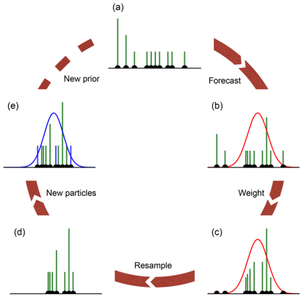
Autonomous Mobile Robotics

Particle Filter

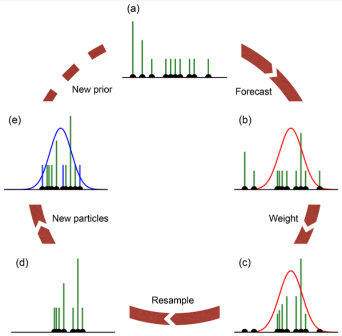
Geesara Kulathunga

March 6, 2023



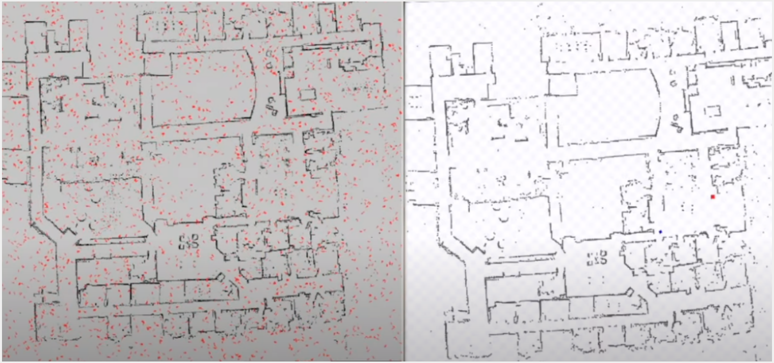
Particle Filter

- A Taxonomy of Particle Filter
- Bayesian Filter
- Monte Carlo Integration (MCI)
- Particle Filter
- Importance Sampling
- Particle Filter Algorithm



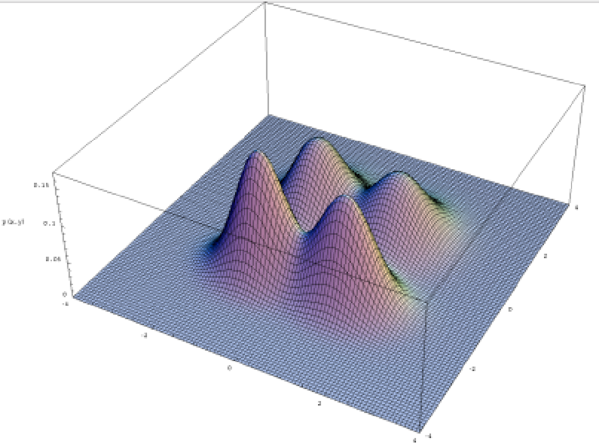
https://hess.copernicus.org/articles/23/1163/2019/

Particle Filter



https://www.youtube.com/watch?v=F6T3dtXviNY

Multi model distribution



https://en.wikipedia.org/wiki/Multimodal_distribution

Particle Filter

Why do we need a particle filter? Let's try to understand the common problems we face in this contest

- Do we need **more than one landmark for localization**? Are we interested in **tracking multiple objects simultaneously**?

Particle Filter

Why do we need a particle filter? Let's try to understand the common problems we face in this contest

- Do we need **more than one landmark for localization**? Are we interested in **tracking multiple objects simultaneously**?
- Are **objects** we are interested in **visible all the time** or have some occlusions?

Particle Filter

Why do we need a particle filter? Let's try to understand the common problems we face in this contest

- Do we need **more than one landmark for localization**? Are we interested in **tracking multiple objects simultaneously**?
- Are **objects** we are interested in **visible all the time** or have some occlusions?
- How do we address the **non-linear behaviour** of the system and measurements?

Particle Filter

Why do we need a particle filter? Let's try to understand the common problems we face in this contest

- Do we need **more than one landmark for localization**? Are we interested in **tracking multiple objects simultaneously**?
- Are **objects** we are interested in **visible all the time** or have some occlusions?
- How do we address the **non-linear behaviour** of the system and measurements?
- Have you ever considered **non-Gaussian noise**? Can you explain a bit more on this?

