Lab 6 Particle Filtering - ECE 8540 Analysis of Tracking Systems

Mayuresh Bhosale

November 24, 2022

1 Introduction

This lab report explains how to use the particle filter on a set of measured data. The measured data is non-linear, with non-gaussian noise. For non-linear data, the extended Kalman filter can be used to get the estimates but requires the noise distribution to be gaussian. For non-linear data with non-gaussian noise Particle Filter is applied for tracking systems. For this lab, the object moving zig-zag along a line is tracked. The system has two magnets at a defined position that provide magnetic strength for the object moving in the line. The approximate position of the object is determined based on the particle filtering applied to the magnetic strength. The measured data and ground truth data are provided to us.

2 Methodology

The particle filter is based on Bayesian estimation, Monte Carlo approximation, and sequential importance sampling. The particle filter algorithm follows a predict-update cycle where the next state of each particle is determined. Using the new measurement, the weight of each particle is updated and normalized later. The expected output is computed based on the normalized weight and the state at that time step. Based on the set threshold, it is decided if resampling of particles is necessary and is accordingly performed. For this problem, following are the equations used for particle filtering:
The two state variables are:

$$X_t = \begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} \tag{1}$$

where x_t is position and $\dot{x}_t$ is velocity of the object.

1. Calculating the next step of each particle from the state transition equation

$$\{x_t^{(m)} = f(x_{t-1}^{(m)}, a_t^{(m)})\}_{m=1}^{M} \tag{2}$$

   $a_t^{(m)}$ represents the dynamic noise from $t-1$ to t and $f(x_{t-1}^{(m)}, a_t^{(m)})$ is given as:

$$f(x_t, a_t) = \begin{bmatrix} x_{t+1} = x_t + \dot{x}_t T \\ \dot{x}_{t+1} = \begin{cases} 2 & \text{if } x_t < -20 \\ \dot{x}_t + |a_t| & \text{if } -20 \le x_t < 0 \\ \dot{x}_t - |a_t| & \text{if } 0 \le x_t \le 20 \\ -2 & \text{if } x_t > 20 \end{cases} \end{bmatrix} \tag{3}$$

   The values a_t are drawn from a zero-mean Gaussian distribution $N(0, \sigma_a^2)$. The data was generated using a value of $\sigma_a = 2^{-4} = 0.0625$. The goal of the state transition equation is to keep the position oscillating about zero but between -20 and 20.

2. Record the sensor-measured data

$$Y_t = \begin{bmatrix} y_t \end{bmatrix} \tag{4}$$

3. Calculate the ideal measurement of the particle

$$g(x_t^{(m)}, 0) = \left[y_t^{(m)} = \frac{1}{\sqrt{2\pi}\sigma_m} \exp(\frac{-(x_t^{(m)} - x_{m1})^2}{2\sigma_m^2}) + \frac{1}{\sqrt{2\pi}\sigma_m} \exp(\frac{-(x_t^{(m)} - x_{m2})^2}{2\sigma_m^2}) \right] \tag{5}$$

where, n_t is measurement noise and a random sample drawn from $N(0, \sigma_n^2)$. $\sigma_m = 4.0$ and $\sigma_n = 0.003906$.

4. Compare the ideal measurement to the actual measurement

$$p(y_t | x_t^{(m)}) = \frac{1}{\sqrt{2\pi}\sigma_n} \exp(\frac{-(y_t^{(m)} - y_t)^2}{2\sigma_n^2}) \tag{6}$$

5. Calculate the updated weights of the particles

$$\tilde{w}_t^{(m)} = w_{t-1}^{(m)} \cdot p(y_t | x_t^{(m)}) \tag{7}$$

6. Normalize the updated weights

$$w_t^{(m)} = \frac{\tilde{w}_t^{(m)}}{\sum_{m=1}^{M} \tilde{w}_t^{(m)}} \tag{8}$$

