ME 456: Intelligent Robotics

Week 3, Lecture 1:
Bayes Filters
(Based on S. Thrun and A. Teichman)

Reading: Sections 3.1, 3.2, 3.6

Concepts

• The robot has to estimate its state and the state of environment to take actions

• Observations reduce uncertainty

• Actions increase uncertainty

• The robot will build a belief of its state based on observations and actions
Measurement Data

Measurement data:

• Provides information of momentary state of environment from sensor observations

• Include camera images, range scans, etc.

Notation:

• $z_t$: measurement data at time $t$

• $z_{t_1:t_2}$: measurement data from time $t_1$ to time $t_2$

Control Data

Control data:

• Carries information about change of state in environment

• Example: if we know robot velocity is 5 cm/s, then robot pose is 50cm ahead of pose from ten seconds ago

Notation:

• $u_t$: control data at time $t$

• $u_{t_1:t_2}$: control data from time $t_1$ to time $t_2$
Probabilistic Generative Laws

Law characterizing the evolution of a state is given by:

\[ p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}) \]

All of our past states, measurements, and actions allow us to predict our current state.

Probabilistic Laws

What if we have a complete state?

\[ p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t \mid x_{t-1}, u_t) \]

Our previous state, plus our current action gives us all the information we need to predict current state, we don’t have to look at history of state evolution.
Model Measurement Process

Model predicting the measurement is given by:

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

All of our past measurements, and past to current state and actions allow us to predict our measurement.

Probabilistic Laws

What if we have a complete state?

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

Our current state allows us to predict our measurement.
State Beliefs

Sensors and actuators are both noisy

- This noise means we don’t know exactly what the state is, we have a belief about the state

- The belief represent the posterior probabilities over state variables

\[ \text{bel}(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) \]

State Beliefs

- What if we calculate belief before measurement?

\[ \overline{\text{bel}}(x_t) = p(x_t \mid z_{1:t-1}, u_{1:t}) \]

- This is the prediction of the state, before incorporating the measurement.

- Calculate bel(x) from \( \overline{\text{bel}}(x) \) is called correction or measurement update.
Example

• A robot has a map of a hallway and wants to determine its state (position)
• The hallway has doors which can be sensed by the robot
• The robot can either take measurements (use sensors) or move along the hallway
• How does the robot determine where it is?

Example

• Initially, the robot has no idea where it is
• It has a uniform belief distribution
Example

- The robot senses it is near a door
  - There is an equal probability that the measurement came from one of the three doors

- The belief distribution is updated and the robot now thinks it is in front of one of the doors

Example

- The robot moves to the right
  - The previous belief distribution is shifted by the same amount the robot moves
Example

• The robot takes another measurement and sees it is in front of a door

• This measurement indicates the robot is in front of one of the three doors
  - However, the robot’s belief incorporates this information and now the robot
    believes it is in front of the second door

Example

• The robot moves, and its belief distribution shifts according to the motion
Summary of Example

• Start out with uniform belief distribution

• Sensing and actions both change belief distribution
  ▪ A sensor detecting a door tells the robot it is in front of one of the three doors
  ▪ The robot moving shifts the belief distribution in the direction of motion

• Each time the robot senses a door, there is equal probability that the measurement was caused by any of the doors
  ▪ We use the prior belief of where the robot is and incorporate these probabilities into the belief distribution
  ▪ This update of the belief distribution is done by using Bayes filters

Bayes Filter Algorithm

1. Algorithm Bayes Filter (bel(x_{t-1}), u_t, z_t):
2. for all x_t do
3. \[ \overline{bel}(x_t) = \int p(x_t | u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1} \]
4. \[ bel(x_t) = \eta \ p(z_t | x_t) \overline{bel}(x_t) \]
5. end loop
6. return bel(x_t)
Bayes Filter Algorithm

1. Algorithm Bayes Filter (\( \text{bel}(x_{t-1}),u_t,z_t \)):
2. for all \( x_t \) do
3. \( \overline{\text{bel}}(x_t) = \int p(x_t | u_t, x_{t-1}) \text{bel}(x_{t-1}) dx_{t-1} \)
4. \( \text{bel}(x_t) = \eta \; p(z_t | x_t) \overline{\text{bel}}(x_t) \)
5. end loop
6. return \( \text{bel}(x_t) \)

