ECE 245 Estimation and Introduction to Control of Stochastic Processes

Final Project

Instructor: Prof. Dejan Milutinovic

Student: Keng-yu Lin 1694084

Submitted on: June 10, 2020

Part 1 Setup the dynamics model

We have a dynamic model form as

$x_1 = x$

$x_2 = y$

$x_3 = v$

$x_4 = \theta$

$$X(t) = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \quad F(X(t)) = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} v(t)\cos\theta(t) \\ v(t)\sin\theta(t) \\ 0 \\ 0 \end{bmatrix}$$

$$\dot{X}(t) = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} v(t)\cos\theta(t) \\ v(t)\sin\theta(t) \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \xi_v(t) \\ \xi_\theta(t) \end{bmatrix}$$

We know that

$$\dot{v} = \xi_v(t) \quad , \quad dv = \delta_v dw_v \quad , \quad w_x \sim N(0, W_x)$$

$$\dot{\theta} = \xi_\theta(t) \quad , \quad d\theta = \delta_\theta dw_\theta \quad , \quad w_y \sim N(0, W_y)$$

We can re-write the dynamic model form in discrete-time model as

$$X_{(k+1)} = \begin{bmatrix} x_{(k+1)} \\ y_{(k+1)} \\ v_{(k+1)} \\ \theta_{(k+1)} \end{bmatrix} = \begin{bmatrix} x_{(k)} \\ y_{(k)} \\ v_{(k)} \\ \theta_{(k)} \end{bmatrix} + \begin{bmatrix} v_{(k)}\cos(\theta_k)\,\Delta T & -\theta_{(k)}v_{(k)}\sin(\theta_k)\,\Delta T \\ v_{(k)}\sin(\theta_k)\,\Delta T & \theta_{(k)}v_{(k)}\cos(\theta_k)\,\Delta T \\ & \delta_v\, W_v \\ & \delta_\theta\, W_\theta \end{bmatrix}$$

$$Z_{(k)} = \begin{bmatrix} X_{m(k)} \\ Y_{m(k)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} X_{(k)} + \begin{bmatrix} W_x \\ W_y \end{bmatrix}$$

For the extended Kalman filter (EKF) form is that

$$\underline{\mathrm{x}}_{k+1} = \underline{\Phi}_k \underline{\mathrm{x}}_k + \underline{\Gamma}_k \underline{\mathrm{w}}_k$$

$$\underline{\mathrm{y}}_k = \underline{\mathrm{H}}_k \underline{\mathrm{x}}_k + \underline{\mathrm{v}}_k$$

For our dynamic system model form can be written as

$$X_{(k+1)} = \begin{bmatrix} 1 & 0 & \cos(x_{4(k)})\,\Delta T & -x_{3(k)}\sin(x_{4(k)})\,\Delta T \\ 0 & 1 & \sin(x_{4(k)})\,\Delta T & x_{3(k)}\cos(x_{4(k)})\,\Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{1(k)} \\ x_{2(k)} \\ x_{3(k)} \\ x_{4(k)} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \delta_v & 0 \\ 0 & \delta_\theta \end{bmatrix} \begin{bmatrix} W_v \\ W_\theta \end{bmatrix}$$

$$Z_{(k)} = \begin{bmatrix} X_{m(k)} \\ Y_{m(k)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} X_{(k)} + \begin{bmatrix} W_x \\ W_y \end{bmatrix}$$

So the matrices can be represented as

$$\underline{\Phi}_k = \begin{bmatrix} 1 & 0 & \cos(x_{4(k)})\,\Delta T & -x_{3(k)}\sin(x_{4(k)})\,\Delta T \\ 0 & 1 & \sin(x_{4(k)})\,\Delta T & x_{3(k)}\cos(x_{4(k)})\,\Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\underline{\Gamma}_k = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \delta_v & 0 \\ 0 & \delta_\theta \end{bmatrix}$$

$$\underline{H}_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\underline{V}_k = \begin{bmatrix} W_x \\ W_y \end{bmatrix}$$

$$E[\,W_k W_k^T\,] = Q_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$E[\,V_k V_k^T\,] = R_k = \begin{bmatrix} w_x & 0 \\ 0 & w_y \end{bmatrix}$$

$$\underline{\mathrm{P}}_k = \begin{bmatrix} var[\,x\,] & 0 & 0 & 0 \\ 0 & var[\,y\,] & 0 & 0 \\ 0 & 0 & var[\,v\,] & 0 \\ 0 & 0 & 0 & var[\,\theta\,] \end{bmatrix}$$

Part 2 Setup the initial guess

Now we need to define the initial state and initial covariance matrices.

2.1 Initial state guess

For x and y, we can choose the first measurements x_{m1} and y_{m1} from the video as the initial guess x_1 and x_2 .

For the velocity, we can use first and second measurement's points x_{m1} x_{m2} y_{m1} y_{m2} to be an initial guess.

$$v_0 = \sqrt{v_x^2 + v_y^2} = \sqrt{(\frac{x_{m2} - x_{m1}}{dT})^2 + (\frac{y_{m2} - y_{m1}}{dT})^2}$$

