Statistical Techniques in Robotics (16-831, F09) Lecture #24 (November 12, 2009)

SLAM, Fast SLAM, and Rao-Blackwellization

Lecturer: Drew Bagnell *Scribe: Bryan Wagenknecht*

1 About SLAM

Simultaneous localization and mapping (SLAM) is one of the canonical problems in robotics. A robot set down in an unknown environment with noisy odometry and sensing needs to figure out where it is in the world and learn details about the world at the same time. It is a sort of chicken-egg problem where the mapping problem is relatively easy given exact knowledge of the robot's location, and localizing the robot is relatively easy given a good map of the environment. The challenge is found in performing these tasks simultaneously.

1.1 Variations on the SLAM problem

SLAM can be approached in two ways:

- Batch: given all data $y_{1:T}$ up front, solve for robot path $x_{1:T}$ and map M all at once (we won't cover this)
- Online/filtering: compute $\{x_t, m_t\}$ at each timestep as data y_t is received (we focus on this)

The mapping part of SLAM can also be approached in two ways:

- Filling an occupancy grid $M = \{m_1, m_2, ...\}$ (we'll come back to this)
- Tracking landmarks $l_i = (m_{xi}, m_{yi})$ (focus on this for now)

2 Basic Approaches to SLAM

2.1 PF over joint

The first naive solution is to build a Particle Filter to sample from the joint distribution

$$p(r_x, r_y, m_{x1}, m_{y1}, m_{x2}, m_{y2}, ...)$$

This becomes bad very quickly due to the high dimensionality of the state space when landmark position is included:

- PF doesn't scale well with dimensionality - particle sparsity means a particle that is good at describing one landmark is probably bad at describing others
- Observation distributions/densities get very sharp due to multiplying distributions

2.2 XKF over joint

The next best solution (often called the "classical solution") might be to use your favorite Kalman Filter (XKF here could mean EKF, UKF, etc.) to track all variables in one joint distribution

$$p(r_x, r_y, m_{x1}, m_{y1}, m_{x2}, m_{y2}, ...)$$

This results in a big covariance matrix with the marginal gaussian distribution of each landmark contained in blocks along the covariance diagonal. Off-diagonal terms are correlations from multiple landmarks appearing in the robot's view at once.

The motion update would look like:

$$\begin{bmatrix} r_x \\ r_y \\ m_{x1} \\ m_{y1} \\ \vdots \end{bmatrix}_{t+1} = \begin{bmatrix} \begin{bmatrix} A_r \end{bmatrix} & 0 \\ 0 & \begin{bmatrix} I \end{bmatrix} \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ m_{x1} \\ m_{y1} \\ \vdots \end{bmatrix}_t + \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \\ 0 \\ \end{bmatrix}$$

and the observation model would look like:

$$\begin{bmatrix} range_i \\ bearing_i \end{bmatrix} = \begin{bmatrix} \sqrt{(r_x - m_x)^2 + (r_y - m_y)^2} \\ atan2((r_y - m_y), (r_x - m_x)) \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \end{bmatrix}$$

2.2.1 Potential problems with this approach

- The solution could be multimodal, especially with bad odometry
- Data association: the filter must be able to distinguish between landmarks, which can cause problems with "loop closure" where the robot returns to a location it has already mapped and landmarks need to matched with those previously observed.

  Maximum Likelihood data association with hard thresholds is often used for this
- Scales as $O(n^2)$ where n is number of landmarks

2.2.2 Some hacks that could be helpful

- Use submaps (segment the large world map into smaller maps) and work with smaller chunks of the covariance matrix.
- Throw out confusing landmarks

3 SLAM as a Bayes Net

If we draw the Bayes Network representation of SLAM we get something like Figure 1. Here the data associations are presumed known and fixed.

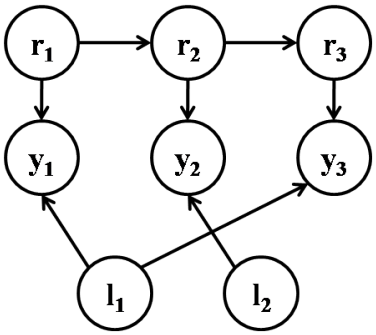

Figure 1: Bayes Network representation of SLAM. Robot states r_i form a Markov chain and data associations (which landmarks l_i are in view at each observation y_i) are presumed known and fixed. Conditioning on robot states causes independence between landmark locations.

