Particle Filter: a Sequential Monte Carlo Filtering method for Dynamic State-space models

Mathilda Nguyen

December 5, 2020

Abstract

In this paper, I tried to introduce the particle filter, it's varieties, their pros and cons, as well as some methods on how to overcome their drawbacks. The second to last section would discuss some python programming packages one can employ to code up a particle filter.

1 Introduction

1.1 Blurb on Particle Filter

Predictive filtering algorithms arise from the need of inferring the internal state of a system and predicting its next state when only partial observations are obtained. Kalman Filter is a widely used filtering method, however, unlike Kalman which is mostly used for linear systems with Gaussian noise, Particle Filter thrives with nonlinear systems whose observations are complicated and/or might get extremely corrupted.

1.2 Background and motivation

I came to learn about particle filter from my Virtually Integrated Project (VIP) research position for Drexel Wireless Systems Lab(DWSL). We need an algorithm like Particle Filter, which can be used to predict when certain things can happens.

1.3 Some usage

Tracking: robot localization (widely applied), simultaneous localization and mapping (SLAM), object tracking

Prediction: Economic data forecasting, predicting the stock market, predicting the weather, predicting the future

2 Explain to me like I'm 5

Here is the situation:
1. We want to know some distribution "a"
2. We can measure "b"
3. We know something about the relationship between "b" and "a"

"a" in this case is the probability density function (PDF) and "b" is what obtained from measurements (such as sensor data). As an example, take Figure 1 below as our PDF, we are trying to estimate the area under the curve.

Step 1 (Prediction): Now conjure up a bunch of points (the particles) and put them in random places in the x-axis of the plane, each of these points is a possibility of where the system can be.
Step 2 (Update): Update weight by calculation
Step 3 (Resample): Change current particles population by flushing out some and multiplying others with respect to the weights

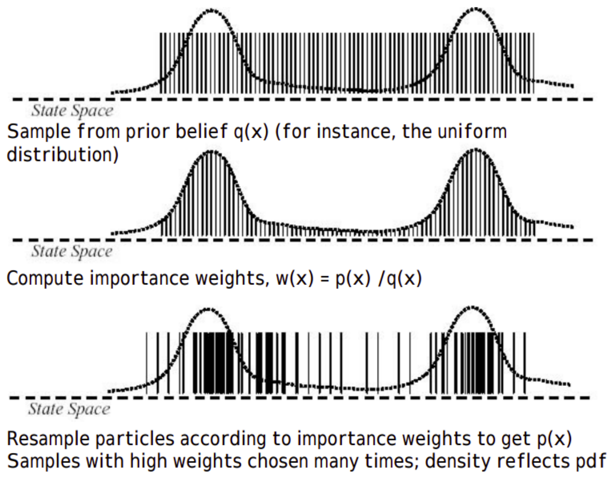
State Space
Sample from prior belief q(x) (for instance, the uniform distribution)
State Space
Compute importance weights, w(x) = p(x) /q(x)
State Space
Resample particles according to importance weights to get p(x)
Samples with high weights chosen many times; density reflects pdf

Figure 1: $p(x)$ is posterior; $q(x)$ is prior belief

Step 4 (Propagate): Uniformly move every particles forward according to a model

Repeating these steps, over time, we expect the particles' concentration (and their weights) to reflect the density of the state.

3 Formalization

The key idea of the Particle Filter is to approximate the target posterior conditional probability density function (PDF) of a system using particles that are recursively generated and filtered as more information becomes available (I wish someone has told me this line before I waded through all the different sources in confusion). This PDF is $P(X_t|Y_{1:t})$. Once obtained, any characteristic of the state, such as the mean, median, credible intervals, kurtosis, and so on, can be easily calculated.

3.1 State-space formulations

Particle Filter is a Sequential Monte Carlo (SMC) method, and is designed for hidden Markov Model.