Why do we need a particle filter? Let's try to understand the common problems we face in this contest

- Do we need **more than one landmark for localization**? Are we interested in **tracking multiple objects simultaneously**?
- Are **objects** we are interested in **visible all the time** or have some occlusions?
- How do we address the **non-linear behaviour** of the system and measurements?
- Have you ever considered **non-Gaussian noise**? Can you explain a bit more on this?
- How many states are to be tracked? How do we define these states? Are they discrete or continuous?

Why do we need a particle filter? Let's try to understand the common problems we face in this contest

- Do we need **more than one landmark for localization**? Are we interested in **tracking multiple objects simultaneously**?
- Are **objects** we are interested in **visible all the time** or have some occlusions?
- How do we address the **non-linear behaviour** of the system and measurements?
- Have you ever considered **non-Gaussian noise**? Can you explain a bit more on this?
- How many states are to be tracked? How do we define these states? Are they discrete or continuous?
- When the process model is unknown, what can we do about that?

$$p(x_k|z_{1:k}, u_{0:k-1}) = \frac{p(z_k|x_k, z_1, ..., z_{k-1}, u_{0:k-1})p(x_k|z_1, ..., z_{k-1}, u_{0:k-1})}{p(z_k|z_{1:k-1}, u_{0:k-1})}$$
$$= \frac{p(z_k|x_k)p(x_k|z_{1:k-1}, u_{0:k-1})}{p(z_k|z_{1:k-1}, u_{0:k-1})} \quad (1)$$

- $p(x_k|z_{1:k}, u_{0:k-1})$ state probability distribution at time step k, updated with measurement data and control inputs
- $p(z_k|x_k, z_{1:k-1}, u_{0:k-1})$ measurement probability distribution
- $p(x_k|z_{k-1}, u_{0:k-1})$ predicted state probability distribution
- $p(z_k|z_{1:k-1}, u_{0:k-1})$ measurement probability distribution

- Prediction step

$$p(x_k|z_{1:k-1}, u_{1:k-1}) = \Sigma_{x_{k-1} \in X}\, p(x_k|x_{k-1}, u_{k-1}) p(x_{k-1}|z_{1:k-1}, u_{0:k-1}) \tag{2}$$

- Correction step

$$p(x_k|z_{1:k}, u_{0:k-1}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1}, u_{0:k-1})}{p(z_k|z_{1:k-1}, u_{0:k-1})} \tag{3}$$

, where

$$p(z_k|z_{1:k-1}, u_{0:k-1}) = \Sigma_{x_k \in X}\, p(z_k|x_k)p(x_k|z_{1:k-1}, u_{0:k-1})$$

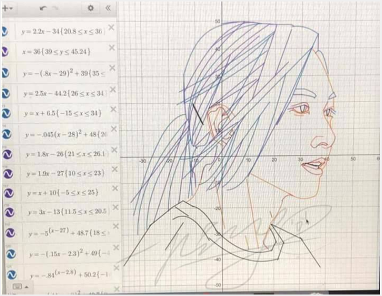
An explicit solution of the Bayesian filter for the continuous state-space variables

$$p(x_k|z_{1:k}, u_{0:k-1}) = \frac{p(z_k|x_k)}{p(z_k|z_{1:k-1}, u_{0:k-1})} \int p(x_k|x_{k-1}, u_{k-1})p(x_{k-1}|z_{1:k-1}, u_{0:k-2})dx_{k-1}$$

With the Markov assumption, the most probable state estimate

$$E\{\hat{x}_{k|k}\} = \int x_{k|k} \cdot p(x_k|z_{1:k}, u_{0:k-1})dx_{k|k}$$

However, this form can be applied only where data distribution is Gaussian and system has to be linear, the result is a Kalman filter. When the system is nonlinear result in EKF. Particle filter on the other had a more general case where noise does not need to be Gaussian.