3.1 Fast SLAM factorization

Investigation of the Bayes Net shows us that we can make the landmark locations independent by conditioning on the robot locations. This allows us to factorize the distribution as

$$p(r, m) = p(r) \cdot p(m|r) = p(r) \cdot \prod_i p(l_i|r)$$

Thus we can track the landmark locations independently by conditioning on the assumed robot position.

Fast SLAM is an algorithm that runs a particle filter in robot location space using augmented particles. Each particle carries a history of its past locations $r_1 ... r_t$ and an individual Kalman filter (consisting of μ_{l_i} and Σ_{l_i}) for each landmark l_i it sees. This is visualized in Figure 2.

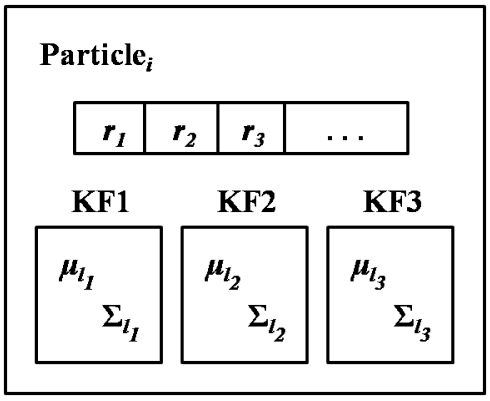

Figure 2: Fast SLAM particles include a history of their positions and individual Kalman filters for each landmark.

4 Smarter Approach: Fast SLAM

4.1 Fast SLAM 1.0

The first version of Fast SLAM runs a particle filter with augmented particles, then applies importance weighting to the particles using the latest estimates of any landmarks within view. The landmarks are tracked independently in Kalman filters within each particle.

4.1.1 Fast SLAM 1.0 Algorithm

1. Motion update: for all $particle_i$, move r_i according to motion model (leave KF's for landmarks alone)
2. Observation update: weight $particle_i$ by $p(y|associations\, l, r_i)$

$$p(y|associations\, l, r_i) = \frac{1}{\sqrt{(2\pi)^2\,|\Sigma_l|}} \exp\left[-\frac{1}{2}(y-\mu_y)^T \Sigma_l (y-\mu_y)\right]$$

3. Run KF update on each particle's estimate of landmark l
4. Resample particles

4.1.2 Timestep Complexity: $O(l)$

- Data association is $O(l)$ (use KD tree)
- Memory copy is $O(l)$

4.1.3 Occupancy Grids

- In theory each particle should carry its own map grid (instead of KF's)
- Practically this is too memory intensive, use data structure tricks: DP-SLAM

  Use a single map grid with data for all relevant particles in each cell

4.2 Fast SLAM 2.0

The problem with Fast SLAM is paricle deprivation, especially for loop closure. The key is to have a smarter motion model (sampling distribution).

4.2.1 Fast SLAM 2.0 Algorithm

1. Motion update: Sample robot position from Kalman filter $\tilde{p}(x_{t+1}|y_{t+1}, x_t$
2. Observation update: Weight $particle_i$ by

$$w_i \propto \frac{p(y_{t+1}|associations\, l, r_i)}{\tilde{p}(x_{t+1}|y_{t+1}, x_t)}$$

5 Wrap-up Topics

5.1 Data association

If we want to be lazy with data association (or we can't label the landmarks a priori) then we can track associations within the particle. Wrong assignments are corrected in retrospect for free through particle resampling.

- For each landmark l_i, sample associations from $p(y|l_i, r)$

5.2 Rao-Blackwell theorem

The Rao-Blackwell theorem says running an algorithm that samples from a distribution on one variable $p(x_1)$ then determines another variable x_2 analytically conditioned on x_1 will do no worse than an algorithm that samples from the joint distribution $p(x_1, x_2)$.

$$var\left[sampler(x_1, x_2)\right] \geq var\left[sampler(x_1) + does\, x_2|x_1\, analytically\right]$$

Fast SLAM is an example of Rao-Blackwellization since by using KF's inside particles, we don't have to sample landmark info from the whole joint distribution. In general Rao-Blackwellized filters make for the most efficient SLAM algorithms.