: State Estimation using Kalman Filter

- Project state and error covariance forward in time:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$$

$$P_k^- = AP_{k-1}A^T + Q$$

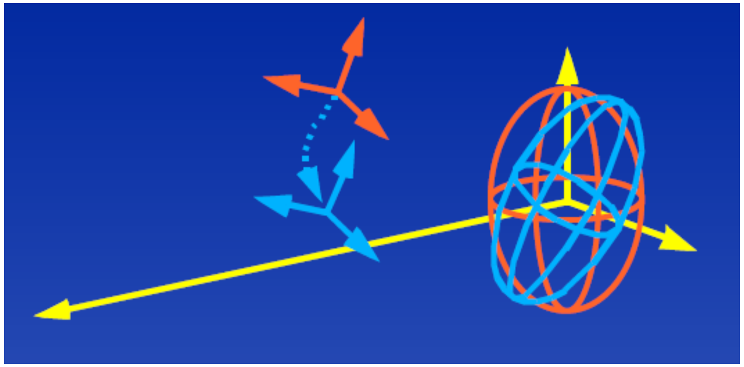


- Update estimate after measurement:

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

$$P_k = (I - K_k H)P_k^-$$



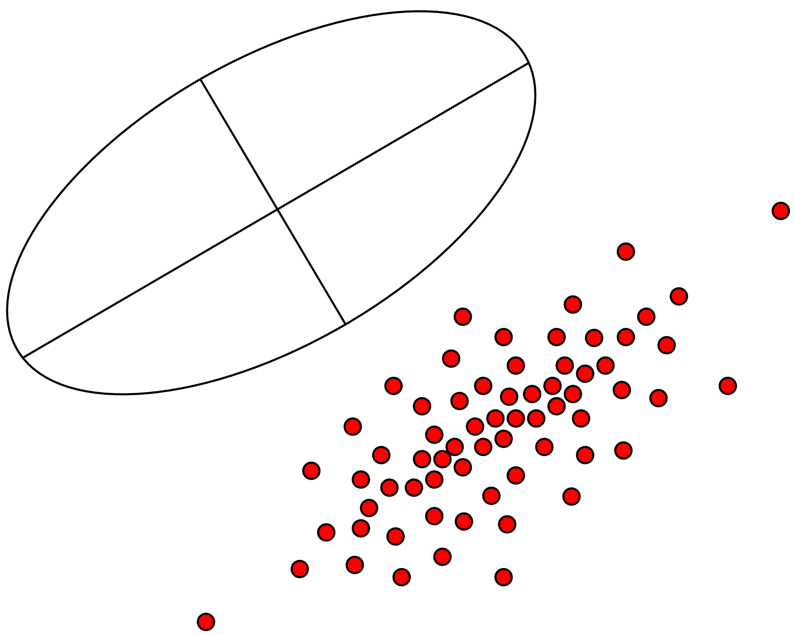
Limitations of the Kalman Filter

- Optimal state estimator for <u>Linear</u> systems & <u>Gaussian</u> noise
  - Most robots involve nonlinear dynamics (simple example: stick-slip friction and slippage in tyres)
  - Many commonly used sensors, e.g., sonar, involve more complex type of noise
  - In complex scenarios (e.g., estimating positions of obstacles in a room), one is dealing with multi-modal distributions

- Standard extensions for nonlinearity may not be satisfactory:
  - If initial state estimate is wrong, or if process is incorrectly modeled, the filter could quickly diverge
  - Covariance is underestimated

Particle Filter

Represent probability distribution as a set of discrete *particles* which occupy the state space – efficient for non-Gaussian distributions



Particle = state hypothesis

Distribution = set of state hypotheses

Sample Based Posterior Probabilities

Set of weighted samples

$$S = \left\{ \left\langle s^{(i)}, w^{(i)} \right\rangle \mid i = 1, \ldots, N \right\}$$

State hypothesis — Importance weight

e.g., distance could be 4, 5 or 6 m – each value is a hypothesis.

But, it is much more likely that the true value is 6 m, and not 4 m.

