Simultaneous Localization and Mapping (Intelligent Autonomous Robotics)

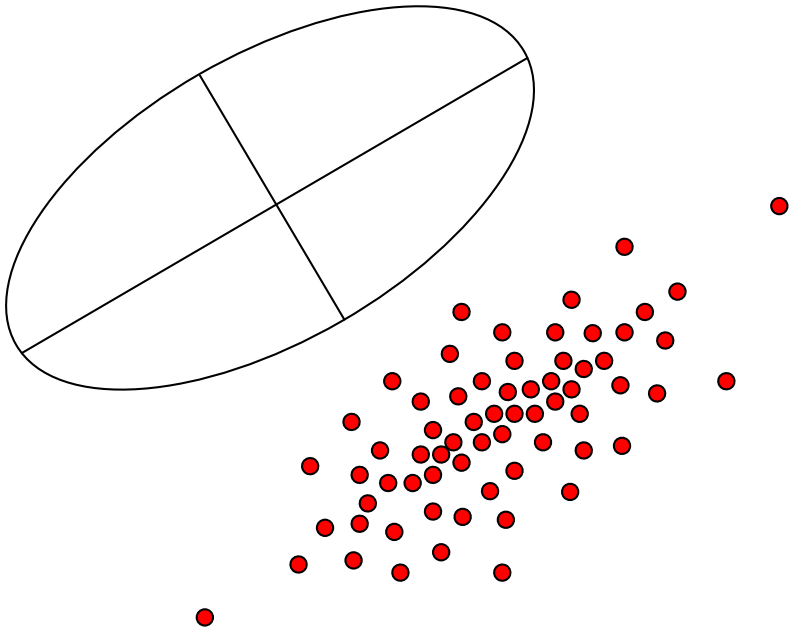
Dr. Subramanian Ramamoorthy
School of Informatics

Story so far, and Agenda

- We looked at basic notion of estimation
  - LS, WLS, recursive formulations, etc.
  - Useful for cleaning up noisy sensor data and fusing them
- Then, we added dynamics – robots tend to move!
  - Kalman filters, EKF, UKF
- A limitation with a lot of these methods – optimal only if noise is Gaussian
  - EKF/UKF did alleviate the problem somewhat
  - Today, we'll see another modern algorithm – Particle Filter
- Then, we'll consider a challenging robotics problem
  - *Simultaneous* localization and mapping – uses all of the above

Particle Filters

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

Particle Filters: Sample Based Posterior

Set of weighted samples

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

State hypothesis

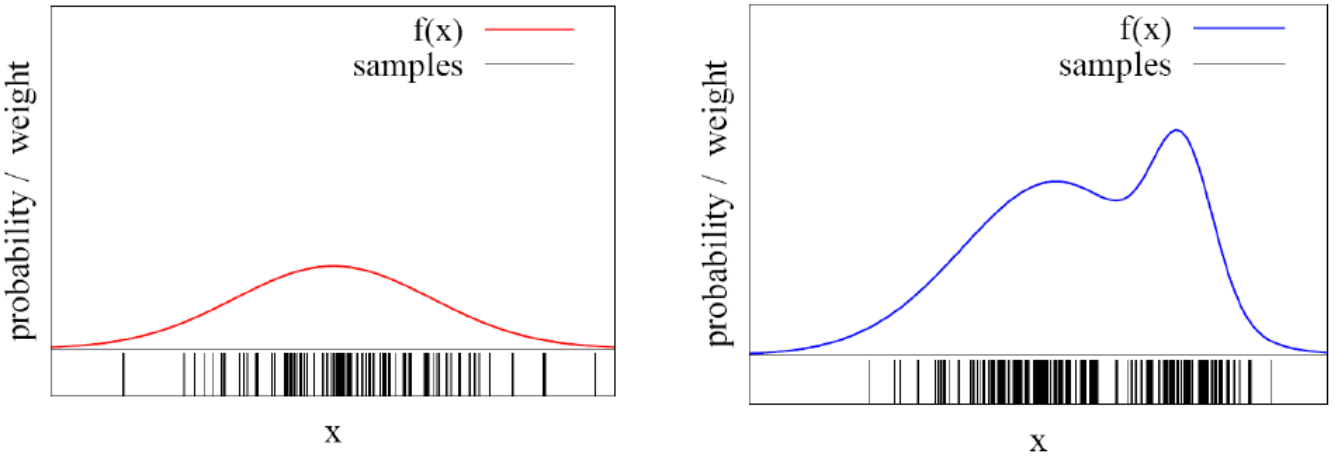
Importance weight

The samples represent the posterior

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

Approximating the Posterior

Particle sets can be used to approximate functions



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

How to draw samples form a function/distribution?

