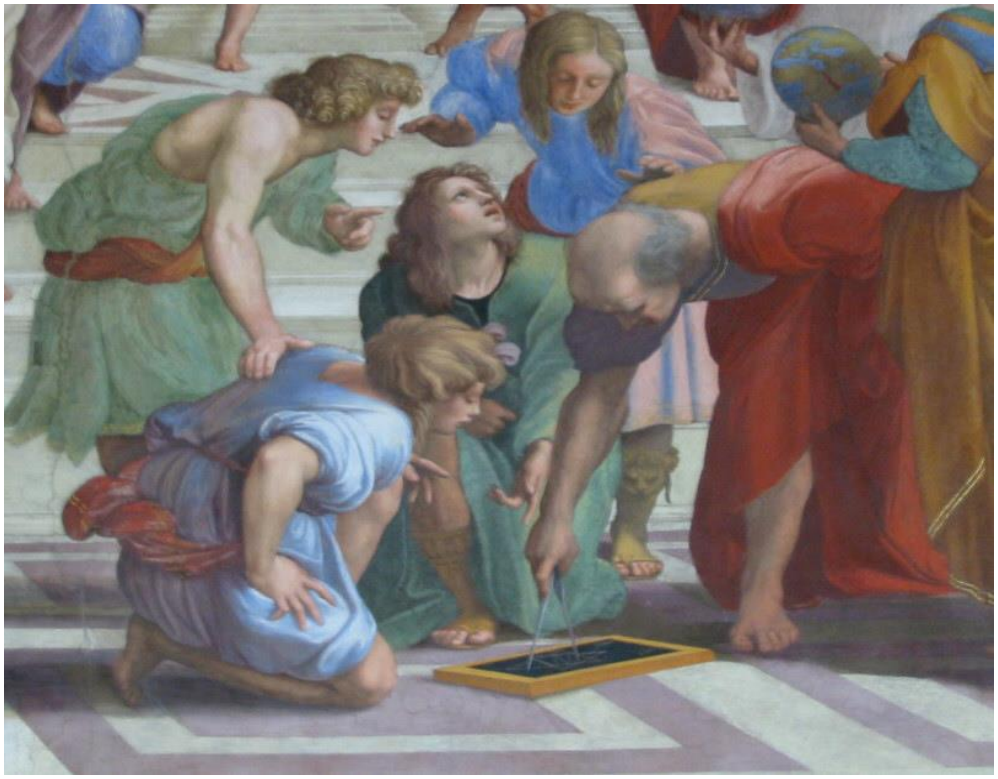
2D TRACKING KALMAN FILTER

A simple tutorial with Matlab



MUHAMMAD ALAHDAB

Table of Contents

1 Introduction

Let's assume we have an object that moves only on a plane, then its motion is defined completely by 3 variables: translation on the x-axis, translation on the y-axis, and a rotation by an angle θ around the z-axis. If we want to track the movement of this object in a specified time interval T in the plane, we must know its pose (x,y, θ) at every moment of time within the time interval T. We can measure the pose of this object at every instant of time. However; sensor's readings are usually noisy, and they can't give us an accurate value of the object's pose. One way to solve this problem is to use a Kalman filter to estimate the pose of the object at every time step in the time interval T.

2 Developing a model

To use Kalman filtering to track an object in a plane, we first need to model the movement of this object. We can't model accurately the object's movement, but we can have an acceptable approximation model of the object movement. Assuming that the motion on the x-axis is uncorrelated to the motion on the y-axis and the motion on both of the x-axis and y-axis are uncorrelated to the angular rotation around the z-axis, and by ignoring the jerk and all the higher derivatives of the pose, we can write the following discrete equations that describe the object's movements as shown below:

$$x(k+1) = x(k) + T_s v_x(k) + \frac{{T_s}^2}{2} a_x(k)$$

$$y(k+1) = y(k) + T_s v_y(k) + \frac{{T_s}^2}{2} a_y(k)$$

$$\theta(k+1) = \theta(k) + T_s \omega(k) + \frac{{T_s}^2}{2} \alpha(k)$$

$$v_x(k+1) = v_x(k) + T_s a_x(k)$$

$$v_y(k+1) = v_y(k) + T_s a_y(k)$$

$$\omega(k+1) = \omega(k) + T_s \alpha(k)$$

And we can write them as a state space model as following:

$$X(k+1) = \begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \\ v_x(k+1) \\ v_y(k+1) \\ \omega(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 & T_s & 0 \\ 0 & 0 & 1 & 0 & 0 & T_s \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x(k) \\ y(k) \\ \theta(k) \\ v_x(k) \\ v_y(k) \\ \omega(k) \end{bmatrix} + \begin{bmatrix} \frac{{T_s}^2}{2} & 0 & 0 \\ 0 & \frac{{T_s}^2}{2} & 0 \\ 0 & 0 & \frac{{T_s}^2}{2} \\ T_s & 0 & 0 \\ 0 & T_s & 0 \\ 0 & 0 & T_s \end{bmatrix} \begin{bmatrix} a_x(k) \\ a_y(k) \\ \alpha(k) \end{bmatrix}$$

