ECE276A: Sensing & Estimation in Robotics
Lecture 10: Gaussian Mixture and Particle Filtering

Lecturer:

Nikolay Atanasov: natanasov@ucsd.edu

Teaching Assistants:

Siwei Guo: s9guo@eng.ucsd.edu

Anwesan Pal: a2pal@eng.ucsd.edu

UC San Diego

JACOBS SCHOOL OF ENGINEERING

Electrical and Computer Engineering

Vectors Notation

$x \in \mathbb{R}^d$ — d-dimensional vector in Euclidean space. It may be deterministic or random. If it is random we describe it with a parametric distribution with parameters θ or w or equivalently with its probability density function $p(x; w)$. **Example**: $x \sim \mathcal{N}(\mu, \Sigma)$ with pdf $\phi(x; \mu, \Sigma)$, where $\mathcal{N}$ is the distribution, ϕ is the pdf, and $\{\mu, \Sigma\}$ are the deterministic vector and matrix parameters.

$A \in \mathbb{R}^{n \times m}$ — An $n \times m$ dimensional matrix. Its transpose is $A^T \in \mathbb{R}^{m \times n}$ and its inverse (if it exists) is $A^{-1} \in \mathbb{R}^{n \times m}$ for $n = m$.

$\|x\|_2 := x^T x$ — vector Euclidean norm

$x(t)$ — a vector that changes in continuous time $t \in \mathbb{R}_{>0}$

x_t — a vector that changes in discrete time $t \in \mathbb{N}$

$\dot{x}(t) := \frac{d}{dt} x(t)$ — time derivative of $x(t)$

Random Vectors Notation

$x \in \mathbb{R}^d$	d-dimensional random vector with CDF $F(\cdot)$ and pdf $p(\cdot)$
$\mathbb{E}[h(x)]$	expectation of a function h of the random vector x: $$\mathbb{E}[h(x)] := \int h(x)p(x)dx$$
$\mu := \mathbb{E}[x]$	mean of x, also called first moment of x
$x^* := \arg\max_x p(x)$	mode of x
$\mathbb{E}\left[xx^T\right]$	second moment of x
$\Sigma := \mathbb{E}\left[(x-\mu)(x-\mu)^T\right] = \mathbb{E}\left[xx^T\right] - \mu\mu^T$	(co)variance of x

Parameter Estimation Notation

$D := \{\mathbf{x}_i, y_i\}_{i=1}^n$	**training data** of examples $\mathbf{x}_i \in \mathbb{R}^d$ and labels $y_i \in \{-1, 1\}$ (classification) or $y_i \in \mathbb{R}$ (regression). The elements $(\mathbf{x}_i, y_i)$ are iid (over i) sampled from an unknown joint (over $\mathbf{x}_i$ and y_i) pdf $p^*(\mathbf{x}_i, y_i)$
$p(\mathbf{x}_i, y_i; \omega)$	**generative model** using pdf parameters ω
$p(y_i \mid \mathbf{x}_i; \omega)$	**discriminative model** using parameters ω
$\max_\omega \prod_{i=1}^n p(\mathbf{x}_i, y_i; \omega)$	**training**: parameter estimation via MLE
$\max_y p(x, y; \omega)$	**testing**: label prediction for given new sample x and already trained parameters ω
$\sigma(z) := \frac{1}{1+\exp(-z)} \in \mathbb{R}$	**sigmoid function**: useful for modeling binary classification
$\text{softmax}(z) := \frac{e^z}{1^T e^z} \in \mathbb{R}^d$	**softmax function**: useful for modeling K-ary classification

Orientations Notation

$a \in \mathbb{R}^3$	an axis-angle representation of orientation/rotation, also called rotation vector. The axis of rotation is $\xi := \frac{a}{\|a\|_2}$ and the angle is $\theta := \|a\|_2$
$\hat{a} \in \mathfrak{so}(3)$	a skew-symmetric matrix associated with cross products $\hat{a}b = a \times b$ for $b \in \mathbb{R}^3$. Skew-symmetric matrices define the Lie algebra (i.e., local linear approximation) of the Lie group of rotations.
$R = \exp(\hat{a}) \in SO(3)$	rotation matrix representation of the rotation vector $a \in \mathbb{R}^3$. The matrix exponential function is $\exp(A) := \sum_{k=0}^{\infty} \frac{A^k}{k!}$ and for rotations can be computed in closed form via the Rodrigues formula.
$q = \exp([0, \frac{1}{2}a]) \in \mathbb{S}^3$	quaternion representation of the rotation vector $a \in \mathbb{R}^3$. The quaternion exponential function is not the same as the matrix exponential function!

