Particle Filter and Monte Carlo Localization (MCL)

EE468/CE468: Mobile Robotics

Dr. Basit Memon

Electrical and Computer Engineering
Habib University

November 13, 2023

Table of Contents

1 Particle Filter

2 How does it work?

3 Sources of error in the PF

4 References

- Markov localization (Bayes Algorithm) is computationally intractable.
    - Closed-form solution seldom exists.

Parametric Filters (Kalman)

+ Exact filter
+ Computationally efficient
- Exact for linear only
- Inaccurate, when true posterior is multi-modal

Grid-based

+ Easy - discretize space.
- Curse of dimensionality
  Dynamic discretization
- Wasted computations
  Lookup table
- Wasted memory
  Update select cells

Monte-Carlo

- Randomly sample space

+ Computationally better.
+ Accommodates arbitrary distribution.
-

Table of Contents

1 Particle Filter

2 How does it work?

3 Sources of error in the PF

4 References

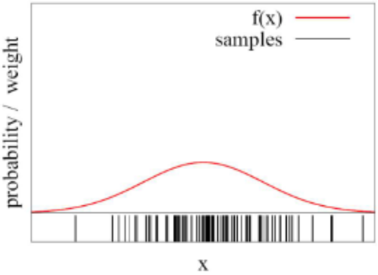
Key idea of a particle filter:

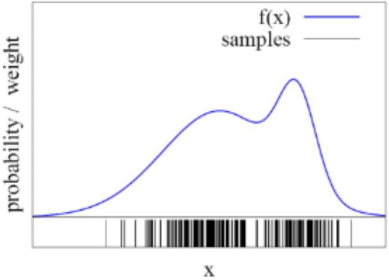
- Recall **Bayes Filter**

$$\overline{bel}(x_t) = \int p(x_t|\, u_t,\, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$$
$$bel(x_t) = \eta\, p(z_t|x_t)\, \overline{bel}(x_t)$$

- Represent posterior by set $\mathcal{M}$ of N samples (particles).

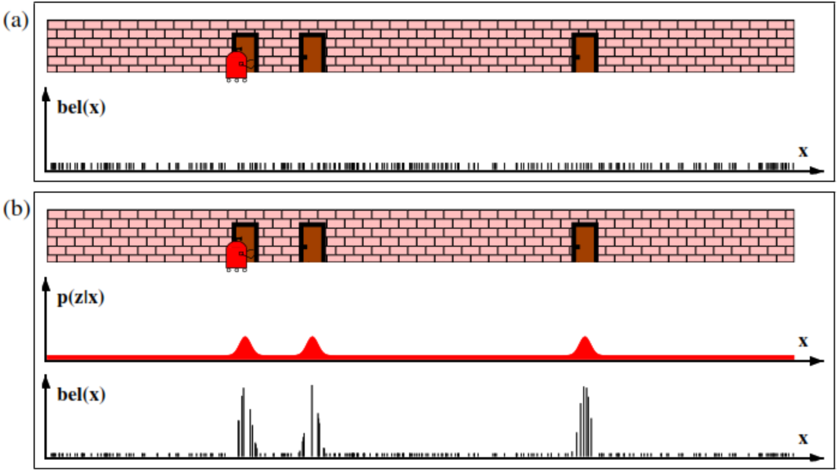
- Each sample, $(x^{[i]}, w^{[i]})$, consists of state hypothesis $x^{[i]}$ and an importance weight $w^{[i]}$.

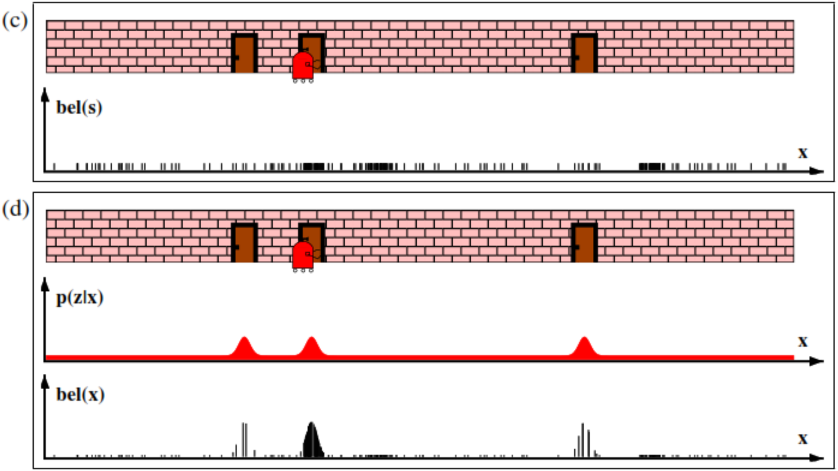




- The more particles fall in an interval, the higher the probability of that interval.
- x can be repeated in the samples.