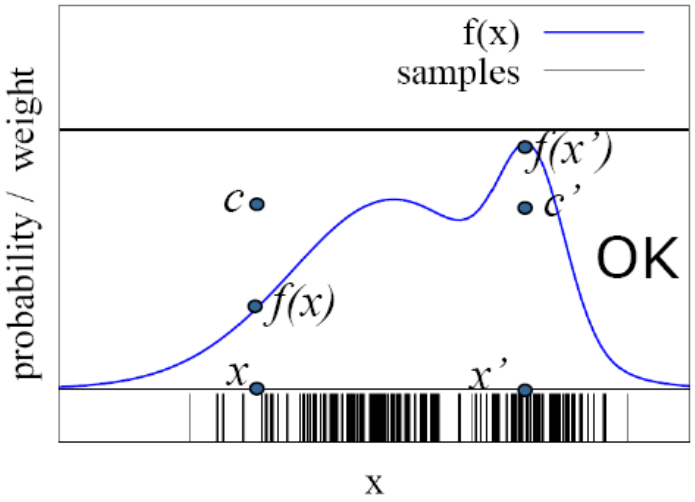
Rejection Sampling

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



Idea of Importance Sampling

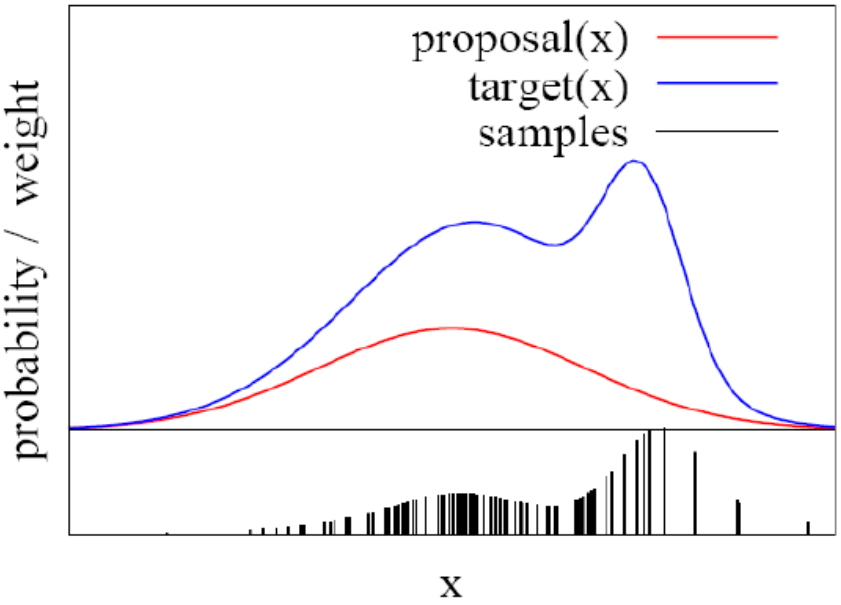
We can even use a different distribution g to generate samples from f