7. Record the desired output (Expected value)

$$E[x_t] \approx \sum_{m=1}^{M} x_t^{(m)} \cdot w_t^{(m)} \tag{9}$$

8. Check if the resampling of weights is required

$$\mathrm{CV} = \frac{\mathrm{VAR}(w^{(m)})}{E^2[w^{(m)}]} = \frac{\frac{1}{M} \sum_{m=1}^{M} \left(w^{(m)} - \frac{1}{M} \sum_{m=1}^{M} w^{(m)} \right)^2}{\left(\frac{1}{M} \sum_{m=1}^{M} w^{(m)} \right)^2} = \frac{1}{M} \sum_{m=1}^{M} (M \cdot w^{(m)} - 1)^2 \tag{10}$$

$$\mathrm{ESS} = \frac{M}{1 + \mathrm{CV}} \tag{11}$$

The coefficient of variance and effective sampling size is used to determine if the particles have appreciable weights. M is the number of particles used, and in this problem, 1000 particles were used for particle filtering. If resampling is necessary, the weights of each particle are replaced to $1/M$.

9. Loop through each time step t

The model equations with data sets are further solved by using MATLAB Scripts. R2021a version of MATLAB was used on a Windows 11 Operating System. All the matrices were constructed in MATLAB, and expected output position of the object was obtained. Graphs were plotted for the given data set and the model equation for visualization.

3 Results

Results for the implementation of particle filters for 1000 particles and a resampling threshold of 0.5 are shown in this section. Figure 1 and figure 2 show the results of the final implementation of the particle filter for object tracking. Both plots depict Estimated values for each time step against the ground truth data and measured data. Figure 1 shows proper tracking of the object where the estimates are in phase with the ground truth data, whereas figure 2 shows the estimates are out of phase with the ground truth data. As

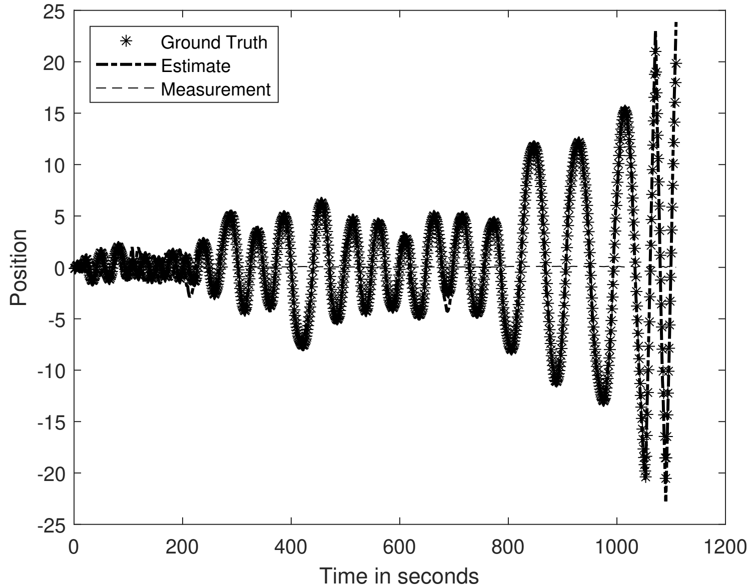

Figure 1: Plot of ground truth data, filtered estimates, and measured data for particle filtering

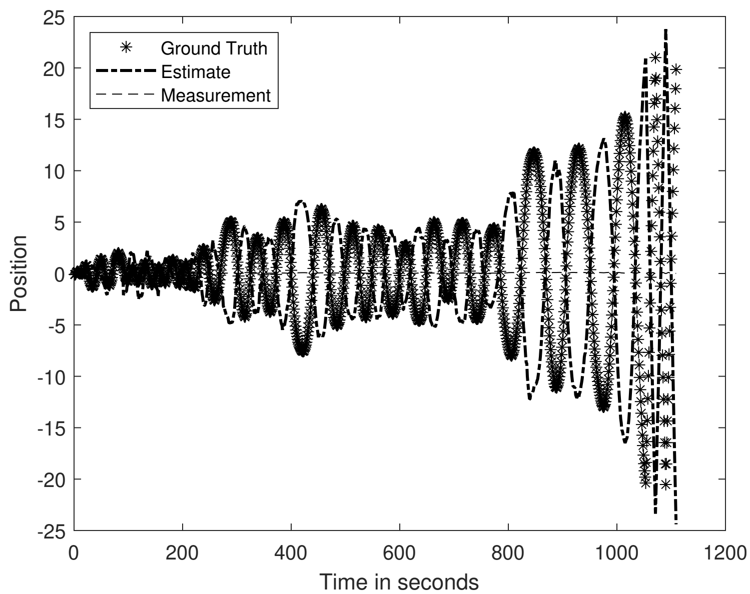

Figure 2: Plot of ground truth data, filtered estimates, and measured data for particle filtering