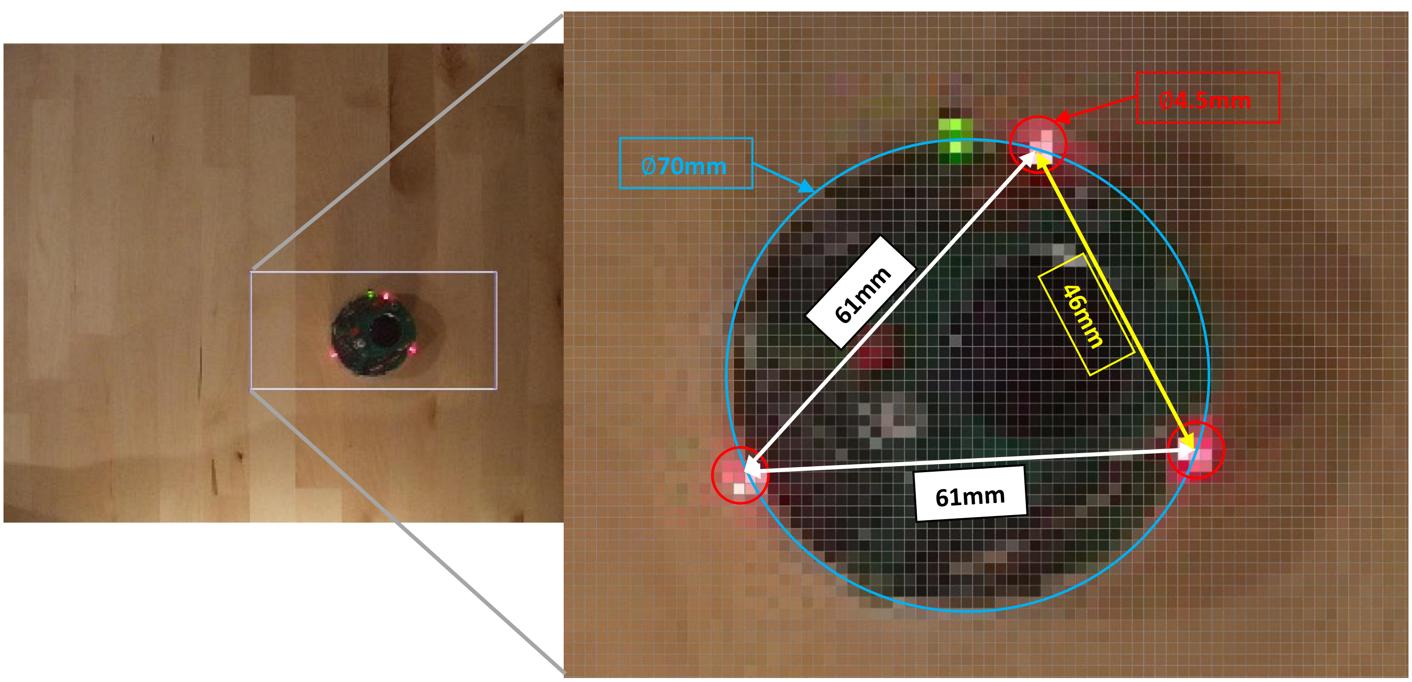
For the θ, we can see the robot is moving from the left side to the right side from the video so that we can guess $\theta_0 = \pi$.

So we can get our $\hat{x}_0$

$$\hat{x}_0 = \begin{bmatrix} x_{(0)} \\ y_{(0)} \\ v_{(0)} \\ \theta_{(0)} \end{bmatrix} = \begin{bmatrix} x_{m(1)} \\ y_{m(1)} \\ v_{(0)} \\ \pi \end{bmatrix}$$

2.2 Initial covariance matrices guess

The data are derived from the video based on **(1)** the image resolution is 320 x 240 **(2)** the robot size is about 70mm diameter **(3)** the distance from the front light to both backlights is 61mm and **(4)** the distance between the two backlights is 46mm.



We can see from the video frames, the diameter of the red LED light is $\emptyset = 4.5\ mm = 0.45\ cm$.

So we can assume the standard deviation (std) of one red LED light is $std = 0.45\ cm$, then the variance of one red LED light is $(0.45)^2$, and we have 3 LED lights, so the variance of $var[\ x\ ]\ and\ var[\ y\ ]$ is

$$var\ [\ x\ ] = var\ [\ y\ ] = \frac{3 * (0.45)^2}{9} = 0.0675$$

For the $var\ [\ v\ ]$, we can found all section's velocity $v_1\ v_2\ v_3\ v_4\ \ldots\ldots v_n$

$$\sum_{i=1}^{n} \sqrt{(\frac{x_{m(i+1)} - x_{mi}}{dT})^2 + (\frac{y_{m(i+1)} - y_{mi}}{dT})^2}$$

And take approximate derivatives from $v_5\ to\ v_{35}$ and take standard deviation for them

We can get

Std_v = Std [diff(v(5:35))];

Std_v = 0.6984

$$var\ [\ v\ ] = (0.9684)^2 = 0.4877$$

For the $var\ [\ \theta\ ]$, we use *HeadingAngle_rad.csv* file data and trim the data range not in $(-\pi,\ \pi\ ]$ but in continuous smooth data. The reason to do this is that it can minimize the approximate derivatives when two data points are near $-\pi\ and\ \pi$.

Then we can take approximate derivatives and retake standard deviation for those data.

Std_θ = Std [diff (*HeadingAngle*)];

Std_ θ = 0.2201

$$var\ [\ \theta\ ] = (0.2201)^2 = 0.0484$$

We can get the matrix $\underline{P_0}$

$$\underline{P_0} = \begin{bmatrix} 0.0675 & 0 & 0 & 0 \\ 0 & 0.0675 & 0 & 0 \\ 0 & 0 & 0.4877 & 0 \\ 0 & 0 & 0 & 0.0484 \end{bmatrix}$$

2.3 Initial Gaussian-distributed noise

Assume W_x and $W_y = var$ [x] and var[y], then

$$R = \begin{bmatrix} 0.0675 & 0 \\ 0 & 0.0675 \end{bmatrix}$$

Assume $\delta_v = \frac{1}{4}\sqrt{dT}$, $\delta_\theta = \frac{20\pi}{180}\sqrt{dT}$ and ($dT = \frac{1}{3}$) than

$$\underline{\Gamma}_k = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0.1443 & 0 \\ 0 & 0.2015 \end{bmatrix}$$

Part 3 Discrete-time Kalman filter model

$$\underline{x}_{k+1} = \underline{\Phi}_k \underline{x}_k + \underline{\Gamma}_k \underline{w}_k$$