The samples represent the posterior

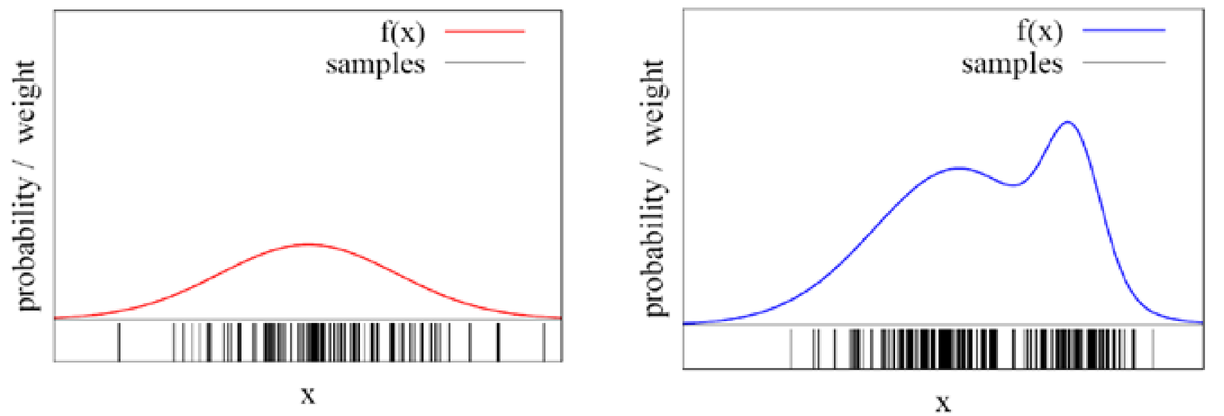
$$p(x) = \sum_{i=1}^{N} w_i \cdot \delta_{s^{(i)}}(x)$$

True measurement is estimated as weighted average of all hypotheses.

Approximating the Posterior

Particle sets can be used to approximate functions

- *Sample from 'proposal' distribution*
- *Update hypotheses using measurements*



The more particles fall into an interval, the higher the probability of that interval

How to draw samples form a function/distribution?

Drawing Samples from a Distribution: Rejection Sampling

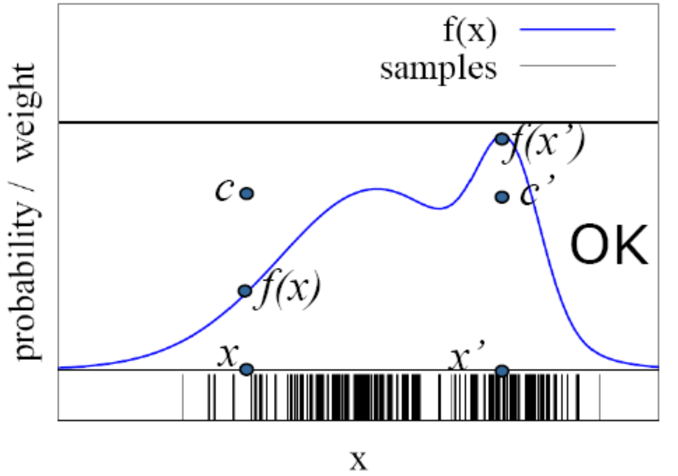
Let us assume that $f(x)<1$ for all x

Sample x from a uniform distribution

Sample c from [0,1]

if $f(x) > c$ keep the sample

otherwise reject the sample



Better Idea: Importance Sampling

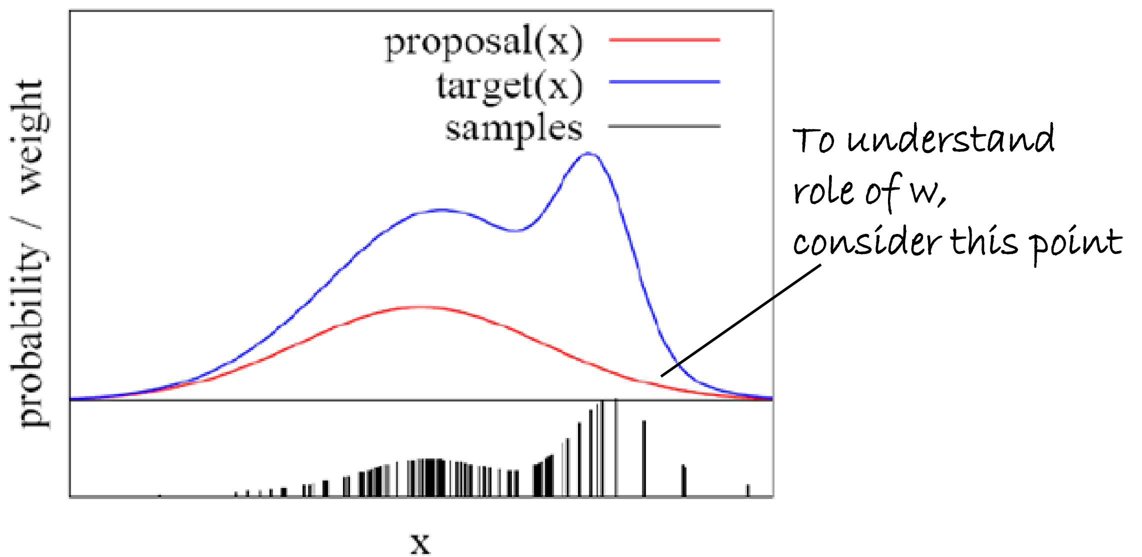
We can use a different distribution g to generate samples from f

