Chapter 9: Bayesian Methods

CS 336/436
Gregory D. Hager
A Simple Example
Why Not Just Use a Kalman Filter?
Recall Robot Localization

• Given
  – Sensor readings $y_1, y_2, \ldots, y_k = y_{1:k}$
  – Known control inputs $u_0, u_2, \ldots, u_k = u_{0:k}$
  – Known model $P(x_{t+1} \mid x_t, u_t)$ with initial $P(x_1 \mid u_0)$
  – Known map $P(y_t \mid x_t)$ Most likely sensor reading given state $x$

• Compute
  – $P(x_t \mid y_{1:t-1}, u_{0:t-1})$ Most likely state $x$ at time $t$ given a sequence of commands $u_{0:t-1}$ and measurements $y_{1:t-1}$

This is just a probabilistic representation of what you’ve already learned!

Let’s try to connect the dots and do a couple of examples
Some Probability Reminders

- \( P(x, y) = P(x | y) \ P(y) \)
- \( P(x) = \sum_y P(x, y) = \sum_y P(x | y) \ P(y) \)
- If \( x \) is independent of \( z \) given \( y \)
  - \( P(x | y, z) = P(x | y) \)
- Two important assumptions
  1. Markov
     \[
     P(x_k | x_{k-1}, \ldots, x_0) = P(x_k | x_{k-1})
     \]
  2. Observation
     \[
     P(y_k | x_k, \ldots, x_0) = P(y_k | x_k)
     \]
Bayes Filter

- Given a sequence of measurements \( y_1, \ldots, y_k \) and a sequence of commands \( u_0, \ldots, u_{k-1} : \{ y_{1:k}, u_{0:k-1} \} \)
- Given a sensor model \( P( y_k | x_k ) \)
- Given a dynamic model \( P( x_k | x_{k-1} ) \)
- Given a prior probability \( P(x_0) \)

Find \( P(x_k | y_{1:k}, u_{0:k-1}) \)
Bayesian Localization

- Recall Bayes Theorem:
  \[ P(x \mid y) = \frac{P(y \mid x) P(x)}{P(y)} \]

- Also remember conditional independence

- Think of \( x \) as the state of the robot and \( y \) as the data we know

\[
P(x_k \mid u_{0:k-1}, y_{1:k}) = \frac{P(y_k \mid x_k, u_{0:k-1}, y_{1:k-1}) P(x_k \mid u_{0:k-1}, y_{1:k-1})}{P(y_k \mid u_{0:k-1}, y_{1:k-1})} = \eta_k P(y_k \mid x_k) \int_{x_{k-1}} P(x_k \mid u_{k-1}, x_{k-1}) P(x_{k-1} \mid u_{0:k-2}, y_{1:k-1})
\]

- Observation
- State prediction
- Recursive instance
A Simple Example
Ways of Representing Probabilities

• We have already seen Kalman filters
  — represent probabilities as Gaussians
  — Gaussians are *conjugate* distributions

• How arbitrary distributions are represented?
  — Mixtures of Gaussians

• Suppose instead state space is partitioned and probability in partition is constant

  - \( P(x \mid y) = P(y \mid x) P(x) / P(y) \approx \eta \frac{P(x_i \mid y)}{P(x)} = \frac{\eta \frac{P(x \mid x_i)}{P(x)}}{P(x)} \)
  - \( \eta = \sum_i P(y \mid x_i) P(x_i) \)
  - Thus, updating from observations is a simple multiplication of prior probability by likelihood of observation

  - \( P(x_i(k) \mid u(k-1:0), y(k-1)) = \sum_j P(x_i(k) \mid u(k-1), x_j(k-1)) P(x_j(k-1) \mid u(k-2:0), y(k-1)) \)
  - Thus, updating using dynamical model is simply a discrete convolution (blurring) of the prior by the driving noise of the planned motion

How Do We Think about Motion?

- Suppose we have $P(x_k)$
- We have $P(x_{k+1} \mid x_k, u_k)$

- Put together
- $P(x_{k+1}) = \int P(x_{k+1} \mid x_k, u_k) P(x_k) \, dx_k$

What is the probability distribution for $x_{k+1}$ given the command $u_k$ and all the previous states $x_k$?
Updating from observations is a simple multiplication of prior probability by likelihood of observation.

Updating using dynamical model is simply a discrete convolution (blurring) of the prior by the driving noise of the planned motion.

Updating from observations is a simple multiplication of prior probability by likelihood of observation.

