ECE276A: Sensing & Estimation in Robotics
Lecture 8: Bayesian Filtering

Instructor:

Nikolay Atanasov: natanasov@ucsd.edu

Teaching Assistants:

Qiaojun Feng: qif007@eng.ucsd.edu

Arash Asgharivaskasi: aasghari@eng.ucsd.edu

Thai Duong: tduong@eng.ucsd.edu

Yiran Xu: y5xu@eng.ucsd.edu

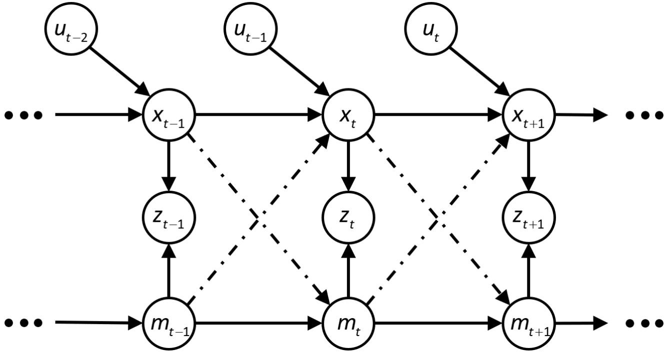
UC San Diego

JACOBS SCHOOL OF ENGINEERING

Electrical and Computer Engineering

Structure of Robotics Problems

- **Time**: t (discrete or continuous)
- **Robot state**: $\mathbf{x}_t$ (e.g., position, orientation, velocity)
- **Control input**: $\mathbf{u}_t$ (e.g., quadrotor thrust and torque)
- **Observation**: $\mathbf{z}_t$ (e.g., image, laser scan, inertial measurements)
- **Environment state**: $\mathbf{m}_t$ (e.g., map of the occupancy of space)



Structure of Robotics Problems

- The sequences of control inputs $\mathbf{u}_{0:t}$ and observations $\mathbf{z}_{0:t}$ are known/observed
- The sequences of robot states $\mathbf{x}_{0:t}$ and environment states $\mathbf{m}_{0:t}$ are unknown/hidden
- **Markov Assumptions**
  - The robot state $\mathbf{x}_{t+1}$ only depends on the previous input $\mathbf{u}_t$ and state $\mathbf{x}_t$, i.e., $\mathbf{x}_{t+1}$ given $\mathbf{u}_t$, $\mathbf{x}_t$ is independent of the history $\mathbf{x}_{0:t-1}$, $\mathbf{z}_{0:t-1}$, $\mathbf{u}_{0:t-1}$
  - The environment state $\mathbf{m}_{t+1}$ only depends on the previous environment state $\mathbf{m}_t$.
  - The environment state $\mathbf{m}_t$ and robot state $\mathbf{x}_t$ may affect each other's motion (e.g., collisions) but we do not make this explicit to simplify the presentation.
  - The observation $\mathbf{z}_t$ only depends on the robot state $\mathbf{x}_t$ and the environment state $\mathbf{m}_t$

Motion and Observation Models

- **Motion Model**: a nonlinear function f or equivalently a probability density function p_f that describes the motion of the robot to a new state $\mathbf{x}_{t+1}$ after applying control input $\mathbf{u}_t$ at state $\mathbf{x}_t$:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t) \sim p_f(\cdot \mid \mathbf{x}_t, \mathbf{u}_t) \qquad \mathbf{w}_t = \text{motion noise}$$

- The robot motion model may also depend on $\mathbf{m}_t$ and the environment may have its own motion model:

$$\mathbf{m}_{t+1} = a(\mathbf{m}_t, \mathbf{x}_t, \text{noise}_t) \sim p_a(\cdot \mid \mathbf{m}_t, \mathbf{x}_t)$$

- **Observation Model**: a function h or equivalently a probability density function p_h that describes the observation $\mathbf{z}_t$ of the robot depending on $\mathbf{x}_t$ and $\mathbf{m}_t$

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{m}_t, \mathbf{v}_t) \sim p_h(\cdot \mid \mathbf{x}_t, \mathbf{m}_t) \qquad \mathbf{v}_t = \text{observation noise}$$

Bayes Filter

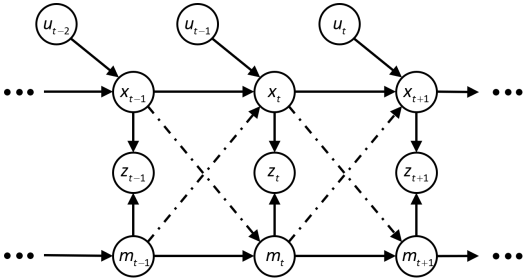
- **Bayes filtering** is a probabilistic inference technique for estimating the state of dynamical systems (robot and/or environment) that combines evidence from control inputs and observations using the **Markov assumptions** and **Bayes rule**:
  - **Total probability**: $p(x) = \int p(x, y) dy$
  - **Conditioning**: $p(x, y) = p(y \mid x) p(x)$
  - **Bayes rule**: $p(x \mid y, z) = \dfrac{p(y \mid x, z) p(x \mid z)}{\int p(y, s \mid z) ds} = \dfrac{p(y \mid x, z) p(z \mid x) p(x)}{p(y \mid z) p(z)}$
- Special cases of the Bayes filter:
  - Particle filter
  - Kalman filter
  - Forward algorithm for Hidden Markov Models (HMMs)

Filtering Examples

- Track the center $\mathbf{c}_t \in \mathbb{R}^2$ and radius $r_t \in \mathbb{R}$ of a ball in images: `http://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/`
- Track the position $\mathbf{p}_t \in \mathbb{R}^3$ and orientation $\mathbf{R}_t \in SO(3)$ of a camera: `https://www.youtube.com/watch?v=CsJkci5lfco`
- Estimate the probability of occupancy of a static environment represented as a grid $\mathbf{m}$: `https://www.youtube.com/watch?v=RhPlzIyTT58`