$y = 2.2x - 34\{20.8 \le x \le 36$
$x = 36\{39 \le y \le 45.24\}$
$y = -(.8x - 29)^2 + 39\{35 \le$
$y = 2.5x - 44.2\{26 \le x \le 34$
$y = x + 6.5\{-15 \le x \le 34\}$
$y = -.045(x - 28)^2 + 48\{2$
$y = 1.8x - 26\{21 \le x \le 26.1$
$y = 1.9x - 27\{10 \le x \le 23\}$
$y = x + 10\{-5 \le x \le 25\}$
$y = 3x - 13\{11.5 \le x \le 20.5$
$y = -5^{(x-27)} + 48.7\{18 \le$
$y = -(.15x - 2.3)^2 + 49\{-$
$y = -.84^{(x-2.8)} + 50.2\{-1$

]

(a)

(b)

Monte Carlo Integration (MCI)

MCI is used to generate a finite number of random samples and pass through a system that we are interested in followed by computing the result on the transformed points. MCI is quite a powerful method for computing the value of complex integrals using probabilistic techniques

- Generating set of points from a chosen distribution, $X_i = U[a, b]$, $i = 1, \ldots, N$

Monte Carlo Integration (MCI)

MCI is used to generate a finite number of random samples and pass through a system that we are interested in followed by computing the result on the transformed points. MCI is quite a powerful method for computing the value of complex integrals using probabilistic techniques

- Generating set of points from a chosen distribution, $X_i = U[a, b]$, $i = 1, ..., N$
- Calculate $Y_i = g(X_i)$, $i = 1, ..., N$ where g(X) is a function which is used to generate a point, in other words, a function to be integrated.

Monte Carlo Integration (MCI)

MCI is used to generate a finite number of random samples and pass through a system that we are interested in followed by computing the result on the transformed points. MCI is quite a powerful method for computing the value of complex integrals using probabilistic techniques

- Generating set of points from a chosen distribution, $X_i = U[a, b]$, $i = 1, ..., N$
- Calculate $Y_i = g(X_i)$, $i = 1, ..., N$ where g(X) is a function which is used to generate a point, in other words, a function to be integrated.
- Calculate the estimation of g(x):

$$\bar{Y} = \frac{(b-a)}{N} \Sigma_{i=1}^{N} Y_i$$

Monte Carlo Integration (MCI)

MCI is used to generate a finite number of random samples and pass through a system that we are interested in followed by computing the result on the transformed points. MCI is quite a powerful method for computing the value of complex integrals using probabilistic techniques

- Generating set of points from a chosen distribution, $X_i = U[a, b]$, $i = 1, ..., N$
- Calculate $Y_i = g(X_i)$, $i = 1, ..., N$ where g(X) is a function which is used to generate a point, in other words, a function to be integrated.
- Calculate the estimation of g(x):

$$\bar{Y} = \frac{(b-a)}{N} \Sigma_{i=1}^{N} Y_i$$

- Use $g(x) = 4\sqrt{1-x^2}$, try to estimate $\bar{Y}$? what will happen when N is increasing?

Example 01

Can you estimate the value of π?

Example 01

Can you estimate the value of π? We can calculate the area of a circle and estimate the value of π
Consider circle radius is 1 and the area that contains such circle $A = 2 \cdot 2 = 4$

- Generate a set of points $x_i = U(-1, 1),$ for $i = 1, .., N$.

Example 01

Can you estimate the value of π? We can calculate the area of a circle and estimate the value of π
Consider circle radius is 1 and the area that contains such circle $A = 2 \cdot 2 = 4$

- Generate a set of points $x_i = U(-1, 1),\ \text{for } i = 1, .., N.$
- Check each x_i is inside a circle by verifying the distance, i.e., if its distance from the center of the circle is less than or equal to the radius

Example 01

Can you estimate the value of π? We can calculate the area of a circle and estimate the value of π
Consider circle radius is 1 and the area that contains such circle $A = 2 \cdot 2 = 4$

- Generate a set of points $x_i = U(-1, 1),\ \text{for } i = 1, .., N$.
- Check each x_i is inside a circle by verifying the distance, i.e., if its distance from the center of the circle is less than or equal to the radius
- Then we can estimate the value of π as follows:

$$\frac{\pi}{4} = \frac{\pi r^2}{4r^2} = \frac{\text{number of points inside the circle}}{\text{number of points inside the square}}$$

None of the filters we have learned so far work well with the following constraints. Comment on KF and EKF with respect to these constraints

The **main steps** of a **generic particle filter**

- Generate enough points for representing states that are interested in such as pose, etc.

