So far, we discussed the Kalman filter: Gaussian, linearization problems. Particle filters are a way to efficiently represent non-Gaussian distributions. Basic principle: Set of state hypotheses ("particles") Survival-of-the-fittest.

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

Function 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.

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 $f$”.
$w = f / g$.
$f$ is often called target.
$g$ is often called proposal.

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**

**Weighted samples**

**After resampling**

**Target distribution**

\[
T: \frac{p(z_1, z_2, \ldots, z_n) \prod p(z_i | x) p(x)}{p(z_1, z_2, \ldots, z_n)}
\]

**Sampling distribution**

\[
S: \frac{p(z_1 | x) p(x)}{p(z_1)}
\]

**Importance weights**

\[
I = \frac{p(z_1, z_2, \ldots, z_n)}{p(z_1) \prod p(z_i | x)}
\]

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

**Resampling Algorithm**

1. Algorithm `systematic_resampling(S, n)`:
   2. \( S = \emptyset, c_i = w_i \)
   3. For \( i = 2 \ldots n \)
      - Generate cdf
   4. \( c_i = c_{i-1} + w_i \)
      - Initialize threshold
   5. \( u_i \sim U(0, n^{-1}), i = 1 \)
   6. For \( j = 1 \ldots n \)
      - Draw samples ...  
      - Skip until next threshold reached
   7. While \( (u_j > c_i) \)
       - Insert
       - Increment threshold
   8. \( S = S \cup \{x', n^{-1}\} \)
   9. \( u_j = u_j + n^{-1} \)
   10. Return \( S' \) Also called stochastic universal sampling

**Particle Filters**

**Sensor Information: Importance Sampling**

\[
\begin{align*}
\text{Bel}(x) & \leftarrow \alpha \ p(z \mid x) \ \text{Bel}(x) \\
\alpha & \leftarrow \frac{\alpha \ p(z \mid x) \ \text{Bel}(x)}{\text{Bel}(x)} = \alpha \ p(z \mid x)
\end{align*}
\]
Robot Motion

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

Sensor Information: Importance Sampling

\[ \text{Bel}(x) \leftarrow \alpha p(z | x) \text{Bel}'(x) \]

\[ w_i \leftarrow \frac{\alpha p(z | x) \text{Bel}'(x)}{\text{Bel}(x)} = \alpha p(z | x) \]

Particle Filter Algorithm

1. Algorithm particle_filter( \( S_{t-1}, u_{t-1}, z_t \)):
2. \( S_t = \emptyset, \quad \eta = 0 \)
3. For \( i = 1...n \) \hspace{1cm} Generate new samples
4. Sample index \( j(i) \) from the discrete distribution given by \( w_{i,t-1} \)
5. Sample \( x'_i \) from \( p(x_t | x_{t-1}, u_{t-1}) \) using \( x^{(0)}_{i,t-1} \) and \( u_{t-1} \)
6. \( w'_i = p(z_t | x'_i) \) \hspace{1cm} Compute importance weight
7. \( \eta = \eta + w'_i \) \hspace{1cm} Update normalization factor
8. \( S_t = S_t \cup \{x'_i, w'_i\} \) \hspace{1cm} Insert
9. For \( i = 1...n \) \hspace{1cm} Normalize weights
10. \( w'_i = w'_i / \eta \)
Particle Filter Algorithm

\[ Bel(x_t) = \eta \int p(z_t | x_t) p(x_t | x_{t-1}, u_{t-1}) Bel(x_{t-1}) \, dx_{t-1} \]

- Draw \( x'_{t-1} \) from \( Bel(x_{t-1}) \)
- Draw \( x'_t \) from \( p(x_t | x'_{t-1}, u_{t-1}) \)

Importance factor for \( x'_t \):

\[ w'_t = \frac{\text{target distribution}}{\text{proposal distribution}} \]

\[ = \frac{\eta \int p(z_t | x_t) p(x_t | x_{t-1}, u_{t-1}) Bel(x_{t-1})}{p(x_t | x_{t-1}, u_{t-1}) Bel(x_{t-1})} \]

\[ = p(z_t | x_t) \]

Proximity Sensor Model Reminder

Laser sensor

Sonar sensor

Motion Model Reminder
Using Ceiling Maps for Localization

[Image: Ceiling Map]

Vision-based Localization

\[ P(z|x) \]

Under a Light

\[ \text{Measurement } z: \quad P(z|x): \]
Next to a Light

Elsewhere

Global Localization Using Vision

Recovery from Failure
Localization for AIBO robots

Hybrid Model for People Tracking

[Ferris-Haehnel-Fox: RSS-06]

WiFi Sensor Model

Tracking Example
Adaptive Sampling

KLD-Sampling Sonar
Adapt number of particles on the fly based on statistical approximation measure

KLD-Sampling Laser

Particle Filter Projection
Density Extraction

Sampling Variance

CSE-571
Probabilistic Robotics

Bayes Filter Implementations

Discrete filters

Piecewise Constant
Discrete Bayes Filter Algorithm

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

Piecewise Constant Representation

Grid-based Localization

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

---

Topological Localization

**XAVIER: Corridor Navigation**

July 1996
School of Computer Science
Carnegie Mellon University

---

Localization Algorithms - Comparison

<table>
<thead>
<tr>
<th></th>
<th>Kalman filter</th>
<th>Multi-hypothesis tracking</th>
<th>Topological maps</th>
<th>Grid-based (fixed/variable)</th>
<th>Particle filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensors</td>
<td>Gaussian</td>
<td>Gaussian</td>
<td>Features</td>
<td>Non-Gaussian</td>
<td>Non-Gaussian</td>
</tr>
<tr>
<td>Posterior</td>
<td>Gaussian</td>
<td>Multi-modal</td>
<td>Piecewise constant</td>
<td>Piecewise constant</td>
<td>Samples</td>
</tr>
<tr>
<td>Efficiency (memory)</td>
<td>+++</td>
<td>+++</td>
<td>+++</td>
<td>-/0</td>
<td>+++</td>
</tr>
<tr>
<td>Efficiency (time)</td>
<td>+++</td>
<td>+++</td>
<td>+++</td>
<td>o/+</td>
<td>+++</td>
</tr>
<tr>
<td>Implementation</td>
<td>+</td>
<td>o</td>
<td>+</td>
<td>+/o</td>
<td>+</td>
</tr>
<tr>
<td>Accuracy</td>
<td>+++</td>
<td>++</td>
<td>-</td>
<td>++/+</td>
<td>++</td>
</tr>
<tr>
<td>Robustness</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+/++</td>
</tr>
<tr>
<td>Global localization</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>