Filtering Problem

- The Markov assumptions are used to decompose the joint pdf of the states $\mathbf{x}_{0:T}$ (robot and map combined), observations $\mathbf{z}_{0:T}$, and controls $\mathbf{u}_{0:T-1}$



- **Joint distribution**:

$$p(\mathbf{x}_{0:T}, \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}) = \underbrace{p_{0|-1}(\mathbf{x}_0)}_{\text{prior}} \prod_{t=0}^{T} \underbrace{p_h(\mathbf{z}_t \mid \mathbf{x}_t)}_{\text{observation model}} \prod_{t=1}^{T} \underbrace{p_f(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1})}_{\text{motion model}}$$

- **Filtering**: keeps track of

$$p_{t|t}(\mathbf{x}_t) := p(\mathbf{x}_t \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1})$$
$$p_{t+1|t}(\mathbf{x}_{t+1}) := p(\mathbf{x}_{t+1} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t})$$

- **Smoothing**: keeps track of

$$p_{t|t}(\mathbf{x}_{0:t}) := p(\mathbf{x}_{0:t} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1})$$
$$p_{t+1|t}(\mathbf{x}_{0:t+1}) := p(\mathbf{x}_{0:t+1} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t})$$

Bayes Filter Prediction and Update Steps

- The Bayes filter relies on two steps to keep track of $p_{t|t}(\mathbf{x}_t)$ and $p_{t+1|t}(\mathbf{x}_{t+1})$
- **Prediction step**: given a prior density $p_{t|t}$ over $\mathbf{x}_t$ and the control input $\mathbf{u}_t$, uses the motion model p_f to compute the predicted density $p_{t+1|t}$ over $\mathbf{x}_{t+1}$:

$$\boxed{p_{t+1|t}(\mathbf{x}) = \int p_f(\mathbf{x} \mid \mathbf{s}, \mathbf{u}_t) p_{t|t}(\mathbf{s}) d\mathbf{s}}$$

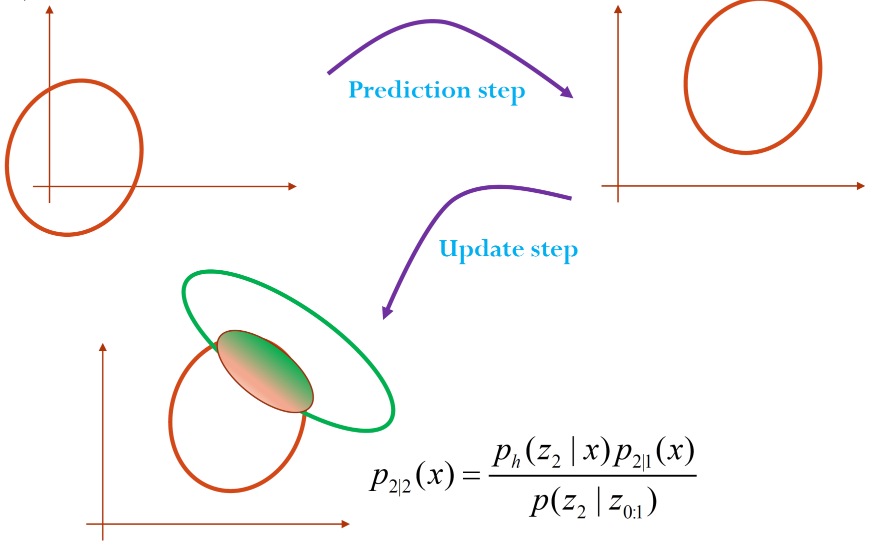
- **Update step**: given the predicted density $p_{t+1|t}$ over $\mathbf{x}_{t+1}$ and the measurement $\mathbf{z}_{t+1}$, uses the observation model p_h to incorporate the measurement information and obtain the posterior $p_{t+1|t+1}$ over $\mathbf{x}_{t+1}$:

$$\boxed{p_{t+1|t+1}(\mathbf{x}) = \frac{p_h(\mathbf{z}_{t+1} \mid \mathbf{x}) p_{t+1|t}(\mathbf{x})}{\int p_h(\mathbf{z}_{t+1} \mid \mathbf{s}) p_{t+1|t}(\mathbf{s}) d\mathbf{s}}}$$

Bayes Filter Illustration

$$p_{1|1}(x) := p(x_1 \mid z_{0:1}, u_0)$$

$$p_{2|1}(x) = \int p_f(x \mid s, u_1)\, p_{1|1}(s)\, ds$$



$$p_{2|2}(x) = \frac{p_h(z_2 \mid x)\, p_{2|1}(x)}{p(z_2 \mid z_{0:1})}$$

Bayes Filter Derivation

$$
\begin{aligned}
p_{t+1|t+1}(\mathbf{x}_{t+1}) &= p(\mathbf{x}_{t+1} \mid \mathbf{z}_{0:t+1}, \mathbf{u}_{0:t}) \\
&\overset{\text{Bayes}}{=\!=} \frac{1}{\eta_{t+1}} p(\mathbf{z}_{t+1} \mid \mathbf{x}_{t+1}, \mathbf{z}_{0:t}, \mathbf{u}_{0:t}) p(\mathbf{x}_{t+1} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t}) \\
&\overset{\text{Markov}}{=\!=} \frac{1}{\eta_{t+1}} p_h(\mathbf{z}_{t+1} \mid \mathbf{x}_{t+1}) p(\mathbf{x}_{t+1} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t}) \\
&\overset{\text{Total prob.}}{=\!=} \frac{1}{\eta_{t+1}} p_h(\mathbf{z}_{t+1} \mid \mathbf{x}_{t+1}) \int p(\mathbf{x}_{t+1}, \mathbf{x}_t \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t}) d\mathbf{x}_t \\
&\overset{\text{Cond. prob.}}{=\!=} \frac{1}{\eta_{t+1}} p_h(\mathbf{z}_{t+1} \mid \mathbf{x}_{t+1}) \int p(\mathbf{x}_{t+1} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_t) p(\mathbf{x}_t \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t}) d\mathbf{x}_t \\
&\overset{\text{Markov}}{=\!=} \frac{1}{\eta_{t+1}} p_h(\mathbf{z}_{t+1} \mid \mathbf{x}_{t+1}) \int p_f(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t) p(\mathbf{x}_t \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1}) d\mathbf{x}_t \\
&= \boxed{\frac{1}{\eta_{t+1}} p_h(\mathbf{z}_{t+1} \mid \mathbf{x}_{t+1}) \int p_f(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t) p_{t|t}(\mathbf{x}_t) d\mathbf{x}_t}
\end{aligned}
$$