Updating using dynamical model is simply a discrete convolution (blurring) of the prior by the driving noise of the planned motion.
Piecewise Constant Representation (Mobile Robot)

Position of a mobile robot: \((x, y, \theta)\)

\( Bel(x_t =< x, y, \theta) \)
Propagating Motion

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

Prior $P(x_k)$

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

Compute

$$P(x_{k+1}) = \int P(x_{k+1} \mid x_k, u_k) P(x_k) dx_k$$
Propagating Motion

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

| $P(x_{k+1} | x_k, u_k)$  | $x_{k+1}=1$ | $x_{k+1}=2$ | $x_{k+1}=3$ | $x_{k+1}=4$ |
|-------------------------|-------------|-------------|-------------|-------------|
| $x_k=1$                 | 0.25        | 0.5         | 0.25        | 0           |
| $x_k=2$                 | 0           | 0.25        | 0.5         | 0.25        |
| $x_k=3$                 | 0           | 0           | 0.25        | 0.75        |
| $x_k=4$                 | 0           | 0           | 0           | 1           |

$$P(x_{k+1} = 1) = \sum_{x' \in X} P(x_{k+1} = 1 | x'_k, u_k) P(x'_k)$$

$$= P(x_{k+1} = 1 | x'_k = 1, u_k) P(x'_k = 1) + \ldots + P(x_{k+1} = 1 | x'_k = 4, u_k) P(x'_k = 4)$$

$$= 0.25 \times 0 + 0 \times 0.5 + 0 \times 0 + 0 \times 0.5$$

$$= 0$$
Propagating Motion

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

\[
\begin{array}{c|cccc}
X_k=1 & X_{k+1}=1 & X_{k+1}=2 & X_{k+1}=3 & X_{k+1}=4 \\
\hline
0.25 & 0.5 & 0.25 & 0 & \\
0 & 0.25 & 0.5 & 0.25 & \\
0 & 0 & 0.25 & 0.75 & \\
0 & 0 & 0 & 1 & \\
\end{array}
\]

Prior \( P(x_k) \)

\[
P(x_{k+1} = 2) = \sum_{x' \in X} P(x_{k+1} = 2 \mid x'_k, u_k) P(x'_k)
\]

\[
= P(x_{k+1} = 2 \mid x'_k = 1, u_k) P(x'_k = 1) + P(x_{k+1} = 2 \mid x'_k = 2, u_k) P(x'_k = 2) + P(x_{k+1} = 2 \mid x'_k = 3, u_k) P(x'_k = 3) + P(x_{k+1} = 2 \mid x'_k = 4, u_k) P(x'_k = 4)
\]

\[
= 0.5 \times 0 + 0.25 \times 0.5 + 0 \times 0 + 0 \times 0.5
\]

\[
= 0.125
\]
Propagating Motion

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

\[
P(x_{k+1} = 3) = \sum_{x' \in X} P(x_{k+1} = 3 \mid x'_k, u_k) P(x'_k)
\]

\[
= P(x_{k+1} = 3 \mid x'_k = 1, u_k) P(x'_k = 1) + P(x_{k+1} = 3 \mid x'_k = 2, u_k) P(x'_k = 2) + P(x_{k+1} = 3 \mid x'_k = 3, u_k) P(x'_k = 3) + P(x_{k+1} = 3 \mid x'_k = 4, u_k) P(x'_k = 4)
\]

\[
= 0.25 \times 0 + 0.5 \times 0.5 + 0.25 \times 0 + 0 \times 0.5
\]

\[
= 0.25
\]
Propagating Motion

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

<table>
<thead>
<tr>
<th>$P(x_{k+1} \mid x_k, u_k)$</th>
<th>$x_{k+1}=1$</th>
<th>$x_{k+1}=2$</th>
<th>$x_{k+1}=3$</th>
<th>$x_{k+1}=4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_k=1$</td>
<td>0.25</td>
<td>0.5</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>$x_k=2$</td>
<td>0</td>
<td>0.25</td>
<td>0.5</td>
<td>0.25</td>
</tr>
<tr>
<td>$x_k=3$</td>
<td>0</td>
<td>0</td>
<td>0.25</td>
<td>0.75</td>
</tr>
<tr>
<td>$x_k=4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Prior $P(x_k)$

$$P(x_{k+1} = 4) = \sum_{x' \in X} P(x_{k+1} = 4 \mid x_k', u_k) P(x'_k)$$

$$= P(x_{k+1} = 4 \mid x_k' = 1, u_k) P(x_k' = 1) +$$

$$P(x_{k+1} = 4 \mid x_k' = 2, u_k) P(x_k' = 2) +$$

$$P(x_{k+1} = 4 \mid x_k' = 3, u_k) P(x_k' = 3) +$$

$$P(x_{k+1} = 4 \mid x_k' = 4, u_k) P(x_k' = 4)$$

$$= 0 \times 0 + 0.25 \times 0.5 + 0.75 \times 0 + 1 \times 0.5$$

$$= 0.625$$
Propagating Motion

\[ P(x_{k+1}) = \int P(x_{k+1} \mid x_k, u_k) P(x_k) dx_k \]
Discrete Bayes Filter Algorithm