Integrate (sum) the impact of your current action on your previous beliefs
Correct your belief based on your observation:
\( \text{posterior} = \text{likelihood} \times \text{prior} \)

Bayes Filter Example

http://www.youtube.com/watch?v=Ry2PXkkMCmg
Bayes Filters

• As we take more measurements, our belief becomes more and more accurate

• As we move, our belief becomes less accurate

Grid-Based Localization
Another Example

http://www.youtube.com/watch?v=ucy8IGOR0-I

Discrete Bayes Filter Algorithm

1. Algorithm Discrete Bayes Filter (bel(x_{k,t-1}), u_t, z_t):
2. for all k do
3. \( \overline{bel}(x_{k,t}) = \sum_{i} p(x_{k,t} \mid u_t, x_{i,t-1}) \text{bel}(x_{i,t-1}) \)
4. \( bel(x_{k,t}) = \eta \ p(z_t \mid x_{k,t}) \overline{bel}(x_{k,t}) \)
5. end loop
6. return bel(x_{k,t})
Summary of Bayes Filters

- Bayes filters are recursive filters which use Bayes rule
- They represent the posterior by a set of weighted samples
- For localization, belief is propagated according to the motion model
- It is then weighted according to the likelihood of the observations

Kalman Filter

- Recursive state estimator
- Beliefs represented by multivariate Gaussian
- Kalman Filters implement the Bayes filter for:
  - Continuous states
  - Linear dynamics
Bayes Filter Reminder

- Prediction
  \[
  \overline{bel}(x_t) = \int p(x_t | u_t, x_{t-1}) \, bel(x_{t-1}) \, dx_{t-1}
  \]

- Measurement Update
  \[
  bel(x_t) = \eta \ p(z_t | x_t) \overline{bel}(x_t)
  \]
Properties of Gaussians

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

\[
X_i \sim N(\mu_i, \sigma_i^2) \quad X_2 \sim N(\mu_2, \sigma_2^2) \quad \Rightarrow \quad p(X_1) \cdot p(X_2) \sim N\left(\frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \mu_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \mu_2, \frac{1}{\sigma_1^2 + \sigma_2^2}\right)
\]

Multivariate Gaussians

\[
X \sim N(\mu, \Sigma) \quad Y = AX + B \quad \Rightarrow \quad Y \sim N(A\mu + B, A\Sigma A^T)
\]

\[
X_i \sim N(\mu_i, \Sigma_i) \quad X_2 \sim N(\mu_2, \Sigma_2) \quad \Rightarrow \quad p(X_1) \cdot p(X_2) \sim N\left(\frac{\Sigma_1^{-1}}{\Sigma_1^{-1} + \Sigma_2^{-1}} \mu_1 + \frac{\Sigma_2^{-1}}{\Sigma_1^{-1} + \Sigma_2^{-1}} \mu_2, \frac{1}{\Sigma_1^{-1} + \Sigma_2^{-1}}\right)
\]

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

Estimates the state $x$ of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$$

with a measurement

$$z_t = C_t x_t + \delta_t$$

Components of a Kalman Filter

- $A_t$: Matrix (nxn) that describes how the state evolves from $t$ to $t-1$ without controls or noise.
- $B_t$: Matrix (nxl) that describes how the control $u_t$ changes the state from $t$ to $t-1$.
- $C_t$: Matrix (kxn) that describes how to map the state $x_t$ to an observation $z_t$.
- $\epsilon_t$: Random variables representing the process and measurement noise that are assumed to be independent and normally distributed with covariance $R_t$ and $Q_t$ respectively.
- $\delta_t$: Random variables representing the process and measurement noise that are assumed to be independent and normally distributed with covariance $R_t$ and $Q_t$ respectively.
Kalman Filter Updates in 1D

New belief mean is between initial belief and measurement.

New belief has less uncertainty than both initial belief and measurement.