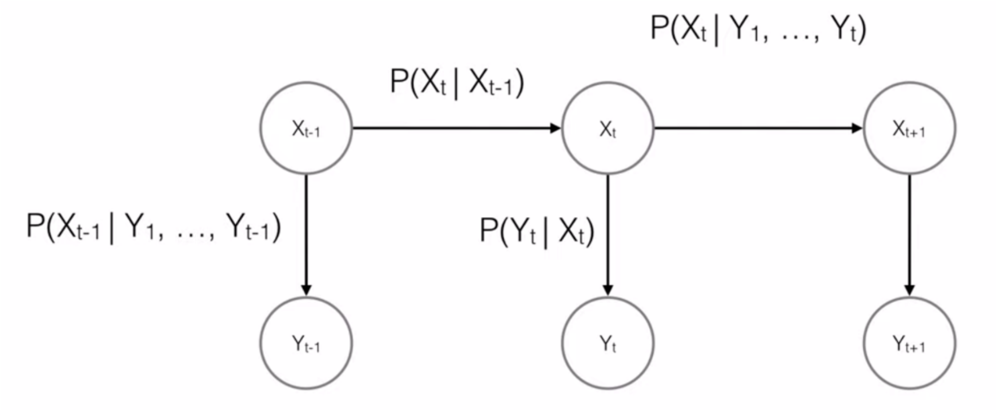

Figure 2: X is the latent (true) state of the system, and Y is the observed/evidence. At any time index, the latent state X can only be accessed through noisy observation Y that is available

Below is a break down of the terms:
$P(X_{t-1}|Y_1, ..., Y_{t-1}) = P(X_{t-1}|Y_{1:t-1})$: Filtering dty. Latent state at time t depends on all the observations that have happened
$P(X_t|X_{t-1})$: Transition dty. Current latent state depends on the state before it

$P(Y_t|X_t)$: Likelihood dty. Observed state at time t depends on the latent state at time t

These are the necessary building blocks to obtain equations needed to do all the aforementioned steps.

3.2 General prediction expression

To do prediction step, we need $P(X_t|Y_{1:t-1})$. Using the Markovian property of the states, we got:

$$P(X_t|Y_{1:t-1}) = \int P(X_t|X_{t-1})P(x_{t-1}|Y_{1:t-1})dX_{t-1} \tag{3.2.1}$$

3.3 General Update/Filtering expression

Once new observation Y_t arrives, we need to update the model. We do it by:

$$P(X_t|Y_{1:t}) \propto P(Y_t|X_t)P(X_t|Y_{1:t-1}) \tag{3.3.1}$$

Plugging (3.2.1) into (3.3.1), we have the recursive resample-update-prediction expression:

$$P(X_t|Y_{1:t}) \propto P(Y_t|X_t)P(X_t|Y_{1:t-1}) = P(Y_t|X_t)\int P(X_t|X_{t-1})P(x_{t-1}|Y_{1:t-1})dX_{t-1} \tag{3.3.2}$$

3.4 Resample

The resampling step is the heart of particle filter. In fact, it is so important that we are going to dedicate a whole section for it.

4 Resamplings and Particle Filter variants

I hope that by now, you'd have an idea of why we need to resample the particle. If not, the resampling process is essentially there so that we can properly adjust the PDF. More specifically, if you keep your old particles around forever, they would move according to transition probabilities, unaffected by your observations (other than their weights). Unlikely particles will only becomes more unlikely, which means that in the end, we can only have one particle in the area of high probability of our posterior. This is what we called "particle depletion".

Needless to say, one particle doesn't represent PDFs very well.

I want to re-state again that the goal of particle filter is approximating the PDF of a system based on the density of particles, with high-probability area having more and low-probability having less particles. By resampling, we can keep high-possible areas high in particle density.

To do this step, in this paper, we relies on the importance sampling (IS) principle.

4.1 Importance sampling (IS)

The IS principle allows us to sample from a *proposal (or imporance) distribution,* $q(z)$ which is an alternative PDF as close as possible to the *true PDF,* $p(z)$ since the true PDF is seldom available.

Particle filter aim to approximate $p(x_t|y_{1:t})$ (the filtering PDF). We can do it by drawing particles from the PDF $q(x_t|y_{1:t})$, by

$$P(x_t|y_{1:t}) \approx \sum_{j=1}^{M} \widetilde{\omega}_t^{(j)}\delta(x_t - x_t^{(j)}) \tag{4.1.1}$$

where
$\widetilde{\omega}_t^{(j)}$: importance weights

$\omega_t^{(j)}$: the corresponding normalized weights

$\delta(.)$: the Dirac function

As the sample size $M \longrightarrow \infty$, the approximation (3.4.1) converges to gives us $p(x_t|y_{1:t})$ - the true filtering PDF.