Transformations Notation

$\omega \in \mathbb{R}^3$	Angular velocity – defines the rate of change of orientation for rotation matrices $\dot{R} = \hat{\omega} R$ or equivalently for quaternions $\dot{q} = \left[0, \frac{\omega}{2}\right] \circ q$
$\zeta := (\omega, v) \in \mathbb{R}^6$	Linear velocity $v \in \mathbb{R}^3$ and angular velocity $\omega \in \mathbb{R}^3$
$\hat{\zeta} \in \mathfrak{se}(3)$	a twist matrix $\begin{bmatrix} \hat{\omega} & v \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4}$ defining the Lie algebra of the Lie group of rigid body transformations. A twist $\hat{\zeta}$ defines the rate of change of pose of position $\dot{p} = \hat{\omega} p + v$ and orientation $\dot{R} = \hat{\omega} R$
$g = \exp(\hat{\zeta}) \in SE(3)$	a matrix $g = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$ representing rigid body transformations, where $R \in SO(3)$ is the rotation/orientation and $p \in \mathbb{R}^3$ is the translation/position

Transformations Notation

${}_W g_B \in SE(3) \qquad x_t \in SE(3)$

Body frame to world frame transformation defined by the body pose ${}_W g_B$ (or robot state x_t)

$$g_1 \oplus g_2 := g_1 g_2 = \begin{bmatrix} R_1 & p_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_2 & p_2 \\ 0 & 1 \end{bmatrix}$$

Transformation composition in $SE(3)$ (equivalent to addition in Euclidean spaces)

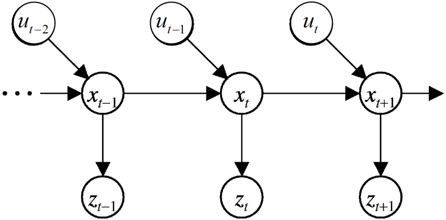
$$g_2 \ominus g_1 := g_1^{-1} g_2 = \begin{bmatrix} R_1^T & -R_1^T p_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_2 & p_2 \\ 0 & 1 \end{bmatrix}$$

Transformation inverse composition in $SE(3)$ (equivalent to subtraction in Euclidean spaces)

Filtering Notation

$x_t \in \mathbb{R}^d$	system/robot state **to be estimated** (e.g., position, orientation, etc.). Usually $x_t \sim \mathcal{N}(\mu_{t\|t}, \Sigma_{t\|t})$ with pdf $\phi(x_t; \mu_{t\|t}, \Sigma_{t\|t})$
$u_t \in \mathbb{R}^{d_u}$	**known** system/robot control input (e.g., rotational velocity)
$z_t \in \mathbb{R}^{d_z}$	**known** system/robot measurement/observation (e.g., pixel coordinates)
$w_t \sim \mathcal{N}(0, W)$	Gaussian motion noise
$v_t \sim \mathcal{N}(0, V)$	Gaussian observation noise
$p_{t\|t}(x_t)$	pdf of the robot state x_t given past measurements $z_{0:t}$ and control inputs $u_{0:t-1}$: $p_{t\|t}(x_t) := p(x_t \mid z_{0:t}, u_{0:t-1})$
$p_{t+1\|t}(x_{t+1})$	predicted pdf of the robot state x_{t+1} given past measurements $z_{0:t}$ and control inputs $u_{0:t}$: $p_{t+1\|t}(x_{t+1}) := p(x_{t+1} \mid z_{0:t}, u_{0:t})$

Bayes Filter



- **Motion model**:

  $x_{t+1} = a(x_t, u_t, w_t) \sim p_a(\cdot \mid x_t, u_t)$

- **Observation model**:

  $z_t = h(x_t, v_t) \sim p_h(\cdot \mid x_t)$

- **Filtering**: keeps track of

$$p_{t|t}(x_t) := p(x_t \mid z_{0:t}, u_{0:t-1})$$
$$p_{t+1|t}(x_{t+1}) := p(x_{t+1} \mid z_{0:t}, u_{0:t})$$