Prediction step

$$\begin{aligned} \hat{\underline{x}}_{k+1} &= \underline{\Phi}_k \hat{\underline{x}}_k \\ \underline{P}_{k+1} &= \underline{\Phi}_k \underline{P}_k \underline{\Phi}_k^T + \underline{\Gamma}_k \underline{Q}_k \underline{\Gamma}_k^T \end{aligned}$$

Observations (general)

$$\underline{y}_k = \underline{H}_k \underline{x}_k + \underline{v}_k$$

Update (general)

$$\hat{\underline{x}}_k(+) = \hat{\underline{x}}_k(-) + \underline{K}(\underline{y}_k - \underline{H}_k \hat{\underline{x}}_k(-))$$

Optimal gain, or Kalman filter gain (general)

$$\begin{aligned} \underline{K} &= \underline{P}(-)\underline{H}^T(\underline{HP}(-)\underline{H}^T + \underline{R})^{-1} \\ \underline{P}(+) &= (I - \underline{KH})\underline{P}(-) \end{aligned}$$

Initial guess: $\hat{\underline{x}}_0$, $\underline{P}_0$
Prediction: $\hat{\underline{x}}_k$, and $\underline{P}_k$ are known

$$\begin{aligned}\hat{\underline{x}}_{k+1}(-) &= \underline{\Phi}_k \hat{\underline{x}}_k \\ \underline{P}_{k+1}(-) &= \underline{\Phi}_k \underline{P}_k \underline{\Phi}_k^T + \underline{\Gamma}_k \underline{Q}_k \underline{\Gamma}_k^T\end{aligned}$$

Gain (Version 1):

$$\underline{K}_{k+1} = \underline{P}_{k+1}(-)\underline{H}_{k+1}^T(\underline{H}_{k+1}\underline{P}_{k+1}(-)\underline{H}_{k+1}^T + \underline{R}_{k+1})^{-1}$$

Update (Version 1):

$$\begin{aligned}\hat{\underline{x}}_{k+1} &= \hat{\underline{x}}_{k+1}(-) + \underline{K}_{k+1}(\underline{y}_{k+1} - \underline{H}_{k+1}\hat{\underline{x}}_{k+1}(-)) \\ \underline{P}_{k+1} &= (I - \underline{K}_{k+1}\underline{H}_{k+1})\underline{P}_{k+1}(-)\end{aligned}$$

Part 4 Second-order filter model

$$\begin{aligned}\underline{x}_{k+1} &= \underline{f}_k(\underline{x}_k, \underline{w}_k) \\ \underline{y}_k &= \underline{h}_k(\underline{x}_k) + \underline{v}_k\end{aligned}$$

1. Linearization for the prediction step, $\underline{x}_k \in R^n$
Let us consider a single state x^i

$$x_{k+1}^i = f_k^i(\underline{x}_k, \underline{w}_k)$$

then the second order Taylor expansion is

$$\begin{aligned}x_{k+1}^i &= f_k^i(\hat{\underline{x}}_k, 0) + \frac{\partial f_k^i(\hat{\underline{x}}_k, 0)}{\partial \underline{x}_k}\delta\underline{x}_k + \frac{\partial f_k^i(\hat{\underline{x}}_k, 0)}{\partial \underline{w}_k}\underline{w}_k \\ &+ \frac{1}{2}\delta\underline{x}_k \frac{\partial^2 f_k^i(\hat{\underline{x}}_k, 0)}{\partial \underline{x}_k^2}\delta\underline{x}_k^T + \frac{1}{2}\underline{w}_k \frac{\partial^2 f_k^i(\hat{\underline{x}}_k, 0)}{\partial \underline{w}_k^2}\underline{w}_k^T + \delta\underline{x}_k \frac{\partial^2 f_k^i(\hat{\underline{x}}_k, 0)}{\partial \underline{x}_k \partial \underline{w}_k}\underline{w}_k^T \\ &+ hot\end{aligned}$$

$$E\{x^i_{k+1}\} = f^i_k(\underline{\hat{x}}_k,0) + E\left\{\frac{1}{2}\delta\underline{x}_k \frac{\partial^2 f^i_k(\underline{\hat{x}}_k,0)}{\partial \underline{x}^2_k}\delta\underline{x}^T_k + \frac{1}{2}\underline{w}_k \frac{\partial^2 f^i_k(\underline{\hat{x}}_k,0)}{\partial \underline{w}^2_k}\underline{w}^T_k\right\}$$

$$E\{x^i_{k+1}\} = f^i_k(\underline{\hat{x}}_k,0)\delta\underline{x}_k + \frac{1}{2}tr\left[\frac{\partial^2 f^i_k(\underline{\hat{x}}_k,0)}{\partial \underline{x}^2_k}E\{\delta\underline{x}^T_k\delta\underline{x}_k\}\right] + \frac{1}{2}tr\left[\frac{\partial^2 f^i_k(\underline{\hat{x}}_k,0)}{\partial \underline{w}^2_k}E\{\underline{w}^T_k\underline{w}_k\}\right]$$

$$E\{x^i_{k+1}\} = f^i_k(\underline{\hat{x}}_k,0) + \frac{1}{2}tr\left[\frac{\partial^2 f^i_k(\underline{\hat{x}}_k,0)}{\partial \underline{x}^2_k}\underline{P}_k\right] + \frac{1}{2}tr\left[\frac{\partial^2 f^i_k(\underline{\hat{x}}_k,0)}{\partial \underline{w}^2_k}Q_k\right]$$

