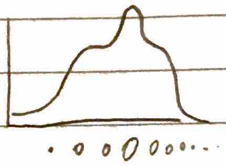
Intro: Particle Filters & Monte Carlo Localization

- estimating position of (localizing) the robot, given a map
- approach for dealing w arbitrary distributions (non-parametric)



→ more sample points & higher weights @ higher probability

Particle Set

- set of weighted samples: $X = \{ \langle x^{[j]}, w^{[j]} \rangle \}_{j=1,\dots,J}$

  state hypothesis ($x^{[j]}$), importance weight ($w^{[j]}$)

↳ every point is one possible state the system might be in

- the samples represent the posterior

$$p(x) = \sum_{j=1}^{J} w^{[j]} \delta_{x^{[j]}}(x)$$

→ Dirac Distribution centred in the state of the sample

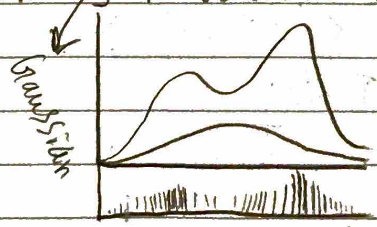
↳ the more particles fall into a region, the higher the prob. of that region

Importance Sampling

→ rejection sampling inefficient w dirac func.

- can use a different distribution g to generate samples from f
- weight = f/g
- f, g = target, proposal
- Pre-condition:

  $f(x) > 0 \rightarrow g(x) > 0$



→ re-weight samples according to this difference

↳ if g is zero, we'll never draw samples from there

Particle Filter - State Estimation

- recursive Bayes filter, non-parametric
- models distribution by samples
- Prediction: draw from proposal
- Correction: weighted by ratio of target & proposal

↳ the more samples, the better the estimate

Particle Filter Algorithm

① Sample particles using the proposal distribution

$$x_t^{[j]} \sim \pi(\underbrace{x_t \mid \ldots}_{\text{typically motion model}})$$

② Compute importance weights

$$w_t^{[j]} = \frac{\text{target}(x_t^{[j]})}{\text{proposal}(x_t^{[j]})}$$

③ Resampling: Draw sample i with probability $w_t^{[i]}$ and repeat J times.
↳ survival of fittest

Particle_filter(X_{t-1}, u_t, z_t):
$\bar{X}_t = X_t = \emptyset$ // empty sets
for $j = 1$ to J do:
Sample $x_t^{[j]} \sim \pi(x_t)$
$w_t^{[j]} = p(x_t^{[j]}) / \pi(x_t^{[j]})$
$\bar{X}_t = \bar{X}_t + \langle x_t^{[j]}, w_t^{[j]} \rangle$
for $j = 1$ to J do:
draw $i \in 1, \ldots, J$ with probability $\propto w_t^{[i]}$
add $x_t^{[i]}$ to X_t
return X_t

Monte Carlo Localization

- each particle is a pose hypothesis
- the motion model is the proposal } apply motion model to every sample
$$x_t^{[j]} \sim p(x_t \mid x_{t-1}, u_t)$$
- correction via the observation model

$$w_t^{[j]} = \frac{\text{target}}{\text{proposal}} \propto p(z_t \mid x_t, m)$$

→ if proposal distribution is the motion model (smart thing to do) then the weight becomes the observation model

Particle Filter for Localization (Monte Carlo Loc. (MCL))

Particle_filter(X_{t-1}, u_t, z_t)

$\bar{X}_t = X_t = \emptyset$

for j in J do:

Sample $x_t^{[j]} \sim P(x_t \mid u_t, x_{t-1}^{[j]})$

$w_t^{[j]} = P(z_t \mid x_t^{[j]})$ } this is what changed

$\bar{X}_t$ += $\langle x_t^{[j]}, w_t^{[j]} \rangle$

for j in J do:

draw $i \in 1, \ldots, J$ with probability $\propto w_t^{[i]}$

add $x_t^{[i]}$ to X_t

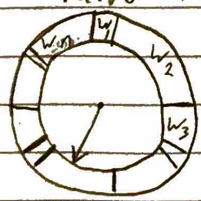
return X_t

- in the resampling step (second loop) we are replacing high weights with high sample frequency. eg, sample with weight 2 replaced with two samples of weight 1

↳ way to avoid many samples covering useless space since we have limited # of samples

Low Variance ReSampling

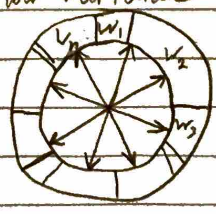
naive → never use



↳ use bin. search to find the random arrow for each sample

- $O(J \log J)$
- Particle depletion; if all weights same, no guarentee that they will all be picked again

low variance → always use → J = 8 here

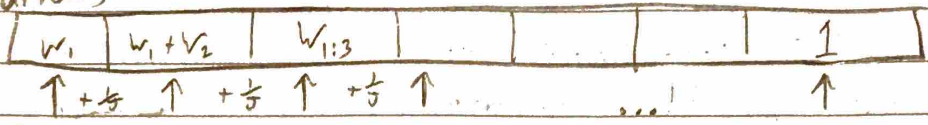


↳ find random arrow once then go around by even numb. sampling J times

- $O(J)$

★ - if every weight is same, this has added benefit of guarentee choosing each one once

Implementation:



1) Pick random num between 0 & $\frac{1}{J}$

2) for $(j = 1, \ldots, J)$:

$U = r + j \cdot \frac{1}{J}$

while (U > cumulative) i++;

Pick Particle i

Summary - Particle Filters Localization.

- Non-Parametric recursive Bayes filters
- Posterior represented by set of weighted samples
- Proposal to draw samples for t+1
- the art is to design appropriate motion & sensor models
- MCL is very commonly used - Golden Standard
- works well in low-dim spaces