Kalman Filter Updates

\[
\text{bel}(x_i) = \begin{cases} 
\mu_i = \overline{\mu}_i + K_i (z_i - \overline{\mu}_i) \\
\sigma_i^2 = (1 - K_i) \sigma_i^2 
\end{cases} \quad \text{with} \quad K_i = \frac{\sigma_i^2}{\sigma_i^2 + \sigma_{\text{old},i}^2}
\]

\[
\text{bel}(x_i) = \begin{cases} 
\mu_i = \overline{\mu}_i + K_i (z_i - C_i \overline{\mu}_i) \\
\Sigma_i = (I - K_i C_i) \Sigma_i 
\end{cases} \quad \text{with} \quad K_i = \Sigma_i^{-1} C_i^T (C_i \Sigma_i^{-1} C_i^T + Q_i)^{-1}
\]
Kalman Filter Updates

\[
\text{bel}(x_t) = \begin{cases} 
\bar{\mu}_t = a_t \mu_{t-1} + b_t u_t \\
\bar{\sigma}_t^2 = a_t^2 \sigma_{t-1}^2 + a^2 
\end{cases}
\]

\[
\text{bel}(x_t) = \begin{cases} 
\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t \\
\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t 
\end{cases}
\]
Linear Gaussian Systems: Initialization

- Initial belief is normally distributed:

\[ \text{bel}(x_0) = N(x_0; \mu_0, \Sigma_0) \]

Linear Gaussian Systems: Dynamics

- Dynamics are linear function of state and control plus additive noise:

\[ x_t = A_t x_{t-1} + B_t u_t + \epsilon_t \]

\[ p(x_t | u_t, x_{t-1}) = N(x_t; A_t x_{t-1} + B_t u_t, R_t) \]

\[ \text{bel}(x_t) = \int p(x_t | u_t, x_{t-1}) \, \text{bel}(x_{t-1}) \, dx_{t-1} \]

\[ \Downarrow \quad \Downarrow \]

\[ \sim N(x_t; A_t x_{t-1} + B_t u_t, R_t) \sim N(x_{t-1}; \mu_{t-1}, \Sigma_{t-1}) \]
Linear Gaussian Systems: Dynamics

\[
\bar{\text{bel}}(x_t) = \int \text{bel}(x_{t-1}) \, dx_{t-1}
\]
\[
\downarrow
\]
\[
\sim N(x_t; A_t x_{t-1} + B_t u_t, R_t)
\]
\[
\downarrow
\]
\[
\sim N(x_t; \mu_{t-1}, \Sigma_{t-1})
\]
\[
\downarrow
\]
\[
\bar{\text{bel}}(x_t) = \eta \int \exp \left\{ -\frac{1}{2} (x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1} (x_t - A_t x_{t-1} - B_t u_t) \right\} 
\exp \left\{ -\frac{1}{2} (x_{t-1} - \mu_{t-1})^T \Sigma_{t-1}^{-1} (x_{t-1} - \mu_{t-1}) \right\} \, dx_{t-1}
\]
\[
\bar{\text{bel}}(x_t) = \begin{cases} 
\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t \\
\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t
\end{cases}
\]

Linear Gaussian Systems: Observations

- Observations are linear function of state plus additive noise:

\[
z_t = C_t x_t + \delta_t
\]
\[
p(z_t \mid x_t) = N(z_t; C_t x_t, Q_t)
\]
\[
\bar{\text{bel}}(x_t) = \eta \int \text{bel}(x_{t-1}) \, dx_{t-1}
\]
\[
\downarrow
\]
\[
\sim N(z_t; C_t x_t, Q_t)
\]
\[
\downarrow
\]
\[
\sim N(x_t; \overline{\mu}_t, \overline{\Sigma}_t)
\]
Linear Gaussian Systems: Observations

\[ \text{bel}(x_i) = \eta \cdot p(z_i | x_i) \]
\[ \downarrow \]
\[ \sim N(z_i; C_i x_i, Q_i) \]
\[ \downarrow \]
\[ \text{bel}(x_i) = \eta \exp\left\{ \frac{1}{2} (z_i - C_i x_i)^T Q_i^{-1} (z_i - C_i x_i) \right\} \exp\left\{ -\frac{1}{2} (x_i - \mu_i)^T \Sigma_i^{-1} (x_i - \mu_i) \right\} \]
\[ \text{bel}(x_i) = \begin{cases} \mu_i = \overline{\mu}_i + K_i (z_i - C_i \overline{\mu}_i) \\ \Sigma_i = (I - K_i C_i) \overline{\Sigma}_i \end{cases} \text{ with } K_i = \overline{\Sigma}_i C_i^T (C_i \overline{\Sigma}_i C_i^T + Q_i)^{-1} \]