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



Going from Sampling to the Particle Filter

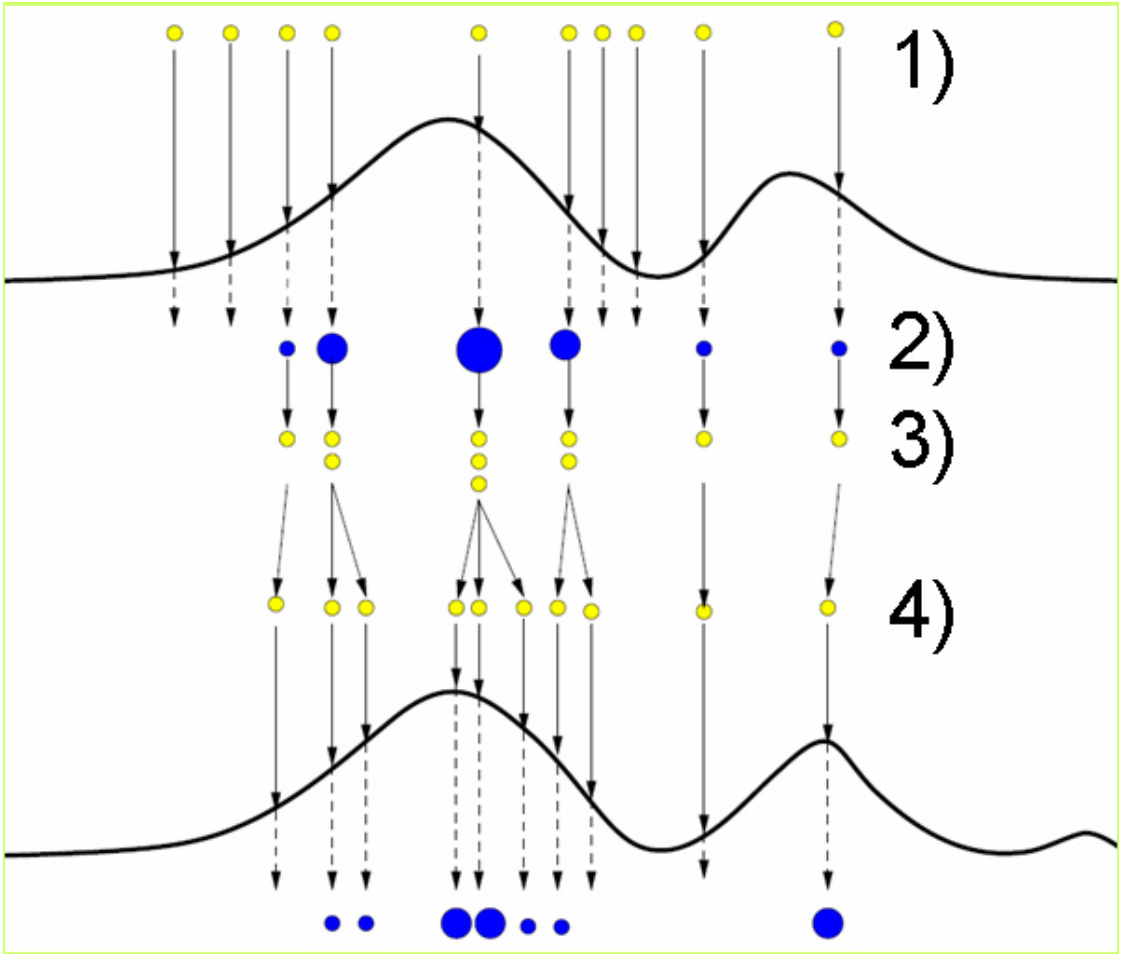
- Posterior = set of samples
- Updates are based on actions and observations (robot!)

- Basic Idea in three sequential steps:
  1. Sampling from the proposal distribution
     - (This is like a 'prediction' step)
  2. Computing the particle weight (importance sampling)
     - (This is like a 'correction' step)
  3. Resampling

Particle Filter Update Cycle: Bit More Detail

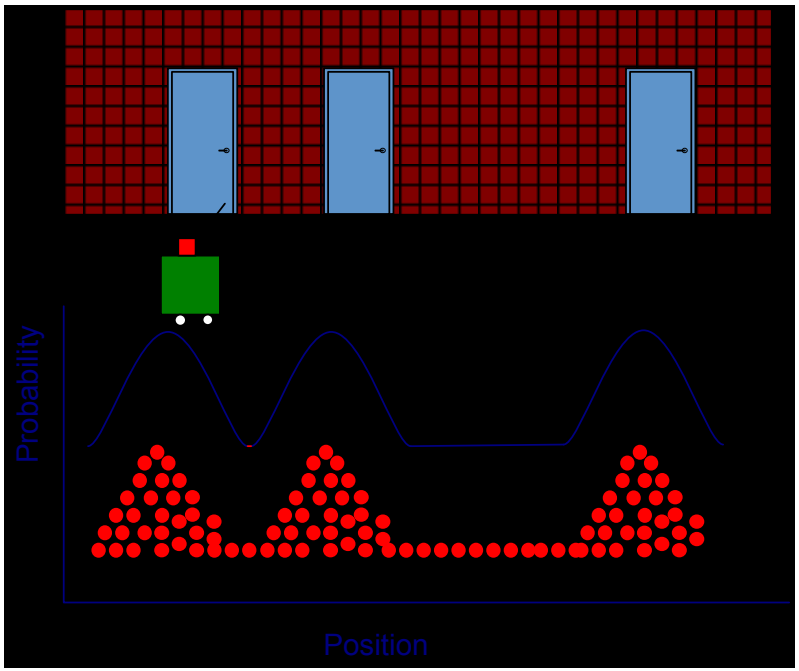
- Generate new particle distribution given motion model and controls applied
- For each particle
  - Compare particle's prediction of measurements with actual measurements
  - Particles whose predictions match the measurements are given a high weight
- Resample particles based on weight
  - Randomly draw particles from previous distribution based on weights creating a new distribution