By introducing an importance weight w, we can account for the "differences between g and f"

$w = f / g$

f is called target

g is called proposal



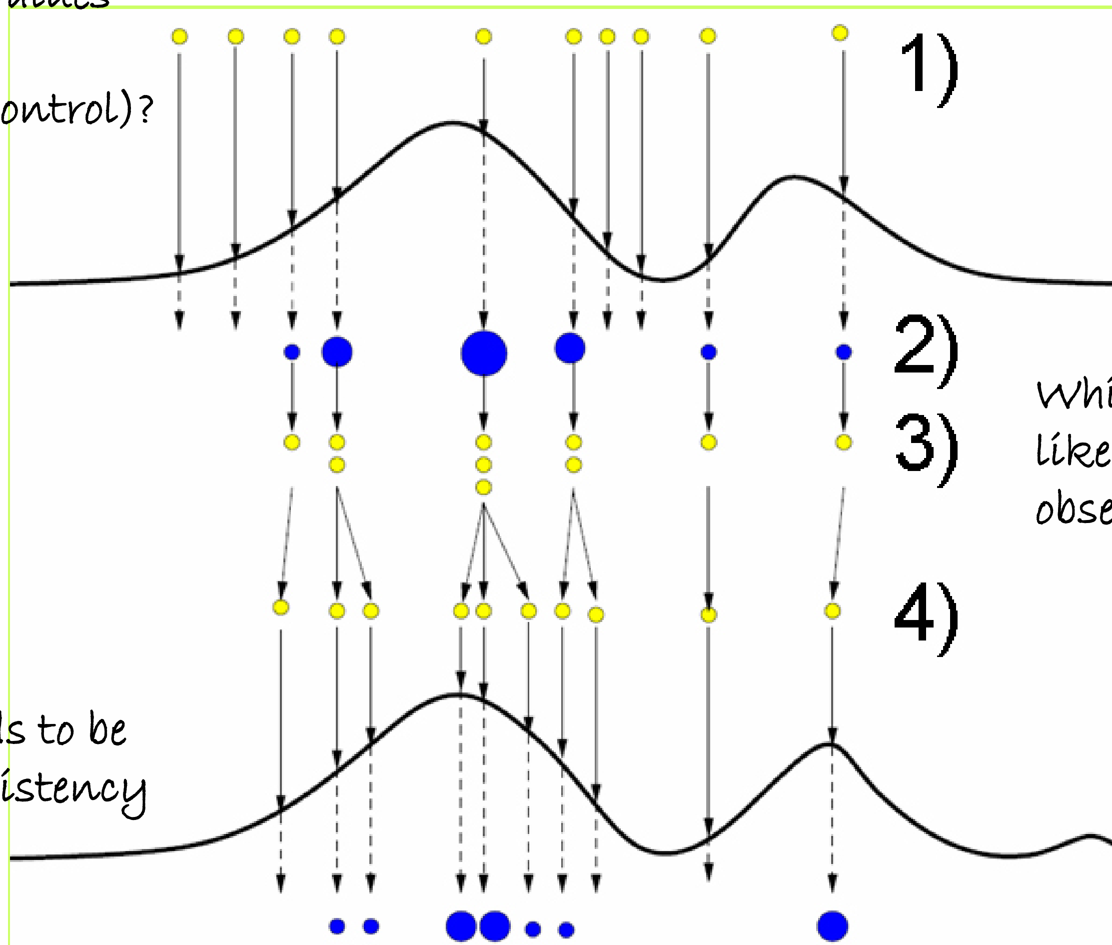
From Sampling to the *Particle Filter*

- Posterior distribution ⇔ set of sample hypotheses
- Filter update (i.e., state estimate) based on actual actions and observations by the robot

- The particle filter algorithms involves three steps:
  1. Sampling particles from a proposal distribution
     - (This is like the 'prediction' step in KF)
  2. Computing the particle weight (importance sampling)
     - (This is like the 'correction' step in KF)
  3. Resampling – an additional 'statistical' correction step

Particle Filter Update Cycle

What are possible values for estimated state (given past state/control)?

