Bayesian Methods
Recall Robot Localization

• Given
  - Sensor readings $z_1, z_2, \ldots, z_t = z_{1:t}$
  - Known control inputs $u_0, u_1, \ldots, u_t = u_{0:t}$
  - Known model $P(x_{t+1} | x_t, u_t)$ with initial $P(x_1 | u_0)$
  - Known map $P(z_t | x_t)$ — Most likely sensor reading given state $x$

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

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
A Simple Example
Why Not Just Use a Kalman Filter?
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 $P(z_k | x_k)$
- Given a dynamic model $P(x_k | x_{k-1})$
- Given a prior probability $P(x_0)$
- Find $P(x_k | z_{1:k}, u_{0:k-1})$
Bayesian Filtering

• 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

\[ P(x_k | u_0^{k-1}, z_{1:k}) \quad \rightarrow \quad Posterior \ Probability \ Distribution \]

\[ P(x_k | u_0^{k-1}, z_{1:k}) = \frac{P(z_k | x_k, u_0^{k-1}, z_{1:k-1})P(x_k | u_0^{k-1}, z_{1:k-1})}{P(z_k | u_0^{k-1}, z_{1:k-1})} \]

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

observation state prediction recursive instance
Observation Model

What is the probability that a given state $x_k$ generates the measurement $z_k$?

$$P(z_k | x_k)$$

Same $z_k$
Different $x_k$

This is more likely than that
but how do we measure?
Observation Model

- Compute the distance of end-point of each beam to nearest obstacle
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.*

$$P(z_k|x_k)$$
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.*
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.

$$P(z_k | x_k)$$
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

\[
P(z_k | x_k)
\]
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

\[ P(z_k | x_k) \]
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

\[ x_k \]

\[ z_k \]

\[ d \]
Observation Model
Beam Model

• Combine all the distances $d_i$ between the measured obstacle and the real obstacle (we do have the map, or part thereof)

• But other things must be considered
Range Finder

Observation Model Beam Model

• For each laser beam we can have (given a map and a state)
  – Correct range with noise
    \[ P_{\text{hit}}(z_k|x_k, m) = \begin{cases} \eta_{\text{hit}} N(z_k^*, \sigma_{\text{hit}}) & \text{if } 0 \leq z_k \leq z_{\text{max}} \\ 0 & \text{otherwise} \end{cases} \]
  – Unexpected object
    \[ P_{\text{short}}(z_k|x_k, m) = \begin{cases} \eta_{\text{short}} \lambda e^{-\lambda z_k} & \text{if } 0 \leq z_k \leq z_k^* \\ 0 & \text{otherwise} \end{cases} \]
  – Failures
    \[ P_{\text{max}}(z_k|x_k, m) = I(z_k = z_{\text{max}}) \begin{cases} 1 & \text{if } z_k = z_{\text{max}} \\ 0 & \text{otherwise} \end{cases} \]
  – Random measurement
    \[ P_{\text{rand}}(z_k|x_k, m) = \begin{cases} 1 \frac{1}{z_{\text{max}}} & \text{if } 0 \leq z_k \leq z_{\text{max}} \\ 0 & \text{otherwise} \end{cases} \]
Range Finder
Observation Model Beam Model

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

(b) Exponential distribution $p_{\text{short}}$

(c) Uniform distribution $p_{\text{max}}$

(d) Uniform distribution $p_{\text{rand}}$

Probabilistic Robotic
Range Finder
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
\[ w_{hit} + w_{short} + w_{max} + w_{rand} = 1 \]
Constants

\[ \eta_{\text{hit}} = \left( \int_{0}^{z_{\text{max}}} N(z_k^*, \sigma^2) dz_k \right)^{-1} \]

\[ \eta_{\text{short}} = \frac{1}{1 - e^{-\lambda_{\text{short}} z_k^*}} \]
Range Finder
Likelihood Field Model

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

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

Probabilistic Robotic
Range Finder
Likelihood Field Model

1: Algorithm likelihood_field_range_finder_model($z_t, x_t, m$):

