So far, we discussed the
Kalman filter: Gaussian, linearization problems, multi-modal beliefs

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

Basic principle
- Set of state hypotheses (“particles”)
- Survival-of-the-fittest

Density Approximation
- Particle sets can be used to approximate densities
  - The more particles fall into an interval, the higher the probability of that interval
  - How to draw samples form a function/distribution?
Let us assume that $f(x) \leq 1$ for all $x$
Sample $x$ from a uniform distribution
Sample $c$ from $[0,1]$
if $f(x) > c$ keep the sample
otherwise reject the sample

Rejection Sampling

We can even use a different distribution $g$ to generate samples from $f$
By introducing an importance weight $w$, we can account for the “differences between $g$ and $f$”
$w = \frac{f}{g}$
$f$ is often called target
$g$ is often called proposal

Importance Sampling Principle

Importance Sampling with Resampling: Landmark Detection Example

Distributions

Wanted: samples distributed according to $p(x \mid Z_1, Z_2, Z_3)$
This is Easy!

We can draw samples from $p(x|z_l)$ by adding noise to the detection parameters.

Importance Sampling with Resampling

Target distribution $f : p(x|z_1, z_2, ..., z_n) = \frac{\prod_{i=1}^{k} p(z_i | x) p(x)}{p(z_1, z_2, ..., z_n)}$

Sampling distribution $g: p(x|z_l) = \frac{p(z_l | x) p(x)}{p(z_l)}$

Importance weights $w : \frac{f}{g} = \frac{p(x|z_1, z_2, ..., z_n)}{p(x|z_l)} = \frac{\prod_{i=1}^{k} p(z_i | x)}{p(z_1, z_2, ..., z_n)}$

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, $n \log n$
- Stochastic universal sampling
- Systematic resampling
- Linear time complexity
- Easy to implement, low variance

Particle Filters

Sensor Information: Importance Sampling

$$Bel(x) \leftarrow \alpha \frac{p(z | x) Bel'(x)}{Bel'(x)} = \alpha \frac{p(z | x)}{Bel'(x)}$$

Robot Motion

$$Bel'(x) \leftarrow \int p(x | u, x') Bel(x') \, dx'$$
Sensor Information: Importance Sampling

Sensor Information: Importance Sampling

\[ \text{Bel}(x) \leftarrow \alpha p(z \mid x) \text{Bel}^\prime(x) \]

\[ w \leftarrow \alpha \frac{p(z \mid x) \text{Bel}^\prime(x)}{\text{Bel}^\prime(x)} = \alpha p(z \mid x) \]

Particle Filter Algorithm

1. Algorithm \texttt{particle\_filter}( S_{t-1}, u_{t-1} z_t):
2. \( S_i = \emptyset, \quad \eta = 0 \)
3. For \( i = 1 \ldots n \)
4. Sample index \( j(i) \) from the discrete distribution given by \( w_{i,j} \)
5. Sample \( x'_i \) from \( p(x_i \mid S_{i-1}, u_{i-1}) \) using \( x_{i-1}^{(j)} \) and \( u_{i-1} \)
6. \( w'_i = p(z_i \mid x'_i) \)
7. \( \eta = \eta + w'_i \)
8. \( S_i = S_i \cup \{ x'_i, w'_i \} \)
9. For \( i = 1 \ldots n \)
10. \( w'_i = w'_i / \eta \)

Robot Motion

Robot Motion

\[ \text{Bel}^\prime(x) \leftarrow \int p(x \mid u, x') \text{Bel}(x') \, dx' \]

Particle Filter Algorithm

Particle Filter Algorithm

\[ \text{Bel}(x_i) = \eta p(z_i \mid x_i) \int p(x_i \mid x_{i-1}, u_{i-1}) \text{Bel}(x_{i-1}) \, dx_{i-1} \]

Importance factor for \( x'_i \):

\[ w'_i = \frac{\eta \, p(z_i \mid x_i) \, p(x_i \mid x_{i-1}, u_{i-1}) \, \text{Bel}(x_{i-1})}{\int p(x_i \mid x_{i-1}, u_{i-1}) \, \text{Bel}(x_{i-1})} \approx p(z_i \mid x_i) \]
Motion Model Reminder

Start

10 meters

Proximity Sensor Model Reminder

Laser sensor

Sonar sensor
Using Ceiling Maps for Localization

[Image: Dellaert et al. 99]

Vision-based Localization

\[ P(z|x) \]

Under a Light

Measurement \( z \):

\[ P(z|x) \]

Next to a Light

Measurement \( z \):

\[ P(z|x) \]
Elsewhere

Measurement $z$: $P(z|x)$

Global Localization Using Vision

Recovery from Failure

Localization for AIBO robots
Adaptive Sampling

Adapt number of particles on the fly based on statistical approximation measure

KLD-Sampling Sonar

KLD-Sampling Laser

Particle Filter Projection
Density Extraction

Sampling Variance

CSE-571
Robotics

Bayes Filter Implementations

Discrete filters

Piecewise Constant

4/22/20
CSE-571
Discrete Bayes Filter Algorithm

1. Algorithm \texttt{Discrete\_Bayes\_filter}\,(\texttt{Bel(x),d }):
2. \( \eta = 0 \)
3. If \( d \) is a perceptual data item \( z \) then
   4. For all \( x \) do
   5. \( \text{Bel}'(x) = P(z|x)\text{Bel}(x) \)
   6. \( \eta = \eta + \text{Bel}'(x) \)
   7. For all \( x \) do
   8. \( \text{Bel}'(x) = \eta^{-1}\text{Bel}'(x) \)
9. Else if \( d \) is an action data item \( u \) then
   10. For all \( x \) do
   11. \( \text{Bel}'(x) = \sum_{x'} P(x|u,x') \text{Bel}(x') \)
12. Return \( \text{Bel}'(x) \)

Piecewise Constant Representation

Sonars and Occupancy Grid Map
**Tree-based Representation**

**Idea:** Represent density using a variant of Octrees

---

**Tree-based Representations**

- Efficient in space and time
- Multi-resolution