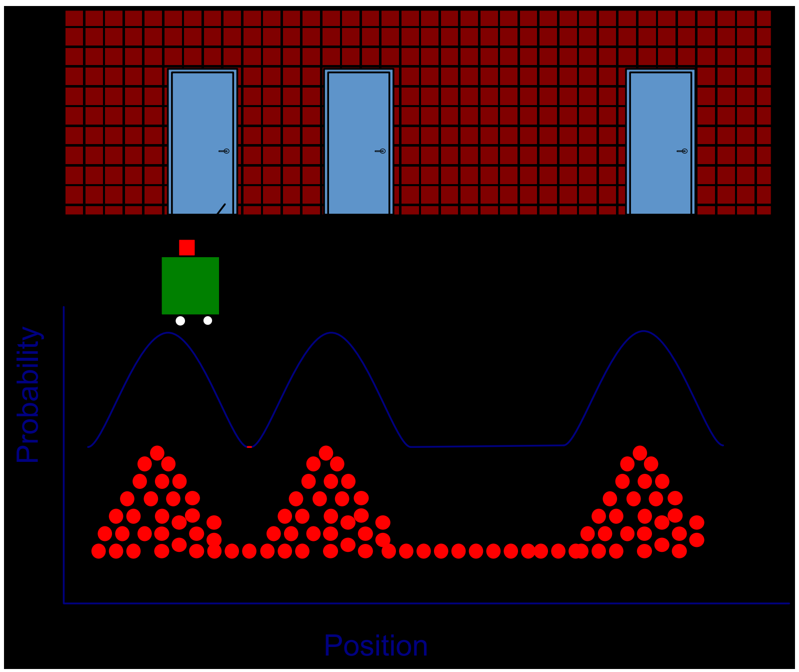


Which states are more likely given sensor observation?

Distribution needs to be adjusted for consistency

Particle Filter – Advantages/Disadvantages

Nonparametric, Handles multi-modal distributions

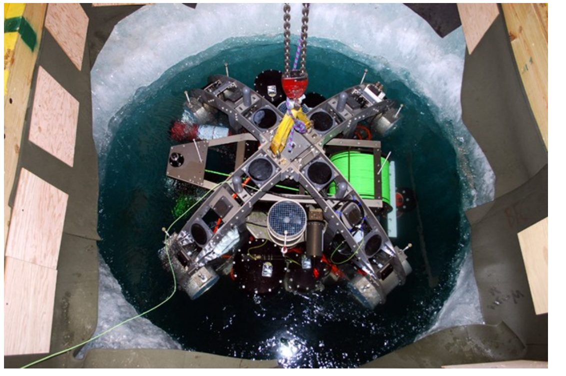


Number of particles grows exponentially with the dimensionality of the state space

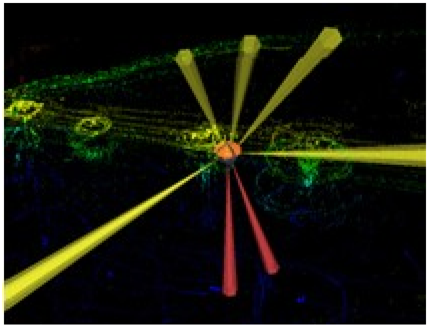
1-dim ⇔ *n* particles
2-dim ⇔ n^2 particles
m-dim ⇔ n^m particles

What is SLAM?

Consider the following scenario:



- Your robot is called upon to explore below ice sheet in a lake in Antarctica
- You *do not* have a map of the terrain
- You may sense your current position using a combination of vision and sonar –very noisy!



- Robot needs to do two things at once:
  - Explore the terrain and draw a *map*
  - Use measurements to *locate* itself within map

[Source: Stone Aerospace, ENDURANCE 2008 Mission]

The SLAM Problem

Estimate the pose and the map of a mobile robot at the same time

$$p(x, m \mid z, u)$$

poses map observations & movements

Let's first think about one piece…

Problem: Location Estimation, *given a map*

Simple question: Where are you (*within the given map*)?

