Announcements

- Final project: 45% of the grade, 10% presentation, 35% write-up
  - Presentations: in lecture Dec 1 and 3
  - If you have constraints, inform me by email by Wednesday night, we will assign the others at random on Thursday
- PS2: due Friday 23:59pm.
- Tuesday 4-5pm Cory: Hadas Kress-Gazit (Cornell)

High-level tasks to correct low-level robot control
In this talk I will present a formal approach to creating robot controllers that ensure the robot satisfies a given high level task. I will describe a framework in which a user specifies a complex and reactive task in Structured English. This task is then automatically translated, using temporal logic and tools from the formal methods world, into a hybrid controller. This controller is guaranteed to control the robot such that its motion and actions satisfy the intended task, in a variety of different environments.

Hidden Markov Models

Joint distribution is assumed to be of the form:

\[
P(X_1 = x_1) P(Z_1 = z_1 | X_1 = x_1) P(X_2 = x_2 | X_1 = x_1, X_2 = x_2) \ldots
\]

\[
\begin{align*}
X_1 & \rightarrow X_2 & \rightarrow X_3 & \rightarrow X_4 & \rightarrow \cdots & \rightarrow X_T \\
Z_1 & \rightarrow Z_2 & \rightarrow Z_3 & \rightarrow Z_4 & \rightarrow \cdots & \rightarrow Z_T
\end{align*}
\]

Filtering in HMM

- Init \(P(x_1)\) [e.g., uniformly]
- Observation update for time 0:
  \[
P(x_1 | z_1) = \frac{P(z_1 | x_1) P(x_1)}{\int P(z_1 | x_1) P(x_1) dx_1}
\]
- For \(t = 1, 2, \ldots\)
  - Time update
    \[
P(x_{t+1} | z_{1:t}) = \int P(x_{t+1} | x_t) P(x_t | z_{1:t}) dx_t
\]
  - Observation update
    \[
P(x_{t+1} | z_{1:t+1}) = \int P(x_{t+1} | x_t) P(x_t | z_{1:t}) dx_t
\]
- For discrete state / observation spaces: simply replace integral by summation

Discrete-time Kalman Filter

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

\[
x_t = Ax_{t-1} + Bu_t + \epsilon_t
\]

with a measurement

\[
z_t = Cx_t + \delta_t
\]

Kalman Filter Algorithm

Algorithm Kalman_filter(\(\mu_0, \Sigma_0, u_t, z_t\)):

Prediction:

\[
\begin{align*}
\bar{\mu}_t &= A\mu_{t-1} + Bu_t \\
\bar{\Sigma}_t &= A\Sigma_{t-1}A^T + R
\end{align*}
\]

Correction:

\[
K_t = \Sigma_tC^T(C\Sigma_tC^T + Q)^{-1}
\]

\[
\begin{align*}
\mu_t &= \bar{\mu}_t + K_t(z_t - C\bar{\mu}_t) \\
\Sigma_t &= (I - K_tC)\Sigma_t
\end{align*}
\]

Return \(\mu_t, \Sigma_t\)
Nonlinear systems

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

\[ z_t = h(x_t) + \text{noise} \]

**EKF Linearization: First Order Taylor Series Expansion**

**Prediction:**

\[ g(u_t, x_{t-1}) = g(u_t, \mu_{t-1}) + \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}} (x_{t-1} - \mu_{t-1}) \]

\[ g(u_t, x_{t-1}) = g(u_t, \mu_{t-1}) + G_t (x_{t-1} - \mu_{t-1}) \]

**Correction:**

\[ h(x_t) = h(\bar{x}_t) + \frac{\partial h(\bar{x}_t)}{\partial \bar{x}_t} (x_t - \bar{x}_t) \]

\[ h(x_t) = h(\bar{x}_t) + H_t (x_t - \bar{x}_t) \]

**EKF Algorithm**

**Extended Kalman filter (EKF):**

\[ \hat{\mu}_t = A \hat{\mu}_{t-1} + B u_t \]

\[ \hat{\Sigma}_t = A \hat{\Sigma}_{t-1} A^T + R_t \]

**Correction:**

\[ K_t = \hat{\Sigma}_t H_t^T (H_t \hat{\Sigma}_t H_t^T + R_t)^{-1} \]

\[ \mu_t = \hat{\mu}_t + K_t (z_t - h(\bar{x}_t)) \]

\[ \Sigma_t = (I - K_t H_t) \hat{\Sigma}_t \]

Return \( \mu_t, \Sigma_t \)