However, we need to recursively approximate p. A recursive version of SIS is the SIS filter.

4.2 The Bayesian Sequential Importance Sampling (SIS) Filter

Drawbacks: SIS approach always fail when the time-horizon t is more than a few tens.

Efforts to combats certain drawbacks from particle filters as a whole gave birth to various particle filtering algorithms, all of which are based on SIS.

4.3 Sequential Importance Sampling with Resampling (SISR)

The first attempt to overcome the degeneracy drawback of SIS filter is by adding a resampling step to the SIS filter. How this resampling step goes is discarding highly improbably particles and multiplying highly probable ones in a way that the total number of particles remain the same (however, when needed, one might resample a greater amount of particles). With this additional resampling step, our filter achieve long-term stability.

This gives rise to the first operational algorithm within the Particle Filtering methodology, named the sequential importance sampling with resampling particle filter variant (SISR PF)

4.4 Sampling Importance Resampling (SIR) Particle Filter

Easy to implement. Most sources called this the "Bootstrap filter". However, some called SISR that.

Drawbacks: resampling must be applied with caution because it may lead to sample impoverishment/attrition (i.e loss of diversity due to frequent sampling, which increases the variance of current estimate, which ultimately reduced accuracy).

There are a lot of research paper written specifically to address various resampling methods to tackle this drawbacks available online. Some popular strategies are: Effective sample size (ESS), stratified resampling, residual resampling,...etc...

4.5 Auxiliary Particle Filter (ASIR)

ASIR simulates from particles that have high likelihood. The second resampling step could be optional. Developed to deal with SIR's tailed deficiencies.

4.6 Extended Particle Filter (EPF)

Used for nonlinear online estimation. A Gaussian EKF (Extended Kalman Filter) approximation is used as proposal distribution for a particle filter.

4.7 Unscented Particle Filter (UPF)

A novel method used for nonlinear, non-Gaussian, online estimation problems. Uses the normal distribution obtained via the UKF as a proposal PDF. As of currently, it seemed like the most superior Particle Filter variance.

5 Programming resources and side notes

Firstly, regarding programming resources, I think that *particles* (https://github.com/nchopin/particles) is the best (and most extensive) python package for particle filters. The author programmed the package according to a book he wrote titled "An introduction to Sequential Monte Carlo".

Aside from that, I find this note book particularly helpful in understanding and programming a particle filter: https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python/blob/master/12-Particle-Filters.ipynb

Secondly, particle filtering is really robust in robot localization problems. In fact, if you googled "Particle Filter (Codes)", most of the resources would points you to some Robotic classes, or attempt to teach particle filter through solving a basic robot localization problem. It is fun but to be honest, it hindered my understanding of how to use particle filters in other problems, such as predicting the next event of a time-series dataset.

Lastly, I would like to state that even though I tried my best, I don't have a lot of experiences with particle filters and as such, there might still be mistakes in this work. If you found any, please don't hesitate to contact me through the email provided above.

6 References

[1] http://web.mit.edu/16.412j/www/html/Advanced%20lectures/Slides/Hsaio_plinval_miller_ParticleFiltersPrint.p

[2] Wikipedia: https://en.wikipedia.org/wiki/Particle_filter

[3] Towardsdatascience.com:
https://towardsdatascience.com/particle-filter-a-hero-in-the-world-of-non-linearity-and-non-gaussian-6d8947f4a3dc

[4] A gentle introduction to predictive filter:
https://www.researchgate.net/publication/220162233_A_Gentle_Introduction_to_Predictive_Filters

[5] Particle Filtering Estimation for Linear and Non-Linear state-space models - Lesly Argueta PhD Thesis (*the resource upon which a large part of this paper is based on*)