- **Bayes filter**:

$$p_{t+1|t+1}(x_{t+1}) = \underbrace{\overbrace{\frac{1}{p(z_{t+1}|z_{0:t}, u_{0:t})}}^{\frac{1}{\eta_{t+1}}} p_h(z_{t+1} \mid x_{t+1}) \overbrace{\int p_a(x_{t+1} \mid x_t, u_t) p_{t|t}(x_t) dx_t}^{\textbf{Predict: } p_{t+1|t}(x_{t+1})}}_{\textbf{Update}}$$

- **Joint distribution**:

$$p(x_{0:T}, z_{0:T}, u_{0:T-1}) = \underbrace{p_{0|0}(x_0)}_{\text{prior}} \prod_{t=0}^{T} \underbrace{p_h(z_t \mid x_t)}_{\text{observation model}} \prod_{t=0}^{T} \underbrace{p_a(x_t \mid x_{t-1}, u_{t-1})}_{\text{motion model}}$$

Gaussian Mixture Filter

- **Prior**: $x_t \mid z_{0:t}, u_{0:t-1} \sim p_{t|t}(x_x) := \sum_k \alpha_{t|t}^{(k)} \phi\left(x_t; \mu_{t|t}^{(k)}, \Sigma_{t|t}^{(k)}\right)$
- **Motion model**: $x_{t+1} = Ax_t + Bu_t + w_t, \quad w_t \sim \mathcal{N}(0, W)$
- **Observation model**: $z_t = Hx_t + v_t, \quad v_t \sim \mathcal{N}(0, V)$
- **Prediction**:

$$p_{t+1|t}(x) = \int p_a(x \mid s, u_t) p_{t|t}(s) ds = \sum_k \alpha_{t|t}^{(k)} \int p_a(x \mid s, u_t) \phi\left(s; \mu_{t|t}^{(k)}, \Sigma_{t|t}^{(k)}\right) ds$$

$$= \sum_k \alpha_{t|t}^{(k)} \phi\left(x; A\mu_{t|t}^{(k)} + Bu_t, A\Sigma_{t|t}^{(k)} A^T + W\right)$$

- **Update**:

$$p_{t+1|t+1}(x) = \frac{p_h(z_{t+1} \mid x) p_{t+1|t}(x)}{p(z_{t+1} \mid z_{0:t}, u_{0:t})} = \frac{\phi(z_{t+1}; Hx, V) \sum_k \alpha_{t+1|t}^{(k)} \phi\left(x; \mu_{t+1|t}^{(k)}, \Sigma_{t+1|t}^{(k)}\right)}{\int \phi(z_{t+1}; Hs, V) \sum_j \alpha_{t+1|t}^{(j)} \phi\left(s; \mu_{t+1|t}^{(j)}, \Sigma_{t+1|t}^{(j)}\right) ds}$$

$$= \sum_k \left(\frac{\alpha_{t+1|t}^{(k)} \phi(z_{t+1}; Hx, V) \phi\left(x; \mu_{t+1|t}^{(k)}, \Sigma_{t+1|t}^{(k)}\right)}{\sum_j \alpha_{t+1|t}^{(j)} \phi\left(z_{t+1}; H\mu_{t+1|t}^{(j)}, H\Sigma_{t+1|t}^{(j)} H^T + V\right)} \times \frac{\phi\left(z_{t+1}; H\mu_{t+1|t}^{(k)}, H\Sigma_{t+1|t}^{(k)} H^T + V\right)}{\phi\left(z_{t+1}; H\mu_{t+1|t}^{(k)}, H\Sigma_{t+1|t}^{(k)} H^T + V\right)} \right)$$

$$= \sum_k \left[\frac{\alpha_{t+1|t}^{(k)} \phi\left(z_{t+1}; H\mu_{t+1|t}^{(k)}, H\Sigma_{t+1|t}^{(k)} H^T + V\right)}{\sum_j \alpha_{t+1|t}^{(j)} \phi\left(z_{t+1}; H\mu_{t+1|t}^{(j)}, H\Sigma_{t+1|t}^{(j)} H^T + V\right)} \right] \phi\left(x; \mu_{t+1|t}^{(k)} + K_{t+1|t}^{(k)} (z_{t+1} - H\mu_{t+1|t}^{(k)}), (I - K_{t+1|t}^{(k)} H) \Sigma_{t+1|t}^{(k)}\right)$$

- **Kalman Gain**: $K_{t+1|t}^{(k)} := \Sigma_{t+1|t}^{(k)} H^T \left(H\Sigma_{t+1|t}^{(k)} H^T + V\right)^{-1}$