- **Normalization constant**: $\eta_{t+1} := p(\mathbf{z}_{t+1} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t})$

Bayes Filter Summary

- **Motion model**: $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t) \sim p_f(\cdot \mid \mathbf{x}_t, \mathbf{u}_t)$
- **Observation model**: $\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{v}_t) \sim p_h(\cdot \mid \mathbf{x}_t)$
- **Joint distribution**:

$$p(\mathbf{x}_{0:T}, \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}) = \underbrace{p_{0|-1}(\mathbf{x}_0)}_{\text{prior}} \prod_{t=0}^{T} \underbrace{p_h(\mathbf{z}_t \mid \mathbf{x}_t)}_{\text{observation model}} \prod_{t=1}^{T} \underbrace{p_f(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1})}_{\text{motion model}}$$

- **Filtering**: recursive computation of $p(\mathbf{x}_T | \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1})$ that tracks:

$$p_{t|t}(\mathbf{x}_t) := p(\mathbf{x}_t \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1})$$
$$p_{t+1|t}(\mathbf{x}_{t+1}) := p(\mathbf{x}_{t+1} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t})$$

- **Bayes filter**:

$$p_{t+1|t+1}(\mathbf{x}_{t+1}) = \underbrace{\overbrace{\frac{1}{p(\mathbf{z}_{t+1}|\mathbf{z}_{0:t}, \mathbf{u}_{0:t})}}^{\frac{1}{\eta_{t+1}}} p_h(\mathbf{z}_{t+1} \mid \mathbf{x}_{t+1}) \overbrace{\int p_f(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t) p_{t|t}(\mathbf{x}_t) d\mathbf{x}_t}^{\textbf{Predict: } p_{t+1|t}(\mathbf{x}_{t+1})}}_{\textbf{Update}}$$

Bayes Smoother

- **Smoothing**: recursive computation of $p(\mathbf{x}_{0:T}|\mathbf{z}_{0:T}, \mathbf{u}_{0:T-1})$ that tracks

$$p_{t|t}(\mathbf{x}_{0:t}) := p(\mathbf{x}_{0:t} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1})$$
$$p_{t+1|t}(\mathbf{x}_{0:t+1}) := p(\mathbf{x}_{0:t+1} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t})$$

- Forward pass (**Bayes filter**): compute $p(\mathbf{x}_{t+1} \mid \mathbf{z}_{0:t+1}, \mathbf{u}_{0:t})$ and $p(\mathbf{x}_{t+1} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t})$ for $t = 0, \ldots, T$

- Backward pass (**Bayes smoother**): for $t = T-1, \ldots, 0$ compute:

$$p(\mathbf{x}_t \mid \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}) \xlongequal[\text{Probability}]{\text{Total}} \int p(\mathbf{x}_t \mid \mathbf{x}_{t+1}, \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}) p(\mathbf{x}_{t+1} \mid \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}) d\mathbf{x}_{t+1}$$

$$\xlongequal[\text{Assumption}]{\text{Markov}} \int p(\mathbf{x}_t \mid \mathbf{x}_{t+1}, \mathbf{z}_{0:t}, \mathbf{u}_{0:t}) p(\mathbf{x}_{t+1} \mid \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}) d\mathbf{x}_{t+1}$$

$$\xlongequal[\text{Rule}]{\text{Bayes}} \underbrace{p(\mathbf{x}_t \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1})}_{\text{forward pass}} \int \left[\frac{\overbrace{p_f(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t)}^{\text{motion model}} p(\mathbf{x}_{t+1} \mid \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1})}{\underbrace{p(\mathbf{x}_{t+1} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t})}_{\text{forward pass}}} \right] d\mathbf{x}_{t+1}$$

Histogram Filter

- Represents the pdfs $p_{t|t}$ and $p_{t+1|t}$ via a histogram over a discrete set $\mathcal{X}$ of possible state values
- Since $\mathcal{X}$ is discrete, the integrals in the prediction and update steps reduce to sums
- **Prediction step**: given a prior probability mass function $p_{t|t}$ over $\mathbf{x}_t$ and control input $\mathbf{u}_t$, uses the motion model p_f to compute the predicted pmf $p_{t+1|t}$ over $\mathbf{x}_{t+1}$:

$$p_{t+1|t}(\mathbf{x}_{t+1}) = \sum_{\mathbf{s}\in\mathcal{X}} p_f(\mathbf{x}_{t+1} \mid \mathbf{s}, \mathbf{u}_t) p_{t|t}(\mathbf{s})$$

- **Update step**: given the predicted pmf $p_{t+1|t}$ over $\mathbf{x}_{t+1}$ and measurement $\mathbf{z}_{t+1}$, uses the observation model p_h to incorporate the measurement information and obtain the posterior $p_{t+1|t+1}$ over $\mathbf{x}_{t+1}$:

$$p_{t+1|t+1}(\mathbf{x}_{t+1}) = \frac{p_h(\mathbf{z}_{t+1} \mid \mathbf{x}_{t+1}) p_{t+1|t}(\mathbf{x}_{t+1})}{\sum_{\mathbf{s}\in\mathcal{X}} p_h(\mathbf{z}_{t+1} \mid \mathbf{s}) p_{t+1|t}(\mathbf{s})}$$