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

1. \( P(x) = P(x_0) \)
2. for i=1:k
3. \quad \text{for all states } x \in X
4. \quad \quad P'(x) = \sum_{x' \in X} P(x \mid u_{i-1}, x') P(x') \quad \text{Prediction given prior dist. and command}
5. \quad \text{end for}
6. \eta = 0
7. \quad \text{for all states } x \in X
8. \quad \quad P(x) = P(y_i \mid x) P'(x) \quad \text{Update using measurement}
9. \quad \eta = \eta + P(x)
10. \quad \text{end for}
11. \quad \text{for all states } x \in X \quad \text{Normalize to 1}
12. \quad P(x) = P(x) / \eta
13. \quad \text{end for}
14. \quad \text{end for}
Note About the Posterior

• It is a probability distribution

\[
P(x)
\]

• What do we do with it?
  – Maximum likelihood: \( \arg \max P(z \mid x) \)
  – Mean Squared Error: \( E[(P(x) - P(\hat{x}))^2] \)
Bayes Filter Ingredients

• Motion model
  \[ \mathcal{P}(x_k \mid x_{k-1}, u_{k-1}) \]

• Observation model
  \[ \mathcal{P}(y_k \mid x_k) \]

• Bayes estimator
  MAP, MSE,
How To Get Likelihoods?

- How do we get $p(y \mid x)$?

- In the discrete case, $x$ is a fixed value

- For a fixed value and *known* map, we can predict/simulate sensor readings $y^*$ (recall $y = h(x) + v$)

- But, we know $y^* - y \sim v$ for whatever distribution $v$ has
  - $v$ is Gaussian with covariance $\Lambda$, then $P(y \mid x) = G(y^*-y ; 0, \Lambda)$
  - $v$ could be represented with an empirical histogram; $P(y \mid x)$ is a table lookup
Grid-based Localization
Sonars and Occupancy Grid Map
# Localization Algorithms - Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>Kalman filter</th>
<th>Multi-hypothesis tracking</th>
<th>Grid-based (fixed/variable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensors</td>
<td>Gaussian</td>
<td>Gaussian</td>
<td>Non-Gaussian</td>
</tr>
<tr>
<td>Posterior</td>
<td>Gaussian</td>
<td>Multi-modal</td>
<td>Piecewise constant</td>
</tr>
<tr>
<td>Efficiency (memory)</td>
<td>++</td>
<td>++</td>
<td>-/+</td>
</tr>
<tr>
<td>Efficiency (time)</td>
<td>++</td>
<td>++</td>
<td>0/+</td>
</tr>
<tr>
<td>Implementation</td>
<td>+</td>
<td>0</td>
<td>+/-</td>
</tr>
<tr>
<td>Accuracy</td>
<td>++</td>
<td>++</td>
<td>+/++</td>
</tr>
<tr>
<td>Robustness</td>
<td>-</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Global localization</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Particle Filters

- Represent belief by random samples
- Estimation of non-Gaussian, nonlinear processes

- Monte Carlo filter, Survival of the fittest, Condensation, Bootstrap filter, Particle filter

- Filtering: [Rubin, 88], [Gordon et al., 93], [Kitagawa 96]
- Computer vision: [Isard and Blake 96, 98]
- Dynamic Bayesian Networks: [Kanazawa et al., 95]
Sample-based Density Representation
Particle Filter Algorithm

\[
\text{Bel} \ (x_t) = p(z_t \mid x_t) \ p(x_t \mid x_{t-1}, u_{t-1}) \ Bel \ (x_{t-1}) \ dx_{t-1}
\]

\[\text{draw } x_{t-1}^i \text{ from Bel}(x_{t-1})\]

\[\text{draw } x_t^i \text{ from } p(x_t \mid x_{t-1}^i, u_{t-1})\]

\[\text{Importance factor for } x_t^i: \]

\[
w_t^i = \frac{\text{target distribution}}{\text{proposal distribution}} = \frac{p(z_t \mid x_t) \ p(x_t \mid x_{t-1}, u_{t-1}) \ Bel \ (x_{t-1})}{p(x_t \mid x_{t-1}, u_{t-1}) \ Bel \ (x_{t-1})} = \mu \ p(z_t \mid x_t)
\]
Monte Carlo Localization