- Instead of a single hypothesis about location, maintain probability distribution over hypotheses
- Use estimation algorithm to improve knowledge given sequence of measurements
- Density function can have arbitrary form (e.g., multiple modes) – so, use algorithm like particle filters

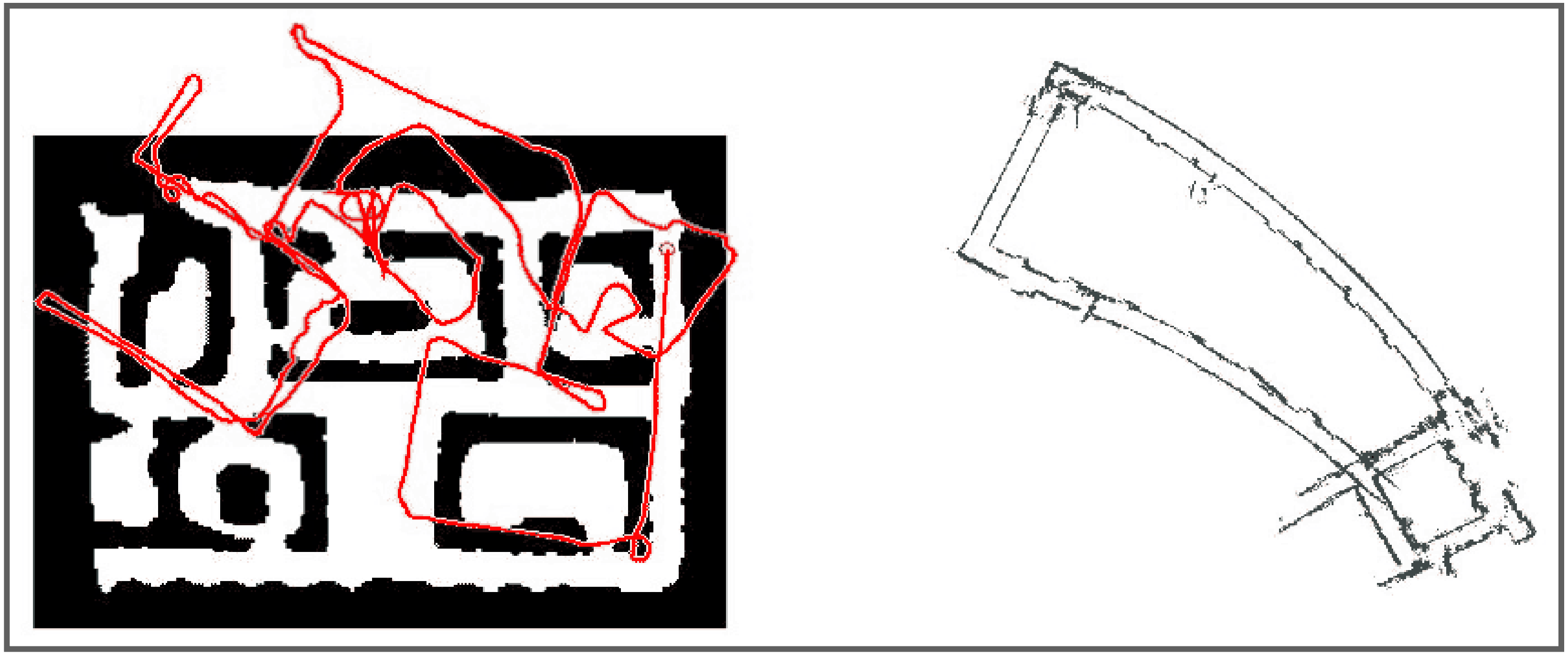
But first, a naïve question: if you have a map and a stream of measurements, couldn't you just trace your path?

View through the robot's "eyes"…

[Source: http://www.cs.washington.edu/ai/Mobile_Robotics]