None of the filters we have learned so far work well with the following constraints. Comment on KF and EKF with respect to these constraints
The **main steps** of a **generic particle filter**

- Generate enough points for representing states that are interested in such as pose, etc.
- Predict the next state of particles

None of the filters we have learned so far work well with the following constraints. Comment on KF and EKF with respect to these constraints

The **main steps** of a **generic particle filter**

- Generate enough points for representing states that are interested in such as pose, etc.
- Predict the next state of particles
- Updating weights of particles based on the measurements

None of the filters we have learned so far work well with the following constraints. Comment on KF and EKF with respect to these constraints

The **main steps** of a **generic particle filter**

- Generate enough points for representing states that are interested in such as pose, etc.
- Predict the next state of particles
- Updating weights of particles based on the measurements
- Resampling

Particle Filter

None of the filters we have learned so far work well with the following constraints. Comment on KF and EKF with respect to these constraints

The **main steps** of a **generic particle filter**

- Generate enough points for representing states that are interested in such as pose, etc.
- Predict the next state of particles
- Updating weights of particles based on the measurements
- Resampling
- Computing weighted mean and covariance of selected particles to estimate the posterior state

Particle Filter

- Initialize a set of N particles x_k^i random or some prior distribution $p(\mathbf{x}_0)$

- Initialize a set of N particles x_k^i random or some prior distribution $p(\mathbf{x}_0)$
  - If there is no clue about the location of the robot, uniform distribution can be used for generating N number of particles

$$\begin{aligned} p_x &= U(x_{min}, x_{max}, N) \\ p_y &= U(y_{min}, y_{max}, N) \\ p_\theta &= U(\theta_{min}, \theta_{max}, N), \end{aligned} \tag{4}$$

- Initialize a set of N particles x_k^i random or some prior distribution $p(\mathbf{x}_0)$
  - If there is no clue about the location of the robot, uniform distribution can be used for generating N number of particles

$$\begin{aligned} p_x &= U(x_{min}, x_{max}, N) \\ p_y &= U(y_{min}, y_{max}, N) \\ p_\theta &= U(\theta_{min}, \theta_{max}, N), \end{aligned} \tag{4}$$

  - If the initial belief $\mathbf{x}_0 = N(\mu_{\mathbf{x}}, \sigma_{\mathbf{x}})$ is known with uncertainty, Gaussian distribution can be used for generating N number of particles

$$\begin{aligned} p_x &= \mu_x + randn(N) \cdot \sigma_x \\ p_y &= \mu_y + randn(N) \cdot \sigma_y \\ p_\theta &= \mu_\theta + randn(N) \cdot \sigma_\theta, \end{aligned} \tag{5}$$

Particle Filter

- Prediction

- Prediction
  - Apply control input u_{k-1} on the state of each particle $\hat{x}^i_{k-1|k-1}$, with random noise using the motion model prediction $p(x_k|x_{k-1}, u_{k-1})$

- Prediction
  - Apply control input u_{k-1} on the state of each particle $\hat{x}^i_{k-1|k-1}$, with random noise using the motion model prediction $p(x_k|x_{k-1}, u_{k-1})$
  - The obtained predicted set of particles $\hat{x}^i_{k|k-1}$

- Prediction
  - Apply control input u_{k-1} on the state of each particle $\hat{x}^i_{k-1|k-1}$, with random noise using the motion model prediction $p(x_k|x_{k-1}, u_{k-1})$
  - The obtained predicted set of particles $\hat{x}^i_{k|k-1}$
- Correction

Particle Filter

- Prediction
  - Apply control input u_{k-1} on the state of each particle $\hat{x}^i_{k-1|k-1}$, with random noise using the motion model prediction $p(x_k|x_{k-1}, u_{k-1})$
  - The obtained predicted set of particles $\hat{x}^i_{k|k-1}$
- Correction
  - Estimate the measurement for each particles $\hat{x}^i_{k|k-1}$