[6] https://irma.math.unistra.fr/ guillou/meeting/cappe.pdf

7 Pseudo-code and Figures

All of the below are taken from [5]

```
Algorithm 4 SIS Filter
Initialization   $t = 0$
  for $j = 1$ to $M$ do
    Sample $\boldsymbol{x}_0^{(j)} \sim p(\boldsymbol{x}_0)$ (Random sample taken from $p(\boldsymbol{x}_0)$ with $\boldsymbol{\omega}_0^{(j)} = \frac{1}{M}$)
  end for
  for $t = 1$ to $N$ do
Step 1   Importance sampling step
    for $j = 1$ to $M$ do
      Prediction Sample particles from proposal PDF $\boldsymbol{x}_t^{(j)} \sim q(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}^{(j)}, \boldsymbol{y}_t)$
      Filtering Assign to each particle $\boldsymbol{x}_t^{(j)}$ the importance weight $\boldsymbol{\omega}_t^{(j)}$ according to (2.38)
    end for
    for $j = 1$ to $M$ do
      Normalize the importance weights: $\tilde{\boldsymbol{\omega}}_t^{(j)} = \frac{\boldsymbol{\omega}_t^{(j)}}{\sum_{i=1}^{M} \boldsymbol{\omega}_t^{(i)}}$
    end for
  end for
```

Figure 3: SIS Filter pseudo-code

```
Algorithm 5 SISR PF
Initialization   $t = 0$
  for $j = 1$ to $M$ do
    Sample $\boldsymbol{x}_0^{(j)} \sim p(\boldsymbol{x}_0)$
  end for
  for $t = 1$ to $N$ do
Step 1   Importance sampling step
    for $j = 1$ to $M$ do
      Prediction Sample $\boldsymbol{x}_t^{(j)} \sim q(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}^{(j)}, \boldsymbol{y}_t)$
      Filtering Assign to each particle $\boldsymbol{x}_t^{(j)}$ the importance weight $\boldsymbol{\omega}_t^{(j)}$ according to (2.38)
    end for
    for $j = 1$ to $M$ do
      Normalize the importance weights: $\tilde{\boldsymbol{\omega}}_t^{(j)} = \frac{\boldsymbol{\omega}_t^{(j)}}{\sum_{i=1}^{M} \boldsymbol{\omega}_t^{(i)}}$
    end for
Step 2   Resampling step
    Resample with replacement the particles $\boldsymbol{x}_t^{(1)}, \ldots, \boldsymbol{x}_t^{(M)}$ according to importance weights
    $\{\tilde{\boldsymbol{\omega}}_t^{(1)}, \ldots, \tilde{\boldsymbol{\omega}}_t^{(M)}\}$
  end for
```

