PRM Flavors

• What’s the main difficulty with PRM
• Sampling and connection strategies
  – Near or inside obstacles
  – Visibility
  – Importance
  – Lazy PRM
• Post-processing
  – Removing cycles
  – Path smoothing
Single vs Multiple Queries

- PRM: long to compute but can be used multiple times
  - as long as the environment stays the same

- Trees: Faster to compute but the algorithms are geared toward a single use (EST, RRT and their different flavors)
  - Also works for non-holonomic robots and kinodynamic planning
  - Using heuristics to “aim” toward the goal
  - Use two or more trees and keep them growing fast
Gaussian (or Normal) Distribution

Univariate

\[ p(x) \sim N(\mu, \sigma^2) \]

\[ p(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2} \frac{(x-\mu)^2}{\sigma^2}} \]

Multivariate

\[ p(x) \sim N(\mu, \Sigma) \]

\[ p(x) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} e^{-\frac{1}{2} (x-\mu)^T \Sigma^{-1} (x-\mu)} \]
Properties of Gaussians

\[ X \sim N(\mu, \sigma^2) \]  
\[ Y = aX + b \] \] \implies Y \sim N(a\mu + b, a^2\sigma^2) 

\[ X_1 \sim N(\mu_1, \sigma_1^2) \]  
\[ X_2 \sim N(\mu_2, \sigma_2^2) \] \] \implies X_1 + X_2 \sim N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2) 

- We stay in the “Gaussian world” as long as we start with Gaussians and perform only linear transformations.

- Same holds for multivariate Gaussians
Covariance, covariance, covariance

• Know what’s a covariance matrix
  – What it does, what it means, why it matters, how you can compute one, how you can visualize them

\[ \bar{x} = \frac{1}{N} \sum_{i=0}^{N} x_i \]
\[ e_i = x_i - \bar{x} \]
\[ \Sigma = E[e_i e_i^T] \]
Kalman Filter

- Recursive solution for discrete linear filtering problems
  - A state $x \in \mathbb{R}^n$
  - A measurement $z \in \mathbb{R}^m$
  - Discrete (i.e. for time $t = 1, 2, 3, \ldots$)
  - Recursive process (i.e. $x_t = f(x_{t-1})$)
  - Linear system (i.e. $x_t = A x_{t-1}$)

- The system is defined by:
  1) *linear* process model
     \[
     x_t = A x_{t-1} + B u_{t-1} + w_{t-1}
     \]
     state transition control Gaussian
     (optional) Input white noise
     How a state transitions into another state
  2) *linear* measurement model
     \[
     z_t = H x_t + v_t
     \]
     observation model Gaussian white noise
     How a state relates to a measurement
Kalman Filter

• Recipe:
  1. Start with an initial guess
     \( \hat{x}_0, \Sigma_0 \)
  2. Compute prior (prediction)
     \[
     \hat{x}'_t = A\hat{x}_{t-1} + Bu_{t-1} \\
     \Sigma'_t = A\Sigma_{t-1}A^T + Q
     \]
  3. Compute posterior (correction)
     \[
     K_t = \Sigma'_tH^T(H\Sigma'_tH^T + R)^{-1} \\
     \hat{x}_t = \hat{x}'_t + K_t(z_t - H\hat{x}'_t) \\
     \Sigma_t = (1 - K_tH)\Sigma'_t
     \]

Time update (predict)
Measurement update (correct)

Minimizes the sum of the expected errors:
\[ \Sigma \sum e_i^2 \]
Initial \( \hat{x}_0, \Sigma_0 \)

Predict \( \hat{x}_1 \) from \( \hat{x}_0 \) and \( u_0 \)
\[
\hat{x}_1' = A\hat{x}_0 + Bu_0 \\
\Sigma_1' = A\Sigma_0 A^T + Q
\]

Correction using \( z_1 \)
\[
K_1 = \Sigma_1'H^T(H\Sigma_1'H^T + R)^{-1} \\
\hat{x}_1 = \hat{x}_1' + K_1(z_1 - H\hat{x}_1') \\
\Sigma_1 = (1 - K_1H)\Sigma_1'
\]