Efficient Histogram Filter Prediction

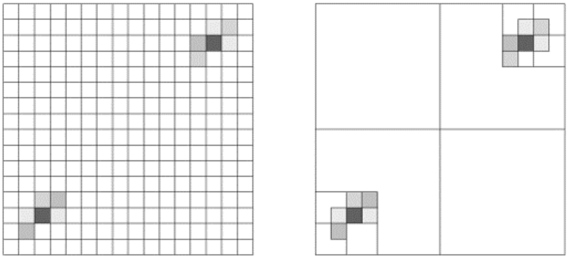
- Assume bounded Gaussian noise in the motion model
- Prediction step:
  - shift the data $p_{t|t}(\mathbf{x})$ in the grid $\mathbf{x} \in \mathcal{X}$ to a new index $\mathbf{x}'$ according to the control input $\mathbf{u}$ and the motion model $\mathbf{x}' = f(\mathbf{x}, \mathbf{u})$
  - convolve the shifted grid values with a **separable** Gaussian kernel:

$$\begin{bmatrix} 1/16 & 1/8 & 1/16 \\ 1/8 & 1/4 & 1/8 \\ 1/16 & 1/8 & 1/16 \end{bmatrix} \cong \begin{bmatrix} 1/4 \\ 1/2 \\ 1/4 \end{bmatrix} + \begin{bmatrix} 1/4 & 1/2 & 1/4 \end{bmatrix}$$

- This reduces the prediction step cost from $O(n^2)$ to $O(n)$ where n is the number of cells in $\mathcal{X}$

Histogram Filter Memory

- The accuracy of the histogram filter is limited by the size of the grid $\mathcal{X}$
- A small-resolution grid becomes very computationally expensive in high dimensional state spaces because the number of cells is exponential in the number of dimensions
- **Adaptive Histogram Filter**: represents the pmf via adaptive discretization, e.g., octrees

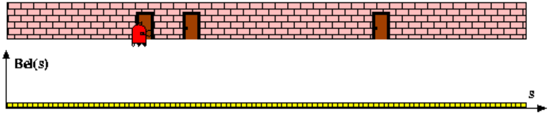


Markov Localization

- **Robot Localization Problem**: Given a map $\mathbf{m}$, a sequence of control inputs $\mathbf{u}_{0:t-1}$, and a sequence of measurements $\mathbf{z}_{0:t}$, infer the state of the robot $\mathbf{x}_t$
- **Approach**: use a Bayes filter with a multi-modal distribution in order to capture multiple hypotheses about the robot state, e.g.:
  - Histogram filter
  - Particle filter
  - Gaussian mixture filter
- **Important considerations**:
  - How is the map $\mathbf{m}$ represented?
  - What are the motion and observation models?
  - Need to keep the number of hypotheses about the value of $\mathbf{x}_t$ under control

Histogram Filter Localization (1-D)

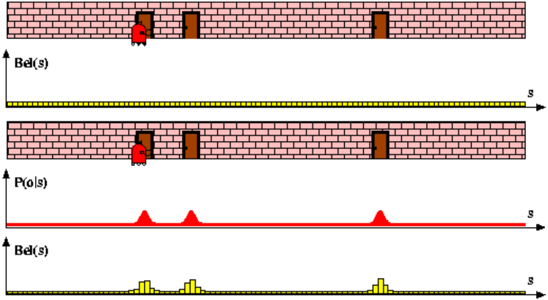
Prior:



Histogram Filter Localization (1-D)

Prior:

Update:

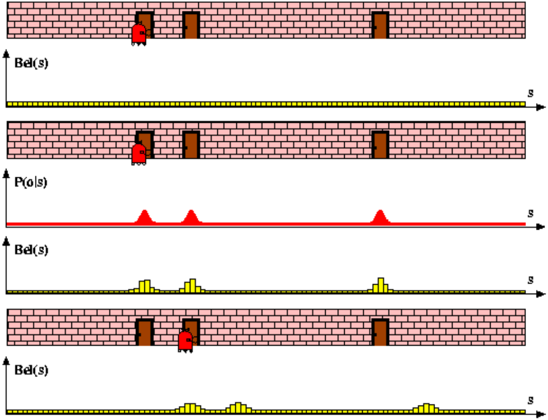


Histogram Filter Localization (1-D)

Prior:

Update:

Predict:



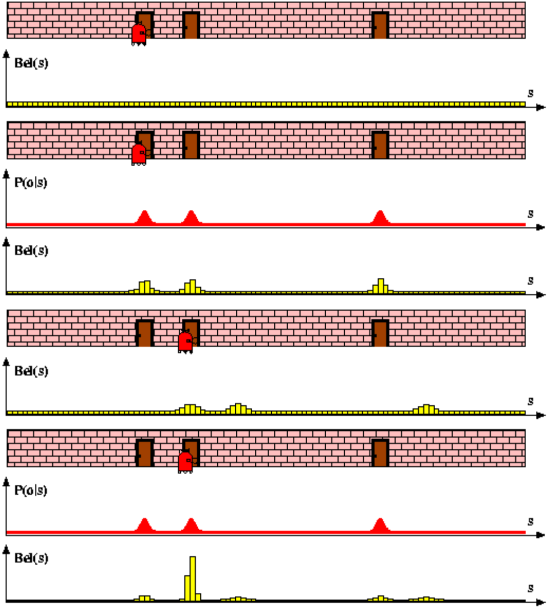
Histogram Filter Localization (1-D)

Prior:

Update:

Predict:

Update:



Particle Filter

- The histogram filter uses pmfs over a grid to represent $p_{t|t}$ and $p_{t+1|t}$
- The particle filter uses a mixture of delta functions (**particles**):

$$\delta(\mathbf{x}; \boldsymbol{\mu}^{(k)}) := \begin{cases} \infty & \mathbf{x} = \boldsymbol{\mu}^{(k)} \\ 0 & \text{else} \end{cases} \qquad \text{for } k = 1, \ldots, N$$

  with weights $\alpha^{(k)}$ to represent $p_{t|t}$ and $p_{t+1|t}$, i.e.,

$$p(\mathbf{x}_t \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1}) = p_{t|t}(\mathbf{x}_t) = \sum_{k=1}^{N_{t|t}} \alpha_{t|t}^{(k)} \delta\left(\mathbf{x}_t; \boldsymbol{\mu}_{t|t}^{(k)}\right)$$

$$p(\mathbf{x}_{t+1} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t}) = p_{t+1|t}(\mathbf{x}_{t+1}) = \sum_{k=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(k)} \delta\left(\mathbf{x}_{t+1}; \boldsymbol{\mu}_{t+1|t}^{(k)}\right)$$

- To derive the filter, substitute the delta mixture pdfs in the Bayes filter prediction and update steps
- The prediction and update steps should maintain the mixture-of-delta-functions form of the pdfs

Particle Filter Prediction

- How do we approximate the prediction step as a delta-mixture pdf?

$$p_{t+1|t}(\mathbf{x}) = \int p_f(\mathbf{x} \mid \mathbf{s}, \mathbf{u}_t) \sum_{k=1}^{N_{t|t}} \alpha_{t|t}^{(k)} \delta\left(\mathbf{s}; \boldsymbol{\mu}_{t|t}^{(k)}\right) d\mathbf{s}$$

$$= \sum_{k=1}^{N_{t|t}} \alpha_{t|t}^{(k)} p_f(\mathbf{x} \mid \boldsymbol{\mu}_{t|t}^{(k)}, \mathbf{u}_t) \overset{??}{\approx} \sum_{k=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(k)} \delta\left(\mathbf{x}; \boldsymbol{\mu}_{t+1|t}^{(k)}\right)$$

- Since $p_{t+1|t}(\mathbf{x})$ is a mixture pdf with components $p_f(\mathbf{x} \mid \boldsymbol{\mu}_{t|t}^{(k)}, \mathbf{u}_t)$, we may approximate it with particles by drawing samples from it
- This process consists of two steps:
  - **Resampling**: given particles $\left\{\boldsymbol{\mu}_{t|t}^{(k)}, \alpha_{t|t}^{(k)}\right\}$ for $k = 1, \ldots, N_{t|t}$, create a new set, $\left\{\bar{\boldsymbol{\mu}}_{t|t}^{(k)}, \bar{\alpha}_{t|t}^{(k)}\right\}$ for $k = 1, \ldots, N_{t+1|t}$ (usually $N_{t+1|t} = N_{t|t}$)
  - **Prediction**: apply the motion model to each $\bar{\boldsymbol{\mu}}_{t|t}^{(k)}$ by drawing $\boldsymbol{\mu}_{t+1|t}^{(k)} \sim p_f\left(\cdot \mid \bar{\boldsymbol{\mu}}_{t|t}^{(k)}, u_t\right)$ and set $\alpha_{t+1|t}^{(k)} = \bar{\alpha}_{t|t}^{(k)}$

Particle Filter Update

- Evaluate Bayes rule with the predicted delta-mixture pdf

$$
\begin{aligned}
p_{t+1|t+1}(\mathbf{x}) &= \frac{p_h\left(\mathbf{z}_{t+1} \mid \mathbf{x}\right) \sum_{k=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(k)} \delta\left(\mathbf{x}; \boldsymbol{\mu}_{t+1|t}^{(k)}\right)}{\int p_h\left(\mathbf{z}_{t+1} \mid \mathbf{s}\right) \sum_{j=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(j)} \delta\left(\mathbf{s}; \boldsymbol{\mu}_{t+1|t}^{(j)}\right) d\mathbf{s}} \\
&= \sum_{k=1}^{N_{t+1|t}} \left[\frac{\alpha_{t+1|t}^{(k)} p_h\left(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}_{t+1|t}^{(k)}\right)}{\sum_{j=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(j)} p_h\left(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}_{t+1|t}^{(j)}\right)}\right] \delta\left(\mathbf{x}; \boldsymbol{\mu}_{t+1|t}^{(k)}\right)
\end{aligned}
$$

- The resulting pdf turns out to be a delta mixture so no approximation is necessary

- The update step does not update the particle positions but only their weights

Particle Filter Summary

- **Prior**: $\mathbf{x}_t \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1} \sim p_{t|t}(\mathbf{x}_t) := \sum_{k=1}^{N} \alpha_{t|t}^{(k)} \delta\left(\mathbf{x}_t; \boldsymbol{\mu}_{t|t}^{(k)}\right)$
- **Resampling**: If $N_{eff} := \frac{1}{\sum_{k=1}^{N}\left(\alpha_{t|t}^{(k)}\right)^2} \leq N_{threshold}$, resample the particle set $\left\{\boldsymbol{\mu}_{t|t}^{(k)}, \alpha_{t|t}^{(k)}\right\}$ via stratified or sample importance resampling
- **Prediction**: let $\boldsymbol{\mu}_{t+1|t}^{(k)} \sim p_f\left(\cdot \mid \boldsymbol{\mu}_{t|t}^{(k)}, u_t\right)$ and $\alpha_{t+1|t}^{(k)} = \alpha_{t|t}^{(k)}$ so that:

$$p_{t+1|t}(\mathbf{x}) \approx \sum_{k=1}^{N} \alpha_{t+1|t}^{(k)} \delta\left(\mathbf{x}; \boldsymbol{\mu}_{t+1|t}^{(k)}\right)$$