Example of MonteCarlo Localization




(c)
bel(s)
x
(d)
p(z|x)
x
bel(x)
x

Example of MonteCarlo Localization

(e)

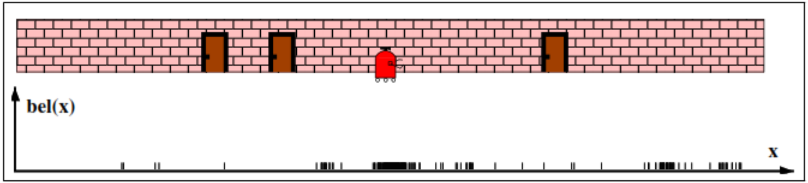


Table of Contents

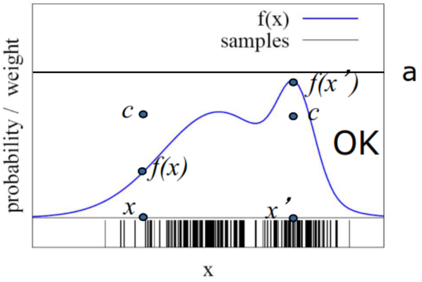
1 Particle Filter

2 How does it work?

3 Sources of error in the PF

4 References

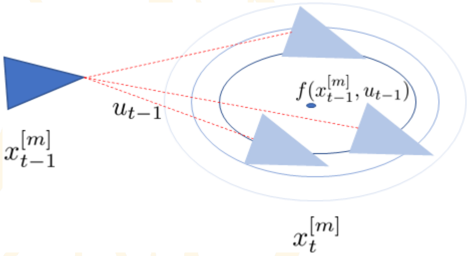
Sample the prior using Rejection Sampling (Acceptance-Rejection M



- Rejection sampling is one method to obtain samples to approximate $f(x)$.

- Assume that $f(x) < a$ for all x. Then,

1. Sample x from a uniform distribution.
2. Sample c from $[0, a]$
3. If $f(x) > c$, then keep the sample, otherwise reject it.



- We currently have

$$bel(x_{t-1}) = \left\{ \begin{array}{c} x_{t-1}^{[1]}, x_{t-1}^{[2]}, \cdots, x_{t-1}^{[N]} \\ w_{t-1}^{[1]}, w_{t-1}^{[2]}, \cdots, w_{t-1}^{[N]} \end{array} \right\}$$

- Simulate what is going to happen to each particle at next time step given the the input, and select one sample from it as the update for the particle.

$$x_t^{[m]} \sim p\left(x_t | x_{t-1}^{[m]}, u_t\right)$$

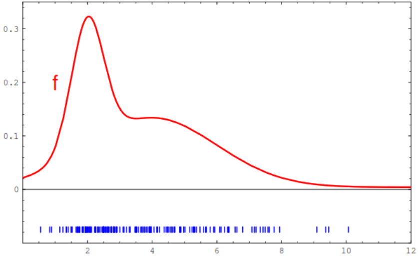
Update Step

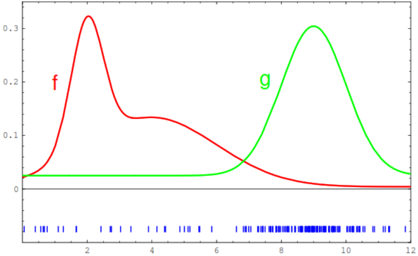
- How do we sample from

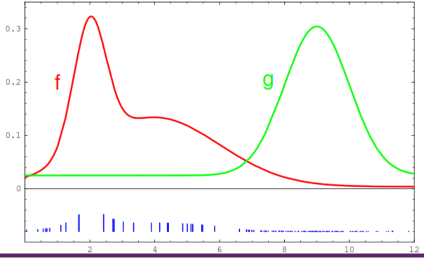
$$bel(x_t) = \eta \, p(z_t | x_t) \int p(x_t | u_t, x_{t-1}) \, bel(x_{t-1}) dx_{t-1} ?$$

- Importance Sampling
  - Sample from an easy proposal distribution
  - Reweigh samples to fix it.

Importance Sampling: Sample g and reweigh to represent f







Why does it work?

$$\begin{aligned} E_{f(x)}[X] &= \sum x\, f(x) \\ &= \sum x\, f(x) \frac{g(x)}{g(x)} \\ &= \sum x\, g(x) \frac{f(x)}{g(x)} \\ &= E_{g(x)}\left[\frac{f(x)}{g(x)}\, x\right] \end{aligned}$$

- Weigh the samples by $f(x)/g(x)$.
- For convergence, wherever $f(x) > 0$, the function $g(x) > 0$.

