1 Particle Filter

Gaussian filtering is a useful technique for performing nonlinear filtering. However, Gaussian filtering methods do not perform well when the models are highly nonlinear or when the posterior distribution is significantly non-Gaussian (e.g. a multimodal density). For such problems we need a different type of approximation to the posterior density.

Idea
Use a non-parametric representation:
$p(x_k|y_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \delta(x_k - x_k^{(i)})$ where $x_k^{(i)}$ are particles and $w_k^{(i)}$ are associated weights.
Filtering is essentially performed by propagating $x_{k-1}^{(i)} \to x_k^{(i)}$ over time and then updating the weights $w_k^{(i)}$.
Basic version: $x_k^{(i)} \sim p(x_k|x_{k-1}^{(i)}), w_k^{(i)} \propto w_{k-1}^{(i)} p(y_k|x_k^{(i)})$

Monte Carlo Approximations
Given independent samples $x^{(1)}, x^{(2)}, \dots x^{(N)} \sim p(x)$, we can approximate:
$E[g(x)] \approx \frac{1}{N}\sum_{i=1}^{N} g(x^{(i)})$
$p(x) \approx \frac{1}{N}\sum_{i=1}^{N} \delta(x - x^{(i)})$

Remarks
Non-parametric approximation to $p(x)$
Approximate all kinds of densities (very flexible)
Does not suffer from the curse of dimensionality, e.g. $\text{Cov}(\frac{1}{N}\sum_{i=1}^{N} x^{(i)}) = \frac{1}{N}\text{Cov}(x)$, independently on $\dim(x)$
Weakness: it is often difficult to generate samples from $p(x)$

Importance Sampling
Generate samples $x^{(1)}, x^{(2)}, \dots x^{(N)}$ from a proposal density $q(x)$:

$$E_{p(x)}[g(x)] = \int g(x)p(x)\,dx = \int \underbrace{g(x)\frac{p(x)}{q(x)}}_{\tilde{g}(x)} q(x)\,dx \approx \frac{1}{N}\sum_{i=1}^{N} g(x^{(i)})\frac{p(x^{(i)})}{q(x^{(i)})} \propto \sum_{i=1}^{N} \underbrace{\frac{p(x^{(i)})}{q(x^{(i)})}}_{\tilde{w}^{(i)}} g(x^{(i)})$$

$p(x) \approx \sum_{i=1}^{N} w^{(i)} \delta(x - x^{(i)})$
$w^{(i)} = \dfrac{\tilde{w}^{(i)}}{\sum_{n=1}^{N} \tilde{w}^{(n)}}$ and $\tilde{w}^{(i)} = \dfrac{p(x^{(i)})}{q(x^{(i)})}$

It can perform very well as long as it is easy to sample from $q(x)$, the support of $q(x)$ contains the support of $p(x)$, and $q(x)$ is similar to $p(x)$

Sequential Importance Sampling
Objective: to recursively and accurately approximate the filtering density $p(x_k|y_{1:k})$
Assumption: both the motion and measurement models, $p(x_k|x_{k-1})$ and $p(y_k|x_k)$, can be easily evaluated point-wise
A common example is:
$x_k = f(x_{k-1}) + q_{k-1} \qquad q_{k-1} \sim \mathcal{N}(0, Q_{k-1})$
$y_k = h(x_k) + r_k \qquad r_k \sim \mathcal{N}(0, R_k)$

Particle filters are also known as sequential importance resampling or sequential Monte Carlo. The basis of these methods is an algorithm called sequential importance sampling (SIS)

SIS
For $i = 1, \dots, N$ and at each time k:
Draw $x_k^{(i)} \sim q(x_k | x_{k-1}^{(i)}, y_k)$
Compute weights $w_k^{(i)} \propto w_{k-1}^{(i)} \dfrac{p(y_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})}{q(x_k^{(i)}|x_{k-1}^{(i)}, y_k)}$
Normalize the weights
Approximate $p(x_k|y_{1:k}) \approx \sum_{i=1}^N w_k^{(i)} \delta(x_k - x_k^{(i)})$