**UKF Sigma-Point Estimate (4)**

**Linearization via Unscented Transform**

**UKF**

**EKF**
UKF intuition why it can perform better

- Assume we know the distribution over $X$ and it has a mean $\mu[x]$
- $Y = f(X)$

$$ f[x] = f[x + \delta x] = f[x] + \nabla f(x) \delta x + \frac{1}{2} \nabla^2 f(x) \delta x^2 + \frac{1}{3!} \nabla^3 f(x) \delta x^3 + \frac{1}{4!} \nabla^4 f(x) \delta x^4 + \cdots $$

$$ y = f[x] + \frac{1}{2} \nabla f(x) \delta x^2 + \cdots $$

$P_m \approx \nabla P_m (\nabla^2 f(x)) \delta x + \frac{1}{3} \nabla^3 f(x) (\nabla^2 P_m + \nabla^2 f(x) \nabla^2 f(x)^T) \delta x^2 + \frac{1}{4} \nabla^4 f(x) (\nabla^2 P_m + \nabla^2 f(x) \nabla^2 f(x)^T) \delta x^3 + \cdots $ 

[Julier and Uhlmann, 1997]

Self-quiz

- When would the UKF significantly outperform the EKF?

Original unscented transform

- Picks a minimal set of sample points that match 1st, 2nd and 3rd moments of a Gaussian:

$$ X_1 = x \quad W_n = \kappa / (x + u) $$

$$ X_2 = x + \left\{ \sqrt{\kappa} (x + u) P_{uu} \right\} \quad W_t = 1 / 2(x + u) $$

$$ X_{2+} = x - \left\{ \sqrt{\kappa} (x + u) P_{uu} \right\} \quad W_{2+} = 1 / 2(x + u) $$

\( \mu[x] = \text{mean, } P_{xx} = \text{covariance, } i \rightarrow i\text{'th row, } x \in \mathbb{R}^n \)

\( \kappa \text{ : extra degree of freedom to fine-tune the higher order moments of the approximation; when } x \text{ is Gaussian, } n + \kappa = 3 \text{ is a suggested heuristic} \)

[Julier and Uhlmann, 1997]

Unscented Kalman filter

- Dynamics update:
  - Can simply use unscented transform and estimate the mean and variance at the next time from the sample points
  - Observation update:
  - Use sigmamoints from unscented transform to compute the covariance matrix between $X_i$ and $Z_i$. Then can do the standard update.

Algorithm UnscentedKalmanFilter($\mu_i, \Sigma_i, X_{i-1}, u_i$):
1. $X_{i-1} = (\mu_{i-1} + \gamma \Sigma_i) \mu_{i-1} - \gamma \Sigma_i \mu_{i-1})$
2. $X_i = g(X_{i-1})$
3. $\mu_i = \Sigma_i \bar{z}(X_i)$
4. $S_i = \Sigma_i \bar{z}(X_i) \Sigma_i - \mu_i \Sigma_i^2 + R_i$
5. $\bar{X}_i = \mu_i + \sqrt{\kappa} \Sigma_i \mu_i - \gamma \Sigma_i \Sigma_i^T$
6. $Z_i = h(\bar{X}_i)$
7. $i_i = \Sigma_i \bar{z}(Z_i) \Sigma_i$
8. $S_i = \Sigma_i \bar{z}(Z_i) \Sigma_i^2 + i_i - i_i^T + Q_i$
9. $\Sigma_s = \Sigma_i \bar{z}(Z_i) \Sigma_i^2 + i_i - i_i^T$
10. $\Sigma_s = \Sigma_s \Sigma_s^T$
11. $\mu_s = \mu_i + K_i (i_i - i_i)$
12. $\Sigma_s = \Sigma_s - K_i \Sigma_s K_i^T$
13. return $\mu_s, \Sigma_s$

[Table 3.4 in Probabilistic Robotics]
UKF Summary

- **Highly efficient**: Same complexity as EKF, with a constant factor slower in typical practical applications
- **Better linearization than EKF**: Accurate in first two terms of Taylor expansion (EKF only first term) + capturing more aspects of the higher order terms
- **Derivative-free**: No Jacobians needed
- **Still not optimal!**

Particle filters motivation

- Particle filters are a way to **efficiently** represent non-Gaussian distribution

Basic principle

- Set of state hypotheses ("particles")
- Survival-of-the-fittest

Sample-based Localization (sonar)

FastSLAM [particle filter + Rao-Blackwellization + occupancy grid mapping + scan matching based odometry]

Sample-based Mathematical Description of Probability Distributions

- Particle sets can be used to approximate functions
- The more particles fall into an interval, the higher the probability of that interval
- If a continuous density needs to be extracted → can place a narrow Gaussian at the location of each particle
- How to efficiently draw samples in an HMM?