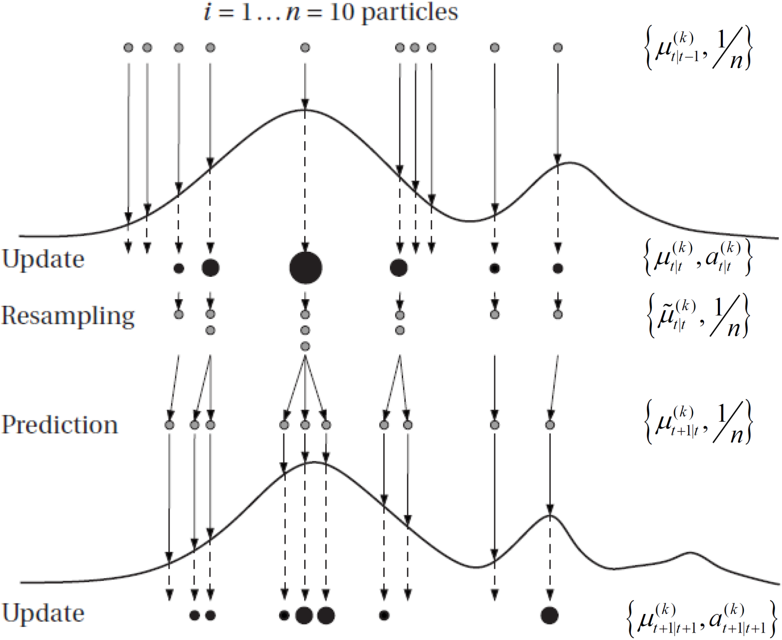
- **Update**: rescale the particles based on the observation likelihood:

$$p_{t+1|t+1}(\mathbf{x}) = \sum_{k=1}^{N} \left[\frac{\alpha_{t+1|t}^{(k)} p_h\left(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}_{t+1|t}^{(k)}\right)}{\sum_{j=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(j)} p_h\left(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}_{t+1|t}^{(j)}\right)}\right] \delta\left(\mathbf{x}; \boldsymbol{\mu}_{t+1|t}^{(k)}\right)$$

Particle Resampling

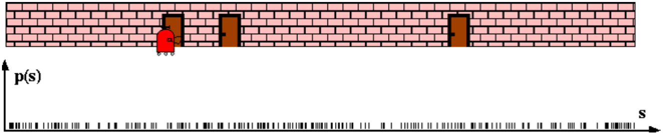
- **Particle depletion**: a situation in which most of the updated particle weights become close to zero because the finite set of particles are not accurate hypotheses, i.e., the observation likelihoods $p_h\left(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}^{(k)}_{t+1|t}\right)$ are small at all $k = 1, \ldots, N$
- The resampling procedure tries to avoid particle depletion
- Given a weighted particle set, resampling creates a new particle set with **equal weights** by adding many particles to the locations that had high weight and few particles to the locations that had low weights
- Resampling focuses the representation power of the particles to likely regions, while leaving unlikely regions with only few particles
- Resampling is applied at time t if the **effective number of particles**: $\boxed{N_{eff} := \frac{1}{\sum_{k=1}^{N}\left(\alpha^{(k)}_{t|t}\right)^2}}$ is less than a threshold

Particle Filter Resampling



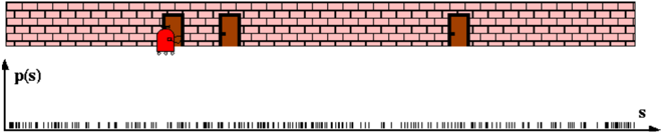
Particle Filter Localization (1-D)

Prior:

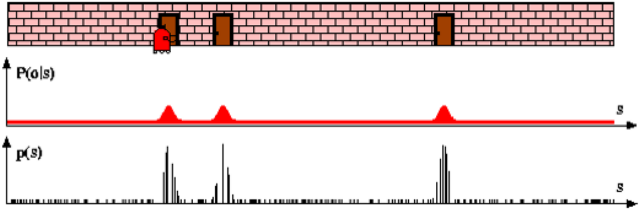


Particle Filter Localization (1-D)

Prior:

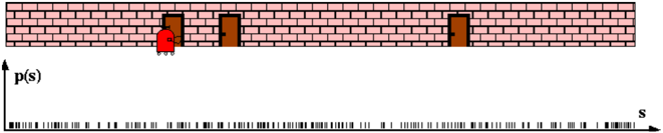


Update:

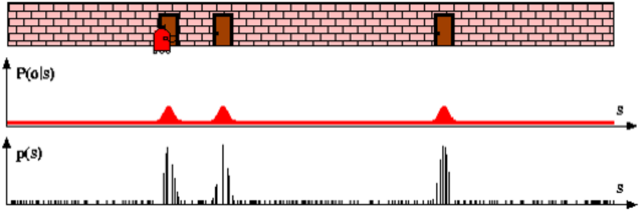


Particle Filter Localization (1-D)

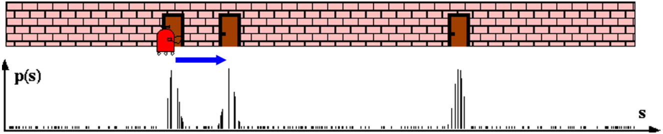
Prior:



Update:



Predict:



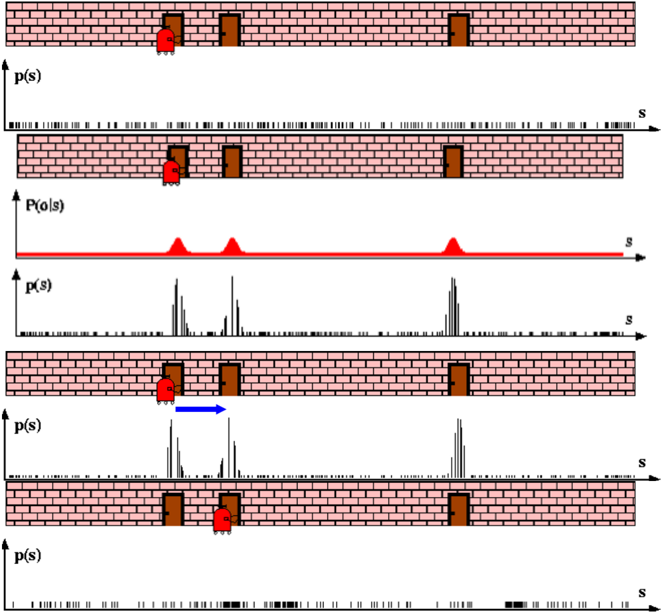
Particle Filter Localization (1-D)