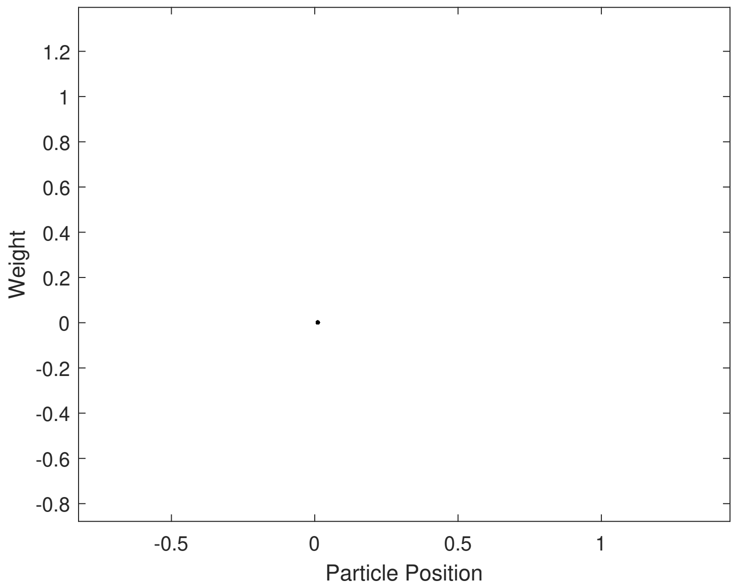

Figure 3: Distribution of particles at time step = 1 second

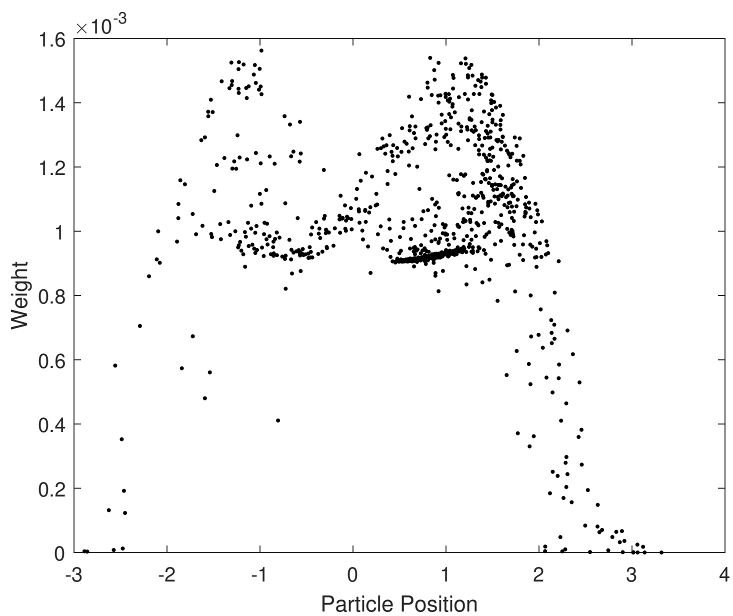

Figure 4: Distribution of particles at time step = 36 second

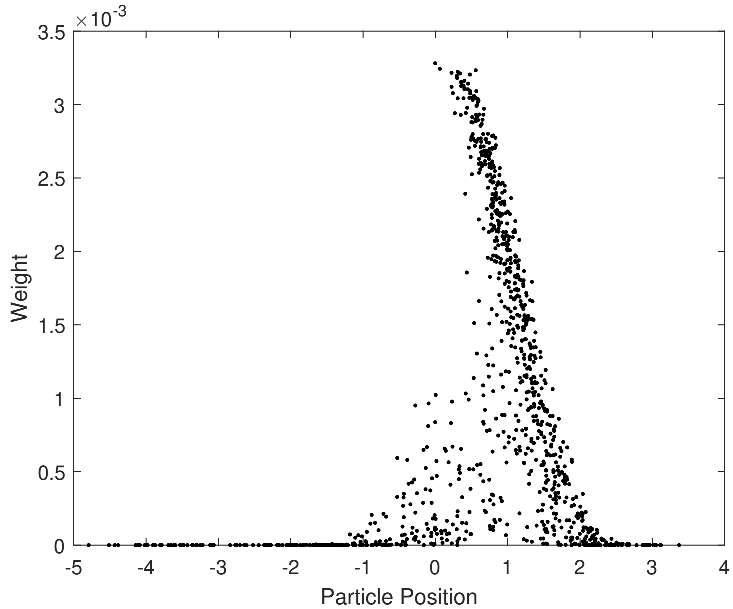

Figure 5: Distribution of particles at time step = 108 second, Before resampling

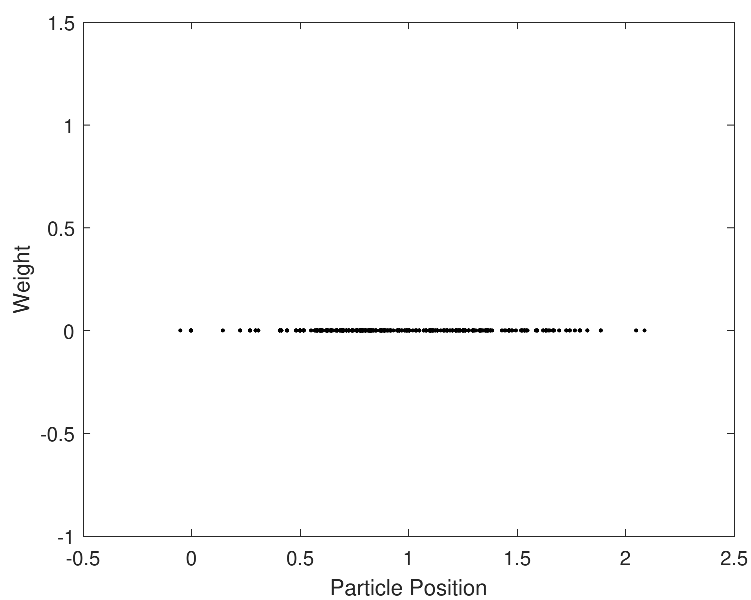

Figure 6: Distribution of particles at time step = 108 second, After resampling

observed, measurement data is too noisy and is deviating largely from the ground truth data.
Figures 3 to 6 represents the distribution of particles at one instance of time. Y-axis represents the weight of the particle, and the x-axis represents the position of the particles. Initialization of 1000 particles is done at the same position with particle weight as $1/M$, where M is the number of particles. Figure 3 shows the initialization of particles at time step 1 second. Figure 4 shows the distribution of particles at time 36 seconds, and as observed, the distribution looks like dual gaussian. Figures 5 and 6 show the distribution of particles before and after resampling, respectively. As there are more particles with inappropriate weight, the resampling is performed, and all the particles are given an equal weight of $1/M$ as shown in figure 6.

4 Conclusion

In this Lab, I learned how to implement Particle filtering for object tracking. Based on the particle distribution change over time, it can be suggested that Particle filtering performs well with non-gaussian noise for object tracking. Due to the symmetrical nature of the experiment, the estimated position is sometimes out of phase and in phase, as shown in the results. The particle filter can be applied to most of the systems for the given model knowledge. I learned that with a number of consecutive iterations over the time steps, the weight of particles decreases, and particles slowly start to become insignificant. Resampling is an important step and is performed based on a set threshold to avoid the increasing number of dying particles.