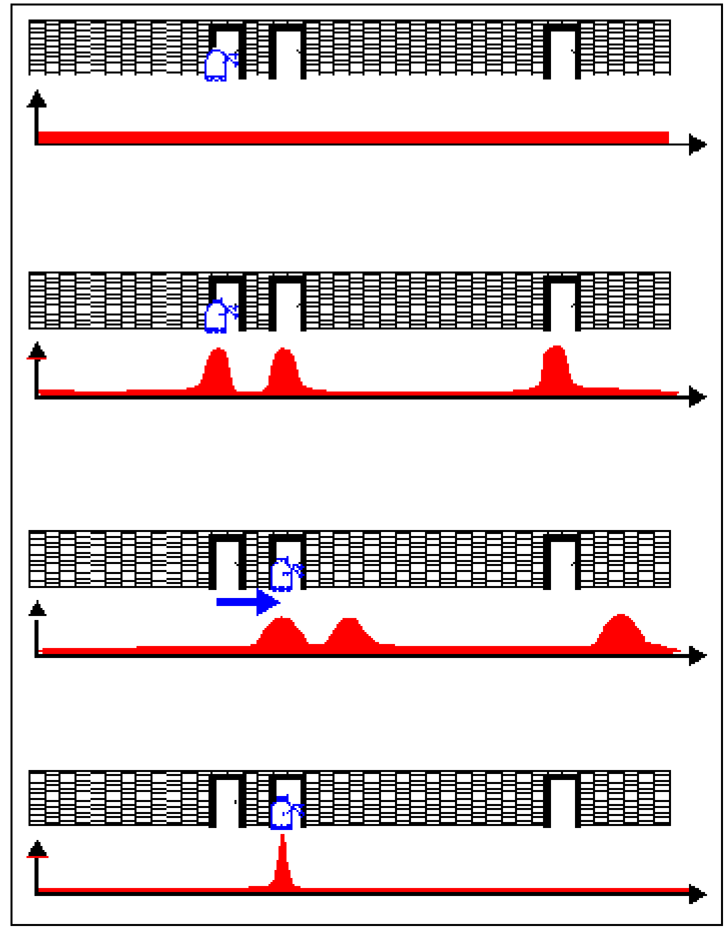
Problem with Dead Reackoning

- Simply integrating robot velocity commands from a known starting point gets the robot hopelessly lost
- Same thing if you integrate on-board odometry (position)



Probabilistic Localization: Basic Idea

- Robot in 1-dim world
- Initially, it is lost: uniform distribution
- Queries sensor to find it is near a door: increase probability near all doors
  - Multimodal distribution, need more information!
- Robot moves, to door #2
  - Move increases uncertainty, squashes state distribution
- Robot queries sensor again and localizes itself!



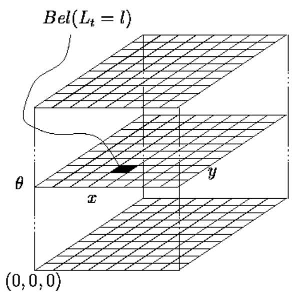
Some Remarks on Localization

- If the doors were uniquely identifiable then the problem is merely that of sensor noise – use a Kalman filter
- In fact, robot can not be sure which door it has sensed – this is the *data association* problem
  - Beliefs are inherently multimodal due to ambiguities
- The benefit of the probabilistic approach lies in the ability to explicitly represent and reason about this ambiguity

- Localization involves two major issues:
  1. Representing the belief *P(x)* (where could I possibly be?)
  2. Computing conditional probabilities (where could I be, given what I see?)

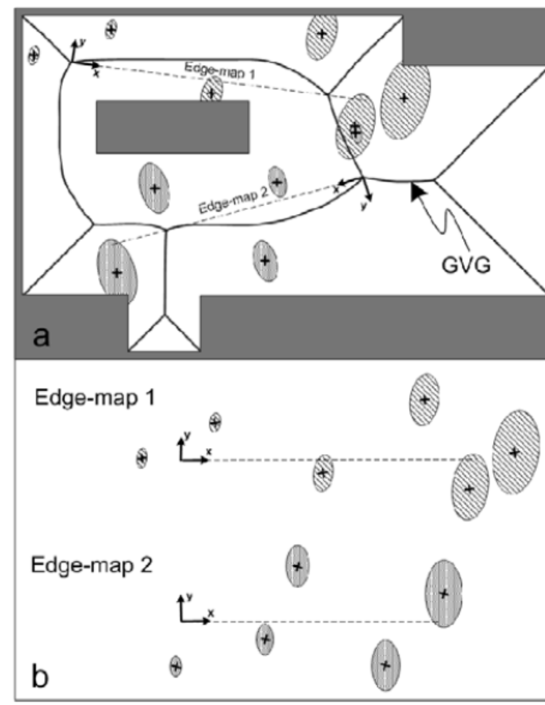
How to *represent* map (configuration space)?

There are a number of choices and they determine how we deal with the computation. Two examples:



- Simple – use a grid

- Landmark based methods:
  e.g., derived from a Voronoi graph



Probabilistic Localization (Recursive Filtering)

Problem: Estimate posterior probability of states $P(x(k)|u(0:k-1), y(1:k))$,
given sensor readings $y(1:k)$ obtained by movements $u(0:k-1)$.

Assume that robot is *given* a map m, all terms are conditioned on this knowledge.