Particle Filter Update Cycle: Visually



Particle Filter – Advantages/Disadvantages

Represents multi-modal distributions



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

- 1D – n particles
- 2D – n^2 particles
- mD – n^m particles

What is SLAM?

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

Simple question: Where are you?

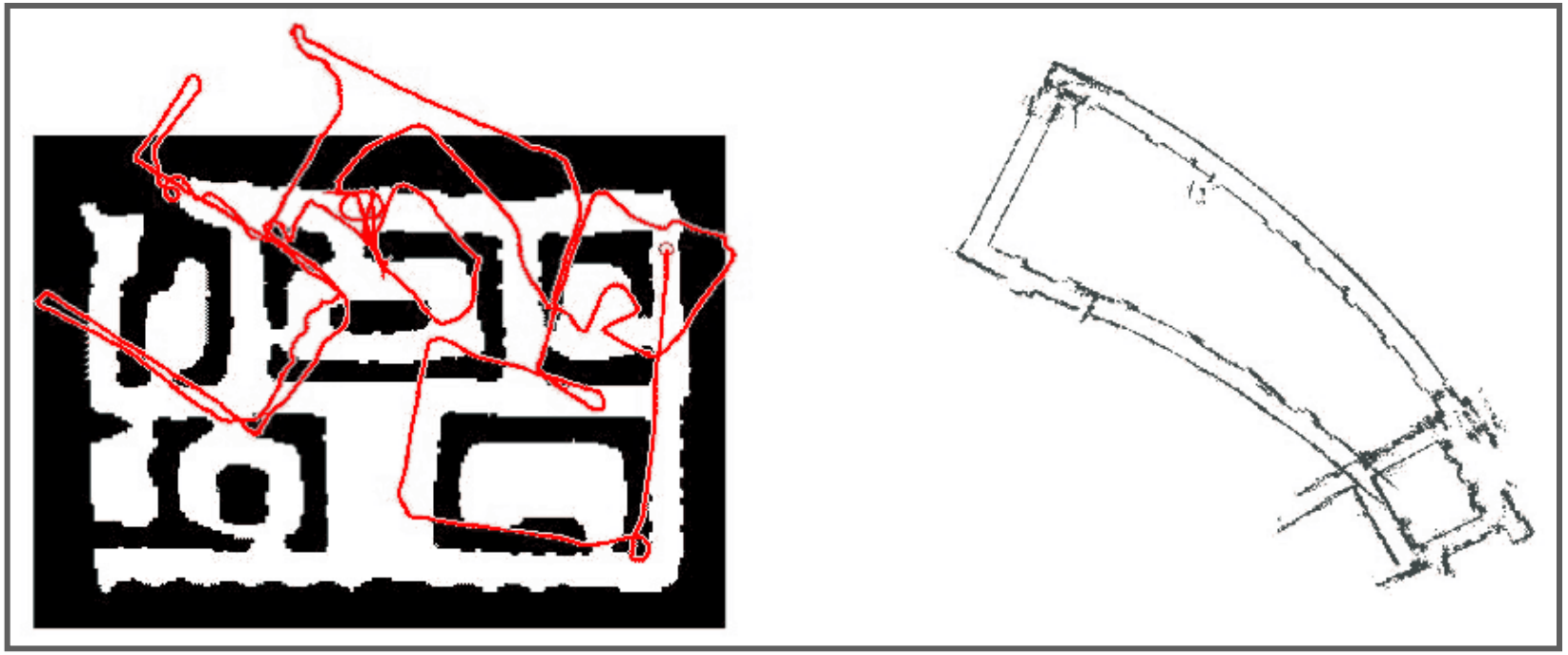
- Instead of maintaining a single hypothesis about location, maintain a probability distribution over the space of all hypotheses
- Then, use estimation algorithms to improve knowledge
- The density function can have arbitrary forms – that is why we need algorithms like particle filters
- But first, let is start with some visuals…