Gaussian Mixture Filter

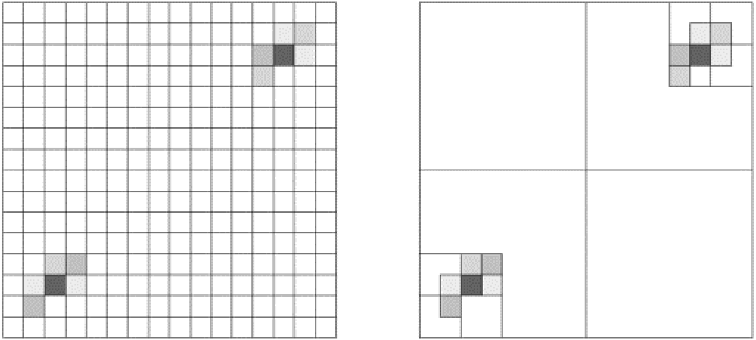
- **pdf**: $x_t \mid z_{0:t}, u_{0:t-1} \sim p_{t|t}(x) := \sum_k \alpha_{t|t}^{(k)} \phi\left(x_t; \mu_{t|t}^{(k)}, \Sigma_{t|t}^{(k)}\right)$
- **mean**: $\mu_{t|t} := \mathbb{E}[x_t \mid z_{0:t}, u_{0:t-1}] = \int x p_{t|t}(x) dx = \sum_k \alpha_{t|t}^{(k)} \mu_{t|t}^{(k)}$
- **cov**: $\Sigma_{t|t} := \mathbb{E}\left[x_t x_t^T \mid z_{0:t}, u_{0:t-1}\right] - \mu_{t|t}\mu_{t|t}^T$
  $= \int x x^T p_{t|t}(x) dx - \mu_{t|t}\mu_{t|t}^T = \sum_k \alpha_{t|t}^{(k)} \left(\Sigma_{t|t}^{(k)} + \mu_{t|t}^{(k)} (\mu_{t|t}^{(k)})^T\right) - \mu_{t|t}\mu_{t|t}^T$
- The GMF is just a **bank of Kalman filters**; sometimes called **Gaussian Sum filter**

Gaussian Mixture Filter

- If the motion or observation models are nonlinear, we can apply the EKF or UKF tricks to get a nonlinear GMF
- Additional operations are needed when strong nonlinearities are present in the motion or observation models:
  - **Refinement**: introduces additional components to reduce the linearization error
  - **Pruning**: approximates the overall distribution with a smaller number of components (e.g., using KL divergence as a measure of accuracy)
- More details:
  - Nonlinear Gaussian Filtering: Theory, Algorithms, and Applications: Huber
  - Bayesian Filtering and Smoothing: Särkkä

Histogram Filter

- Represent the pdf via a histogram over a discrete set of possible locations
- The accuracy is limited by the grid size
- A small grid becomes very computationally expensive in high dimensional state spaces because the number of cells is exponential in the number of dimensions
- **Idea**: represent the pdf via adaptive discretization, e.g., octrees



Histogram Filter

- **Prediction step**
  - Assume bounded Gaussian noise in the motion model
  - The prediction step can be realized by shifting the data in the grid according to the control input and convolving the grid 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
- **Update step**
  - To update and normalize the pdf upon sensory input, one has to iterate over all cells
  - Is it possible to monitor which part of the state space is affected by the observations and only update that?

Particle Filter

- A Gaussian mixture filter with $\Sigma_{t|t}^{(k)} \to 0$ so that $\phi\left(x_t; \mu_{t|t}^{(k)}, \Sigma_{t|t}^{(k)}\right) \to \delta\left(x_t; \mu_{t|t}^{(k)}\right) := \mathbb{1}\{x_t = \mu_{t|t}^{(k)}\}$
- **Prior**: $x_t \mid z_{0:t}, u_{0:t-1} \sim p_{t|t}(x_x) := \sum_{k=1}^{N_{t|t}} \alpha_{t|t}^{(k)} \delta\left(x_t; \mu_{t|t}^{(k)}\right)$
- **Motion model**: $x_{t+1} \sim p_a(\cdot \mid x_t, u_t)$
- **Observation model**: $z_t \sim p_h(\cdot \mid x_t)$
- **Prediction**:

$$p_{t+1|t}(x) = \int p_a(x \mid s, u_t) \sum_{k=1}^{N_{t|t}} \alpha_{t|t}^{(k)} \delta\left(s; \mu_{t|t}^{(k)}\right) ds = \sum_{k=1}^{N_{t|t}} \alpha_{t|t}^{(k)} p_a(x \mid \mu_{t|t}^{(k)}, u_t) \overset{??}{\approx} \sum_{k=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(k)} \delta\left(x; \mu_{t+1|t}^{(k)}\right)$$