- Monte-Carlo-Localization(a, z, N, map)
  - S = N samples from P(X(t)) from previous call
  - for i = 1 to N
    - S[i] = sample from P(X(t+1) | X(t) = S[i], A = a)
    - W[i] = 1
    - for j = 1 to M do
      - z* = expected-sensor-reading(j, S[i], map)
      - W[i] = W[i] * P(Z = z(j) | Z* = Z*)
  - S = weighted-sample-with-replacement(N, S, W)
  - return S

- Note that S is a discrete representation of the probability of robot location
The Likelihood Function

- Generating the sensor likelihood is essentially a sensor simulation
  - can be expensive
  - pre-compute
  - approximate

- A good fast approximation is often a weighted sum of
  - a nominal model that is fast to compute
  - other deviations that are modeled as random elements
Proximity Sensor Model

Tuned model that takes into account normal reflection, unexpected returns and randomness and out of range

Laser sensor

Sonar sensor
Resampling

- **Given**: Set $S$ of weighted samples.

- **Wanted**: Random sample, where the probability of drawing $x_i$ is given by $w_i$.

- Typically done $n$ times with replacement to generate new sample set $S'$. 
Resampling

- Roulette wheel
- Binary search, log n

- Stochastic universal sampling
- Systematic resampling
- Linear time complexity
- Easy to implement, low variance
Resampling Algorithm

1. Algorithm `systematic_resampling(S,n)`:

2. \( S' = c_1 = w_1 \)

3. **For** \( i = 2 \ldots n \) **Generate cdf**

4. \( c_i = c_{i-1} + w_i \)

5. \( u_1 \sim U[0, 1/n], i = 1 \) **Initialize threshold**

6. **For** \( j = 1 \ldots n \) **Draw samples ...**

7. **While** \( u_j > c_i \) **Skip until next threshold reached**

8. \( i = i + 1 \)

9. \( S' = S' \{ < x_i, 1/n > \} \) **Insert**

10. \( u_j = u_j + 1/n \) **Increment threshold**

11. **Return** \( S' \)

Also called **stochastic universal sampling**
Motion Model Reminder

Start

10 meters
Sample-based Localization (sonar)
The Minerva Experience
Using Ceiling Maps for Localization

[Dellaert et al. 99]
Ceiling Light Localization
Vision-based Localization

\[ P(z|x) \]

\[ h(x) \]
Under a Light

Measurement $z$: $P(z|x)$:
Next to a Light

Measurement $z$:  

$P(z|x)$:
Elsewhere

Measurement $z$: $P(z|x)$:
Sample-based Localization Demos

http://www.cs.washington.edu/ai/Mobile_Robotics//mcl/
# Localization Algorithms - Comparison

<table>
<thead>
<tr>
<th></th>
<th>Kalman filter</th>
<th>Multi-hypothesis tracking</th>
<th>Topological maps</th>
<th>Grid-based (fixed/variable)</th>
<th>Particle filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensors</td>
<td>Gaussian</td>
<td>Gaussian</td>
<td>Features</td>
<td>Non-Gaussian</td>
<td>Non-Gaussian</td>
</tr>
<tr>
<td>Posterior</td>
<td>Gaussian</td>
<td>Multi-modal</td>
<td>Piecewise constant</td>
<td>Piecewise constant</td>
<td>Samples</td>
</tr>
<tr>
<td>Efficiency (memory)</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>-/+</td>
<td>+/-/++</td>
</tr>
<tr>
<td>Efficiency (time)</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>o/+</td>
<td>+/-/++</td>
</tr>
<tr>
<td>Implementation</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>+/-/o</td>
<td>++</td>
</tr>
<tr>
<td>Accuracy</td>
<td>++</td>
<td>++</td>
<td>-</td>
<td>+/++</td>
<td>++</td>
</tr>
<tr>
<td>Robustness</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>+/++</td>
</tr>
<tr>
<td>Global localization</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Bayes Filters for Robot Localization

**Discrete**
- **Grid**
  - Fixed/variable resolution
  - Piecwise constant approximation
  - Arbitrary posteriors
  - Non-linear dynamics/observations
  - Optimal, converges to true posterior
  - Exponential in state dimensions
  - Global localization

- **Topological**
  - Abstract state space
  - Arbitrary, discrete posteriors
  - Abstract dynamics/observations
  - One-dimensional graph
  - Global localization

- **Particle filter**
  - Sample-based approximation
  - Arbitrary posteriors
  - Non-linear dynamics/observations
  - Optimal, converges to true posterior
  - Exponential in state dimensions
  - Global localization

**Continuous**
- **Kalman filter**
  - First and second moment
  - Linear dynamics/observations
  - Optimal (linear, Gaussian)
  - Quadratic in state dimension
  - Position tracking

- **Extended / unscented Kalman filter**
  - First and second moment
  - Non-linear dynamics/observations
  - Not optimal (linear approx.)
  - Quadratic in state dimension
  - Position tracking

- **Multi-hypothesis tracking (EKF)**
  - Multi-modal Gaussian
  - Non-linear dynamics/observations
  - Not optimal (linear approx.)
  - Polynomial in state dimension
  - Global localization
Some Maps
Problems in Mapping

• Sensor interpretation
  – How do we extract relevant information from raw sensor data?
  – How do we represent and integrate this information over time?
  – Do we map for the purpose of localization or do we map for “human consumption” (dense vs sparse)?