And it can be written as:

$$X(k+1) = AX(k) + Bu(k)$$

to account for the uncertainty that results from the inaccuracy of the model or the inaccuracy of the values of the accelerations (the inputs), we introduce a white noise to the model:

$$X(k+1) = AX(k) + Bu(k) + w$$

assuming w is a Gaussian distribution noise with a mean 0 and a variance $Q = BQ_bB^T = B\begin{bmatrix} {\sigma_{a_x}}^2 & 0 & 0 \\ 0 & {\sigma_{a_y}}^2 & 0 \\ 0 & 0 & {\sigma_\alpha}^2 \end{bmatrix} B^T$. In practice, the value of Q is unknown, and we will have to estimate it. Notice also how the off-diagonal elements of Q_B are zeros, this is due to our assumption that the motions on the x-axis and the y-axis, and the rotation around the z-axis are uncorrelated.

Now, since we are measuring the x, y and θ, we can write:

$$Y(k+1) = CX(k+1) + v$$

where: $C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$ and v is the measurement noises that are introduced by the means of measurements. We assume the measurements noises as a Gaussian distribution with a mean of 0 and a variance $R = CC^TV = CC^T\begin{bmatrix} {\sigma_{z_x}}^2 & 0 & 0 \\ 0 & {\sigma_{z_y}}^2 & 0 \\ 0 & 0 & {\sigma_{z_\theta}}^2 \end{bmatrix}$.

3 Kalman filter

3.1 The Kalman filter algorithm

The Kalman filter has two main stages: Prediction stage, and a correction stage.

For the prediction state, we predict the state of the object as well as the covariance matrix (you can think of it as an indication of how well our estimation is, or as an estimation error). Before mentioning any equations, the (-) superscript indicates a predicted value and the (+) superscript indicates an estimated value. The prediction stage is illustrated by the following equations:

$$X^-(k+1) = AX^+(k) + Bu(k)\ predicting\ the\ state$$

$$P^-(k+1) = APA^{-1} + Q\ predicting\ the\ covariance\ matrix$$

We need to calculate the values of the Kalman gains Before moving to the correction stage:

$$K = P^-C^{-1}inv(CP^-C^{-1} + R)$$

where: $R = VCC^{-1}$.

The correction stage:

$$X^{+}(k+1) = X^{-}(k+1) + K[Y(k+1) - CX^{-}(k+1)]$$

And to estimate the covariance matrix:

$$P^{+} = (I - KC)P^{-}$$

At the beginning of the process, the Kalman filter must be given a correct initial state and an initial covariance matrix.

3.2 Some notes on the Kalman filter

Unlike other kinds of filters such as Markov filter, the Kalman filter requires us to provide it with a correct initial state of the object and a correct initial covariance. Therefore, if you can't provide an accurate initial pose and a covariance matrix for the Kalman filter, it will fail. This is considered a problem when dealing with an object that starts at a random unknown pose, or with an object which has a sudden and a great change in its pose, for example: someone carried the object away and put it in another place, this problem is known as the kidnapped robot problem.