Probabilistic *localization* uses current measurements,
to update our most recent (*prior*) estimate
in order to obtain an improved (*posterior*) estimate,

Sensor Model: Probability of current Observation given current state

$$
\begin{aligned}
&P(x(k)|u(0:k-1), y(1:k)) \\
&\quad = \eta(k) P(y(k)|x(k)) \\
&\quad \sum_{x_{k-1} \in X} (P(x(k)|u(k-1), x(k-1)) \cdot P(x(k-1)|u(0:k-2), y(1:k-1)))
\end{aligned}
$$

Motion model: Probability of current state, given previous state and action

Probability of state, given past history

Localization – procedurally…

- When you get odometry reading *u(k-1)*, prediction step:

$$P(x(k)|u(0:k-1),y(1:k-1)) \\ = \sum_{x_{k-1}\in X}(P(x(k)|u(k-1),x(k-1))P(x(k-1)|u(0:k-2),y(1:k-1)))$$

- Then, when you get a measurement *y(k)*, update step:

$$\eta(k) = \left[\sum_{x(k)\in X} P(y(k)|x(k))P(x(k)|u(0:k-1),y(1:k-1))\right]^{-1}$$

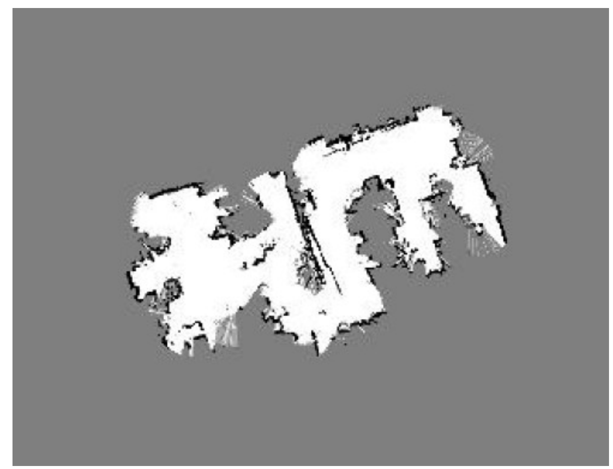
$$P(x(k)|u(0:k-1),y(1:k)) \\ = \eta(k)P(y(k)|x(k))P(x(k)|u(0:k-1),y(1:k-1))$$

Note: All discrete sums may be replaced by integrals.

The Mapping Problem
(structure of the environment)

- Based on a trace of observations, can we build a map?
- Robot must cope with two forms of uncertainty: noise in perception *(y)* and noise in odometry (*u*).
- Assume 'localization is solved' – robot knows where it is

- A simple way to build a map:
  Occupancy grid - Each cell in a 2-dim grid *m* stores the probability that it is occupied



Occupancy Grids

- Impose grid on space to be mapped
- Identify an *inverse sensor model*

$$p(m_x \mid y_t)$$

- Update odds that grid cells are occupied

Simultaneous Localization and Mapping

Major approaches:

- Historical: Given a set of landmarks (e.g., I know goal posts in a stadium), use Kalman Filter type algorithms to estimate joint posterior probability over maps and robot locations
- Global optimization: Consider locations as random variables and derive constraints between locations using overlapping measurements (parameter optimization to minimize error)
- Neither one good for truly on-line applications, so many variations based on Bayesian statistics have been developed.

Bayesian SLAM: Posterior Probability of Map and Location

Motion model: Given what I just did and where I have just been, where am I?

Sensor model: Given where I am in a map, what could I expect to see?

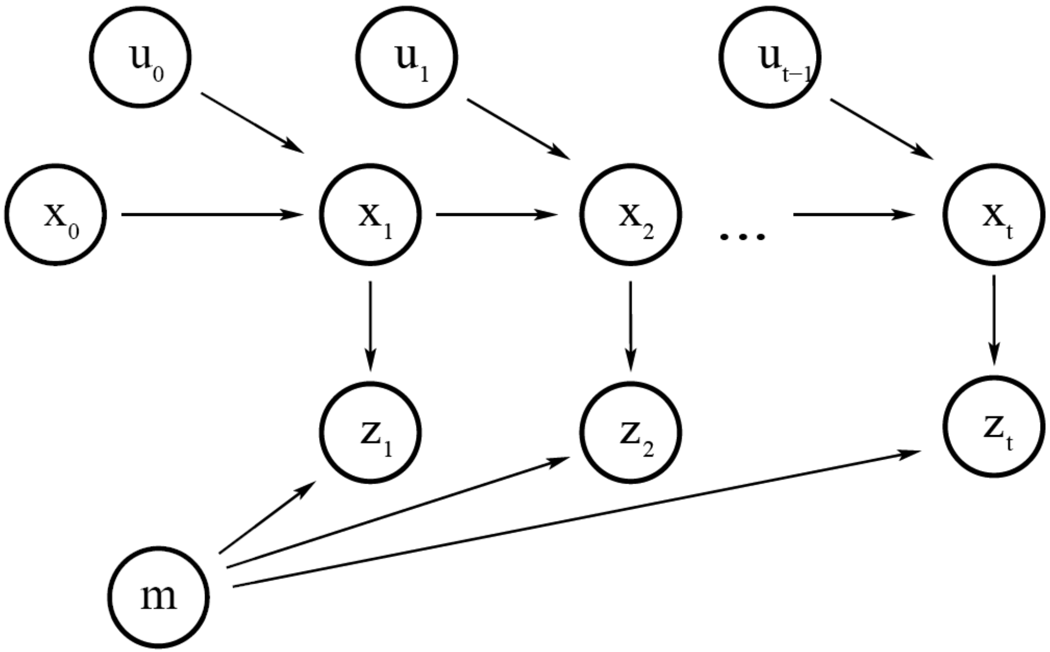
$$P(x(1:k), m | u(0:k-1), y(1:k)) = \alpha P(y(k) | x(k), m)$$

