• SLAM exercise?

• Google Cartographer

https://opensource.googleblog.com/2016/10/introducing-cartographer.html?m=1
Example 2

The probability distribution no longer sums to 1!

Normalize (divide by total)

Sums to 0.36

The probability distribution no longer sums to 1!
Kalman Filter

\[
\begin{align*}
\mu' &= \left( \frac{x r^2 + ν σ^2}{σ^2 + r^2} \right) \\
σ^2' &= \left( \frac{σ^2 + r^2}{σ^2 r^2} \right)
\end{align*}
\]

\[\mu' = μ + u \quad σ^2' = σ^2 + r^2\]
Implementing a Kalman Filter example

VERY simple model of robot movement:

\[
x' = x + \dot{x}
\]

\[
\dot{x}' = \dot{x}
\]

\[
\begin{bmatrix} x' \\ \dot{x}' \end{bmatrix} \leftarrow [?] \begin{bmatrix} x \\ \dot{x} \end{bmatrix}
\]

What information do our sensors give us?

\[
Z \leftarrow [?] \begin{bmatrix} x \\ \dot{x} \end{bmatrix}
\]
Implementing a Kalman Filter example

\[ x' = x + \dot{x} \]
\[ \dot{x}' = \dot{x} \]

\[
\begin{bmatrix}
  x' \\
  \dot{x}'
\end{bmatrix} \leftarrow \begin{bmatrix}
  F & \? \\
  \? & \? 
\end{bmatrix}
\begin{bmatrix}
  x \\
  \dot{x}
\end{bmatrix}
\]

\[ Z \leftarrow \begin{bmatrix}
  \? & \? \\
  \? & \? 
\end{bmatrix}
\begin{bmatrix}
  x \\
  \dot{x}
\end{bmatrix}
\]

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

\[ H = [1 \ 0] \]
Implementing a Kalman Filter

Motion Prediction

\[ x' = Fx + u \]
\[ P' = FF^T + Q \]

Estimate

- \( P' \): uncertainty covariance
- \( F \): state transition matrix
- \( u \): motion vector
- \( F \): motion noise
Implementing a Kalman Filter

Motion Prediction

\[ x' = Fx + u \]
\[ P' = FPFT + Q \]

Measurement Update

\[ y = z - Hx \]
\[ S = PHHT + R \]
\[ K = PHHTS^{-1} \]

Estimate

- \( P' \): uncertainty covariance
- \( F \): state transition matrix
- \( u \): motion vector
- \( F \): motion noise
Implementing a Kalman Filter

Motion Prediction

\[ x' = Fx + u \]
\[ P' = FPF^T + Q \]

Measurement Update

\[ y = z - Hx \]
\[ S = HPH^T + R \]
\[ K = PH^T S^{-1} \]

End Result

\[ x' = x + (Ky) \]
\[ P' = (I - KH)P \]

Estimate

- \( P' \): uncertainty covariance
- \( F \): state transition matrix
- \( u \): motion vector
- \( F \): motion noise
Particle Filter Localization (using sonar)

Particules

• Each particle is a guess about where the robot might be
Robot Motion

move each particle according to the rules of motion
+ add random noise

$n$ particles

$n$ particles
Incorporating Sensing
Incorporating Sensing

Difference between the actual measurement and the estimated measurement

Importance weight
def Gaussian(self, mu, sigma, x):
    # calculates the probability of x for 1-dim Gaussian with mean mu and var. sigma
    return exp(- ((mu - x) ** 2) / (sigma ** 2) / 2.0) / sqrt(2.0 * pi * (sigma ** 2))

def measurement_prob(self, measurement):
    # calculates how likely a measurement should be
    prob = 1.0;
    for i in range(len(landmarks)):
        dist = sqrt((self.x - landmarks[i][0]) ** 2 + (self.y - landmarks[i][1]) ** 2)
        prob *= self.Gaussian(dist, self.sense_noise, measurement[i])
    return prob

def get_weights(self):
    w = []
    for i in range(N): #for each particle
        w.append(p[i].measurement_prob(Z)) #set it’s weight to p(measurement Z)
    return w
Importance weight pseudocode

Calculate the probability of a sensor measurement for a particle:

```
prob = 1.0;
  for each landmark
    d = Euclidean distance to landmark
    prob *= Gaussian probability of obtaining a reading at
distance d for this landmark from this particle

return prob
```
Incorporating Sensing
Resampling

<table>
<thead>
<tr>
<th>$n$ original particles</th>
<th>Importance Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
</tr>
</tbody>
</table>