• Robot locations have to be known
  – How can we estimate them during mapping?
Occupancy Grid Maps

• Introduced by Moravec and Elfes in 1985
• Represent environment by a grid.
• Estimate the probability that a location is occupied by an obstacle.

• Key assumptions
  – Occupancy of individual cells is independent
  – Robot positions $x(1:k)$ are known!

\[
Bel(m_t) = P(m_t \mid x(1:k), y(1:k))
= Bel(m_t^{[xy]} \mid x(1:k), y(1:k))
\]

  – Robot positions $x(1:k)$ are known!
The Basic Intuition

• A ray that flies through an area of space indicates that the space is probably free

• A ray where that reflects at a point indicates something there

• Otherwise, we don’t know
One Idea: Simple Counting

- For every cell count
  - hits\((x,y)\): number of cases where a beam ended at \(<x,y>\)
  - misses\((x,y)\): number of cases where a beam passed through \(<x,y>\)

\[
Bel(m^{[xy]}) = \frac{\text{hits}(x, y)}{\text{hits}(x, y) + \text{misses}(x, y)}
\]

- Assumption: \(P(\text{occupied}(x,y)) = P(\text{reflects}(x,y))\)

- Many cases where this is not a good approximation ... e.g. sonar reflection model
Updating Occupancy Grid Maps

• Note that the following also can be derived:

\[
P(m^{xy} | x(1:k), y(1:k)) =
\]

\[
P(m^{xy} | y(k), x(k))P(y(k) | x(k))P(m^{xy} | x(1:k -1), y(1:k -1))
\]

• Now, consider the odds ratio:

\[
Odds(x) = \frac{P(x)}{1 - P(x)}
\]

\[
= \frac{P(x)}{P(\neg x)}
\]

\[
O(m^{xy} | x(1:k), y(1:k)) =
\]

\[
O(m^{xy} | y(k), x(k))O(m^{xy} | x(1:k -1), y(1:k -1))
\]

inverse sensor model Prior odds
Updating Occupancy Grid Maps

• Typically updated using inverse sensor model and log odds ratio \((\log ab = \log a + \log b)\)

\[
\log O(m \mid x(1:k), y(1:k)) = \log O(m \mid x(k), y(k)) + \log O(m \mid x(1:k-1), y(1:k-1)) + \log \eta
\]

and then recover the probability with

\[
P(x) = \left[ 1 + \frac{1}{\text{Odds}(x)} \right]^{-1}
\]

\[
P(m \mid x(1:k), y(1:k)) = \left[ 1 + \frac{1 - P(m \mid x(k), y(k))}{P(m \mid x(k), y(k))} \frac{\eta}{1 - \eta} \frac{1 - P(m \mid x(1:k-1), y(1:k-1))}{P(m \mid x(1:k-1), y(1:k-1))} \right]^{-1}
\]
The Inverse Sensor Model

- The probability a cell is occupied given observation and localization \( P(m \mid x(k), y(k)) \)

Assume that all cells are independent \( P(m) = \prod_l P(m_l) \)

then specify \( P(m_l \mid x(k), y(k)) \), which is the probability of cell \( m_l \) is occupied given the measurement \( y(k) \) in position \( x(k) \)
Learning Inverse Sensor Model

- Learn probabilities with neural network.
- Consider four beams simultaneously.

[Thrun ’98]
Algorithm **Occupancy_grid**($x_{1:k}$, $y_{1:k}$, $P_0(m)$):

1. $P_m = P_0(m)$
2. for $i=1$ to $k$
3. $P_m = \left[1 + \frac{1 - P(m \mid x(k), y(k))}{P(m \mid x(k), y(k))} \frac{\eta}{1 - \eta} \frac{1 - P_m}{P_m}\right]^{-1}$
Occupancy Grids: From scans to maps
Tech Museum, San Jose

CAD map

occupancy grid map
Concurrent Mapping and Localization

• Chicken-and-egg problem
  – Mapping with known poses is “simple”
  – Localization with known map is “simple”
  – But in combination the problem is hard!
Mapping with Expectation Maximization

- **Idea**: Maximum likelihood with unknown data association.