2: $q = 1$
3: for all $k$ do
4:     if $z_t^k \neq z_{max}$
5:     $x_{z_t^k} = x + x_{k, sens} \cos \theta - y_{k, sens} \sin \theta + z_t^k \cos(\theta + \theta_{k, sens})$
6:     $y_{z_t^k} = y + y_{k, sens} \cos \theta + x_{k, sens} \sin \theta + z_t^k \sin(\theta + \theta_{k, sens})$
7:     $dist = \min_{x', y'} \left\{ \sqrt{(x_{z_t^k} - x')^2 + (y_{z_t^k} - y')^2} \mid (x', y') \text{ occupied in } m \right\}$
8:     $q = q \cdot (z_{hit} \cdot \text{prob}(dist, \sigma_{hit}) + \frac{z_{random}}{z_{max}})$
9: return $q$
Range Finder
Likelihood Field Model

• Precompute the shortest distance to obstacle
  – Distance transformation
  – Brushfire algorithm
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

Prior probability distribution $P(x_k)$

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.

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

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 0.5 0.25 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_k=2 )</td>
<td>0 0.25 0.5 0.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_k=3 )</td>
<td>0 0 0.25 0.75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_k=4 )</td>
<td>0 0 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
P(x_{k+1} = 1) = \sum_{x_k \in X} P(x_{k+1} = 1 \mid x_k, u_k) P(x_k)
\]
System Model

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>

$$P(x_{k+1} = 2) = \sum_{x_k \in X} P(x_{k+1} = 2 \mid x_k, u_k)P(x_k)$$

**Prior $P(x_k)$**
System Model

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

$$P(x_{k+1} = 3) = \sum_{x_k \in X} P(x_{k+1} = 3 | x_k, u_k)P(x_k)$$

| $P(x_{k+1} | x_k, u_k)$ | $X_{k+1} = 1$ | $X_{k+1} = 2$ | $X_{k+1} = 3$ | $X_{k+1} = 4$ |
|------------------------|--------------|--------------|--------------|--------------|
| $X_k = 1$              | 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            |

Prior $P(x_k)$
System Model

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

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

| $P(x_{k+1} | x_k, u_k)$ | $x_{k+1}=1$ | $x_{k+1}=2$ | $x_{k+1}=3$ | $x_{k+1}=4$ |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| $x_k=1$         | 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              |
System Model

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

Algorithm \texttt{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 \( x \in X \)
4. \[ P'(x) = \sum_{x' \in X} P(x \mid u, x') P(x') \] Prediction given prior dist. and command
5. end for
6. \( \eta = 0 \) Normalization constant
7. for all states \( x \in X \)
8. \[ P(x) = P(z \mid x) P'(x) \] Update using measurement
9. \( \eta = \eta + P(x) \)
10. end for
11. for all states \( x \in X \)
12. \[ P(x) = P(x) / \eta \] Normalize to 1
13. end for
14. end for
Note About the Posterior Distribution

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

- It is a probability distribution

- What do we do with it?
  - Maximum likelihood: \( \arg \max_{x} P(x_k | z_k, m) \)
  - Mean Squared Error: \( E[(P(x) - P(\hat{x}))^2] \)
Bayes Filter Recap

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

- Given a measurement \( z_k \), compute the probability that \( z_k \) was generated from the robot in state \( x_k \)
- Prior distribution (recursive)
  - The probability of all previous states
- State transition:
  - Given a prior distribution over all states \( x_{k-1} \) and an action \( u_{k-1} \), compute the probability of the robot transitioning to state \( x_k \)
Bayes Filter Recap

\[ P(x_k \mid u_{0:k-1}, z_{1:k}) \]

Obtain \( z_k \) and apply \( P(z_k \mid x_k) \)

\[ P(x_k = A \mid u_{k-1}, z_{k-1}) \]

Apply command \( u_{k-1} \)

\[ P(x_{k-1} = A) \]

\[ P(x_k = B \mid u_{k-1}, z_{k-1}) \]

\[ P(x_{k-1} = B) \]

\[ P(x_k = Z \mid u_{k-1}, z_{k-1}) \]

\[ P(x_{k-1} = Z) \]
Kalman vs Bayes

Kalman Filter

Bayes Filter

Isard 1998
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
- 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]
Particles/Samples

- Given a distribution $P(x)$, randomly sample the distribution $N$ times.
- Assign a weight $w$ to each sample according to the probability.
- Ensure that $\sum_{i=1}^{N} w_i = 1$