Let us define $\underline{e}_1 = [1\ 0\ 0...\ 0]^T$, $\underline{e}_2 = [0\ 1\ 0...\ 0]^T$, ... $\underline{e}_n = [0\ 0\ 0...\ 1]^T$, then the predition step can be written as

$$\underline{\hat{x}}_{k+1}(-) = \underline{f}_k(\underline{\hat{x}}_k,0) + \underline{b}_k$$

$$\underline{b}_k = \frac{1}{2}\sum_{i=1}^{n}\underline{e}_i\left(tr\left[\underbrace{\frac{\partial^2 f^i_k(\underline{\hat{x}}_k,0)}{\partial \underline{x}^2_k}}_{\underline{F}^i_k}\underline{P}_k\right] + tr\left[\underbrace{\frac{\partial^2 f^i_k(\underline{\hat{x}}_k,0)}{\partial \underline{w}^2_k}}_{\underline{G}^i_k}Q_k\right]\right), \underline{e}_i \in R^n$$

$$\underline{b}_k = \frac{1}{2}\sum_{i=1}^{n}\underline{e}_i\left(tr\left[\underline{F}^i_k\underline{P}_k\right] + tr\left[\underline{G}^i_k Q_k\right]\right), \underline{e}_i \in R^n$$

$$\underline{P}_{k+1}(-) = \underline{\Phi}_k\underline{P}_k\underline{\Phi}^T_k + \underline{\Gamma}_k\underline{Q}_k\underline{\Gamma}^T_k$$

$$\underline{\Phi}_k = \frac{\partial \underline{f}_k(\underline{\hat{x}}_k,0)}{\partial \underline{x}_k}, \quad \underline{\Gamma}_k = \frac{\partial \underline{f}_k(\underline{\hat{x}}_k,0)}{\partial \underline{w}_k}$$

2. Linearization for the update step, $\underline{y}_k \in R^r$

$$\underline{y}_k = \underline{h}_k(\underline{\hat{x}}_k(-),0) + \frac{\partial \underline{h}_k(\underline{\hat{x}}_k(-),0)}{\partial \underline{x}_k}\delta\underline{x}_k + \frac{1}{2}\delta\underline{x}^T_k\frac{\partial^2 \underline{h}_k(\underline{\hat{x}}_k(-),0)}{\partial \underline{x}^2_k}\delta\underline{x}_k + \underline{v} + hot$$

Obviously the innovation term for $i$the measurement will have correction term

$$E\left\{\frac{1}{2}\delta\underline{x}^T_k\frac{\partial^2 h^i_k(\underline{\hat{x}}_k(-),0)}{\partial \underline{x}^2_k}\delta\underline{x}_k\right\} = \frac{1}{2}tr\left[\frac{\partial^2 h^i_k(\underline{\hat{x}}_k(-),0)}{\partial \underline{x}^2_k}\underline{P}_k(-)\right]$$

Therefore, the correction term also has to be included in the update of the covariance matrix.

Predicition step:

$$
\begin{aligned}
\underline{\hat{x}}_{k+1}(-) &= \underline{f}_k(\underline{\hat{x}}_k, 0) + \underline{b}_k \\
\underline{b}_k &= \frac{1}{2}\sum_{i=1}^{n} \underline{e}_i \left(tr\left[\underline{F}_k^i \underline{P}_k\right] + tr\left[\underline{G}_k^i Q_k\right]\right), \underline{e}_i \in R^n \\
\underline{P}_{k+1}(-) &= \underline{\Phi}_k \underline{P}_k \underline{\Phi}_k^T + \underline{\Gamma}_k \underline{Q}_k \underline{\Gamma}_k^T
\end{aligned}
$$

Update step:

$$
\begin{aligned}
[S_{k+1}]_{ij} &= \frac{1}{2} tr\left[\underline{M}_{k+1}^i \underline{P}_{k+1}(-) \underline{M}_{k+1}^j \underline{P}_{k+1}(-)\right] \\
\underline{P}_{k+1} &= \underline{P}_{k+1}(-) - \underline{P}_{k+1}(-)\underline{H}_{k+1}^T \left[\underline{H}_{k+1}\underline{P}_{k+1}(-)\underline{H}_{k+1}^T + \underline{R}_{k+1} + \underline{S}_{k+1}\right]^{-1} \underline{H}_{k+1}\underline{P}_{k+1}(-) \\
\underline{K}_{k+1} &= \underline{P}_{k+1}\underline{H}_{k+1}^T(\underline{R}_{k+1} + \underline{S}_{k+1})^{-1} \\
\underline{s}_{k+1} &= -\frac{1}{2} K_{k+1} \sum_{i=1}^{r} \underline{e}_i tr\left[\underline{M}_{k+1}^i \underline{P}_{k+1}(-)\right], \underline{e}_i \in R^r \\
\underline{\hat{x}}_{k+1} &= \underline{\hat{x}}_{k+1}(-) + \underline{K}_{k+1}(\underline{y}_{k+1} - \underline{h}_{k+1}(\underline{\hat{x}}_{k+1}(-)) + \underline{s}_{k+1}
\end{aligned}
$$