Figure 4: SISR PF pseudo-code

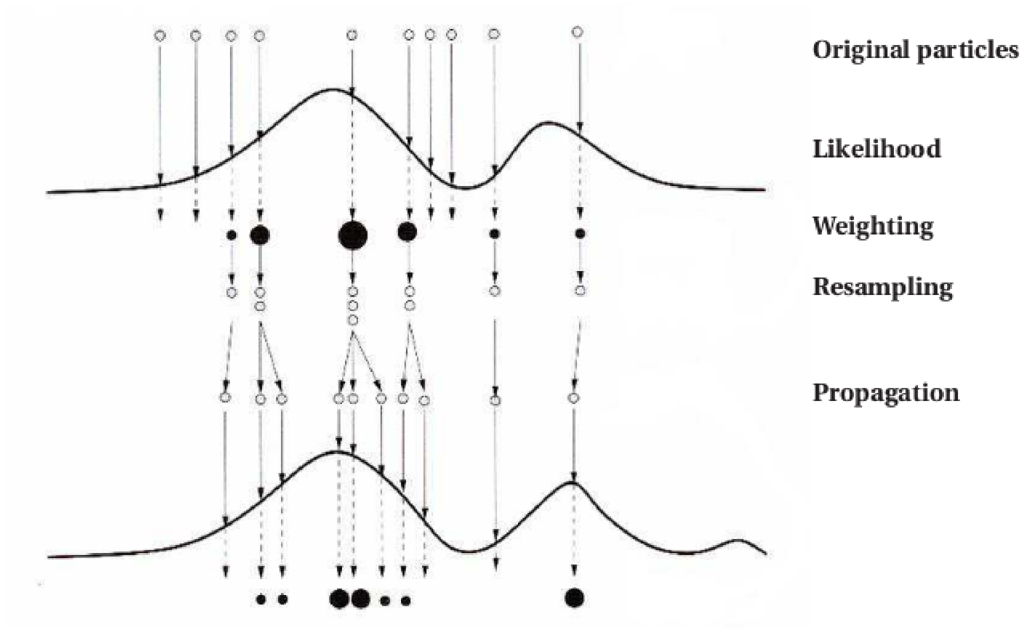

Figure 5: SISR illustration

```
Algorithm 6 SIR PF
Initialization  $t = 0$
  for $j = 1$ to $M$ do
    Sample $\boldsymbol{x}_0^{(j)} \sim p(\boldsymbol{x}_0)$
  end for
  for $t = 1$ to $N$ do
Step 1  Importance sampling step
    for $j = 1$ to $M$ do
      Prediction Sample $\boldsymbol{x}_t^{(j)} \sim q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}^{(j)}, \boldsymbol{y}_t) = p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}^{(j)})$ by
        • generating $\boldsymbol{\eta}_t^{(j)}$ according to the state-noise density (2.1)
        • setting $\boldsymbol{x}_t^{(j)} = f(\boldsymbol{x}_{t-1}^{(j)}, \boldsymbol{\eta}_t^{(j)})$
      Filtering: Assign to each particle $\boldsymbol{x}_t^{(j)}$ the weight $\boldsymbol{\omega}_t^{(j)}$ according to (2.44)
    end for
    for $j = 1$ to $M$ do
      Normalize the importance weights: $\tilde{\boldsymbol{\omega}}_t^{(j)} = \frac{\boldsymbol{\omega}_t^{(j)}}{\sum_{i=1}^{M} \boldsymbol{\omega}_t^{(i)}}$
    end for
Step 2  Resampling step
    Resample with replacement the particles $\boldsymbol{x}_t^{(1)}, \dots, \boldsymbol{x}_t^{(M)}$ according to importance weights
    $\{\tilde{\boldsymbol{\omega}}_t^{(1)}, \dots, \tilde{\boldsymbol{\omega}}_t^{(M)}\}$
  end for
```

Figure 6: SIR PF pseudo-code

Algorithm 8 ASIR PF

```
Initialization   $t = 0$
  for $k = 1$ to $M$ do
    Sample $\boldsymbol{x}_0^{(k)} \sim p(\boldsymbol{x}_0)$
  end for
  for $t = 1$ to $N$ do
Step 1   Auxiliary variable resampling step
    for $k = 1$ to $M$ do
      Select and calculate $\mu_t^{(k)}$ associated to the conditional PDF of $(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}^{(k)})$
      Calculate the first stage weights $\boldsymbol{\lambda}_t^{(k)} = q(k|\boldsymbol{y}_{1:t})$ in (2.50)
    end for
    for $k = 1$ to $M$ do
      Normalize the first stage weights $\tilde{\boldsymbol{\lambda}}_t^{(k)} = \frac{\boldsymbol{\lambda}_t^{(k)}}{\sum_{i=1}^{M} \boldsymbol{\lambda}_t^{(k)}}$
    end for
    Sample with replacement the index $k_{j=1}^{M}$ according to the computed first stage weights.
Step 2   Importance sampling step
    for $j = 1$ to $M$ do
      Sample $\boldsymbol{x}_t^j \sim q(\boldsymbol{x}_t|k^{(j)}, \boldsymbol{y}_{1:t}) = p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}^{k^{(j)}})$ as in the SIR filter, that is
      Calculate the second stage weights $\boldsymbol{\omega}_t^{(j)}$ using (2.52)
    end for
    for $k = 1$ to $M$ do
      Normalize the second stage weights $\tilde{\boldsymbol{\omega}}_t^{(k)} = \frac{\boldsymbol{\omega}_t^{(k)}}{\sum_{k=1}^{M} \boldsymbol{\omega}_t^{(k)}}$
    end for
Step 3   Second Resampling step
    Resample with replacement the particles $\boldsymbol{x}_t^{(1)}, \dots, \boldsymbol{x}_t^{(M)}$ according to importance weights
    $\{\tilde{\boldsymbol{\omega}}_t^{(1)}, \dots, \tilde{\boldsymbol{\omega}}_t^{(M)}\}$
  end for
```