\[
Bel(m, x(k)) \quad P(x(1:k), m \mid u(0:k-1), y(1:k)) \\
Bel(m, x(k)) = p(y(k) \mid m, x(k)) \ p(x(k) \mid x(k-1), u(k-1))Bel(m, x(k-1)) \ dx_{k-1}
\]

Take this apart: first estimate location given map, then estimate map given location

- **EM**: Maximize log-likelihood by iterating

  **E-step**: 
  \[
  Q[x(1:k) \mid m^{[k]}] = E_{m^{[k]}} [\log p(x(1:k) \mid m^{[k]}, u(0:k-1), y(1:k))] \\
  \rightarrow \text{Localization (bi-directional)}
  \]

  **M-step**: 
  \[
  m^{[k+1]} = \arg\max_{m} \ Q \ [x(1:k) \mid m^{[k]}] \\
  \rightarrow \text{Mapping with known poses}
  \]
EM Mapping, Example (width 45 m)
SLAM

- Idea: Given the true trajectory of the robot, all landmark detections are independent.

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

- The first term is “easy” (mapping given location and data)
- The second term is “easy” (predict location from prior data)
- The “hard” part: we know have to represent distributions on *trajectories*!

- We can use Rao-Blackwellised particle filters to estimate robot locations and landmark locations. (FastSLAM, Montemerlo)

- Update can be done efficiently (\(O(m \log n)\)).
Sufficient Statistic

• A statistic is a function $T(X_1, X_2, \ldots, X_n)$ of the random samples $X_1, X_2, \ldots, X_n$

Examples:

$$
\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i
$$

$$
s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X})^2
$$

$$
T = \max \{ X_1, X_2, \ldots, X_n \}
$$
Sufficient Statistic

• Given \( X_1, X_2, \ldots, X_n \), is there a small set of statistics that can contain all the information about the samples?

• If \( f_\theta(x) \) is a probability distribution (\( \theta \) is parameter), then \( T \) is a sufficient statistic for \( \theta \) if we can factorize

\[
f_\theta(x) = h(x) g_\theta(T(x))
\]

  Function of the statistic

  Not a function of \( \theta \)

• If \( T \) is a sufficient statistic then it contain all the information to compute an estimate of \( \theta \)
Sufficient Statistic

• Given $X_1, X_2, ..., X_n$, is there a statistic that can contain all the information about the samples?

  – Poisson distribution:
    • $X_1, X_2, ..., X_n$ are independent samples from a Poisson distribution with parameter $\lambda$. Then a sufficient statistic $T(X)$ of $\lambda$ is
      \[ T(X) = \sum_{i=1}^{n} X_i \]
    • Note that $T(X)$ does not depend on $\lambda$
Sufficient Statistic

- Given $X_1, X_2, \ldots, X_n$, is there a statistic that can contain all the information about the samples?
  - Exponential distribution:
    - $X_1, X_2, \ldots, X_n$ are independent and exponentially distributed with expected value $\lambda$. Then a sufficient statistic $T(X)$ of $\lambda$ is
      \[ T(X) = \sum_{i=1}^{n} X_i \]
    - Note that $T(X)$ does not depend on $\lambda$
Rao-Blackwell Theorem

- **Improve** the efficiency of an *estimator* by taking its *conditional expectation* with respect to a *sufficient statistic*
  - Estimator $\hat{\theta}(X)$: Is a statistic (rule) used to estimate an unobservable parameter $\theta$ in a population
    - The average of $N$ random samples is an estimator of the population’s average (i.e. height)
  - Sufficient statistic $T(X)$: Given $T(X)$, the distribution of observable samples $X$ does not depend on the unobservable parameter $\theta$
  - Rao Blackwell estimator of $\theta$ is defined by
    \[
    \hat{\theta}_R(X) = E[\hat{\theta}(X) | T(X)]
    \]
    and has better mean squared error than $\hat{\theta}(X)$
What About SLAM?

- Recall the Bayes filtering problem. The posterior satisfies the recursion

\[
P(z_{0:k} \mid y_{1:k}) = \eta_k P(y_k \mid z_k) \int P(z_k \mid z_{k-1}) P(z_{0:k-1} \mid y_{1:k-1}) \mathrm{d}z_{k-1}
\]

where \( z_k \) is hidden (put up with “z” instead of “x” for now)

- That integral is often not tractable and numeric approximations with samples is often used
Marginalize the State Space in Two

- Suppose we can divide $z_k$ in two groups: $x_k$ and $m_k$ such that
  \[ P(z_k | z_{k-1}) = P(m_k | x_{k-1:k}, m_{k-1})P(x_k | x_{k-1}) \]
  and assume that
  \[ P(m_{0:k} | y_{1:k}, x_{0:k}) \]
  is tractable.