Part 5 Result

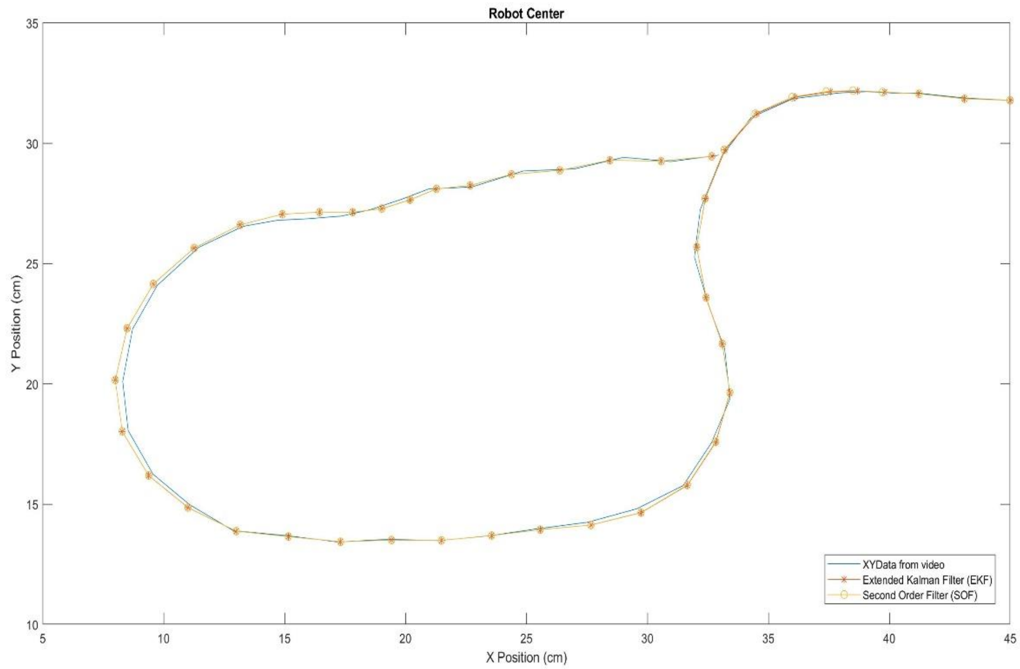


Figure 1. Robot Center

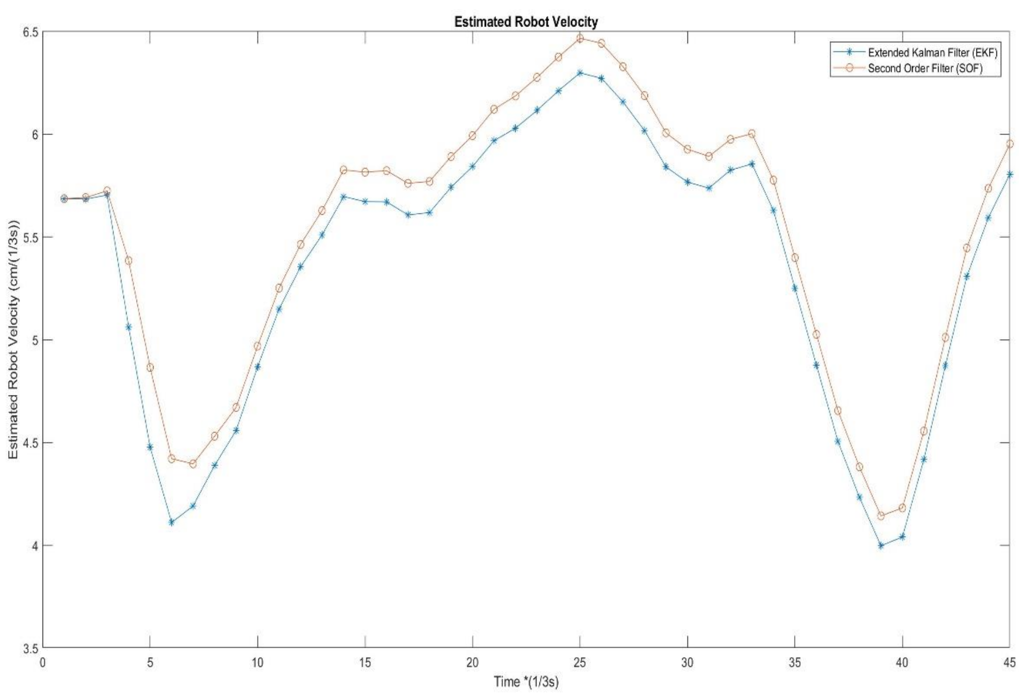


Figure 2. Estimated Robot Velocity

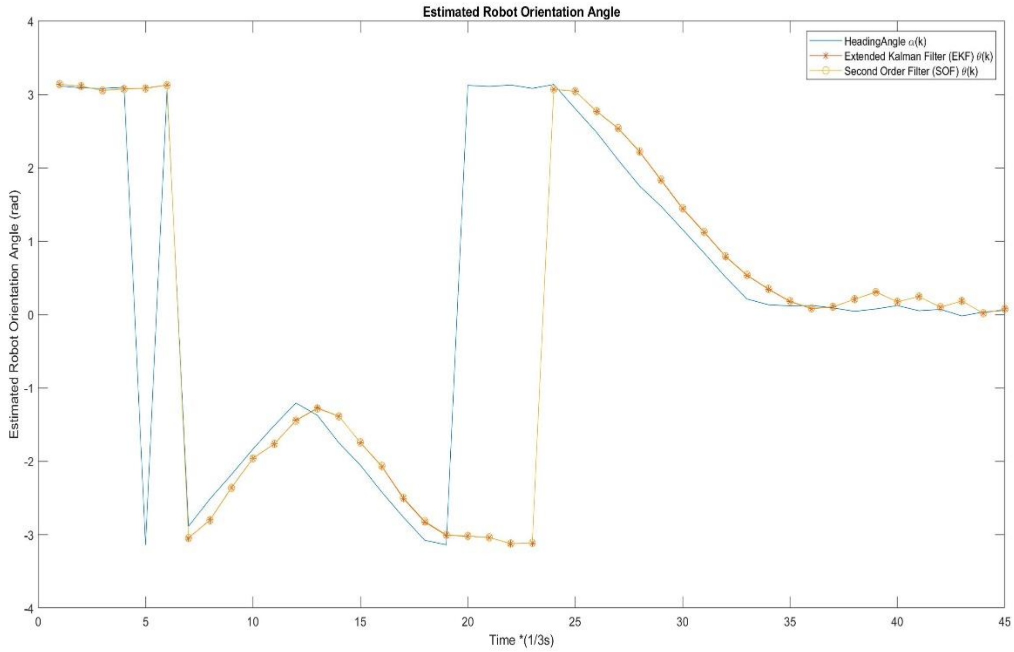


Figure 3. Estimated Robot Orientation Angle

From the plots, we can see that the EKF and second-order filter method have a very similar result; the most obvious difference is on estimated robot velocity.