Particle Filter

- Prediction
  - Apply control input u_{k-1} on the state of each particle $\hat{x}^i_{k-1|k-1}$, with random noise using the motion model prediction $p(x_k|x_{k-1}, u_{k-1})$
  - The obtained predicted set of particles $\hat{x}^i_{k|k-1}$
- Correction
  - Estimate the measurement for each particles $\hat{x}^i_{k|k-1}$
  - Evaluate the particle importance: difference between obtained measurement z_k and estimated particle measurements $\hat{z}^i_k$, i.e., $innov^i_k = z_k - \bar{z}^i_k$ (innovation or measurement residual)

Particle Filter

- Prediction
  - Apply control input u_{k-1} on the state of each particle $\hat{x}^i_{k-1|k-1}$, with random noise using the motion model prediction $p(x_k|x_{k-1}, u_{k-1})$
  - The obtained predicted set of particles $\hat{x}^i_{k|k-1}$
- Correction
  - Estimate the measurement for each particles $\hat{x}^i_{k|k-1}$
  - Evaluate the particle importance: difference between obtained measurement z_k and estimated particle measurements $\hat{z}^i_k$, i.e., $innov^i_k = z_k - \bar{z}^i_k$ (innovation or measurement residual)
  - Importance sampling: way to select importance particles $w^i_k = det(2\pi R)^{-1/2} e^{1/2(innov^i_k)^\top R^{-1}(innov^i_k)}$ to estimate the $p(z_k|\hat{x}^i_{k|k})$

Particle Filter

- Prediction
  - Apply control input u_{k-1} on the state of each particle $\hat{x}^i_{k-1|k-1}$, with random noise using the motion model prediction $p(x_k|x_{k-1}, u_{k-1})$
  - The obtained predicted set of particles $\hat{x}^i_{k|k-1}$
- Correction
  - Estimate the measurement for each particles $\hat{x}^i_{k|k-1}$
  - Evaluate the particle importance: difference between obtained measurement z_k and estimated particle measurements $\hat{z}^i_k$, i.e., $innov^i_k = z_k - \bar{z}^i_k$ (innovation or measurement residual)
  - Importance sampling: way to select importance particles $w^i_k = det(2\pi R)^{-1/2} e^{1/2(innov^i_k)^\top R^{-1}(innov^i_k)}$ to estimate the $p(z_k|\hat{x}^i_{k|k})$
  - Estimate $\hat{x}^i_{k|k}$ as the average value of the all the particles $\frac{1}{N}\sum_{i=1}^{N} w^i_k \hat{x}^i_{k|k-1}$

How can we draw a sample from a probability distribution that is unknown?

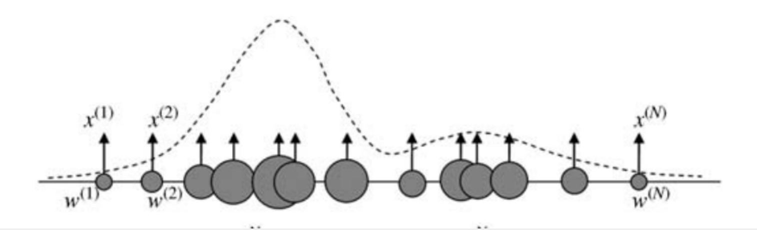


Figure: Importance of weighting the generated points

First, generate a set of samples from a known probability distribution, but weight the samples according to the distribution that is interested in

Importance Sampling

$$E(f(x)) = \int_{-\infty}^{\infty} f(x) \cdot g(x) = \int_{-\infty}^{\infty} x \cdot p(x)$$

- Let expected value of a function f(x) for the distribution g(x) be $E(f(x)) = \int f(x)g(x)dx$, where g(x) is the unknown distribution whose samples are required. However, $E(f(x))$ value can not be found without knowing g(x)

Importance Sampling

- Let expected value of a function f(x) for the distribution g(x) be $E(f(x)) = \int f(x)g(x)dx$, where g(x) is the unknown distribution whose samples are required. However, $E(f(x))$ value can not be found without knowing g(x)
- Let q(x) be a known chosen probability distribution, with that $E(f(x))$ can be reformulated as

$$E(f(x)) = \int f(x)g(x)\frac{q(x)}{q(x)}dx = \int f(x)q(x)\frac{g(x)}{q(x)}dx$$