Isard 1998
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

- Monte-Carlo-Localization \((\mathbf{u}_k, \mathbf{z}_k, N, m)\)
  - \(S\): \(N\) samples from \(P(X_k)\)
  - for \(i = 1: N\)
    - \(s[i] = \text{random sample from } P(x_{k+1}|x_k, \mathbf{u}_k)\)
    - \(w[i] = P(x_{k+1}|x_k, \mathbf{u}_k)\)
    - \(w[i] = w[i]P(z_k | s[i], m)\)
    - \(S = \text{weighted-sample-with-replacement}(N, S, W)\)
  - return \(S\)

- Note that \(S\) is a discrete representation of the probability of robot location
Sample-based Localization (sonar)

rse-lab.cs.washington.edu
Particles/Samples

- The more particles $M$, the better is the approximation
- Curse of dimensionality

$$\sup E |x_k - \hat{x}_k| \leq \frac{C}{\sqrt{M}}$$

- Particle filter works well for low dimensional problems
  - Localization dimension: 3
  - SLAM dimension: $3 + 2N$ ($N$ can be large!)

Independent of time $k$

Exponential in the dimension of the state space

Bound on expected error
Tour of the Smithsonian National Museum of American History
Using Ceiling Maps for Localization

[Dellaert et al. 99]
Vision-Based Localization

$P(z|x)$
Under a Light

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

Measurement \( z \):  

\[ P(z|x) : \]

![Image](image_url)
Elsewhere

Measurement $z$:  

$P(z|x)$:
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
Mapping with Probabilities

- 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 landmarks, scan vs map)?

- Robot locations have to be known
  - How can we estimate them during mapping?
Occupancy Grid Map

- Represent environment by a grid.
- Estimate the probability that a location (cell) is occupied by an obstacle.
- Key assumptions
  - Occupancy of an individual cell is independent from the occupancy of others
- 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” for free
The Basic Idea

- \( P \left( m \mid z_{1:k}, x_{1:k} \right) = \prod_i P \left( m_i \mid z_{1:k}, x_{1:k} \right) \)
- The “log-odds” of \( m_i \) being occupied at time \( t \) is
  \[
  l_{t,i} = \log \frac{P(m_i = 1 \mid z_{1:k}, x_{1:k})}{1 - P(m_i = 1 \mid z_{1:k}, x_{1:k})}
  \]
  \[\Leftrightarrow P(m_i = 1 \mid z_{1:k}, x_{1:k}) = 1 - \frac{1}{1 - e^{l_{t,i}}}
  \]

```plaintext
1: Algorithm occupancy_grid_mapping(\{l_{t-1,i}\}, x_t, z_t):
2:     for all cells \( m_i \) do
3:         if \( m_i \) in perceptual field of \( z_t \) then
4:             \[ l_{t,i} = l_{t-1,i} + \text{inverse} \_ \text{sensor} \_ \text{model}(m_i, x_t, z_t) - l_0 \]
5:         else
6:             \[ l_{t,i} = l_{t-1,i} \]
7:         endif
8:     endfor
9: return \{l_{t,i}\}
```

Occancy prior:
- Default probability of a cell being occupied
Inverse Sensor Model

1: Algorithm inverse_range_sensor_model(m_i, x_t, z_t):

2: Let x_i, y_i be the center-of-mass of m_i
3: r = \sqrt{(x_i - x)^2 + (y_i - y)^2}
4: \phi = \text{atan2}(y_i - y, x_i - x) - \theta
5: k = \text{argmin}_j |\phi - \theta_j,\text{sens}| → Find the nearest beam k
6: if r > \min(z_{\text{max}}, z_k^r + \alpha/2) or |\phi - \theta_k,\text{sens}| > \beta/2 then
7: return l_0 → Return occupancy prior if m_i is outside the beam k
8: if z_k^r < z_{\text{max}} and |r - z_k^r| < \alpha/2
9: return l_{\text{occ}} → range of m_i is within ±\alpha/2 of the detected range z_k
10: if r \leq z_k^r
11: return l_{\text{free}}
12: endif

Robot pose: x_t = [x, y, \theta]
Obstacle thickness: \alpha
Sensor beam width: \beta
Sensor maximum range: z_{\text{max}}
CSIRO
Occupancy Grids: From scans to maps
Tech Museum, San Jose

CAD map

occupancy grid map
SLAM with Particle Filter

- Idea: Given the true trajectory of the robot, the location of landmarks are independent from each other