Looking through robot's "eyes"…

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

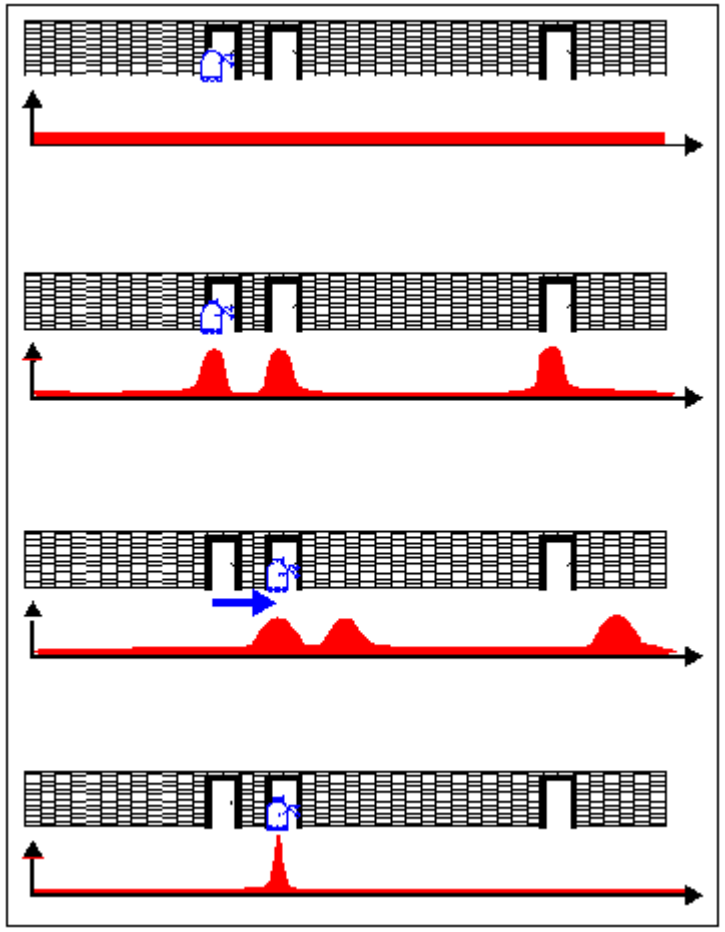
Problem with Dead Reackoning

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



Probabilistic Localization: Basic Idea

- Robot in 1-dim world
- Initially, it has no clue where it is: uniform distribution
- Queries sensor and finds it is near a door: increase probability near doors
  - Multimodal distribution, need more information
- Robot moves, comes near door #2 – movement squashes distribution
- But, robot queries sensor again and localizes itself!



Some nuances…

- The final belief actually has five peaks (although picture is coarse) – corresponding to sequence of two observations and one motion
  - The smaller peaks represent some mismatch between the "map" and the sensor readings
- If the doors are uniquely identifiable then the problem is merely that of sensor noise – use a Kalman filter
- In fact, it can't 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

Probabilistic Localization (Recursive Filtering)

Configuration (position and orientation), i.e., state $x \in X$.

At time step k,
robot needs to estimate posterior $P(x(k)|u(0:k-1), y(1:k))$ over all states,
given sensor readings $y(1:k)$ obtained by movements $u(0:k-1)$.

$u(k)$ is a general definition of movement (could be commanded velocity, odometry measurement or result of filtering and fusing the two).

For now, we will also *assume* that the robot has been given a map m and every term is conditioned on this knowledge.

Then, probabilistic localization is the process of using the current measurement to update out most recent (*prior*) estimate to obtain an improved (*posterior*) estimate,

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

Probabilistic Localization: Key Terms

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

Motion model
(transition probability)

Observation Model (likelihood)

Note that these two terms correspond to the two equations we are familiar with from the Kalman filter.
More general form of the above equation:

$$P(x(k)|u(0:k-1), y(1:k))$$
$$= \eta(k) P(y(k)|x(k))$$
$$\int_X (P(x(k)|u(k-1), x(k-1)) P(x(k-1)|u(0:k-2), y(1:k-1))) dx(k-1)$$

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 measurement *y(k)*, update step:

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

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

Localization – two key decisions

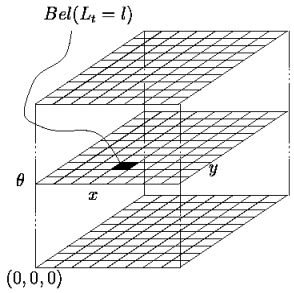
1. How do you represent the belief *P(x)*
2. How do you compute conditional probabilities

You are free to use any of the estimation strategies we have discussed so far. Popular ones are the EKF and Particle filters – the latter being particularly well suited to representing multi-modal distributions.