- Then we can marginalize $x_{0:k}$ from the posterior and focus on estimating
  \[ P(x_{0:k} | y_{1:k}) \]
  which is a “smaller” problem. Essentially we convert the problem to
  \[ P(x_{0:k}, m_{0:k} | y_{1:k}) = P(m_{0:k} | y_{1:k}, x_{0:k}) P(x_{0:k} | y_{1:k}) \]

Optimal Filt. Particle Filt.

and the posterior distribution of $P(x_{0:k} | y_{1:k})$ is given

\[
P(x_{0:k} | y_{1:k}) = n_k P(y_k | x_k) \int_{x_{k-1}} P(x_k | x_{k-1}) P(x_{0:k-1} | y_{1:k-1})
\]

The dimension of $P(x_{0:k} | y_{1:k})$ is smaller than $P(x_{0:k}, m_{0:k} | y_{1:k})$. 

Rao-Blackwellized SLAM

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

\[ p(x_{1:k}, m \mid y_{1:k}, u_{0:k-1}) = p(m \mid x_{1:k}, y_{1:k}, u_{0:t-1}) p(x_{1:k} \mid y_{1:k}, u_{0:k-1}) \]
A Graphical Model of Rao-Blackwellized SLAM

- If we know the map
  - Estimate localize at each step $x_{1:k}$
- If we know locations $x_{1:k}$
  - Compute the map
- Particle filtering
  - Each particle represent the posterior trajectory
  - Compute the map corresponding to the particle’s trajectory
  - Particle’s weight is given by the most likelihood of the most recent observation given the map

Courtesy Dieter Fox
Rao-Blackwellized SLAM

• Break it down even further if a map $m_k$ consists of $N$ individual landmarks $l_i = N(\mu_i, \Sigma_i)$ then

$$P(x_{1:k}, m_k | y_{1:k}, u_{0:k-1}) = P(x_{1:k} | y_{1:k}, u_{0:k-1})P(m_k | x_{1:k}, y_{1:k}, u_{0:k-1})$$

$$\prod_i N(\mu_i, \Sigma_i)$$

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

• Landmark are filtered individually and have low dimensionality

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

Robot Pose

- 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: \( x, y, \theta \), \( \mu_1, \Sigma_1 \), \( \mu_2, \Sigma_2 \), \( \mu_N, \Sigma_N \)
- ... 

Kalman Filters

- \( \mu_1, \Sigma_1 \)
- \( \mu_2, \Sigma_2 \)
- \( \mu_N, \Sigma_N \)
FastSLAM Algorithm $O(MN)$

- Sample a new robot pose for each particle
  - $x_k = g(x_{k-1}, u_k) + \nu$
  - add this to trajectory giving $x_{1:k}$

- Update the landmark EKFs in each particle
  - We have a "known" (estimate) trajectory; run EKFs for each landmark

- Calculate an importance weight (difference between actual observation, $y_k$, and expected observation, with covariance $Z$)
  $$w_k = \frac{1}{\sqrt{2\pi Y_{n,k}}} \exp \left( -\frac{1}{2} (y_k - \hat{y}_{n,k})^T Y_{n,k}^{-1} (y_k - \hat{y}_{n,k}) \right)$$

- Resample particle set
SLAM Data Association

Angle increment = 0.00436940183863

Index1 = [0, 1, 2, ..., 532, 533, 534, ..., 717, 718, 719]
Range1 = [6.0000, 5.9996, 5.9985, ..., 1.8759, 1.8771, 1.8784, ..., 3.0107, 3.0284, 3.0447]

Index2 = [0, 1, 2, ..., 302, 303, 304, ..., 717, 718, 719]
Range2 = [2.0254, 2.0294, 2.0347, ..., 2.0254, 2.0294, 2.0347, ..., 6.0000, 6.0000, 6.0000]
In practice we use a variable $n_k$ to associate each measurement to a landmark number (id#)

- For example $n_3 = 8$ means that measurement at time $k=3$, $y_3$, is associated with the landmark #8

How do we get this $n_k$?

$$
\hat{n}_k = \arg \max_{n_k} P(y_k \mid n_k, y_{k-1}, \hat{n}_{k-1}, \hat{x}_k, u_{k-1})
$$

This is a “Maximum Likelihood” estimator.
FastSLAM Data Association

- In FastSLAM, each landmark is estimated with an EKF and the likelihood can be estimated from the EKF “innovation”

\[
P(y_k \mid y_{k-1}, \hat{n}_{k-1}, \hat{x}_k, u_{k-1}) = \frac{1}{\sqrt{2\pi Y_{n,k}}} \exp\left( -\frac{1}{2} \left( y_k - \hat{y}_{n,k} \right)^T Y_{n,k}^{-1} \left( y_k - \hat{y}_{n,k} \right) \right)
\]