Kalman Filter Algorithm

1. Algorithm **Kalman_filter**( \( \mu_{t-1}, \Sigma_{t-1}, u_t, z_t \)):

2. Prediction:
3. \( \overline{\mu}_i = A_i \mu_{t-1} + B_i u_i \)
4. \( \overline{\Sigma}_i = A_i \Sigma_{t-1} A_i^T + R_i \)

5. Correction:
6. \( K_i = \overline{\Sigma}_i C_i^T (C_i \overline{\Sigma}_i C_i^T + Q_i)^{-1} \)
7. \( \mu_i = \overline{\mu}_i + K_i (z_i - C_i \overline{\mu}_i) \)
8. \( \Sigma_i = (I - K_i C_i) \overline{\Sigma}_i \)
9. Return \( \mu_i, \Sigma_i \)
Conceptual Example

- Prediction
  \[ \hat{x}(k+1|k) = f(\hat{x}(k|k), u(k+1)) \]
  \[ P(k+1|k) = \nabla_{\hat{x}} f(k|k) \nabla_{\hat{x}} f^T(k|k) + \nabla_{\hat{x}} U(k+1) \nabla_{\hat{x}} U^T(k+1) \]

- Observation
  \[ z(k+1), R(k+1) \]

- Matching
  \[ w(k+1) = z(k+1) - h(\hat{x}(k+1|k), m(k+1)) \]

- Correction
  \[ \hat{x}(k+1|k+1) = \hat{x}(k+1|k) + \frac{1}{\lambda} \left[ P(k+1|k) + S(k+1) \right] \left[ z(k+1) - h(\hat{x}(k+1|k), m(k+1)) \right] \]
  \[ P(k+1|k+1) = \left[ P(k+1|k) - \frac{P(k+1|k) S(k+1) P(k+1|k)}{\lambda} \right] \left[ 1 + \frac{S(k+1) P(k+1|k)}{\lambda} \right] \]
The Prediction-Correction-Cycle

Prediction

\[
\pi_1(x_1) = \begin{cases} 
\mu = A\mu + Bu \\
\sigma^2 = A\sigma^2 + \sigma^2_{crf}
\end{cases}
\]

\[
\pi_2(x_2) = \begin{cases} 
\mu = A\mu + Bu \\
\sigma^2 = A\sigma^2 + \sigma^2_{crf}
\end{cases}
\]

Correction

\[
\psi(x) = \begin{cases} 
\mu = \mu + K(x - \mu) \\
\sigma^2 = (1-K)\sigma^2
\end{cases}
\]

\[
\psi(x) = \begin{cases} 
\mu = \mu + K(x - \mu) \\
\sigma^2 = (1-K)\sigma^2
\end{cases}
\]

\[
\psi(x) = \begin{cases} 
\mu = \mu + K(x - \mu) \\
\sigma^2 = (1-K)\sigma^2
\end{cases}
\]
The Prediction-Correction-Cycle

\[
\begin{align*}
\text{Prediction:} \\
\mu_k &= \mu_{k-1} + K_k(z_k - \mu_{k-1}) \\
\Sigma_k &= (I - K_k)\Sigma_{k-1}
\end{align*}
\]

\[
\begin{align*}
\text{Correction:} \\
\mu_k &= \mu_{k-1} + K_k(z_k - C_k\mu_{k-1}) \\
\Sigma_k &= (I - K_k C_k)\Sigma_{k-1}
\end{align*}
\]

Kalman Filter Summary

- Highly efficient: Polynomial in measurement dimensionality \(k\) and state dimensionality \(n\): \(O(k^{2.376} + n^2)\)
- Optimal for linear Gaussian systems!
- Most robotics systems are nonlinear!
- Linearize (extended Kalman Filter)
- Use non-parametrized filters: Particle Filter
Nonlinear System Dynamics

- Most realistic robotic problems involve nonlinear functions

\[ x_t = g(u_t, x_{t-1}) \]

\[ z_t = h(x_t) \]
Next Time

• Extended Kalman Filters
• Particle Filters