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

- The first distribution is “easy” (mapping given location and data)
- The second distribution 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)).
\[ P(x_{0:k}, m_{0:k} \mid z_{1:k}, u_{0:k-1}) \]
Marginalize the State Space in Two

• Suppose we have $P(y_k | y_{k-1})$, we divide $y_k$ in two groups: $x_k$ and $m_k$ such that

$$P(y_k | y_{k-1}) = P(m_k | x_k, m_{k-1})P(x_k | x_{k-1})$$

• Using Rao-Blackwell we convert the SLAM problem to

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

Optimal Filter    Particle Filter

• The dimension of $P(x_{0:k} | z_{1:k}, u_{0:k-1})$ is smaller than

$$P(x_{0:k}, m_{0:k} | z_{1:k}, u_{0:k-1})$$
Rao-Blackwellized SLAM

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

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

• If we know the true path $x_{1:k}$
  – Then we can estimate the coordinates of each landmark independently of each other!
  – Dependencies in the landmarks coordinates is the result of robot pose uncertainty

• Rao-Blackwell Particle Filter
  – Each particle represent the posterior path $x_{1:k}$
  – Compute the map corresponding to the particle’s path $x_{1:k}$
  – Particle’s weight is given by the likelihood of the most recent observation given the map
Rao-Blackwellized SLAM

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

$$P(x_{1:k}, m_k | z_{1:k}, u_{0:k-1}) = P(x_{1:k} | z_{1:k}, u_{0:k-1}) P(m | x_{1:k}, z_{1:k}, u_{0:k-1}) \prod_{i} P(l_i | x_{1:k}, z_{1:k}, u_{0:k-1})$$

• 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

<table>
<thead>
<tr>
<th>Robot Pose</th>
<th>Kalman Filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>x, y, θ</td>
<td>µ₁, Σ₁</td>
</tr>
<tr>
<td></td>
<td>µ₂, Σ₂</td>
</tr>
<tr>
<td></td>
<td>…</td>
</tr>
<tr>
<td></td>
<td>µₙ, Σₙ</td>
</tr>
</tbody>
</table>

Particle #1

Particle #2

Particle #3

[Begin courtesy of Mike Montemerlo]
FastSLAM
FastSLAM Algorithm $O(MN)$

- Sample a new robot pose for each particle
  - $x'_k \sim P(x_k | x_{k-1}, u_k)$
  - add this to trajectory resulting in $x_{1:k}$
- For each measured landmark $z_{k,i}$, update the landmark’s EKFs
  - We have a “known” (estimate) trajectory
  - Run EKFs for each landmark
- Calculate an importance weight (difference between actual observation, $z_{k,i}$, and expected observation $h_i(x'_k)$, with covariance $\Sigma$)
  
  $$w_{k,i} = \frac{1}{\sqrt{2\pi \Sigma_{k,i}}} e^{-\frac{1}{2} (z_{k,i} - h_i(x'_k))^T \Sigma_{k,i}^{-1} (z_{k,i} - h_i(x'_k))}$$
- Resample particle set
FastSLAM Data Association

- In FastSLAM, each landmark is estimated with an EKF and the likelihood can be estimated from the EKF “innovation”
- If the likelihood falls below a threshold, a new landmark is added

<table>
<thead>
<tr>
<th>Robot Pose</th>
<th>Kalman Filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Particle #1</td>
<td>μ₁, Σ₁    μ₂, Σ₂ ... μ_N, Σ_N</td>
</tr>
<tr>
<td>Particle #2</td>
<td>μ₁, Σ₁    μ₂, Σ₂ ... μ_N, Σ_N</td>
</tr>
<tr>
<td>Particle #3</td>
<td>μ₁, Σ₁    μ₂, Σ₂ ... μ_N, Σ_N</td>
</tr>
<tr>
<td>Particle #M</td>
<td>μ₁, Σ₁    μ₂, Σ₂ ... μ_N, Σ_N</td>
</tr>
</tbody>
</table>

\[ Z_k \quad \hat{Z}_{n,k} \]
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)
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

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)
Grid-Based FastSLAM

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

Before

Start at “you are here” (no uncertainty)
When the loop is
Close you are “back here”

After

[Newman 2005]
Closing the Loop
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
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?
EM Mapping, Example (width 45 m)