Derivation
At each time k:
Draw particles $x_{0:k}^{(i)} \sim q(x_{0:k}|y_{1:k})$
Update weights $w_k^{(i)} \propto \dfrac{p(x_{0:k}^{(i)}|y_{1:k})}{q(x_{0:k}^{(i)}|y_{1:k})}$
Assume that $q(x_{0:k}|y_{1:k}) = q(x_k|x_{k-1}, y_k)q(x_{0:k-1}|y_{1:k-1})$ and we generate $x_{0:k-1}^{(i)} \sim q(x_{0:k-1}|y_{1:k-1})$ at time $k-1$
It is sufficient to generate $x_k^{(i)} \sim q(x_k|x_{k-1}^{(i)}, y_k)$ and append that to $x_{1:k-1}^{(i)}$

$$w_k^{(i)} \propto \frac{p(x_{0:k}^{(i)}|y_{1:k})}{q(x_{0:k}^{(i)}|y_{1:k})} \propto p(x_{0:k-1}^{(i)}, x_k^{(i)}, y_k|y_{1:k-1}) \propto \frac{p(y_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})}{q(x_k^{(i)}|x_{k-1}^{(i)}, y_k)} \cdot \underbrace{\frac{p(x_{0:k-1}^{(i)}|y_{1:k-1})}{q(x_{0:k-1}^{(i)}|y_{1:k-1})}}_{w_{k-1}^{(i)}} \propto w_{k-1}^{(i)} \frac{p(y_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})}{q(x_k^{(i)}|x_{k-1}^{(i)}, y_k)}$$

A simple choice of importance density is:
$q(x_k|x_{k-1}, y_k) = p(x_k|x_{k-1})$ for which $w_k^{(i)} \propto w_{k-1}^{(i)} p(y_k|x_k^{(i)})$

One can show that all SIS filters suffer from degeneracy: after a few time steps all but one particle will have negligible weight

Consequences of degeneracy: the filter believes that it knows the true state exactly, we obtain very poor state estimates, and most of our calculations are wasted on insignificant particles

Sequential Importance Resampling (SIR)
Challenge: we have $p(x_k|y_{1:k}) \approx \sum_{i=1}^N w_k^{(i)} \delta(x_k - x_k^{(i)})$ where most weights are very small
Generate independent samples $\tilde{x}_k^{(i)}, \dots, \tilde{x}_k^{(N)}$ from $p(x_k|y_{1:k})$ and set $p(x_k|y_{1:k}) \approx \sum_{i=1}^N \frac{1}{N} \delta(x_k - \tilde{x}_k^{(i)})$
After resampling we get equal weights ($1/N$) and multiple copies of high probability particles

Resampling Algorithm
Draw N samples with replacement where the probability of selecting $x_k^{(i)}$ is $w_k^{(i)}$
Replace the old sample set with the new one and set all weights to $1/N$
We can use samples from the uniform distribution to draw samples from the discrete distribution $p(x_k|y_{1:k})$

Resampling costs some calculations and introduces some errors, but improves performance immensely over time. An estimate for the effective number of particles is $N_{\text{eff}} = \dfrac{1}{\sum_{i=1}^N (w_k^{(i)})^2}$
Many algorithms only resample when N_{eff} is below some threshold, e.g. $N/4$

Choice of Importance Distribution
A carefully selected importance distribution $q(x_k|x_{k-1}, y_k)$, can slow down the degeneracy and improve performance

Intuition: if most particles are placed in high probability regions, there is less need to get rid of useless particles.
Optimal importance density: $q(x_k|x_{k-1}, y_k) = p(x_k|x_{k-1}, y_k)$
Unfortunately, in most nonlinear settings, $p(x_k|x_{k-1}, y_k)$, is difficult to both draw samples from and to evaluate

The most common choice is still $q(x_k|x_{k-1}, y_k) = p(x_k|x_{k-1})$, and the bootstrap algorithm.