Predict \( \hat{x}_2 \) from \( \hat{x}_1 \) and \( u_1 \)
\[
\hat{x}_2' = A\hat{x}_1 + Bu_1 \\
\Sigma_2' = A\Sigma_1 A^T + Q
\]
Kalman Filter Limitations

• Assumptions:
  – Linear process model
    \[ x_{t+1} = Ax_t + Bu_t + w_t \]
  – Linear observation model
    \[ z_t = Hx_t + v_t \]
  – White Gaussian noise
    \[ N(0, \Sigma) \]

• What can we do if system is not linear?
  – Non-linear state dynamics
    \[ x_{t+1} = f(x_t, u_t, w_t) \]
  – Non-linear observations
    \[ z_t = h(x_t, v_t) \]
Extended Kalman Filter

- Kalman Filter Recipe:
  - Given \( \hat{x}_0, \Sigma_0 \)
  - Prediction
    \[
    \hat{x}_t' = A\hat{x}_{t-1} + Bu_{t-1} \\
    \Sigma_t' = A\Sigma_{t-1}A^T + Q
    \]
  - Measurement correction
    \[
    K_t = \Sigma_t'H^T(\Sigma_t'H^T + R)^{-1} \\
    \hat{x}_t = \hat{x}_t' + K(z_t - H \hat{x}_t') \\
    \Sigma_t = (I - K_tH)\Sigma_t'
    \]

- Extended Kalman Filter Recipe:
  - Given \( \hat{x}_0, \Sigma_0 \)
  - Prediction
    \[
    \hat{x}_t' = f(\hat{x}_{t-1}, u_{t-1}, 0) \\
    \Sigma_t' = A_t\Sigma_{t-1}A_t^T + W_tQW_t^T
    \]
  - Measurement correction
    \[
    K_t = \Sigma_t'H_t^T(H_t\Sigma_t'H_t^T + V_tR_tV_t^T)^{-1} \\
    \hat{x}_t = \hat{x}_t' + K_t(z_t - h(\hat{x}_t', 0)) \\
    \Sigma_t = (I - K_tH_t)\Sigma_t'
    \]
EKF for Range-Bearing Localization

- State $s_t = [x_t \ y_t \ \theta_t]^T$ 2D position and orientation
- Input $u_t = [v_t \ \omega_t]^T$ linear and angular velocity
- Process model

$$f(s_t, u_t, w_t) = \begin{bmatrix} x_{t-1} + (\Delta t)v_{t-1}\cos(\theta_{t-1}) \\ y_{t-1} + (\Delta t)v_{t-1}\sin(\theta_{t-1}) \\ \theta_{t-1} + (\Delta t)\omega_{t-1} \end{bmatrix} + \begin{bmatrix} w_{x_t} \\ w_{y} \\ w_{\theta_t} \end{bmatrix}$$

- Given a map, the robot sees $N$ landmarks with coordinates $l_1 = [x_{l_1} \ y_{l_1}]^T, \cdots, l_N = [x_{l_N} \ y_{l_N}]^T$

The observation model is

$$z_t = \begin{bmatrix} h_1(s_t, v_1) \\ \vdots \\ h_N(s_t, v_N) \end{bmatrix} \quad h_i(s_t, v_t) = \begin{bmatrix} \sqrt{(x_t - x_{l_i})^2 + (y_t - y_{l_i})^2} \\ \tan^{-1}\left(\frac{y_t - y_{l_i}}{x_t - x_{l_i}}\right) - \theta_t \end{bmatrix} + \begin{bmatrix} v_r \\ v_b \end{bmatrix}$$
Kalman Filters and SLAM

- Localization: state is the location of the robot
- Mapping: state is the location of 2D landmarks
- SLAM: state combines both
- If the state is $s_t = [x_t \; y_t \; \theta_t \; l_{1_t}^T \; \ldots \; l_{N_t}^T]^T$
  then we can write a linear observation system
  - note that if we don’t have some fixed landmarks, our system is *unobservable* (we can’t fully determine all unknown quantities)