- **Update**:

$$p_{t+1|t+1}(x) = \frac{p_h(z_{t+1} \mid x) \sum_{k=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(k)} \delta\left(x; \mu_{t+1|t}^{(k)}\right)}{\int p_h(z_{t+1} \mid s) \sum_{j=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(j)} \delta\left(s; \mu_{t+1|t}^{(j)}\right) ds} = \sum_{k=1}^{N_{t+1|t}} \left[\frac{\alpha_{t+1|t}^{(k)} p_h\left(z_{t+1} \mid \mu_{t+1|t}^{(k)}\right)}{\sum_{j=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(j)} p_h\left(z_{t+1} \mid \mu_{t+1|t}^{(j)}\right)}\right] \delta\left(x; \mu_{t+1|t}^{(k)}\right)$$

Particle Filter Resampling

- How do we approximate the prediction step?

$$p_{t+1|t}(x) = \sum_{k=1}^{N_{t|t}} \alpha_{t|t}^{(k)} p_a(x \mid \mu_{t|t}^{(k)}, u_t) \overset{??}{\approx} \sum_{k=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(k)} \delta\left(x; \mu_{t+1|t}^{(k)}\right)$$

- How do we avoid **particle depletion** - a situation in which most of the particle weights are close to zero?

- Just like the GMF uses refinement and pruning, the particle filter uses a procedure called **resampling** to:
  1. approximate the prediction step
  2. avoid particle depletion during the update step

- Resampling is applied at time t if the **effective number of particles**:

$$\boxed{N_{eff} := \frac{1}{\sum_{k=1}^{N_{t|t}} \left(\alpha_{t|t}^{(k)}\right)^2}}$$ is less than a threshold

Particle Filter Prediction

- How do we approximate the prediction step?

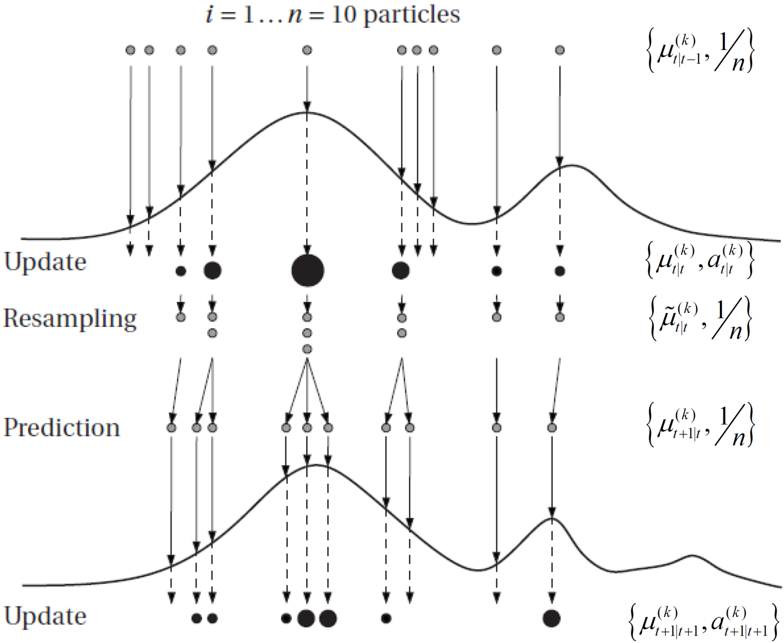
$$p_{t+1|t}(x) = \sum_{k=1}^{N_{t|t}} \alpha_{t|t}^{(k)} p_a(x \mid \mu_{t|t}^{(k)}, u_t) \overset{??}{\approx} \sum_{k=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(k)} \delta\left(x; \mu_{t+1|t}^{(k)}\right)$$

- Since $p_{t+1|t}(x)$ is a mixture pdf, we can approximate it with particles by drawing samples directly from it

- Let $N_{t+1|t}$ be the number of particles in the approximation (usually, $N_{t+1|t} = N_{t|t}$)