References

[1] Milutinovic, D., ECE 245 Lecture Notes 2020 spring

[2] Milutinovic, D., ECE 245 homework 6 solution

[3] Arthur Gelb and The Analytic Sciences Corporation. 1974. Applied Optimal Estimation. The MIT Press.

```
%ECE245 KengyuLin Initial covariance matrices guess & measure video frame
clear;
clc;
XYdata = csvread('XYData_cm.csv');
HeadingAngle = csvread('HeadingAngle_rad.csv');
xm = XYdata(:,1);
ym = XYdata(:,2);
% find the initial variance of velocity var[v]
T=45;
dT=(1/3);
for i=2:length(xm)
    v(i)= sqrt(((ym(i)-ym(i-1))/dT)^2 + ((xm(i)-xm(i-1))/dT)^2);
end
for i=1:45
 if HeadingAngle(i)<-1
       HeadingAngle(i) = HeadingAngle(i)+2*pi;
 end
end
% find the initial variance of velocity var[v]
std_v = std(diff(v(5:35)))
var_v = (std_v)^2
% find the initial variance of theta var[theta]
std_theta=std(diff(HeadingAngle))
var_theta = (std_theta)^2
% measure video frame
% v = VideoReader('ProjectMovie.AVI');
% implay('ProjectMovie.AVI');
```

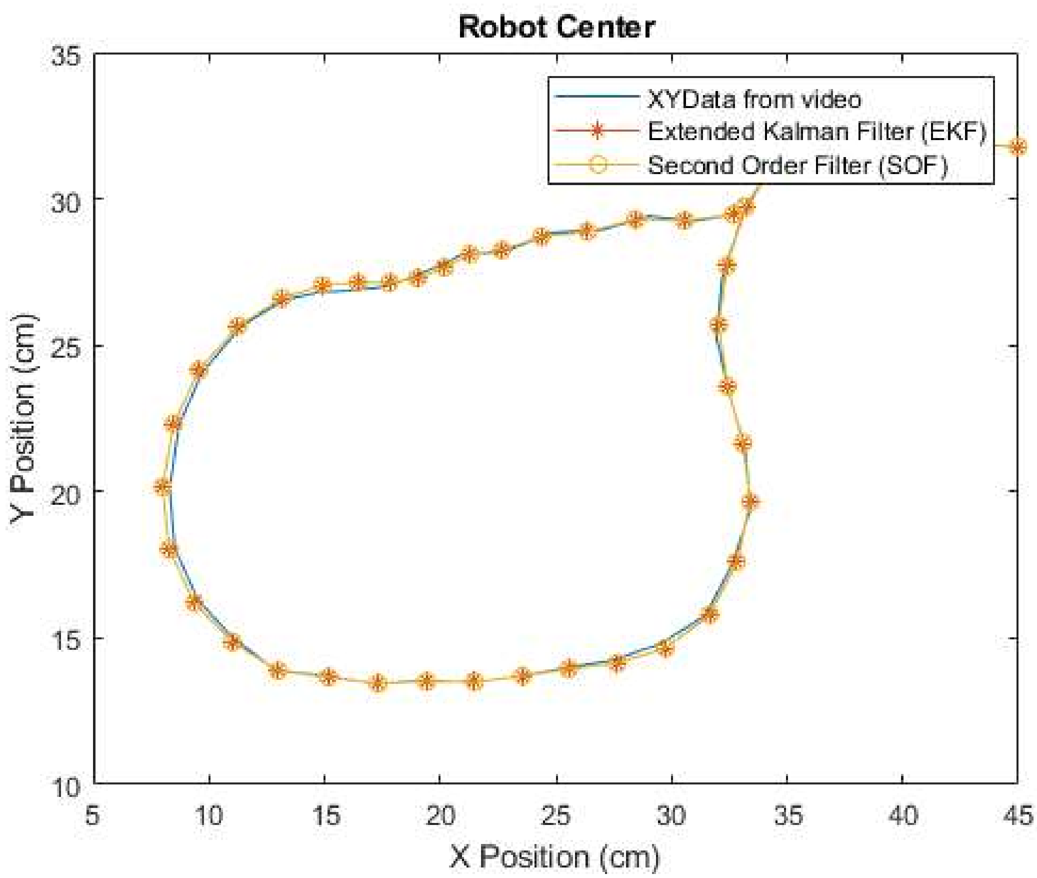
```
std_v =

    0.6984


var_v =

    0.4877


std_theta =

    0.2201


var_theta =

    0.0484
```