- Let expected value of a function f(x) for the distribution g(x) be $E(f(x)) = \int f(x)g(x)dx$, where g(x) is the unknown distribution whose samples are required. However, $E(f(x))$ value can not be found without knowing g(x)
- Let q(x) be a known chosen probability distribution, with that $E(f(x))$ can be reformulated as

$$E(f(x)) = \int f(x)g(x)\frac{q(x)}{q(x)}dx = \int f(x)q(x)\frac{g(x)}{q(x)}dx$$

- Use MCI to get approximate solution to $\int f(x)q(x)dx$ and the unknown term $\frac{g(x)}{q(x)}$ is defined as the weighting factor $w(x)$

$$E(f(x)) = \Sigma_{i=1}^{N} f(x^i)w(x^i)$$

- Importance sampling: way to select importance particles

$$w_k^i = det(2\pi R)^{-1/2} e^{1/2(innov_k^i)^\top R^{-1}(innov_k^i)}$$

- Importance sampling: way to select importance particles

$$w_k^i = det(2\pi R)^{-1/2} e^{1/2(innov_k^i)^\top R^{-1}(innov_k^i)}$$

- Normalized the particle weights

$$wn_k^i = \frac{w_k^i}{\sum_{j=1}^{N} w_k}$$

- Importance sampling: way to select importance particles

$$w_k^i = det(2\pi R)^{-1/2} e^{1/2(innov_k^i)^\top R^{-1}(innov_k^i)}$$

- Normalized the particle weights

$$wn_k^i = \frac{w_k^i}{\Sigma_{j=1}^N w_k}$$

- For the aforementioned example, estimated value would be $\hat{x}_{k|k}^i = \frac{1}{N}\Sigma_{i=1}^N wn_k^i \hat{x}_{k|k-1}^i$

Resampling

How to select a optimal set of particles?

- should preferentially select particles that have a higher probability

How to select a optimal set of particles?

- should preferentially select particles that have a higher probability
- should include enough lower probability particles to give the filter a chance of detecting strongly nonlinear behaviour

Resampling

How to select a optimal set of particles?

- should preferentially select particles that have a higher probability
- should include enough lower probability particles to give the filter a chance of detecting strongly nonlinear behaviour
- Several ways to do resampling, including multinomial resampling, residual resampling, stratified resampling, and systematic resampling

Resampling

How to select a optimal set of particles?

- should preferentially select particles that have a higher probability
- should include enough lower probability particles to give the filter a chance of detecting strongly nonlinear behaviour
- Several ways to do resampling, including multinomial resampling, residual resampling, stratified resampling, and systematic resampling
- Multinomial resampling

Resampling

How to select a optimal set of particles?

- should preferentially select particles that have a higher probability
- should include enough lower probability particles to give the filter a chance of detecting strongly nonlinear behaviour
- Several ways to do resampling, including multinomial resampling, residual resampling, stratified resampling, and systematic resampling
- Multinomial resampling
  - cumulative sum of normalized weights $wc_k^i = \Sigma_{j=1}^{i} wn_k^i$

Resampling

How to select a optimal set of particles?

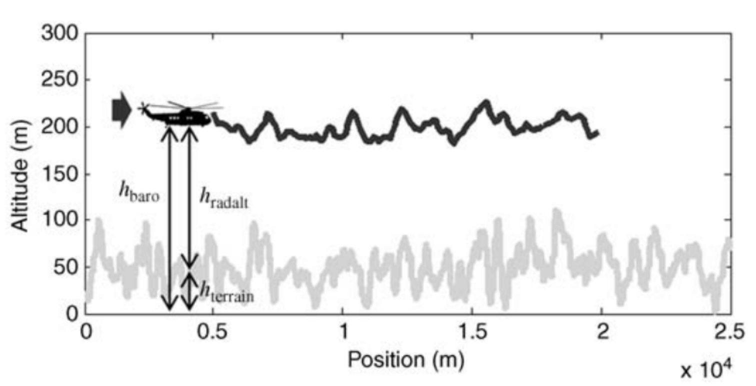
- should preferentially select particles that have a higher probability
- should include enough lower probability particles to give the filter a chance of detecting strongly nonlinear behaviour
- Several ways to do resampling, including multinomial resampling, residual resampling, stratified resampling, and systematic resampling
- Multinomial resampling
  - cumulative sum of normalized weights $wc_k^i = \Sigma_{j=1}^{i} wn_k^i$
  - randomly pick some of the weights and corresponding samples

Resampling

How to select a optimal set of particles?