Sum is 2.8
### Resampling

<table>
<thead>
<tr>
<th>$n$ original particles</th>
<th>Importance Weight</th>
<th>Normalized Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.07</td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>0.21</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>0.07</td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>0.29</td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>0.29</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>0.07</td>
<td></td>
</tr>
</tbody>
</table>

Sum is 2.8
Resampling

$n$ original particles

<table>
<thead>
<tr>
<th>Importance Weight</th>
<th>Normalized Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.07</td>
</tr>
<tr>
<td>0.6</td>
<td>0.21</td>
</tr>
<tr>
<td>0.2</td>
<td>0.07</td>
</tr>
<tr>
<td>0.8</td>
<td>0.29</td>
</tr>
<tr>
<td>0.8</td>
<td>0.29</td>
</tr>
<tr>
<td>0.2</td>
<td>0.07</td>
</tr>
</tbody>
</table>

Sum is 2.8

Sample $n$ new particles from previous set
Each particle chosen with probability $p$, with replacement
### Resampling

<table>
<thead>
<tr>
<th>$n$ original particles</th>
<th>Importance Weight</th>
<th>Normalized Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.07</td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>0.21</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>0.07</td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>0.29</td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>0.29</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>0.07</td>
<td></td>
</tr>
</tbody>
</table>

Sum is 2.8

**Is it possible that one of the particles is never chosen?**
Yes!

**Is it possible that one of the particles is chosen more than once?**
Yes!

Sample $n$ new particles from previous set
Each particle chosen with probability $p$, with replacement.
What is the probability that this particle is not chosen during the resampling of the six new particles?

Sample $n$ new particles from previous set
Each particle chosen with probability $p$, with replacement
Question

• What happens if there are no particles near the correct robot location?
Question

• What happens if there are no particles near the correct robot location?

• Possible solutions:
  – Add random points each cycle
  – Add random points each cycle, where \( a \) is proportional to the average measurement error
  – Add points each cycle, located in states with a high likelihood for the current observations
Summary

• Kalman Filter
  – Continuous
  – Unimodal
  – Harder to implement
  – More efficient
  – Requires a good starting guess of robot location

• Particle Filter
  – Continuous
  – Multimodal
  – Easier to implement
  – Less efficient
  – Does not require an accurate prior estimate
SLAM

• Simultaneous localization and mapping:

Is it possible for a mobile robot to be placed at an unknown location in an unknown environment and for the robot to incrementally build a consistent map of this environment while simultaneously determining its location within this map?
Three Basic Steps

• The robot moves
  – increases the uncertainty on robot pose
  – need a mathematical model for the motion
  – called motion model
Three Basic Steps

• The robot discovers interesting features in the environment
  – called landmarks
  – uncertainty in the location of landmarks
  – need a mathematical model to determine the position of the landmarks from sensor data
  – called inverse observation model
Three Basic Steps

• The robot \textit{observes previously mapped landmarks}:
  – uses them to correct both self localization and the localization of all landmarks in space
  – uncertainties decrease
  – need a model to predict the measurement from predicted landmark location and robot localization
  – called \textit{direct observation model}
How to do SLAM

- Use internal representations for
  - the positions of landmarks (: map)
  - the camera parameters

- Assumption: Robot’s uncertainty at starting position is zero

Start: robot has zero uncertainty
How to do SLAM

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations
How to do SLAM

- The robot observes a feature which is mapped with an uncertainty related to the measurement model

On every frame:
- **Predict** how the robot has moved
- Measure
- **Update** the internal representations
How to do SLAM

- As the robot moves, its pose uncertainty increases, obeying the robot’s **motion model**.

On every frame:
- **Predict** how the robot has moved
- Measure
- **Update** the internal representations

Robot moves forwards: uncertainty grows

© R. Siegwart, D. Scaramuzza and M. Chli, ETH Zurich - ASL
How to do SLAM

- Robot observes two new features.

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

Robot makes first measurements of B & C
How to do SLAM

- Their position uncertainty results from the combination of the measurement error with the robot pose uncertainty.
  - map becomes correlated with the robot pose estimate.

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

Robot makes first measurements of B & C
How to do SLAM

- Robot moves again and its uncertainty increases (motion model)

On every frame:
- **Predict** how the robot has moved
- Measure
- **Update** the internal representations

Robot moves again: uncertainty grows more
How to do SLAM

- Robot re-observes an old feature
  - Loop closure detection

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

Robot re-measures A: “loop closure”
How to do SLAM

- Robot updates its position: the resulting position estimate becomes correlated with the feature location estimates.
- Robot’s uncertainty shrinks and so does the uncertainty in the rest of the map.

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

Robot re-measures A: “loop closure” uncertainty shrinks