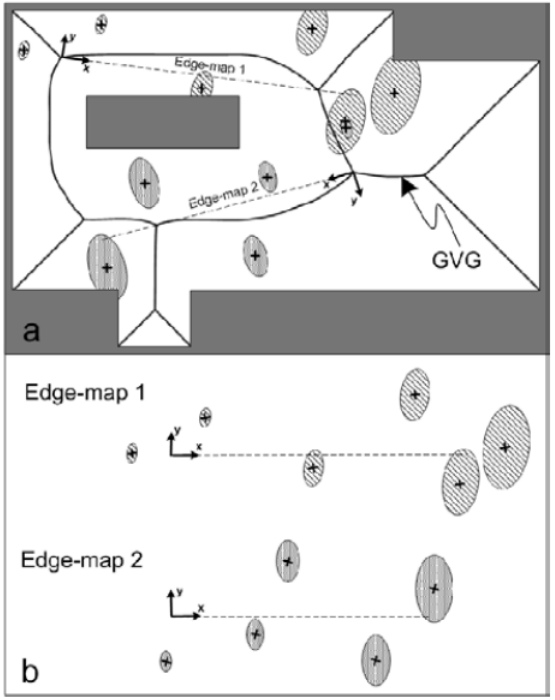
How to represent configuration space?

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

- Simple – use a grid



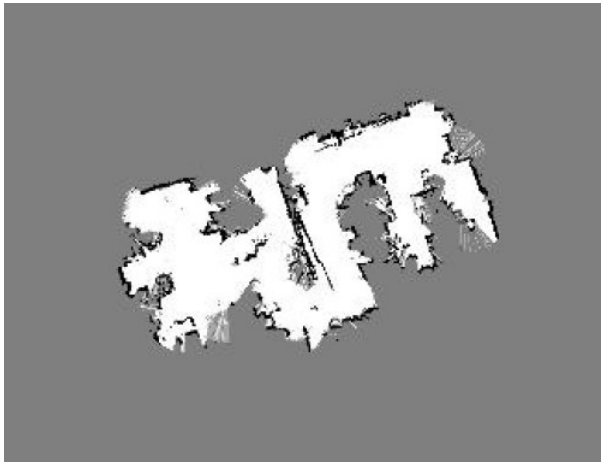
- Landmark based methods:
  e.g., landmark comes from a structure like (generalized) Voronoi graph



The Mapping Problem

- Robot must cope with two forms of uncertainty: noise in perception *(y)* and noise in odometry (*u*).
- Let is first assume that location is known – localization is solved
- Then we can come back and relax that assumption.
- 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
- Find *inverse sensor model*

$$p(m_x \mid y_t)$$

- Update odds that grid cells are occupied

Solving the complete SLAM problem

- Need to do localization and mapping simultaneously
- Two approaches:
  - Historical: Use EKF to estimate joint posteriors over maps and robot locations (Needs predefined landmarks)
  - Global optimization: e.g., consider locations as random variables and derive constraints between locations using overlapping measurements (parameter optimization, EM, etc.)
  - Neither one handles truly online applications, so many variations based on the Bayesian computational theme.

Bayesian SLAM

- Recursive filter for estimating robot positions and map:

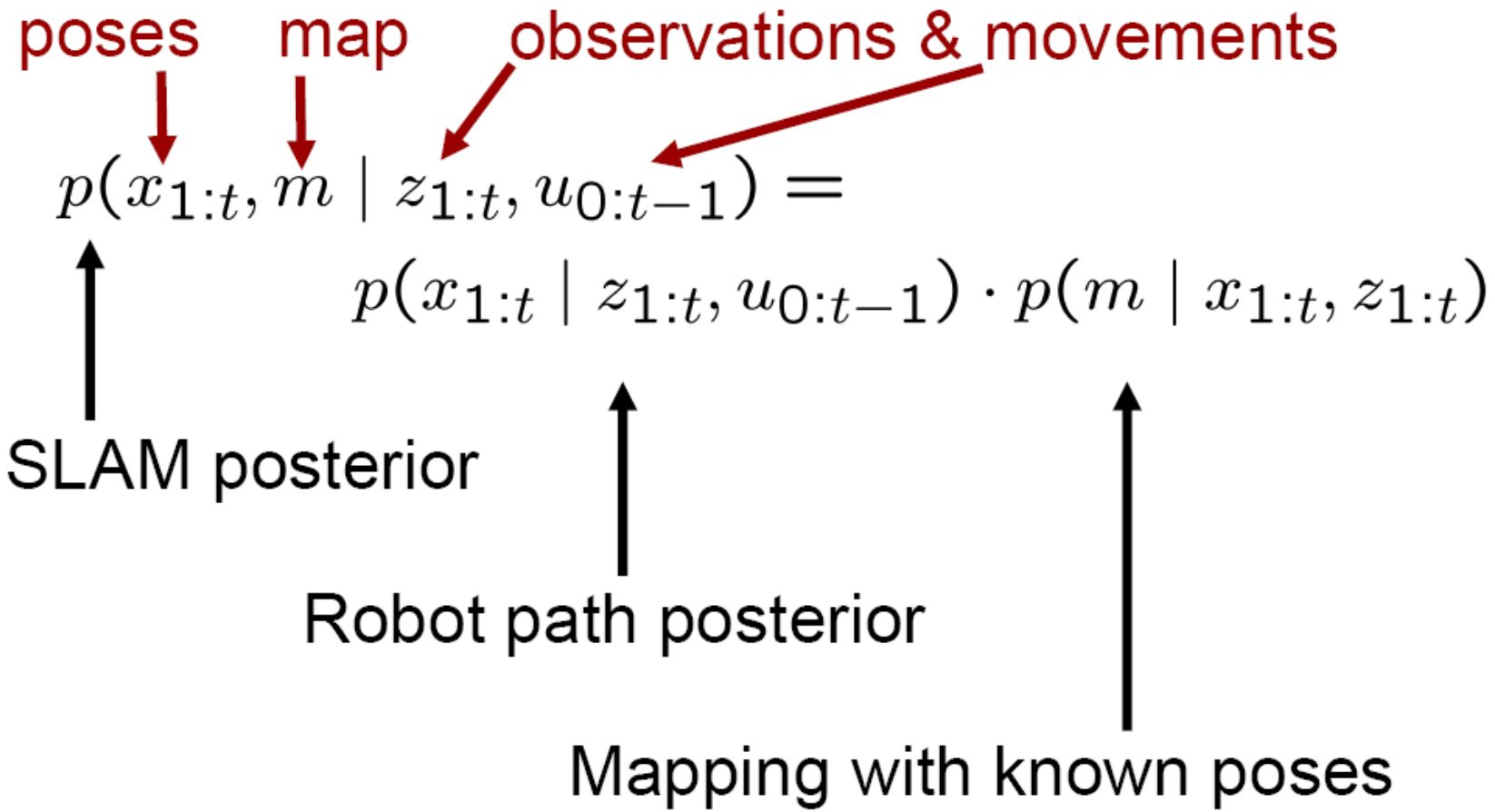
Sensor model

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

$$\int \Big(P(x(k)|u(k-1), x(k-1)) \quad \text{Motion model}$$

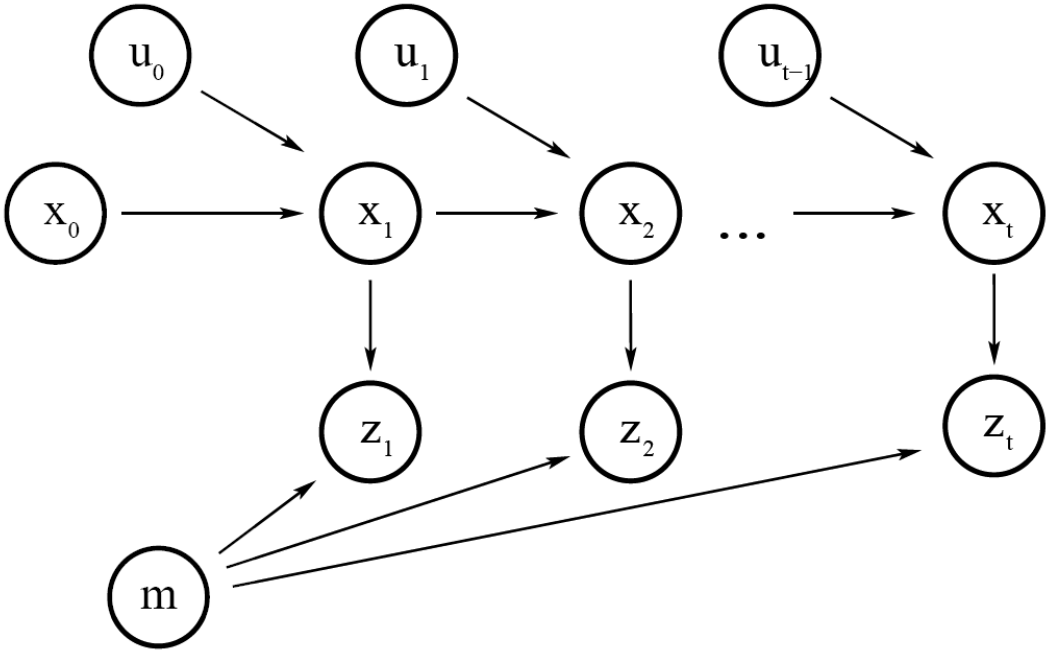
$$P(x(1:k-1), m | u(0:k-2), y(1:k-1)) \Big) dx(1:k-1)$$

Factorization: Rao-Blackwellization



Note: change in variable name – *z* for *y*

Graphical Model for Probabilistic SLAM