- should preferentially select particles that have a higher probability
- should include enough lower probability particles to give the filter a chance of detecting strongly nonlinear behaviour
- Several ways to do resampling, including multinomial resampling, residual resampling, stratified resampling, and systematic resampling
- Multinomial resampling
  - cumulative sum of normalized weights $wc_k^i = \Sigma_{j=1}^{i} wn_k^i$
  - randomly pick some of the weights and corresponding samples
  - after selecting particles, get the average as the posterior $\hat{x}_{k|k} = \frac{1}{N}\Sigma_{i=1}^{N} wn_k^i \hat{x}_{k|k-1}^i$

Example 02



300
250
200
150
100
50
0
Altitude (m)
h_{baro}
h_{radalt}
$h_{terrain}$
0
0.5
1
1.5
2
2.5
Position (m)
x 10^4

Example 02

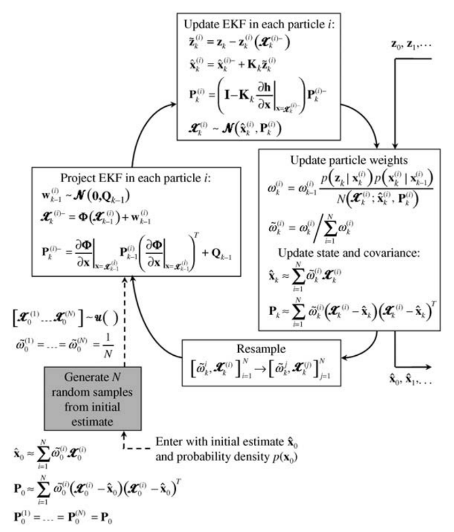
Let's define the process model and measurement model as follow,

$$x_{k+1} = x_k + u_k + w_k \tag{6}$$

$$z_k = h(x_k) + v_k \tag{7}$$

while assuming the process model is linear and the measurement model is non-linear with addictive measurement noise.,Eqs. 7 can be rewritten as