$$\int \Big(P(x(k) | u(k-1), x(k-1))$$

$$P(x(1:k-1), m | u(0:k-2), y(1:k-1)) \Big) dx(1:k-1)$$

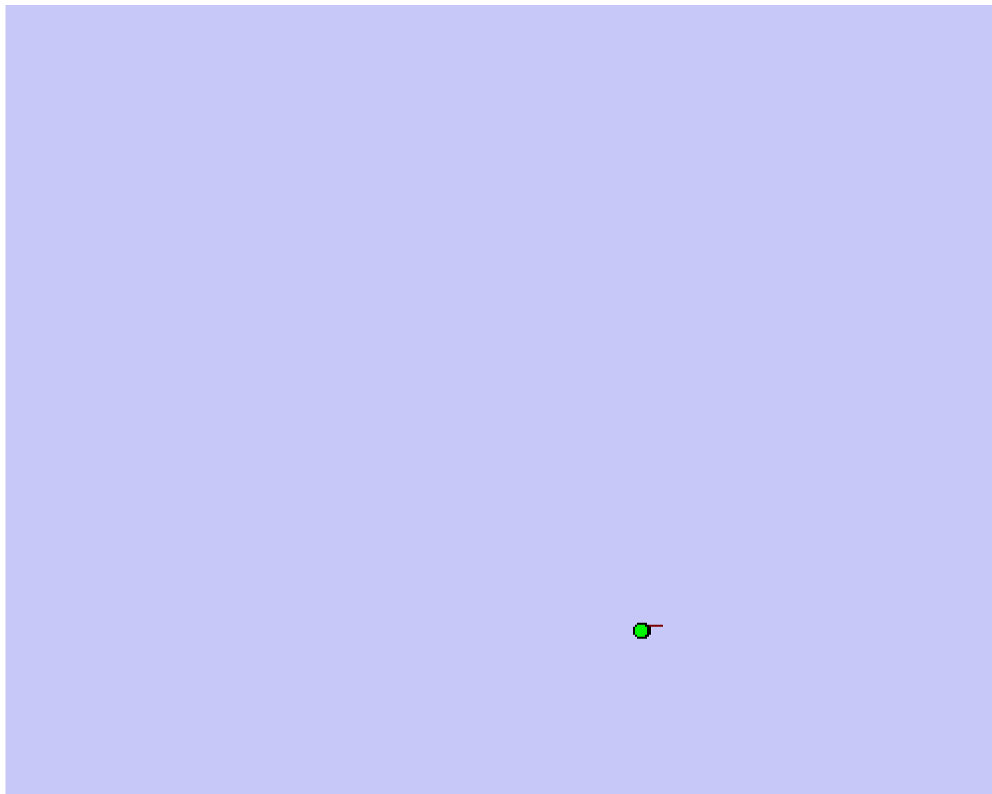
Map Refinement: Given everything I've done and seen so far, what is my best guess of the map and my place in it?

Graphical Model for Probabilistic SLAM



Each node represents variable to be estimated and each arrow represents conditional dependence. (*Note*: observations are denoted z instead of y)

Demo (using FastSLAM Algorithm)



[Source: http://www.cs.washington.edu/ai/Mobile_Robotics/]

References:

- H. Durrant-Whyte and T. Bailey, Simultaneous localization and mapping, IEEE Robotics and Automation Magazine, pp. 99 – 108, June 2006.
- S. Thrun et al., Probabilistic Robotics, MIT Press, 2005.

Acknowledgement:

Many pictures and equations in this presentation are taken from Giorgio Grisetti and Cyrill Stachniss, ECMR 2007 Tutorial