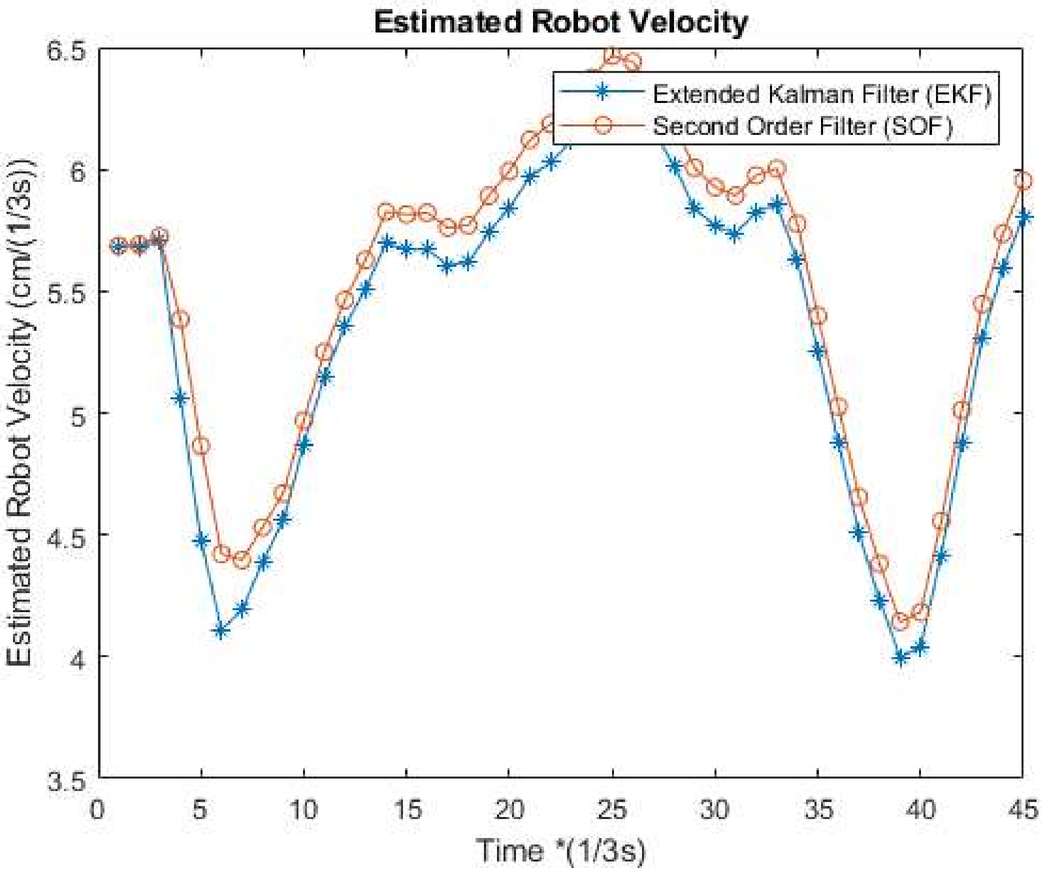
```
% ECE245Final Keng-yu Lin
clear;
clc;

XYdata = csvread('XYData_cm.csv');
HeadingAngle = csvread('HeadingAngle_rad.csv');
xm = XYdata(:,1);
ym = XYdata(:,2);

T=45;
dT=1/3;

v0=sqrt(((ym(2)-ym(1))/dT)^2 + ((xm(2)-xm(1))/dT)^2);  %initial velocity
x_ekf(:,1)=[xm(1);ym(1);v0;pi];                         %initial state
Wx=0.0675;
Wy=0.0675;
R=[Wx 0;0 Wy];

P(:,:,1)=[0.0675 0 0 0;
          0 0.0675 0 0;
          0 0 0.4877 0;
          0 0 0 0.0484];
phi(:,:,1) = [1 0 dT*cos(x_ekf(4,1)) -dT*x_ekf(3,1)*sin(x_ekf(4,1));
              0 1 dT*sin(x_ekf(4,1))  dT*x_ekf(3,1)*cos(x_ekf(4,1));
              0 0 1 0;
              0 0 0 1];
gamma =[0 0;
        0 0;
        0.25*sqrt(dT) 0;
        0 (20*pi/180)*sqrt(dT)];

Q = [1 0;0 1];
H = [1 0 0 0;0 1 0 0];




%%%=============================EKF==============================EKF
for i=1:1:T-1
  % Prediction
    xn(:,i+1) = x_ekf(:,i)+[x_ekf(3,i)*cos(x_ekf(4,i));
                        x_ekf(3,i)*sin(x_ekf(4,i));
                        0;
                        0]  *dT;

    phi(:,:,i) =  [1 0 dT*cos(x_ekf(4,i)) -dT*x_ekf(3,i)*sin(x_ekf(4,i));
                   0 1 dT*sin(x_ekf(4,i))  dT*x_ekf(3,i)*cos(x_ekf(4,i));
                   0 0 1 0;
                   0 0 0 1];

    Pn(:,:,i+1) = phi(:,:,i)*P(:,:,i)*phi(:,:,i)' + gamma*Q*gamma';

  % Gain
    K(:,:,i+1) = Pn(:,:,i+1)*H'*inv(H*Pn(:,:,i+1)*H' + R);
```

