Bel(x_t) = p(x_t | z_{1:t}, u_{1:t})

= \sum \{ p(z_t | x_t) \int p(x_t | x_{t-1}, u_t) \text{Bel}(x_{t-1}) dx_{t-1} \}

= \sum \{ p(z_t | x_t) \text{Bel}(x_t) \}

Sensor model

After motion update

Belief represented as particles:

\{ x_1^*, x_2^* \ldots x_M^* \} : M particles

Expected state (pose, e.g.) of the robot:

E(x_t) = \int \text{Bel}(x_t)x_t dx_t \rightarrow \text{Sampling from Bel}(x_t) \text{ is hard!}

(product of complex distributions)

How to sample?

1. Exhaustive Discretization: [Measure values for all x, y, \theta]
   for x in \{ x_i \}
   \text{A}[x] = \text{Bel}(x) \rightarrow \text{Very expensive!}
   \rightarrow \text{Sample from this array.}

2. Uniform Sampling: Fixed set of M particles
   \rightarrow \text{Need a lot of particles for accurate representation.}

3. Rejection Sampling:

4. Importance Sampling:

   E(x_t) = \int y R(x) dy = \int y \frac{p(x)}{q(x)} q(x) dy = \int y \frac{w(x)}{q(x)} dy
   \text{w(x) \rightarrow Importance weight = E[y w(x)]}

   \frac{w(x)}{q(x)} = \frac{p(x)}{q(x)} = \frac{\text{Target}}{\text{Proposal}}

   \text{Target: Hard to sample from}
   \text{Proposal: Easier to sample from}
To sample, you have to enumerate all possibilities. So, sometimes even if sampling is hard, evaluating may be easy. e.g. \( \Phi(t) \sim \text{depth} \)

Let \( g(x_t) = \text{proposal distribution} \).

Let \( g(x_t) = \text{Bel}(x_t) \rightarrow \text{Easy to sample from! (Proposal distribution)} \)

\( \omega(x_t) = \frac{\text{Bel}(x_t)}{\text{Bel}(x_t)} = \eta \cdot p(x_t | x_{t-1}) \rightarrow \text{Easier to evaluate!} \)

Caveat: \( \text{Bel}(x_t) > 0; g(x_t) > 0 \) otherwise.

Algorithm:

1. Proposal distribution \( \rightarrow \text{from motion model (sampling is easy)} \)
2. Target distribution \( \rightarrow \text{Bel}(x_t) \)

\( \text{PF (Bel} (x_{t-1}), z_t, u_t) \)

\( \propto (\{ x_{t-1}, t_{t-1}, ..., x_{t-1} \}, z_t, u_t) \)

Sequential Importance Sampling

\( \text{Bel} (x_{t-1}) \Rightarrow \langle x_{t-1}, ..., x_{t-1} \rangle \)

for \( i = 1 \) to \( M \)

\( x_t \sim \text{Bel} (x_{t-1}) \)

Sample \( [ \int p(x_t | x_{t-1}, u_t) \text{Bel} (x_{t-1}) dx_{t-1} \) \)

(only one part \( x_t \) for each particle)

\( p(x_t | x_{t-1}, u_t) \rightarrow \text{sampling directly from motion model (for PF) because it is just one particle, and one part } x_{t-1} \)

Importance weighting

\( \omega_t = \eta \cdot p(x_t | z_t, x_{t-1}) \omega_{t-1} \)

for \( i = 1 \) to \( M \)

\( \omega_t = \frac{\omega_t}{\sum \omega_t} \)

Normality

return \( \langle x_t, \omega_t \rangle \)
Why is sequential importance sampling not good?

- Infinite particles: no problem
- Finite particles: over time, sample with very low probability will keep increasing.

What change? → Resampling (selecting likely samples)

After importance weighting step, resample (sample with replacement)

\[ \text{for } i = 1 \text{ to } M \]
\[ \tilde{x}_t \sim \text{sample proportional to weights} \]
\[ w_t = \frac{1}{M} \quad (\text{no need double weighting}) \]

Sequential Importance Resampling Algorithm

So, Importance weighting step \( \Rightarrow \text{for } i = 1 \text{ to } M \)
\[ w_t = \eta P(z_t | \tilde{x}_t) \]

So, PF \( \langle \tilde{x}_t, \lambda_t, ... \tilde{x}_t \rangle \)

Sample
\[ \text{for } i = 1 \text{ to } M \]
\[ \tilde{x}_t \sim \text{Bel}(x_t) \]
\[ \sim P(x_t | x_{t-1}, \lambda_t) \]

Importance
\[ \text{for } i = 1 \text{ to } M \]
\[ w_t \sim \eta P(z_t | \tilde{x}_t) \]

Normalize
\[ \text{for } i = 1 \text{ to } M \]
\[ w_t = \frac{w_t}{\sum_{i} w_t} \]

Resample
Resample according to \( w_t \) (selecting likely samples)

Recompute \( \langle \tilde{x}_t^*, \lambda_t^*, ... \tilde{x}_t^* \rangle \)
So, how to resample?