Dynamics update with sample representation of distribution: sample from $P(x_{t+1} \mid x_t)$
Observation update

- Goal: go from a sample-based representation of
  \( P(x_{t+1} | z_1, ..., z_t) \)
  to a sample-based representation of
  \( P(x_{t+1} | z_1, ..., z_t, z_{t+1}) = C \cdot P(x_{t+1} | z_1, ..., z_t) \cdot P(z_{t+1} | x_{t+1}) \)

- Interested in estimating:

  \[ \text{Hence we could sample from an alternative distribution } Q \text{ and simply re-weight the samples == Importance Sampling} \]

Importance sampling

- Interested in estimating:
  \( \mathbb{E}_{X \sim P}(f(X)) = \int f(x)P(x)dx \)
  \( = \int f(x)P(x)\frac{Q(x)}{Q(x)}dx \) if \( Q(x) = 0 \Rightarrow P(x) = 0 \)
  \( = \mathbb{E}_{X \sim Q}(f(X)) \)
  \( \approx \frac{1}{m} \sum_{i=1}^{m} P(x^{(i)}|z_1, ..., z_t)f(x^{(i)}) \text{ with } x^{(i)} \sim Q \)

Sequential Importance Sampling (SIS) Particle Filter

- Sample \( x^{(1)}, x^{(2)}, ..., x^{(N)} \) from \( P(X_1) \)
- Set \( w_{t=1}^{(i)} = 1 \) for all \( i=1, ..., N \)
- For \( t=1, 2, ... \)
  - Dynamics update:
    - For \( i=1, 2, ..., N \)
      - Sample \( x^{(i)} \) from \( P(X_{t+1} | X_t = x^{(i)}_t) \)
    - Observation update:
      - For \( i=1, 2, ..., N \)
        - \( w_{t+1}^{(i)} = w_t^{(i)} \cdot P(z_{t+1} | X_{t+1} = x^{(i)}_{t+1}) \)
  - At any time \( t \), the distribution is represented by the weighted set of samples \( \{ < x^{(i)}_t, w^{(i)}_t > ; i=1, ..., N \} \)

SIS particle filter major issue

- The resulting samples are only weighted by the evidence
- The samples themselves are never affected by the evidence
  \( \Rightarrow \) Fails to concentrate particles/computation in the high probability areas of the distribution \( P(x_t | z_1, ..., z_t) \)

Resampling

- At any time \( t \), the distribution is represented by the weighted set of samples \( \{ < x^{(i)}_t, w^{(i)}_t > ; i=1, ..., N \} \)
- Sample \( N \) times from the set of particles
- The probability of drawing each particle is given by its importance weight
  \( \Rightarrow \) More particles/computation focused on the parts of the state space with high probability mass

1. Algorithm particle_filter( \( S_{t-1}, u_t, z_t \)):
   2. \( S_t = \emptyset, \quad \eta = 0 \)
   3. For \( i = 1, ..., n \)
     - Sample index \( j(i) \) from the discrete distribution given by \( w_s \)
     - Sample \( x^{(i)} \) from \( p(x_{t+1} | x_t = x^{(j(i))}, u_t) \) using \( x^{(j(i))} \) and \( u_t \)
   4. Compute importance weight
     - \( \eta = \eta + w_s \)
   5. Update normalization factor
     - \( \eta = \eta + w_s \)
   6. \( S_t = S_t \cup \{ < x^{(i)}, w_s > \} \)
   7. Insert
   8. \( S_t = S_t \cup \{ < x^{(i)}, w_s > \} \)
   9. Normalize weights
     - \( w_s = w_s / \eta \)
Particle Filters

\[ \text{Sensor Information: Importance Sampling} \]
\[
\begin{align*}
    \text{Bel}(x) & \leftarrow \alpha p(z | x) \text{Bel}'(x) \\
    w & \leftarrow \frac{\alpha p(z | x) \text{Bel}'(x)}{\text{Bel}'(x)} = \alpha p(z | x)
\end{align*}
\]

Robot Motion

\[ \text{Robot Motion} \]
\[
\begin{align*}
    \text{Bel}'(x) & \leftarrow p(x | u, x') \text{Bel}(x') \, dx'
\end{align*}
\]

Resampling issue

- Loss of samples ...
Low Variance Resampling

Advantages:
- More systematic coverage of space of samples
- If all samples have same importance weight, no samples are lost
- Lower computational complexity

Figure 4.2 Principle of the low variance resampling procedure. We choose a random number \( r \) and then select those particles that correspond to \( u = r + (m - 1) \cdot M^{-1} \) where \( m = 1, \ldots, M \).

Regularization

- If no dynamics noise \( \rightarrow \) all particles will start to coincide
- \( \rightarrow \) regularization: resample from a (narrow) Gaussian around the particle

Mobile Robot Localization

- Each particle is a potential pose of the robot
- Proposal distribution is the motion model of the robot (prediction step)
- The observation model is used to compute the importance weight (correction step)

Motion Model Reminder

Start

Proximity Sensor Model Reminder

Note: sensor model is not Gaussian at all!
Summary – Particle Filters

- Particle filters are an implementation of recursive Bayesian filtering
- They represent the posterior by a set of weighted samples
- They can model non-Gaussian distributions
- Proposal to draw new samples
- Weight to account for the differences between the proposal and the target
  - Monte Carlo filter, Survival of the fittest, Condensation, Bootstrap filter

Summary – PF Localization

- In the context of localization, the particles are propagated according to the motion model.
- They are then weighted according to the likelihood of the observations.
- In a re-sampling step, new particles are drawn with a probability proportional to the likelihood of the observation.