- **Bootstrap approximation**: repeat $N_{t+1|t}$ times and <u>normalize</u> the weights at the end:
  - Draw $j \in \{1, \ldots, N_{t|t}\}$ with probability $\alpha_{t|t}^{(j)}$
  - Draw $\mu_{t+1|t}^{(j)} \sim p_a\left(\cdot \mid \mu_{t|t}^{(j)}, u_t\right)$
  - Add the weighted sample $\left(\mu_{t+1|t}^{(j)}, p_{t+1|t}\left(\mu_{t+1|t}^{(j)}\right)\right)$ to the new particle set

Particle Filter

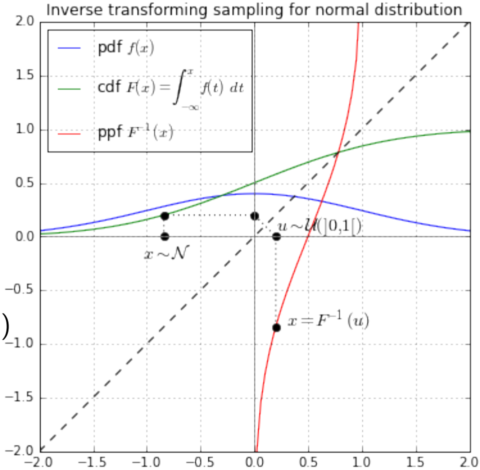


Inverse Transform Sampling

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

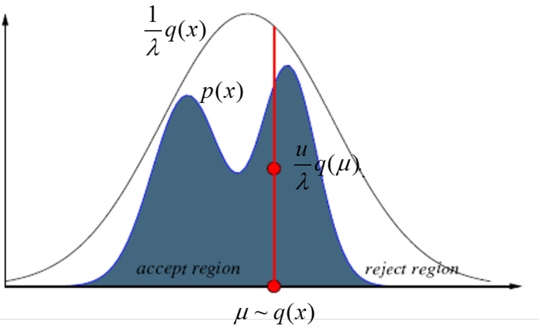
- **Inverse Transform Sampling**:
  1. Draw $u \sim \mathcal{U}(0, 1)$
  2. Return inverse CDF value: $\mu = F^{-1}(u)$
  3. The CDF of $F^{-1}(u)$ is:

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



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 $\mu^{(1)}, \ldots, \mu^{(N)} \sim q(\cdot)$
  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 $\{\mu^{(1)}, \ldots, \mu^{(N)}\}$ 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** methods (e.g., Metropolis-Hastings and Gibbs sampling):
  - 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

Stratified Resampling

- In the last step of SIR, the weighted sample set $\{\mu^{(k)}, \alpha^{(k)}\}$ is resampled 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

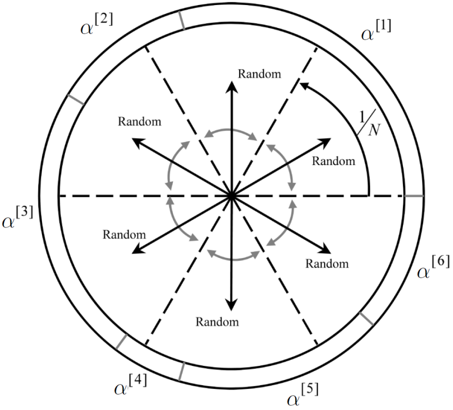
- **Stratified resampling**: guarantees that samples with large weights appear at least once and those with small weights – at most once. Stratified resampling is **optimal in terms of variance** (Thrun et al. 2005)

- 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 Resampling

Stratified (low variance) resampling

Input: particle set $\{\mu^{(k)}, \alpha^{(k)}\}_{k=1}^{N}$
Output: resampled particle set
$j \leftarrow 1$, $c \leftarrow \alpha^{(1)}$
for $k = 1, \ldots, N$ **do**
    $u \sim \mathcal{U}(0, \frac{1}{N})$
    $\beta = u + \frac{k-1}{N}$
    while $\beta > c$ **do**
        $j = j + 1$, $c = c + \alpha^{(j)}$
    add $(\mu^{(j)}, \frac{1}{N})$ to the new set



- **Systematic resampling**: the same as stratified resampling except that the **same** uniform is used for each piece, i.e., $u \sim \mathcal{U}(0, \frac{1}{N})$ is sampled only once before the for loop above.
- **Sample importance resampling (SIR)**: draw $\mu^{(k)}$ independently with replacement from $\{\mu^{(1)}, \ldots, \mu^{(N)}\}$ with probability $\alpha^{(k)}$ and add to the final sample set with weight $\frac{1}{N}$