Figure 7: ASIR PF pseudo-code

```
Algorithm 9 Extended PF (EPF)
Initialization   $t = 0$
  for $j = 1$ to $M$ do
    Sample $x_0^{(j)} \sim p(x_0)$ and fixed known parameters.
  end for
  for $t = 1$ to $N$ do
Step 1    Importance sampling step
    for $j = 1$ to $M$ do
Step 2      Prediction step
      Compute $J_{\boldsymbol{x}_{t-1}}$, and $J_{\boldsymbol{\eta}_t}$ as in equation (2.19).
      Compute the predictive expectation $\bar{\boldsymbol{x}}_{t|t-1}^{(j)}$ and covariance $\boldsymbol{\Sigma}_{\boldsymbol{x}_{t|t-1}}^{(j)}$ using
      $\bar{\boldsymbol{x}}_{t|t-1}^{(j)} = f(\boldsymbol{x}_{t-1}^{(j)}, 0)$ and
      $\boldsymbol{\Sigma}_{\boldsymbol{x}_{t|t-1}}^{(j)} = J_{\boldsymbol{x}_{t-1}}^{(j)} \boldsymbol{\Sigma}_{\boldsymbol{x}_{t-1|t-1}}^{(j)} J_{\boldsymbol{x}_{t-1}}^{\prime(j)} + J_{\boldsymbol{\eta}_t}^{(j)} Q_t J_{\boldsymbol{\eta}_t}^{\prime(j)}$, respectively.
Step 3      Kalman Gain step
      Compute $J_{\boldsymbol{x}_t}$, and $J_{\boldsymbol{v}_t}$ as in equation (2.20).
      Compute the prediction estimate $\boldsymbol{y}_{t|t-1}^{(j)}$ and covariance $\boldsymbol{\Sigma}_{\boldsymbol{y}_{t|t-1}}^{(j)}$ using equations (2.26) and (2.27),
      respectively.
      $\boldsymbol{y}_{t|t-1}^{(j)} = h_{t|t-1}^{(j)} = h(\bar{\boldsymbol{x}}_{t|t-1}^{(j)}, 0)$
      $\boldsymbol{\Sigma}_{\boldsymbol{y}_{t|t-1}} = J_{\boldsymbol{x}_t}^{(j)} \boldsymbol{\Sigma}_{\boldsymbol{x}_{t|t-1}}^{(j)} J_{\boldsymbol{x}_t}^{\prime(j)} + J_{\boldsymbol{v}_t}^{(j)} R_t J_{\boldsymbol{v}_t}^{\prime(j)}$
      Compute the Kalman Gain $K_t$ with equation (2.28).
      $K_t = \boldsymbol{\Sigma}_{\boldsymbol{x}_{t|t-1}}^{(j)} J_{\boldsymbol{x}_t}^{\prime(j)} \boldsymbol{\Sigma}_{\boldsymbol{y}_{t|t-1}}^{-1(j)}$
Step 4      Filtering step
      Compute the filtering expectation $\bar{\boldsymbol{x}}_{t|t}$ and covariance $\boldsymbol{\Sigma}_{\boldsymbol{x}_{t|t}}$ using (2.29) and (2.30), respectively.
      $\bar{\boldsymbol{x}}_t^{(j)^{EKF}} = \bar{\boldsymbol{x}}_{t|t-1}^{(j)} + K_t(\boldsymbol{y}_t - \boldsymbol{y}_{t|t-1}^{(j)})$
      $\boldsymbol{\Sigma}_t^{(j)^{EKF}} = \boldsymbol{\Sigma}_{\boldsymbol{x}_{t|t-1}}^{(j)} - K_t J_{\boldsymbol{x}_t}^{(j)} \boldsymbol{\Sigma}_{\boldsymbol{x}_{t|t-1}}^{(j)}$
      Sample $\boldsymbol{x}_t^{(j)} \sim q(\boldsymbol{x}_t^{(j)} | \boldsymbol{x}_{t-1}^{(j)}, \boldsymbol{y}_{1:t}) \triangleq N(\bar{\boldsymbol{x}}_t^{(j)^{EKF}}, \boldsymbol{\Sigma}_t^{(j)^{EKF}})$
    end for
    for $j = 1$ to $M$ do
      Evaluate the importance weights up to a normalizing constant.
      $\boldsymbol{\omega}_t^{(j)} \propto \frac{p(\boldsymbol{y}_t | \boldsymbol{x}_t^{(j)^{EKF}})\, p(\boldsymbol{x}_t^{(j)^{EKF}} | \boldsymbol{x}_{t-1}^{(j)})}{q(\boldsymbol{x}_t^{(j)} | \boldsymbol{x}_{t-1}^{(j)}, \boldsymbol{y}_t)}$ using equation (2.53) or equation (2.54), as appropriate.
    end for
    for $j = 1$ to $M$ do
      Normalize the importance weights $\tilde{\boldsymbol{\omega}}_t^{(j)} = \frac{\boldsymbol{\omega}_t^{(j)}}{\sum_{i=1}^{M} \boldsymbol{\omega}_t^{(i)}}$.
    end for
Step 5    Resample the discrete PDF to obtain a sample of size $M$.
    Multiply/Supress particles $\boldsymbol{x}_t^{(j)^{\mathrm{EKF}}}$ according to high/low importance weights, $\tilde{\boldsymbol{\omega}}_t^{(j)}$
  end for
```