Bootstrap PF
At each time k: Draw $x_k^{(i)} \sim p(x_k|x_{k-1}^{(i)})$ for $i = 1, \dots, N$
Calculate $w_k^{(i)} \propto w_{k-1}^{(i)} p(y_k|x_k^{(i)})$ and normalize to 1
Resample

Note: if we sample at every time step, we get $w_k^{(i)} \propto p(y_k|x_k^{(i)})$ since $w_{k-1}^{(i)} = 1/N$ after resampling
Note: the Auxiliary PF (APF) is a variation of the SIR algorithm that makes use of y_k

Particle filters can handle highly nonlinear and non-Gaussian systems. Particle filters are asymptotically exact as you increase N. The complexity is roughly $O(N)$ but the gain in performance flattens out as you increase N. Unfortunately, PFs suffer from the curse of dimensionality and are intractable in higher dimensions.

The output from a PF is an approximation $p(x_k|y_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \delta(x_k - x_k^{(i)})$ which implies that $E[g(x_k)|y_{1:k}] \approx \sum_{i=1}^{N} w_k^{(i)} g(x_k^{(i)})$

Rao-Blackwellized PF
Background: particle filters are intractable in high dimensions, many systems are linear in some dimensions
Idea 1: combine a particle filter for nonlinear states with a Kalman filter for the linear states
Idea 2: if $x_k = \begin{bmatrix} x_k^l & u_k \end{bmatrix}^T$ where x_k^l and u_k are the linear and nonlinear states:
$p(x_k^l, u_{1:k}|y_{1:k}) = \underbrace{p(x_k^l|u_{0:k}, y_{1:k})}_{\text{Gaussian}} \underbrace{p(u_{0:k}|y_{1:k})}_{\text{PF}}$

Rao-Blackwellized PFs are often used for models of the form:
$x_k^l = f_{k-1}^l(u_{k-1}) + A_{k-1}^l(u_{k-1}) x_{k-1}^l + q_{k-1}^l$
$u_k = f_{k-1}^U(u_{k-1}) + A_{k-1}^U(u_{k-1}) x_{k-1}^l + q_{k-1}^U$
$y_k = h_k(u_k) + H_k(u_k) x_k^l + r_k$
where all the noises are Gaussian

Single recursion of the Rao-Blackwellized PF
$p(x_{k-1}^l|u_{0:k-1}, y_{1:k-1}) p(u_{0:k-1}|y_{1:k-1}) \longrightarrow p(x_k^l|u_{0:k}, y_{1:k}) p(u_{0:k}|y_{1:k})$
1. PF-pred: $p(u_{1:k-1}|y_{1:k-1}) \longrightarrow p(u_{1:k}|y_{1:k-1})$
2. KF, dyn. update: $p(x_{k-1}^l|u_{1:k-1}, y_{1:k-1}) \longrightarrow p(x_{k-1}^l|u_{1:k}, y_{1:k-1})$
3. KF-pred: $p(x_{k-1}^l|u_{1:k-1}, y_{1:k-1}) \longrightarrow p(x_k^l|u_{1:k-1}, y_{1:k-1})$
4. PF update: $p(u_{1:k}|y_{1:k-1}) \longrightarrow p(u_{1:k}|y_{1:k})$
5. KF, meas. update: $p(x_k^l|u_{1:k}, y_{1:k-1}) \longrightarrow p(x_k^l|u_{1:k}, y_{1:k})$

Step 2 makes use of the motion model for u_k to update x_{k-1}^l
The linear states are marginalized from step 1 and 4, similarly to how we normally handle noise.

Rao-Blackwellized particle filters are useful to reduce the number of particles. These filters enable us to handle higher dimensions than normal PFs. They are particularly useful if Kalman gains and posterior covariances are independent of the nonlinear states. Sufficient to compute them one time in each recursion.