- Covariance $\Sigma$ is represented by http://ais.informatik.uni-freiburg.de

| $\sigma_{xx}$ | $\sigma_{xy}$ | $\sigma_{x\theta}$ |
| $\sigma_{yx}$ | $\sigma_{yy}$ | $\sigma_{y\theta}$ |
| $\sigma_{\theta x}$ | $\sigma_{\theta y}$ | $\sigma_{\theta \theta}$ |
| $\sigma_{m_1,x}$ | $\sigma_{m_1,y}$ | $\sigma_{\theta}$ |
| $\sigma_{m_1,y}$ | $\sigma_{m_1,y}$ | $\sigma_{\theta}$ |
| $\vdots$ | $\vdots$ | $\vdots$ |
| $\sigma_{m_n,x}$ | $\sigma_{m_n,y}$ | $\sigma_{\theta}$ |
| $\sigma_{m_n,y}$ | $\sigma_{m_n,y}$ | $\sigma_{\theta}$ |
EKF Range Bearing SLAM

Prior State Estimation

- State $s_t = [x_t \ y_t \ \theta_t \ l_1^{T} \ \cdots \ l_N^{T}]$ position/orientation of robot and landmarks coordinates
- Input $u_t = [v_t \ \omega_t]^T$ forward and angular velocity
- The process model for localization is

$$s'_t = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \Delta tv_{t-1}\cos(\theta_{t-1}) \\ \Delta tv_{t-1}\sin(\theta_{t-1}) \\ \Delta t\omega_{t-1} \end{bmatrix}$$

This model is augmented for $2N+3$ dimensions to accommodate landmarks. This results in the process equation

$$\begin{bmatrix} x'_t \\ y'_t \\ \theta'_t \\ l'_{1t} \\ \vdots \\ l'_{Nt} \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \\ l_{1t-1} \\ \vdots \\ l_{Nt-1} \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta tv_{t-1}\cos(\theta_{t-1}) \\ \Delta tv_{t-1}\sin(\theta_{t-1}) \\ \Delta t\omega_{t-1} \end{bmatrix}$$

Landmarks don’t depend on external input.
EKF Range Bearing SLAM
Prior Covariance Update

We assume static landmarks. Therefore, the function \( f(s,u,w) \) only affects the robot’s location and not the landmarks.

\[
\Sigma'_t = A_t \Sigma_{t-1} A_t^T + W_t Q W_t^T
\]

The motion of the robot does not affect the coordinates of the landmarks.

Jacobian of the process model

\[
A = \begin{bmatrix}
\frac{\partial x}{\partial x} & \frac{\partial x}{\partial y} & \frac{\partial x}{\partial \theta} \\
\frac{\partial x}{\partial x} & \frac{\partial y}{\partial y} & \frac{\partial y}{\partial \theta} \\
\frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial y} & \frac{\partial \theta}{\partial \theta} \\
\frac{\partial x}{\partial x} & \frac{\partial y}{\partial y} & \frac{\partial \theta}{\partial \theta}
\end{bmatrix}
\]

Jacobian of the robot motion

\[
A = \begin{bmatrix}
\frac{\partial x}{\partial x} & \frac{\partial x}{\partial y} & \frac{\partial x}{\partial \theta} \\
\frac{\partial x}{\partial x} & \frac{\partial y}{\partial y} & \frac{\partial y}{\partial \theta} \\
\frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial y} & \frac{\partial \theta}{\partial \theta} \\
\frac{\partial x}{\partial x} & \frac{\partial y}{\partial y} & \frac{\partial \theta}{\partial \theta}
\end{bmatrix}
\]

2Nx2N identity

\[
A = \begin{bmatrix}
\frac{\partial x}{\partial x} & \frac{\partial x}{\partial y} & \frac{\partial x}{\partial \theta} \\
\frac{\partial x}{\partial x} & \frac{\partial y}{\partial y} & \frac{\partial y}{\partial \theta} \\
\frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial y} & \frac{\partial \theta}{\partial \theta} \\
\frac{\partial x}{\partial x} & \frac{\partial y}{\partial y} & \frac{\partial \theta}{\partial \theta}
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]
Bayes Filter