- Target distribution:

$$bel(x_t) = \eta\, p(z_t|x_t)\, \overline{bel}(x_t)$$

- Proposal distribution:

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

- Weighing ratio:

$$\frac{bel(x_t)}{\overline{bel}(x_t)} = \eta\, p(z_t|x_t)$$

- Weights are proportional to measurement likelihood.

- $$bel(x_{t-1}) = \left\{ \begin{array}{c} x_{t-1}^{[1]}, x_{t-1}^{[2]}, \cdots, x_{t-1}^{[N]} \\ w_{t-1}^{[1]}, w_{t-1}^{[2]}, \cdots, w_{t-1}^{[N]} \end{array} \right\}$$
- for $i = 1$ to N
  $$\text{sample } \bar{x}_t^{[i]} \sim p\left(x_t | u_t, x_{t-1}^{[i]}\right)$$
- for $i = 1$ to N
  $$w_t^{[i]} = p\left(z_t | \bar{x}_t^{[i]}\right) w_{t-1}^{[i]}$$
- $$bel(x_t) = \left\{ \begin{array}{c} \bar{x}_t^{[1]}, \bar{x}_t^{[2]}, \cdots, \bar{x}_t^{[N]} \\ w_t^{[1]}, w_t^{[2]}, \cdots, w_t^{[N]} \end{array} \right\}$$

$$w_t^{[i]} = \frac{w_t^{[i]}}{\sum w_t^{[i]}}$$

Problem: Unlikely particles consuming compute.

- Waste of computational resources if we propagate particles with too low weights.
- Solution: Survival of the fittest! Focus on most likely particles - resample.

Solution: Resampling

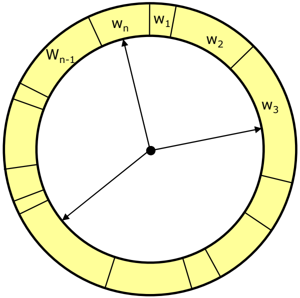
- Given: Set $\mathcal{M}$ of weighted samples
- Required: N samples where the probability of drawing $x^{[i]}$ is proportional to $w^{[i]}$.
- Create a bag of $x^{[i]}$ from $\mathcal{M}$, with instances of $x^{[i]}$ proportional to $w^{[i]}$.
- Draw N samples from the bag with replacement.
- Weights get reflected in the frequency of particles.

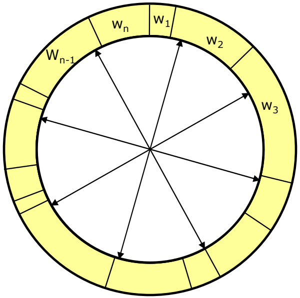
IS with resampling Particle Filter

- $$bel(x_{t-1}) = \left\{ \begin{array}{l} x_{t-1}^{[1]}, x_{t-1}^{[2]}, \cdots, x_{t-1}^{[N]} \\ w_{t-1}^{[1]}, w_{t-1}^{[2]}, \cdots, w_{t-1}^{[N]} \end{array} \right\}$$
- for $i = 1$ to N
  $$\text{sample } \bar{x}_t^{[i]} \sim p\left(x_t | u_t, x_{t-1}^{[i]}\right)$$
- for $i = 1$ to N
  $$w_t^{[i]} = p\left(z_t | \bar{x}_t^{[i]}\right) w_{t-1}^{[i]}$$
- for $i = 1$ to N, sample $x_t^{[i]} \sim w_t^{[i]}$
  $$bel(x_t) = \left\{ \begin{array}{l} x_t^{[1]}, x_t^{[2]}, \cdots, x_t^{[N]} \\ 1, 1, \cdots, 1 \end{array} \right\}$$

$$w_t^{[i]} = \frac{w_t^{[i]}}{\sum w_t^{[i]}}$$

Resampling





- Roulette wheel
- Binary search, n log n

- Stochastic universal sampling
- Systematic resampling
- Linear time complexity
- Easy to implement, low variance

Slide Credit: Probabilistic Robotics

- $\eta = w^1 + w^2 + \cdots + w^N$.
- Create cumulative weights array c, such that

$$c_i = (w^1 + w^2 + \cdots w^i)/\eta.$$

- Draw N random numbers $u_1, \cdots, u_N \sim U[0, 1]$.
- For each j = 1 to N, find the first normalized sum entry i such that $u_j < c_i$.
- Add sample x^{i-1} to new particle set.