Figure 8: EPF pseudo-code

```
Algorithm 10 Unscented PF (UPF)
Initialization   $t = 0$
  Set parameter values: $\alpha$, $\beta$ and $\kappa$.
  Compute the dimension of the augmented state: $n_a = n_x + n_\eta + n_\nu$
  for $j = 1$ to $M$ do
    Sample $x_0^{(j)} \sim p(x_0)$ and set:
    $\bar{\boldsymbol{x}}_0^{(j)} = \mathrm{E}(\boldsymbol{x_0}^{(j)})$
    $\Sigma_0^{(j)} = E\left((\boldsymbol{x_0}^{(j)} - \bar{\boldsymbol{x}}_0^{(j)})(\boldsymbol{x_0}^{(j)} - \bar{\boldsymbol{x}}_0^{(j)})'\right)$
    $\bar{\boldsymbol{x}}_0^{(j)a} = \mathrm{E}(\boldsymbol{x}_0^{(j)a}) = ((\bar{\boldsymbol{x}}_0^{(j)})', \mathbf{0}, \mathbf{0})'$
    $\Sigma_0^{(j)a} = E\left((\boldsymbol{x}_0^{(j)a} - \bar{\boldsymbol{x}}_0^{(j)a})(\boldsymbol{x}_0^{(j)a} - \bar{\boldsymbol{x}}_0^{(j)a})'\right)$ and fixed known parameters.
  end for
  for $t = 1$ to $N$ do
Step 1   Importance sampling step
    for $j = 1$ to $M$ do
Step 2   Update the particles with the UKF
      Compute the sigma points

      $\boldsymbol{\chi}_{t-1}^{(j)a} = [\bar{\boldsymbol{x}}_{t-1}^{(j)a}, \bar{\boldsymbol{x}}_{t-1}^{(j)a} \pm \left(\sqrt{(n_a + \lambda)\Sigma_{t-1}^{(j)a}}\right)]$

      Time update (propagate particles into the future)

      $\boldsymbol{\chi}_{t|t-1}^{(j)x} = f(\boldsymbol{\chi}_{t-1}^{(j)x}, \boldsymbol{\chi}_{t-1}^{(j)\eta})$      $\bar{\boldsymbol{x}}_{t|t-1}^{(j)} = \sum_{i=0}^{2n_a} \boldsymbol{\omega}_i^{(m)} \boldsymbol{\chi}_{i,t|t-1}^{(j)x}$
      $\Sigma_{t|t-1}^{(j)} = \sum_{i=0}^{2n_a} \boldsymbol{\omega}_i^{(c)} (\boldsymbol{\chi}_{i,t|t-1}^{(j)x} - \bar{\boldsymbol{x}}_{t|t-1}^{(j)})(\boldsymbol{\chi}_{i,t|t-1}^{(j)x} - \bar{\boldsymbol{x}}_{t|t-1}^{(j)})'$
      $\boldsymbol{y}_{t|t-1}^{(j)} = h(\boldsymbol{\chi}_{t|t-1}^{(j)x}, \boldsymbol{\chi}_{t-1}^{(j)v})$      $\bar{\boldsymbol{y}}_{t|t-1}^{(j)} = \sum_{i=0}^{2n_a} \boldsymbol{\omega}_i^{(m)} \boldsymbol{y}_{i,t|t-1}^{(j)}$

      Measurement update (incorporate new observation)