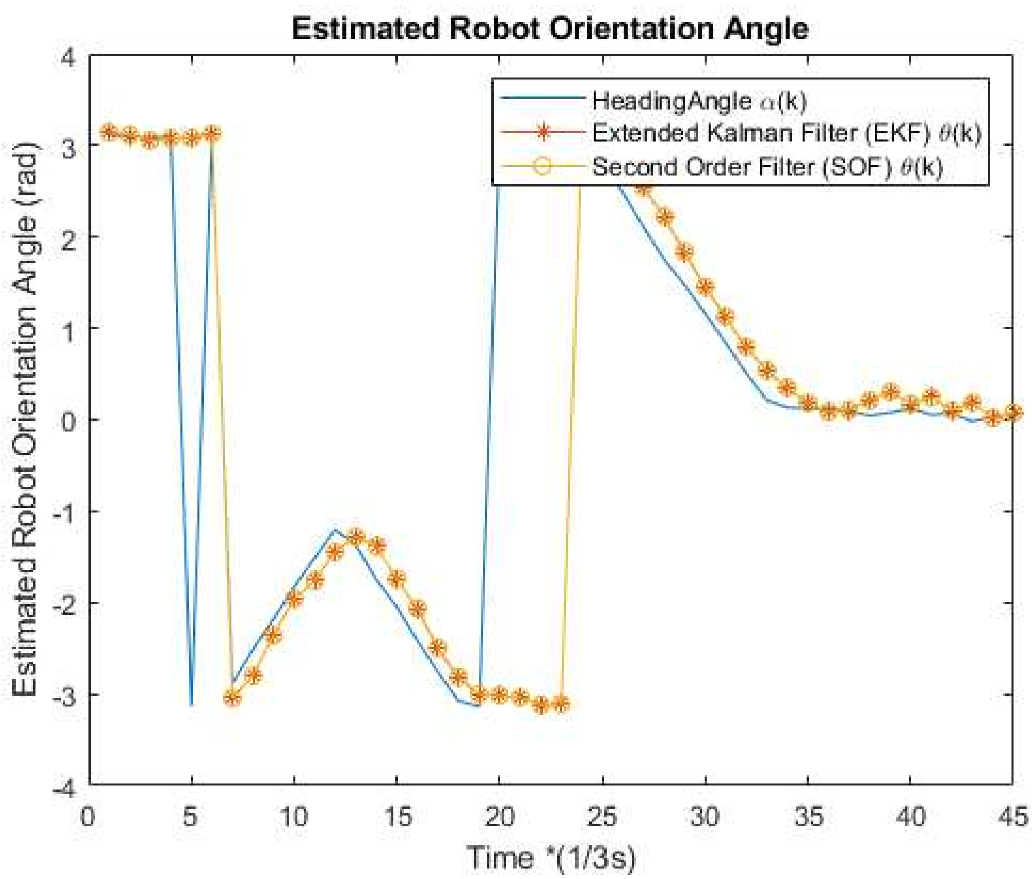
```
  % Update
    x_ekf(:,i+1) = xn(:,i+1) + K(:,:,i+1)*([xm(i+1);ym(i+1)] - H*xn(:,i+1));
    P(:,:,i+1) = (eye(4) - K(:,:,i+1)*H)*Pn(:,:,i+1);

  % To fix the angle theta(x(4,:)) in range (-pi,pi]
    if x_ekf(4,i)>pi
       x_ekf(4,i) = x_ekf(4,i)-2*pi;
    end
    if x_ekf(4,i)<-pi
       x_ekf(4,i) = x_ekf(4,i)+2*pi;
    end
end





%%%=============================SOF=============================SOF
x_sof=x_ekf(:,1);
P=[0.0387 0 0 0;
   0 0.0387 0 0;
   0 0 0.0698 0;
   0 0 0 0.03];

for k=2:45
  %F matrix
        F = zeros(4,4,4);
        F(:,:,1)= [0 0 0 0;
                   0 0 0 0;
                   0 0 0 -sin(x_sof(4,end))*dT;
                   0 0 -sin(x_sof(4,end))*dT, -x_sof(3,end)*cos(x_sof(4,end))*dT];
        F(:,:,2)= [0 0 0 0;
                   0 0 0 0;
                   0 0 0 cos(x_sof(4,end))*dT;
                   0 0 cos(x_sof(4,end))*dT, -x_sof(3,end)*sin(x_sof(4,end))*dT];
        F(:,:,3)= zeros(4,4);
        F(:,:,4)= zeros(4,4);

  %G matrix
        G = zeros(2,2,4);

  % Prediction
        b=0;
        e=eye(4);
            for i=1:4
                b = b + (e(:,i)*(trace(F(:,:,i)*P)+trace(G(:,:,i)*Q)));
            end
        b=b/2;

        x_sofn = [x_sof(1,end)+x_sof(3,end)*cos(x_sof(4,end))*dT;
                  x_sof(2,end)+x_sof(3,end)*sin(x_sof(4,end))*dT;
                  x_sof(3,end);
                  x_sof(4,end)]+b;

        phi = [1 0 cos(x_sofn(4))*dT, -x_sofn(3)*sin(x_sofn(4))*dT;
               0 1 sin(x_sofn(4))*dT,  x_sofn(3)*cos(x_sofn(4))*dT;
               0 0 1 0;
```

```
              0 0 0 1];
        Pn = phi*P*phi' + gamma*Q*gamma';

  %M matrix
        M = zeros(4,4,2);

  % Update
        S = zeros(2,2);
            for i=1:2
                for j=1:2
                    S(i,j)=1/2*trace(M(:,:,i)*Pn*M(:,:,j)*Pn);
                end
            end

        P = Pn - Pn*H'*inv(H*Pn*H'+R+S)*H*Pn;
        K = P*H'*inv(R+S);
        y = [xm(k) ym(k)]';
        h = [x_sofn(1);x_sofn(2)];

        s = 0;
        e = eye(2);
            for i=1:2
                s = s +e(:,i)*trace(M(:,:,i)*Pn);
            end
        s = -1/2*K*s;

        x_sof(:,end+1) = x_sofn + K*(y-h)+ s;

  % To fix the angle theta(x(4,:)) in range (-pi,pi]
        if x_sof(4,end)>pi
            x_sof(4,end) = x_sof(4,end)-2*pi;
        end
        if x_sof(4,end)<-pi
            x_sof(4,end) = x_sof(4,end)+2*pi;
        end
end


figure(1)
plot(xm,ym)
hold on
plot(x_ekf(1,:),x_ekf(2,:),'-*')
hold on
plot(x_sof(1,:),x_sof(2,:),'-o')
title('Robot Center')
xlabel('X Position (cm)');
ylabel('Y Position (cm)');
legend('XYData from video','Extended Kalman Filter (EKF)','Second Order Filter (SOF)');

figure(2)
plot(x_ekf(3,:),'-*')
hold on
plot(x_sof(3,:),'-o')
title('Estimated Robot Velocity')
xlabel('Time *(1/3s)');
ylabel('Estimated Robot Velocity (cm/(1/3s))');
legend('Extended Kalman Filter (EKF)','Second Order Filter (SOF)');
```

```
figure(3)
plot(HeadingAngle)
hold on
plot(x_ekf(4,:),'-*')
hold on
plot(x_sof(4,:),'-o')
title('Estimated Robot Orientation Angle')
xlabel('Time *(1/3s)');
ylabel('Estimated Robot Orientation Angle (rad)');
legend('HeadingAngle \alpha(k)','Extended Kalman Filter (EKF) \theta(k)','Second Order Fil
ter (SOF) \theta(k)');
```




Estimated Robot Velocity
Extended Kalman Filter (EKF)
Second Order Filter (SOF)
Estimated Robot Velocity (cm/(1/3s))
6.5
6
5.5
5
4.5
4
3.5
0
5
10
15
20
25
30
35
40
45
Time *(1/3s)


Estimated Robot Orientation Angle
HeadingAngle α(k)
Extended Kalman Filter (EKF) θ(k)
Second Order Filter (SOF) θ(k)
Estimated Robot Orientation Angle (rad)
4
3
2
1
0
-1
-2
-3
-4
0
5
10
15
20
25
30
35
40
45
Time *(1/3s)

Published with MATLAB® R2018b