Prior:

Update:

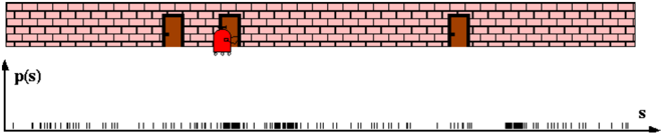
Predict:

Resample:



Particle Filter Localization (1-D)

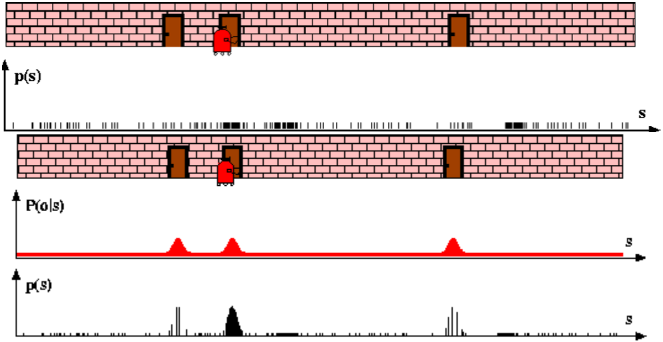
Prior:



Particle Filter Localization (1-D)

Prior:

Update:

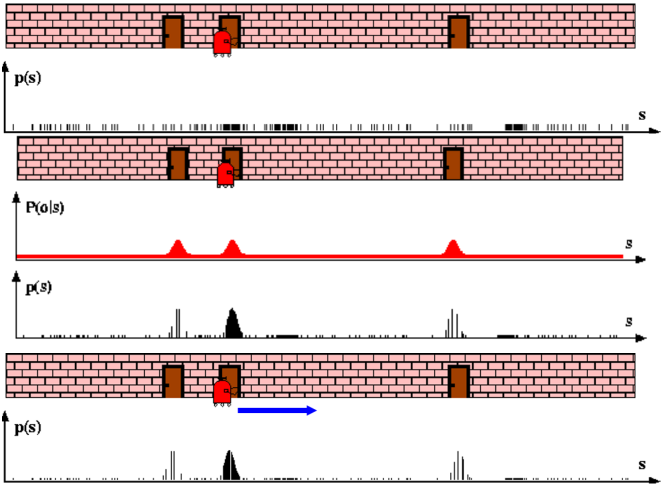


Particle Filter Localization (1-D)

Prior:

Update:

Predict:



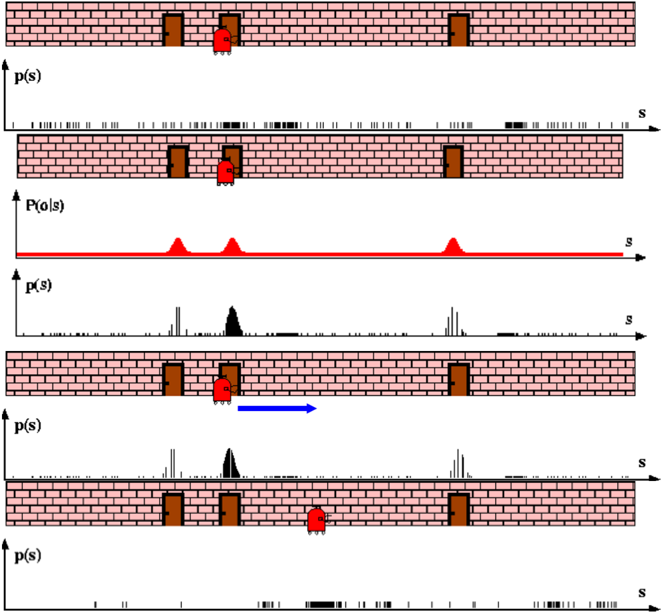
Particle Filter Localization (1-D)

Prior:

Update:

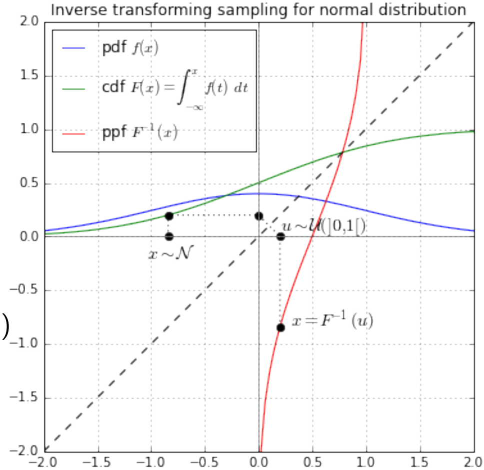
Predict:

Resample:



Inverse Transform Sampling

- **Target distribution**: How do we sample from a distribution with pdf $p(x)$ and CDF $F(x) = \int_{-\infty}^{x} p(s)ds$?