Resampling Algorithm

1. Algorithm **systematic_resampling**(*S*,*n*):

2. $S' = \emptyset, c_1 = w^1$
3. **For** $i = 2 \ldots n$ — ***Generate cdf***
4. $\quad c_i = c_{i-1} + w^i$
5. $u_1 \sim U]0, n^{-1}], i = 1$ — ***Initialize threshold***

6. **For** $j = 1 \ldots n$ — ***Draw samples …***
7. $\quad$ **While** ($u_j > c_i$) — ***Skip until next threshold reached***
8. $\qquad i = i + 1$
9. $\quad S' = S' \cup \{ < x^i, n^{-1} > \}$ — ***Insert***
10. $\quad u_{j+1} = u_j + n^{-1}$ — ***Increment threshold***

11. **Return** *S'*

Also called **stochastic universal sampling**

Table of Contents

1 Particle Filter

2 How does it work?

3 Sources of error in the PF

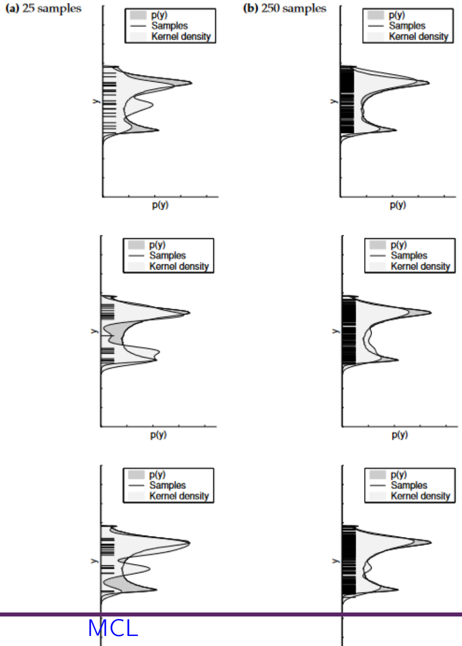
4 References

Problem: Sampling Variance

- Whenever a finite number of samples are drawn from any PDF, the statistics of the samples will differ from the underlying PDF.

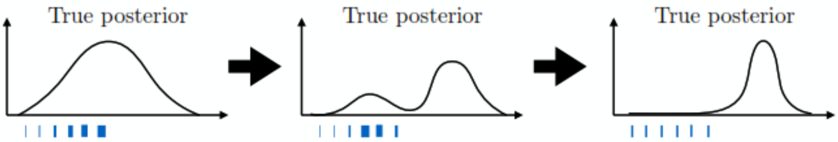
- **Sampling variance is amplified through repetitive resampling.**

- Suppose a robot does not move, but we don't know its location. Prior is uniform and PF starts with particles spread out. During resampling, PF will occasionally fail to reproduce a sample, $x^{[m]}$. Over time, with probability 1 we'll be left with identical copies of a single particle.


(a) 25 samples
(b) 250 samples
p(y)
Samples
Kernel density
y
p(y)

- Stochastic universal sampling is a low-variance sampling method.

Problem: Particle Deprivation



- Particles can get stuck in regions of low probability. No particles in the vicinity of correct state.
- Occurs mostly when not enough particles.
- Could happen because of sampling variance or an unlucky series of random numbers wipe out particles near true state.

Solution: Injection of random particles

- Robot might get kidnapped with a small probability at any time step.
- How many particles? From which distribution?

Algorithm Augmented_MCL$(\mathcal{X}_{t-1}, u_t, z_t, m)$**:**

static $w_{\text{slow}}, \ w_{\text{fast}}$
$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
for $m = 1$ *to* M *do*
  $x_t^{[m]} = \textbf{sample\_motion\_model}(u_t, x_{t-1}^{[m]})$
  $w_t^{[m]} = \textbf{measurement\_model}(z_t, x_t^{[m]}, m)$
  $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
  $w_{\text{avg}} = w_{\text{avg}} + \frac{1}{M}\, w_t^{[m]}$
endfor
$w_{\text{slow}} = w_{\text{slow}} + \alpha_{\text{slow}}(w_{\text{avg}} - w_{\text{slow}})$
$w_{\text{fast}} = w_{\text{fast}} + \alpha_{\text{fast}}(w_{\text{avg}} - w_{\text{fast}})$
for $m = 1$ *to* M *do*
  with probability $\max\{0.0,\ 1.0 - w_{\text{fast}}/w_{\text{slow}}\}$ *do*
    add random pose to $\mathcal{X}_t$
  else
    draw $i \in \{1, \ldots, N\}$ *with probability* $\propto w_t^{[i]}$
    add $x_t^{[i]}$ *to* $\mathcal{X}_t$
  endwith
endfor
return $\mathcal{X}_t$

Table 8.3 An adaptive variant of MCL that adds random samples. The number of random samples is determined by comparing the short-term with the long-term likelihood of sensor measurements.

Table of Contents

1 Particle Filter

2 How does it work?

3 Sources of error in the PF

4 References