- Given a sequence of measurements $z_1, \ldots, z_k$
- Given a sequence of commands $u_0, \ldots, u_{k-1}$
- Given a sensor model $P(z_k \mid x_k)$
- Given a dynamic model $P(x_k \mid x_{k-1})$
- Given a prior probability $P(x_0)$
- Find $P(x_k \mid z_{1:k}, u_{0:k-1})$
Bayesian Localization

- Recall Bayes Theorem:
  \[ P(x | z) = \frac{P(z|x)P(x)}{P(z)} \]
- Also remember conditional independence
- Think of \(x\) as the state of the robot and \(z\) as the data we know

\[
P(x_k | u_{0:k-1}, z_{1:k}) = \frac{P(z_k | x_k, u_{0:k-1}, z_{1:k-1})P(x_k | u_{0:k-1}, z_{1:k-1})}{P(z_k | u_{0:k-1}, z_{1:k-1})}
\]

\[
= \eta_k P(z_k | x_k) \int_{x_{k-1}} P(x_k | u_{k-1}, x_{k-1})P(x_{k-1} | u_{0:k-2}, z_{1:k-1}) dx_{k-1}
\]

- observation
- state prediction
- recursion
Predict Motion (Prior Distribution)

- Suppose we have $P(x_k)$
- We have $P(x_{k+1}|x_k, u_k)$
- Put together

$$P(x_{k+1}) = \int_{x_k} P(x_{k+1}|x_k, u_k)P(x_k)dx_k$$

What is the probability distribution for $x_{k+1}$ given the command $u_k$ and all the previous states $x_k$?
System Model

State space \( X = \{ 1, 2, 3, 4 \} \)

Transition matrix: The probability \( P(j|i) \) of moving from \( i \) to \( j \) is given by \( P_{i,j} \). Each row must sum to 1.

Prior probability distribution \( P(x_k) \)

Compute \( P(x_{k+1}) = \int_{x_k} P(x_{k+1}|x_k, u_k) \, dx_k \)
Observation Model

- How likely is the measurement $z_k$ given a state $x_k$?
- The measurement $z_k$ is what we get. So we have to stick by it.
- However, not all states will likely produce the measurement.
- We’re not interested in finding the most likely state. *We want the whole distribution*.

$$P(z_k|x_k)$$ Doesn’t have to be Gaussian
Observation Model
Beam Model

• Mixing all these cases together we get

\[
P(z_k | x_k, m) = \begin{bmatrix} w_{hit} \\ w_{short} \\ w_{max} \\ w_{rand} \end{bmatrix}^T \begin{bmatrix} P_{hit}(z_k | x_k, m) \\ P_{short}(z_k | x_k, m) \\ P_{max}(z_k | x_k, m) \\ P_{rand}(z_k | x_k, m) \end{bmatrix}
\]

where the weights are parameters

\[w_{hit} + w_{short} + w_{max} + w_{rand} = 1\]
Likelihood Field Model

\[ P_{\text{hit}}(z_k | x_k, m) = \epsilon \sigma_{\text{hit}}^2 (d^2) \]

(a) Gaussian distribution \( p_{\text{hit}} \)

Probabilistic Robotic
Discrete Bayes Filter Algorithm

Algorithm Discrete_Bayes_filter( \( u_{0:k-1}, y_{1:k}, P(x_0) \) )

1. \( P(x) = P(x_0) \)  
   (if you don’t know: uniform distribution)

2. for \( i=1:k \)

3. for all states \( x \in X \)
   \[ P'(x) = \sum_{x' \in X} P(x | u, x') P(x') \]  
   Prediction given prior dist. and command

4. end for

5. \( \eta = 0 \)  
   constant

6. for all states \( x \in X \)
   \[ P(x) = P(z | x) P'(x) \]  
   Update using measurement