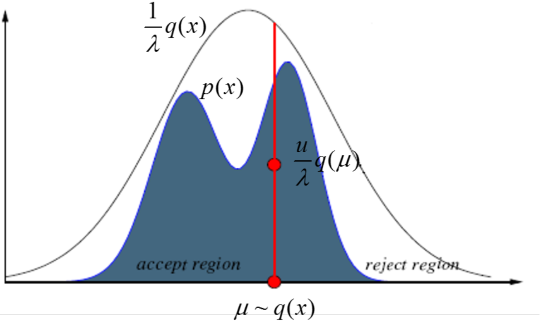


- **Inverse Transform Sampling**:
  1. Draw $u \sim \mathcal{U}(0,1)$
  2. Return inverse CDF value: $\mu = F^{-1}(u)$
  3. The CDF of $F^{-1}(u)$ is:

$$\mathbb{P}(F^{-1}(u) \leq x) = \mathbb{P}(u \leq F(x)) \\ = F(x)$$

Rejection Sampling

- **Target distribution**: How do we sample from a complicated pdf $p(x)$?
- **Proposal distribution**: use another pdf $q(x)$ that is easy to sample from (e.g., Uniform, Gaussian) and: $\lambda p(x) \leq q(x)$ with $\lambda \in (0, 1)$
- **Rejection Sampling**:
  1. Draw $u \sim \mathcal{U}(0, 1)$ and $\mu \sim q(\cdot)$
  2. Return μ only if $u \leq \frac{\lambda p(\mu)}{q(\mu)}$. If λ is small, many rejections are necessary
- Good $q(x)$ and λ are **hard to choose** in practice



Sample Importance Resampling (SIR)

- How about rejection sampling without λ?
- **Sample Importance Resampling** for a target distribution $p(\cdot)$ with proposal distribution $q(\cdot)$
  1. Draw $\mu^{(1)}, \dots, \mu^{(N)} \sim q(\cdot)$
  2. Compute importance weights $\alpha^{(k)} = \frac{p(\mu^{(k)})}{q(\mu^{(k)})}$ and normalize: $\alpha^{(k)} = \frac{\alpha^{(k)}}{\sum_j \alpha^{(j)}}$
  3. Draw $\mu^{(k)}$ independently with replacement from $\{\mu^{(1)}, \dots, \mu^{(N)}\}$ with probability $\alpha^{(k)}$ and add to the final sample set with weight $\frac{1}{N}$
- If $q(\cdot)$ is a poor approximation of $p(\cdot)$, then the best samples from q are not necessarily good samples for resampling

Markov Chain Monte Carlo Resampling

- The main drawback of rejection sampling and SIR is that choosing a good proposal distribution $q(\cdot)$ is hard
- **Idea**: let the proposed samples μ depend on the last accepted sample μ', i.e., obtain correlated samples from a conditional proposal distribution $\mu^{(k)} \sim q\left(\cdot \mid \mu^{(k-1)}\right)$
- Under certain conditions, the samples generated from $q(\cdot \mid \mu')$ form an ergodic Markov chain with $p(\cdot)$ as its stationary distribution
- MCMC methods include Metropolis-Hastings and Gibbs sampling

SIR applied to the Particle Filter

- Let $\left\{\boldsymbol{\mu}_{t|t}^{(k)}, \alpha_{t|t}^{(k)}\right\}$ for $k = 1, \ldots, N$ be the particle set at time t
- If $N_{eff} := \frac{1}{\sum_{k=1}^{N}\left(\alpha_{t|t}^{(k)}\right)^2} \leq N_{threshold}$, create a new set $\left\{\bar{\boldsymbol{\mu}}_{t|t}^{(k)}, \bar{\alpha}_{t|t}^{(k)}\right\}$ for $k = 1, \ldots, N$ as follows
- Repeat N times:
  - Draw $j \in \{1, \ldots, N\}$ independently with resplacement with discrete probability $\alpha_{t|t}^{(j)}$
  - Add the sample $\boldsymbol{\mu}_{t|t}^{(j)}$ with weight $\frac{1}{N}$ to the new particle set

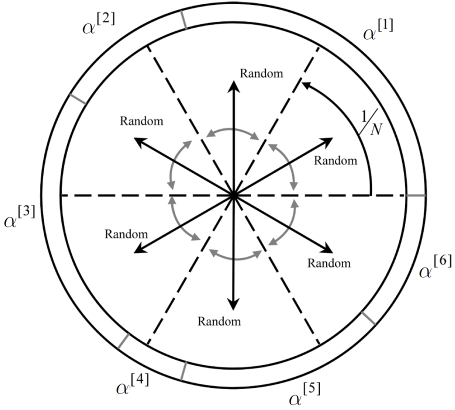
Stratified Resampling

- In SIR, the weighted set $\{\boldsymbol{\mu}^{(k)}, \alpha^{(k)}\}$ is sampled independently with replacement
- This might result in high variance resampling, i.e., sometimes some samples with large weights might not be selected or samples with very small weights may be selected multiple times
- **Stratified resampling**: guarantees that samples with large weights appear at least once and those with small weights – at most once. Stratified resampling is **optimal in terms of variance** (Thrun et al. 2005)
- Instead of selecting samples independently, use a sequential process:
  - Add the weights along the circumference of a circle
  - Divide the circle into N equal pieces and sample a uniform on each piece
  - Samples with large weights are chosen at least once and those with small weights – at most once

Stratified and Systematic Resampling

Stratified (low variance) resampling

1: **Input**: particle set $\{\boldsymbol{\mu}^{(k)}, \alpha^{(k)}\}_{k=1}^{N}$
2: **Output**: resampled particle set
3: $j \leftarrow 1$, $c \leftarrow \alpha^{(1)}$
4: **for** $k = 1, \ldots, N$ **do**
5: $\quad u \sim \mathcal{U}\left(0, \frac{1}{N}\right)$
6: $\quad \beta = u + \frac{k-1}{N}$
7: $\quad$ **while** $\beta > c$ **do**
8: $\quad\quad j = j + 1$, $c = c + \alpha^{(j)}$
9: $\quad$ add $\left(\boldsymbol{\mu}^{(j)}, \frac{1}{N}\right)$ to the new set



- **Systematic resampling**: the same as stratified resampling except that the **same** uniform is used for each piece, i.e., $u \sim \mathcal{U}\left(0, \frac{1}{N}\right)$ is sampled only once before the for loop above.