5 Appendix

The MATLAB code for this lab can be found below:

```
clear all
clc
close all

Adata = importdata("magnets-data.txt");

%Data input as defined
pos_t = Adata(:,1); %Ground truth position
vel_t = Adata(:,2); %Velocity Ground truth
pos_yt = Adata(:,3); %Measurement output
time = 1:length(Adata);
totaldata = length(Adata);

% plot(time,pos_t);
% hold on
% plot(time,pos_yt);
% legend('GT','Meas');
% hold off
% plot(time,pos_yt);

% Magnet positions as defined
x_m1 = -10;
x_m2 = 10;

%Sigma as defined
sigma_a = 0.0625;
sigma_m = 4;
sigma_n = 0.003906;

%Sampling time
```

```
Time = 1; %Second

%Number of Particles
M = 1000;

%Initialize
%State transition matrix/eqs
x_t = ones(1,M)*pos_yt(1);
x_t1 = ones(1,M)*pos_yt(1);
x_t_dot = ones(1,M)*pos_yt(1);
x_t1_dot = ones(1,M)*pos_yt(1);

%weight & variables
w_t = ones(1,M)*(1/M);
norm_w_t = ones(1,M)*(1/M);
w_t1 = ones(1,M)*(1/M);
yt = [];
Pr=[];
ESS = [];
CV = 0;
threshold=0.5;
count = 20;
resample_ct=0;
xaxis = 1:M;
switch_on = 0;
Estimates = zeros(1,totaldata);
Pr = zeros(1,M);
y1 = zeros(1,M);
y2 = zeros(1,M);
yt = zeros(1,M);

for t = 1:totaldata

    %New states
    for i = 1:M

        x_t(i) = x_t1(i) + x_t1_dot(i)*Time;

        if (x_t1(i) < (-20))
            x_t_dot(i) = 2;

        elseif (-20 <= x_t1(i) && x_t1(i) < 0)
            x_t_dot(i) = x_t1_dot(i) + abs(normrnd(0,sigma_a));

        elseif (0 <= x_t1(i) && x_t1(i) <= 20)
            x_t_dot(i) = x_t1_dot(i) - abs(normrnd(0,sigma_a));

        elseif (x_t1(i)>20)
            x_t_dot(i) = -2;
        end

        %Calculate ideal y
        y1(i) = ( (1/(sqrt(2*pi)*sigma_m)) * exp(-((x_t1(i)-x_m1)^2)/(2*(
            sigma_m^2)) ) );
```

```
        y2(i) = ( (1/(sqrt(2*pi)*sigma_m)) * exp(-((x_t1(i)-x_m2)^2)/(2*(sigma_m^2)) ) );

        yt(i) = y1(i)  +  y2(i);

        %Calculate trhe probability
        Pr(i) = (1/(sqrt(2*pi)*sigma_n)) * exp(-((yt(i)-pos_yt(t))^2)/(2*(sigma_n^2)) );

        %Update the wiights
        w_t(i) = w_t1(i)*Pr(i);
%           w_t1(i) = w_t(i);

        x_t1(i) = x_t(i);
        x_t1_dot(i) = x_t_dot(i);
    end

    Exp=0;
    for ii = 1:M
        %calculate normal weight
        norm_w_t(ii)=w_t(ii)/sum(w_t);

        %Filter output
        Exp = Exp + (x_t(ii)*norm_w_t(ii));

        %Resampling
        w_t1(ii)= norm_w_t(ii);
    end
    Estimates(t)=Exp; %Recording estimates values

    CV = var(norm_w_t) / (mean(norm_w_t) ^ 2);

    ESS = M/(CV+1);


%       figure(t)
%       plot(x_t,norm_w_t,'k.');
%       xlabel('Particle Position');
%       ylabel('Weight');
%       t
%       hold off


    Index=zeros(1,M);

    %Resampling code
    if(ESS<M*threshold)
        %Gathering the resampling number
        resample_ct = resample_ct + 1;

        Q=cumsum(norm_w_t);
        t1=rand(M+1,1);
        T=sort(t1);
        T(M+1)=1.0;
```

```
            k=1;
            j=1;
            while (k<=M)
                if (T(k) < Q(j))
                    Index(k)=j;
                    k=k+1;
                else
                    j=j+1;
                end
            end

            for a=1:M
                x_t(a) = x_t(Index(a));
                x_t1(a) = x_t1(Index(a));
                x_t_dot(a) = x_t_dot(Index(a));
                x_t1_dot(a) = x_t1_dot(Index(a));
                norm_w_t(a) = 1/M;
                w_t1(a) = 1/M;
            end
        end

        if(resample_ct == count)
            figure(1+t)
            plot(x_t,norm_w_t,'k.');
            xlabel('Particle Position');
            ylabel('Weight');
            t
            hold off
        end


end

plot((1:length(pos_t)) ,pos_t, 'k*') ;
hold on
plot( (1:length(pos_t)) ,Estimates, 'k-.', 'Linewidth', 1.5 ) ;
plot((1:length(pos_t)), pos_yt,'k--')
xlabel ('Time in seconds');
ylabel ('Position');
legend ('Ground Truth' , 'Estimate', 'Measurement','Location',"northwest");
hold off
```