      $\Sigma_{\bar{\boldsymbol{y}}_t, \bar{\boldsymbol{y}}_t} = \sum_{i=0}^{2n_a} \boldsymbol{\omega}_i^{(c)} (\boldsymbol{y}_{i,t|t-1}^{(j)} - \bar{\boldsymbol{y}}_{t|t-1}^{(j)})(\boldsymbol{y}_{i,t|t-1}^{(j)} - \bar{\boldsymbol{y}}_{t|t-1}^{(j)})'$
      $\Sigma_{\boldsymbol{x}_t, \bar{\boldsymbol{y}}_t} = \sum_{i=0}^{2n_a} \boldsymbol{\omega}_i^{(c)} (\boldsymbol{\chi}_{i,t|t-1}^{(j)x} - \bar{\boldsymbol{x}}_{t|t-1}^{(j)})(\boldsymbol{y}_{i,t|t-1}^{(j)} - \bar{\boldsymbol{y}}_{t|t-1}^{(j)})'$
          $K_t = \Sigma_{\boldsymbol{x}_t, \bar{\boldsymbol{y}}_t} \Sigma_{\bar{\boldsymbol{y}}_t, \bar{\boldsymbol{y}}_t}^{-1}$
      $\bar{\boldsymbol{x}}_t^{(j)^{UKF}} = \bar{\boldsymbol{x}}_{t|t-1}^{(j)} + K_t(\boldsymbol{y}_t - \bar{\boldsymbol{y}}_{t|t-1}^{(j)})$   $\Sigma_t^{(j)^{UKF}} = \Sigma_{t|t-1}^{(j)} - K_t \Sigma_{\bar{\boldsymbol{y}}_t, \bar{\boldsymbol{y}}_t} K_t'$

      Sample $\boldsymbol{x}_t^{(j)} \sim q(\boldsymbol{x}_t^{(j)} | \boldsymbol{x}_{t-1}^{(j)}, \boldsymbol{y}_{1:t}) \doteq N(\bar{\boldsymbol{x}}_t^{(j)^{UKF}}, \boldsymbol{\Sigma}_t^{(j)^{UKF}})$
    end for
    for $j = 1$ to $M$ do
      Evaluate the importance weights up to a normalizing constant.
      $\boldsymbol{\omega}_t^{(j)} \propto \frac{p(\boldsymbol{y}_t | \boldsymbol{x}_t^{(j)^{UKF}})\, p(\boldsymbol{x}_t^{(j)^{UKF}} | \boldsymbol{x}_{t-1}^{(j)})}{q(\boldsymbol{x}_t^{(j)} | \boldsymbol{x}_{t-1}^{(j)}, \boldsymbol{y}_t)}$ using equation (2.53) or equation (2.54), as appropriate.
    end for
    for $j = 1$ to $M$ do
      Normalize the importance weights $\tilde{\boldsymbol{\omega}}_t^{(j)} = \frac{\boldsymbol{\omega}_t^{(j)}}{\sum_{i=1}^{M} \boldsymbol{\omega}_t^{(i)}}$.
    end for
Step 3   Resample the discrete PDF to obtain a sample of size $M$.
    Multiply/Supress particles $\boldsymbol{x}_t^{(j)^{\mathrm{UKF}}}$ according to high/low importance weights, $\tilde{\boldsymbol{\omega}}_t^{(j)}$
  end for
```

Figure 9: UPF pseudo-code