Pose Correction using Scan Matching

Maximize the likelihood of the i-th pose relative to the (i-1)-th pose

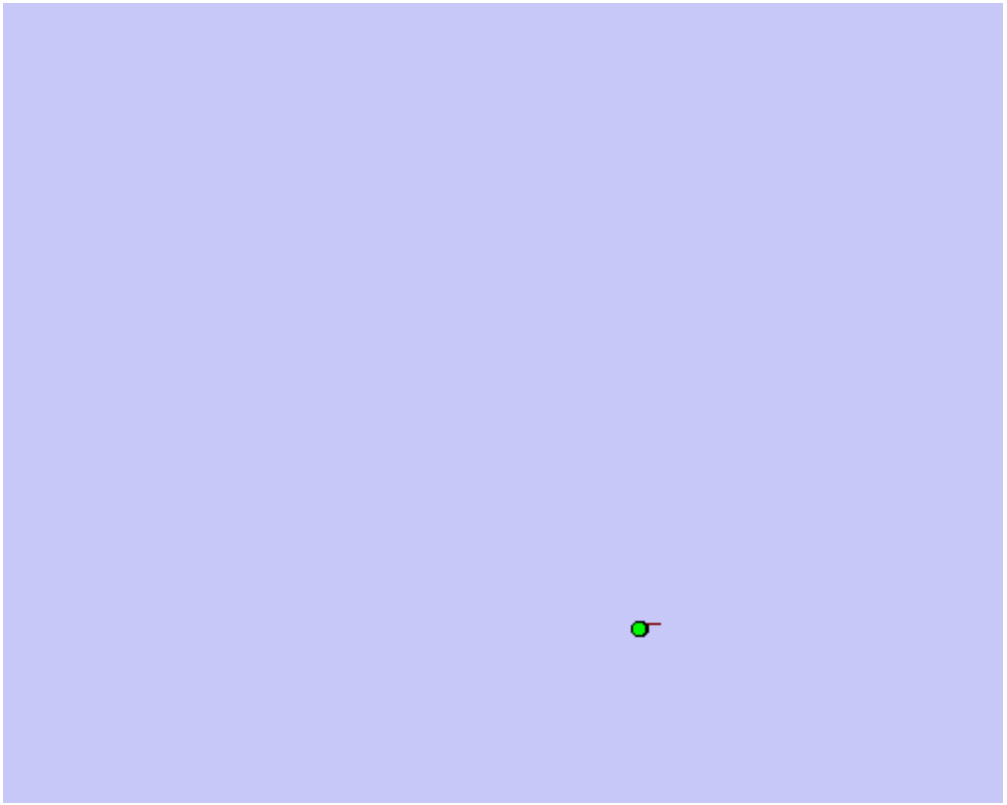
$$x_t^* = \operatorname*{argmax}_{x_t} p(z_t \mid x_t, m_{t-1}) \cdot p(x_t \mid x_{t-1}^*, u_{t-1})$$

current measurement

map constructed so far

robot motion

Demo



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