1. Naive algorithm
   - compute CDF
   - for $i = 1$ to $M$
     - $\sigma = \text{rand}(0, 1)$
     - choose particle whose bin $\sigma$ falls into
   - PF with random resampling

Advantages
- Parallelizable
- Any motion/sensor model
- Sample/evaluate via variance dual PF

Disadvantages
- Low-variance sampler (stochastic)
  - 1 random sample $\gamma$ (not all)
  - $\gamma \sim U(0, \frac{1}{M})$

-> Then, increment it by $\frac{1}{M}$.
-> Not iid.
-> Guaranteed to sample a particle with weight $\frac{1}{M}$.
-> Can show ensemble variance is lower than naive random sampling.

P. T. O
Some Issues with PF

1) lack of Diversity / Particle starvation

\[ \text{Random sample?} \]

Solutions:

a) low-variance sampler ✓

b) Add random particle (when to add?)

\[ \min \frac{\hat{w}_i}{\max \hat{w}_i} \rightarrow 2 \] (threshold)

\[ \frac{\hat{w}_i}{\max \hat{w}_i} \rightarrow \text{resample!} \]

2) Perfect (really good) sensor model causes PF to fail!

⇒ extreme case example

\[ p(2|z) = \begin{cases} 1 & \text{if at } \cdot \cdot \cdot \\ 0 & \text{otherwise} \end{cases} \]

So, most particles weightless ⇒ last back!

e.g. bumper sensor (0 or 1 contact)

or LIDAR

Solutions:

a) sample directly from sensor model (but hard!)

or make "proposal" closer to sensor model

b) add more particles

b) Make sensor model noisy

i) \[ p(\bar{x} | x) = q \rightarrow q_p \] (squash it)

\[ 0 < p < 1 \]
GLOBAL LOCALIZATION (Active research area)
You don't know where you start (can't use GPS indoor)

Solution:
1. Make an inverse sensor model
   - \( p(z|x) \) → sample form (Belief − difficult)

   \[ w_i = p(z|x^i) \]

2. Uniform discretization (sample)
   - for \( i = 1 \) to \( M \)
   - \( w_i = p(z|x^i) \)
   - Choose max \( \hat{x} \)

3. for \( i = 1 \) to \( M \), \( M_s < M \)
   - Choose \( x^i \) as location's probability

   (any of the places in the three corridors)

   Sensor data

PRELOCALIZATION (KIDNAPPED ROBOT) PROBLEM (Active research area)

You don't know that you don't know
(you have been kidnapped)
\( \rightarrow \) wrong belief

1) How to know? \( \rightarrow \) don't care
2) Solution? \( \rightarrow \) Add random particles? Naïve
   a) Add random particles at each step? Naïve?
   b) \( \frac{1}{M} \sum_{i=1}^{M} p(z|x^i) \) have law observation probability

So:
\[ \hat{w}_i^j = \frac{1}{M} \sum_{i=1}^{M} p(z|x^i) \]

P. T. O
e.g.

\[\text{Wsey} \quad \text{(Some)} \quad \text{Wsey} \quad \text{(Better)}\]

In practice:

\[\text{Wsey} \quad \text{(Noise)} \quad \text{Need smoothing} \quad \text{Wslow} \quad \text{(Bad)}\]

\[\text{Wfast} \quad \text{Wslow} \quad \text{Wfast} > \text{Wslow} \quad \text{Wfast} \quad \text{Wslow}\]

Algorithm:

Repeat

\[\text{Compute } W_i \text{ for all particles, } i = 1 \text{ to } M\]

\[\text{Wsey} = \frac{1}{M} \sum_{i=1}^{M} W_i\]

\[\text{Wfast} = (1-\alpha_1) \text{Wfast} + \alpha_1 \text{Wsey}\]

\[\text{Wslow} = (1-\alpha_2) \text{Wslow} + \alpha_2 \text{Wsey}\]

\[\text{if } \left(\max \{0, 1 - \left(\frac{\text{Wfast}}{\text{Wslow}}\right)\} > 0\right) \text{ add random particle, or else nothing.}\]

Another variation:

Adaptively resample particle set.

(Random particle Monte-Carlo localization)