Particle Filter Summary

- **Prior**: $x_t \mid z_{0:t}, u_{0:t-1} \sim p_{t|t}(x_x) := \sum_{k=1}^{N_{t|t}} \alpha_{t|t}^{(k)} \delta\left(x_t; \mu_{t|t}^{(k)}\right)$
- **Motion model**: $x_{t+1} \sim p_a(\cdot \mid x_t, u_t)$
- **Observation model**: $z_t \sim p_h(\cdot \mid x_t)$
- **Prediction**: approximate the mixture by sampling:

$$p_{t+1|t}(x) = \int p_a(x \mid s, u_t) \sum_{k=1}^{N_{t|t}} \alpha_{t|t}^{(k)} \delta\left(s; \mu_{t|t}^{(k)}\right) ds = \sum_{k=1}^{N_{t|t}} \alpha_{t|t}^{(k)} p_a(x \mid \mu_{t|t}^{(k)}, u_t) \approx \sum_{k=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(k)} \delta\left(x; \mu_{t+1|t}^{(k)}\right)$$

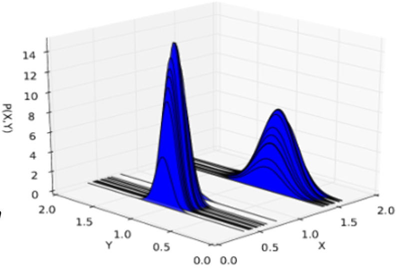
- **Update**: rescale the particles based on the observation likelihood:

$$p_{t+1|t+1}(x) = \frac{p_h(z_{t+1}; x) \sum_{k=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(k)} \delta\left(x; \mu_{t+1|t}^{(k)}\right)}{\int p_h(z_{t+1}; s) \sum_{j=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(j)} \delta\left(s; \mu_{t+1|t}^{(j)}\right) ds} = \sum_{k=1}^{N_{t+1|t}} \left[\frac{\alpha_{t+1|t}^{(k)} p_h\left(z_{t+1} \mid \mu_{t+1|t}^{(k)}\right)}{\sum_{j=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(j)} p_h\left(z_{t+1} \mid \mu_{t+1|t}^{(j)}\right)} \right] \delta\left(x; \mu_{t+1|t}^{(k)}\right)$$

- If $N_{eff} := \frac{1}{\sum_{k=1}^{N_{t|t}} \left(\alpha_{t|t}^{(k)}\right)^2} \leq N_{threshold}$, **resample** the particle set $\left\{\mu_{t+1|t+1}^{(k)}, \alpha_{t+1|t+1}^{(k)}\right\}$ via stratified or sample importance resampling

Rao-Blackwellized Particle Filter

- The Rao-Blackwellized (**marginalized**) particle filter is applicable to conditionally linear-Gaussian models:



$$x_{t+1}^n = f_t^n(x_t^n) + A_t^n(x_t^n)x_t^l + G_t^n(x_t^n)w_t^n$$
$$x_{t+1}^l = f_t^l(x_t^n) + A_t^l(x_t^n)x_t^l + G_t^l(x_t^n)w_t^l$$
$$z_t = h_t(x_t^n) + C_t(x_t^n)x_t^l + v_t$$

Nonlinear states: x_t^n
Linear states: x_t^l

- **Idea**: exploit linear-Gaussian sub-structure to handle high dim. problems

$$p\left(x_t^l, x_{0:t}^n \mid z_{0:t}, u_{0:t-1}\right) = \underbrace{p\left(x_t^l \mid z_{0:t}, u_{0:t-1}, x_{0:t}^n\right)}_{\text{Kalman Filter}} \underbrace{p\left(x_{0:t}^n \mid z_{0:t}, u_{0:t-1}\right)}_{\text{Particle Filter}}$$
$$= \sum_{k=1}^{N_{t|t}} \alpha_{t|t}^{(k)} \delta\left(x_{0:t}^n; m_{t|t}^{(k)}\right) \phi\left(x_t^l; \mu_{t|t}^{(k)}, \Sigma_{t|t}^{(k)}\right)$$

- The RBPF is a combination of the particle filter and the Kalman filter, in which each particle has a Kalman filter associated to it