7. end for

8. \( \eta = \eta + P(x) \)

9. end for

10. for all states \( x \in X \)

11. \[ P(x) = P(x) / \eta \]  
   Normalize to 1

12. end for
Particle Filter

- Computing $P(z_k|x_k, m)$ and $P(x_{k+1}|x_k, u_k)$ is not easy
  - In practice, it is never directly computable
  - Need to propagate an entire conditional distribution, not just one state like we did with Kalman,
- Represent probability distribution by random samples
- Estimation of non-Gaussian, nonlinear processes
Condensation Algorithm

\[ P(x_k | u_{0:k-1}, z_{1:k}) = \eta_k P(z_k | x_k) \int_{x_{k-1}} P(x_k | u_{k-1}, x_{k-1}) P(x_{k-1} | u_{0:k-2}, z_{1:k-1}) dx_{k-1} \]
Monte Carlo Localization

- Note that $\bar{X}$ is a discrete representation of the probability of robot location

```
1: Algorithm MCL($\mathcal{X}_{t-1}, u_t, z_t, m$):
2: $\bar{X}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to $M$ do
4:     $x_t^{[m]} = \text{sample\_motion\_model}(u_t, x_t^{[m-1]})$
5:     $w_t^{[m]} = \text{measurement\_model}(z_t, x_t^{[m]}, m)$
6:     $\bar{X}_t = \bar{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m = 1$ to $M$ do
9:     draw $i$ with probability $\propto w_t^{[i]}$
10:     add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$
```
Occupancy Grid Map

- Represent environment by a grid.
- Estimate the probability that a location is occupied by an obstacle.
- Key assumptions
  - Occupancy of individual cells is independent
- Gold standard
  \[ P(m | z_{1:k}, x_{1:k}) \]
- Partition the space
  \[ m = \{m_i\} \]
  where \( m_i \) is the grid cell with index \( i \) and is “1” for occupied or “0”
Rao-Blackwellized SLAM

Compute a posterior over the map and possible trajectories of the robot:

\[
p(x_{1:k}, m | z_{1:k}, u_{0:k-1})
= p(m | x_{1:k}, z_{1:k}, u_{0:t-1}) p(x_{1:k} | z_{1:k}, u_{0:k-1})
\]

mapping

map

robot motion

trajectory

measurements

Localization

Compute a posterior over the map and possible trajectories of the robot.
Rao-Blackwellized SLAM

- Break it down even further if a map $m_k$ consists of $N$ individual landmarks $l_i$

\[
P(x_{1:k}, m_k | z_{1:k}, u_{0:k-1}) = P(x_{1:k} | z_{1:k}, u_{0:k-1}) P(m | x_{1:k}, z_{1:k}, u_{0:k-1}) \left( \prod_i P(l_i | x_{1:k}, z_{1:k}, u_{0:k-1}) \right)
\]

- Rao-Blackwellized particle filter (RBPF) maintains an individual map for each sample and updates this map based on the trajectory estimate of the sample

- Landmarks are filtered individually and have low dimensionality

- If $M$ particles with $N$ landmarks there is $NM$ landmark filters
FastSLAM

Robot Pose | Kalman Filters
---|---
Particle #1: \(x, y, \theta\) | \(\mu_1, \Sigma_1\) | \(\mu_2, \Sigma_2\) | ... | \(\mu_N, \Sigma_N\)
Particle #2: \(x, y, \theta\) | \(\mu_1, \Sigma_1\) | \(\mu_2, \Sigma_2\) | ... | \(\mu_N, \Sigma_N\)
Particle #3: \(x, y, \theta\) | \(\mu_1, \Sigma_1\) | \(\mu_2, \Sigma_2\) | ... | \(\mu_N, \Sigma_N\)
... | ... | ... | ... | ...
Particle M: \(x, y, \theta\) | \(\mu_1, \Sigma_1\) | \(\mu_2, \Sigma_2\) | ... | \(\mu_N, \Sigma_N\)

[Begin courtesy of Mike Montemerlo]