If the likelihood falls below a threshold, a new landmark is added.
Tree of Landmarks

- FastSLAM complexity is $\log(MN)$
  - $M$ number of particles
  - $N$ number of landmarks
- When we resample (with replacement) the same particle may be duplicated several times
  - Copying is linear in the size of the map
  - Most of the landmarks remain unchanged during a map update (only the visible landmarks are updated)

From Montemerlo 2003
Tree of Landmarks

- Use a tree of landmarks that is shared between particles
- If the tree is balanced then accessing a landmark takes \( \log(N) \) and FastSLAM runs in \( \log(M \log N) \)

Balanced binary tree with 8 landmarks

[Courtesy of Mike Montemerlo]
Tree of Landmarks

- Modifying a landmark #3. Only $\mu_3$ and $\Sigma_3$ are modified
  - Avoid duplicating the entire tree
  - Create a new path from the root to landmark #3
  - Copy the missing pointers to the rest of the old tree
  - Keep the old pointers so the other particles can use the old values

[Courtesy of Mike Montemerlo]
FastSLAM: Victoria Park Results

- 4 km traverse
- 100 particles
- GPS ground truth
- Uses negative evidence to remove spurious landmarks
- Uneven terrain

[Courtesy of Mike Montemerlo]
FastSLAM: Victoria Park Results

100 meters away from its true position.

100 particles
RMS error over 4km is ~4m

(a) Vehicle path predicted by the odometry
(b) True path (dashed line) and FastSLAM path (solid line)

[End courtesy of Mike Montemerlo]
Grid-Based FastSLAM (occupancy grid)

3 particles

map of particle 1

map of particle 2

map of particle 3

Each particle must carry its entire map

[Courtesy of Mike Montemerlo]
FastSLAM Example

- 500 particles
- 28mx28m
- Length of trajectory 491m
- Map resolution 10cm
Closing the Loop

- Recognize a previously landmark
- Typically SLAM will drift, after a long drive the position and landmarks will be off
- Once an previously located object is seen
  - Make the correction
  - Propagate the correction back
  - Uncertainties collapse
    - With great powers comes great responsibility
    - Uncertainty is not a “bad” thing

[Newman 2005]
Visual Landmarks

- Use camera images to detect landmarks
  - Single camera (mono) is similar to “bearing only” SLAM
  - Two cameras (stereo) will give you depth as well
  - Detect landmarks (aka “features”) in the image(s)
  - Build an appearance vector around the landmark to describe its “appearance”
Visual Landmarks

Extract landmarks and their descriptors from two images. Then match the descriptors

Landmarks have a scale and orientation
Visual SLAM

Bird’s eye view of the 3D SIFT map

SIFT landmarks from stereo camera
Disparities are indicated by lines

Se, Lowe and Little 2005
RGBD SLAM
(Red Green Blue Depth)

- Use camera and depth
- Provide a dense point cloud
RGBD SLAM

• Build dense, 3D, colored maps
• Lots of data so maps are usually small (small rooms or scanning objects)
• Use 3D data for localization
• Associate visual landmarks to 3D coordinates
<table>
<thead>
<tr>
<th></th>
<th>SLAM (Kalman)</th>
<th>EM</th>
<th>ML*</th>
<th>FastSLAM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output</strong></td>
<td>Posterior</td>
<td>ML/MAP</td>
<td>ML/MAP</td>
<td>Posterior</td>
</tr>
<tr>
<td><strong>Convergence</strong></td>
<td>Strong</td>
<td>Weak?</td>
<td>No</td>
<td>Strong</td>
</tr>
<tr>
<td><strong>Local minima</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>Real time</strong></td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Odom. Error</strong></td>
<td>Unbounded</td>
<td>Unbounded</td>
<td>Unbounded</td>
<td>Unbounded</td>
</tr>
<tr>
<td><strong>Sensor Noise</strong></td>
<td>Gaussian</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td><strong># Features</strong></td>
<td>$10^3$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$&gt;10^3$</td>
</tr>
<tr>
<td><strong>Feature uniq</strong></td>
<td>Yes</td>
<td>No</td>
<td>~No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Raw data</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Localization and Mapping Summary

• There are several methods for localization and mapping; two dominant are
  – Kalman filter
    • fast and efficient; very well understood
    • local convergence
    • strong assumptions
  – Hypothesis-based methods (particle filters/Monte Carlo methods)
    • not as fast or efficient; not as well understood
    • global convergence
    • very weak assumptions

• The best methods today are hybrids
  – use hypotheses as necessary
  – use KF-like techniques whenever possible

• The largest revolution in mapping and localization has been data: “It’s all in the likelihood function”
  – laser scanners have really revolutionized the trade
  – vision is next?