$$z_k = h(x_k) + v_k = h_{radalt} - h_{baro} = h_{terrain}(x_k) + v_k \tag{8}$$


Update EKF in each particle i:
$\tilde{\mathbf{z}}_k^{(i)} = \mathbf{z}_k - \mathbf{z}_k^{(i)}(\boldsymbol{\mathcal{X}}_k^{(i)-})$
$\hat{\mathbf{x}}_k^{(i)} = \hat{\mathbf{x}}_k^{(i)-} + \mathbf{K}_k \tilde{\mathbf{z}}_k^{(i)}$
$\mathbf{P}_k^{(i)} = \left(\mathbf{I} - \mathbf{K}_k \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\boldsymbol{\mathcal{X}}_k^{(i)-}} \right) \mathbf{P}_k^{(i)-}$
$\boldsymbol{\mathcal{X}}_k^{(i)} \sim \boldsymbol{\mathcal{N}}(\hat{\mathbf{x}}_k^{(i)}, \mathbf{P}_k^{(i)})$
$\mathbf{z}_0, \mathbf{z}_1, \ldots$
Update particle weights
$\omega_k^{(i)} = \omega_{k-1}^{(i)} \frac{p(\mathbf{z}_k \mid \mathbf{x}_k^{(i)}) p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)})}{N(\boldsymbol{\mathcal{X}}_k^{(i)}; \hat{\mathbf{x}}_k^{(i)}, \mathbf{P}_k^{(i)})}$
$\tilde{\omega}_k^{(i)} = \omega_k^{(i)} \Big/ \sum_{i=1}^{N} \omega_k^{(i)}$
Update state and covariance:
$\hat{\mathbf{x}}_k \approx \sum_{i=1}^{N} \tilde{\omega}_k^{(i)} \boldsymbol{\mathcal{X}}_k^{(i)}$
$\mathbf{P}_k \approx \sum_{i=1}^{N} \tilde{\omega}_k^{(i)} (\boldsymbol{\mathcal{X}}_k^{(i)} - \hat{\mathbf{x}}_k)(\boldsymbol{\mathcal{X}}_k^{(i)} - \hat{\mathbf{x}}_k)^T$
Project EKF in each particle i:
$\mathbf{w}_{k-1}^{(i)} \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{Q}_{k-1})$
$\boldsymbol{\mathcal{X}}_k^{(i)-} = \boldsymbol{\Phi}(\boldsymbol{\mathcal{X}}_{k-1}^{(i)}) + \mathbf{w}_{k-1}^{(i)}$
$\mathbf{P}_k^{(i)-} = \frac{\partial \boldsymbol{\Phi}}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\boldsymbol{\mathcal{X}}_{k-1}^{(i)}} \mathbf{P}_{k-1}^{(i)} \left(\frac{\partial \boldsymbol{\Phi}}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\boldsymbol{\mathcal{X}}_{k-1}^{(i)}} \right)^T + \mathbf{Q}_{k-1}$
$[\boldsymbol{\mathcal{X}}_0^{(1)} \ldots \boldsymbol{\mathcal{X}}_0^{(N)}] \sim \boldsymbol{u}(\,)$
$\tilde{\omega}_0^{(1)} = \ldots = \tilde{\omega}_0^{(N)} = \frac{1}{N}$
Resample
$[\tilde{\omega}_k^i, \boldsymbol{\mathcal{X}}_k^{(i)}]_{i=1}^N \rightarrow [\tilde{\omega}_k^j, \boldsymbol{\mathcal{X}}_k^{(j)}]_{j=1}^N$
$\hat{\mathbf{x}}_0, \hat{\mathbf{x}}_1, \ldots$
Generate N random samples from initial estimate
Enter with initial estimate $\hat{\mathbf{x}}_0$ and probability density $p(\mathbf{x}_0)$
$\hat{\mathbf{x}}_0 \approx \sum_{i=1}^{N} \tilde{\omega}_0^{(i)} \boldsymbol{\mathcal{X}}_0^{(i)}$
$\mathbf{P}_0 \approx \sum_{i=1}^{N} \tilde{\omega}_0^{(i)} (\boldsymbol{\mathcal{X}}_0^{(i)} - \hat{\mathbf{x}}_0)(\boldsymbol{\mathcal{X}}_0^{(i)} - \hat{\mathbf{x}}_0)^T$
$\mathbf{P}_0^{(1)} = \ldots = \mathbf{P}_0^{(N)} = \mathbf{P}_0$

Example 02

$p(z_k|x_k^i), p(x_k^i|x_{k-1}^i)$ and $N(x_k^i; \bar{x_k}^i, P_k^i)$ are defined as follow,

$$p(z_k|x_k^i) \propto e^{\frac{-1}{2}(z_k - H_k \bar{x_k}^-)R^{-1}(z_k - H_k \bar{x_k}^-)^t} \tag{9}$$

$$p(x_k|x_{k-1}^i) \propto e^{\frac{-1}{2}(x_k - \Phi_k(x_{k-1}^-))(P_k^{(i)-})^{-1}(x_k - \Phi_k(x_{k-1}^-))^t} \tag{10}$$

$$N(x_k^i; \bar{x_k}^i, P_k^i) \propto e^{\frac{-1}{2}(x_k^i - \bar{x_k}^-)(P_k^{(i)-})^{-1}(x_k^i - \bar{x_k}^-)^t} \tag{11}$$

Example 02

1. Let's say the initial position of the robot is somewhere in between 3000m and 7000m. What can you say about the initial distribution?
2. What would you say about the initial distribution if we know the initial position approximately, i.e., 5100$\pm$10 m?

References

Robert Grover Brown, Patrick YC Hwang, et al.
Introduction to random signals and applied Kalman filtering, volume 3.
Wiley New York, 1992.

Gregor Klancar, Andrej Zdesar, Saso Blazic, and Igor Skrjanc.
Wheeled mobile robotics: from fundamentals towards autonomous systems.
Butterworth-Heinemann, 2017.

Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza.
Introduction to autonomous mobile robots.
MIT press, 2011.

Sebastian Thrun.
Probabilistic robotics.
Communications of the ACM, 45(3):52–57, 2002.