4 Matlab Code for an example with results

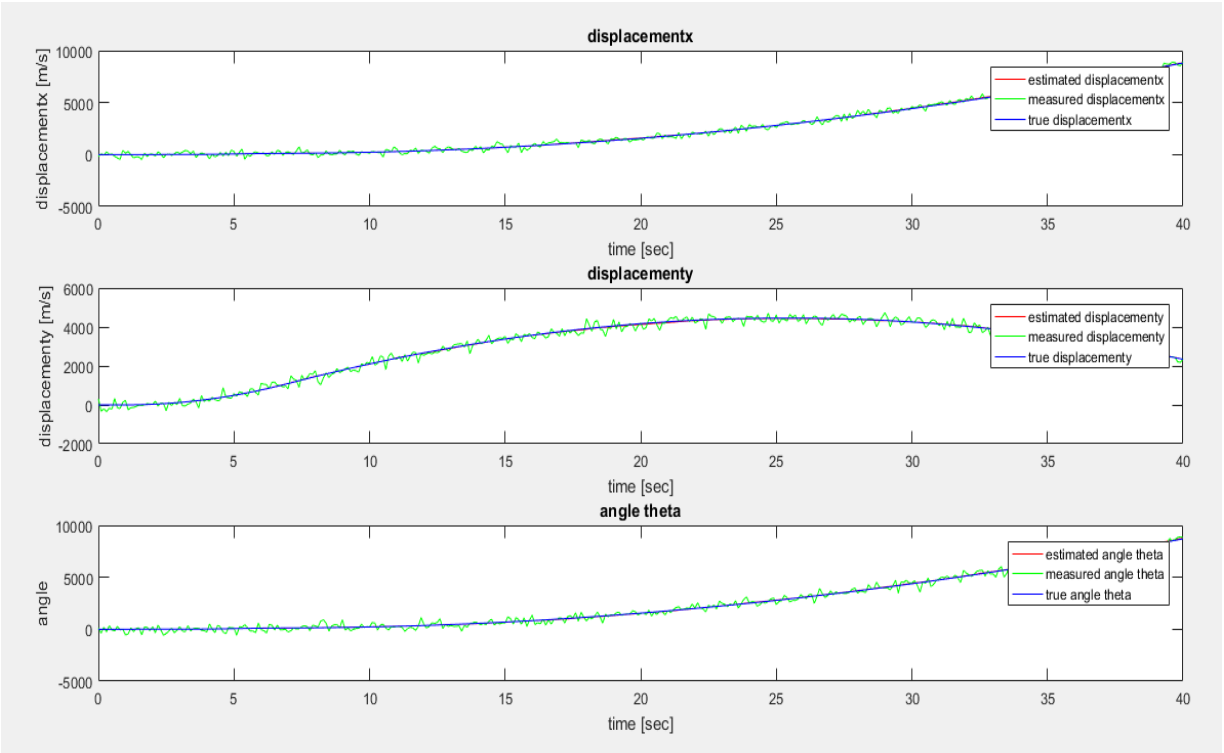
4.1 The code

```
Ts=0.1; %define the sample time
A=[1 0 0 Ts 0 0; 0 1 0 0 Ts 0; 0 0 1 0 0 Ts; 0 0 0 1 0 0
; 0 0 0 0 1 0; 0 0 0 0 0 1]; %define the state matrix
C=[1 0 0 0 0 0 ; 0 1 0 0 0 0 ; 0 0 1 0 0 0]; %define the
output matrix
B=[0.5*Ts^2 0 0;0 0.5*Ts^2 0;0 0 0.5*Ts^2;Ts 0 0;0 Ts 0;
0 0 Ts]; %define the input matrix
x0=[0;0;0;0;0;0]; %define the initial conditions
sys =ss(A,B,eye(6),[],Ts); %define a system to generate
true data
t=0:Ts:40; %define the time interval
%assuming that the uncertanities in the accelerations are
equal, we define
%them as follow:
segmaux=5; %standard deviation ax
segmauy=5; %standard deviation ay
segmaualpha=5; %standard deviation angular acceleration
%In practice, these values are determined experimentally.
%define the input(accelerations):
```

```
ux=[zeros(1,30) 25*ones(1,20) -20*ones(1,20)
15*ones(1,length(t)-70)]+normrnd(0,segmaux,1,length(t));
uy=[zeros(1,10) 60*ones(1,60) -20*ones(1,length(t)-
70)]+normrnd(0,segmauy,1,length(t));
ualpha=[zeros(1,30) 25*ones(1,20) -20*ones(1,20)
15*ones(1,length(t)-
70)]+normrnd(0,segmaualpha,1,length(t));
u=[ux;uy;ualpha];
%generating the true data:
Xtrue=lsim(sys,u,t,x0);
xtrue=Xtrue(:,1);
ytrue=Xtrue(:,2);
thtrue=Xtrue(:,3);
vxtrue=Xtrue(:,4);
vytrue=Xtrue(:,5);
wtrue=Xtrue(:,6);
%defining V:
measurmentsV=[200.^2 0 0; 0 200.^2 0; 0 0 300.^2];
%generating measurment data by adding noise to the true
data:
xm=xtrue+normrnd(0,200,length(xtrue),1);
ym=ytrue+normrnd(0,200,length(ytrue),1);
thm=thtrue+normrnd(0,300,length(ytrue),1);
%initializing the matricies for the for loop (this will
make the matlab run
%the for loop faster.
Xest=zeros(6,length(t));
Xest(:,1)=x0;
%defining R and Q
R=measurmentsV*C*C';
Q=[segmaux.^2 0 0 ; 0 segmauy.^2 0 ;0 0 segmaualpha.^2];
%Initializing P
P=B*Q*B';
for(i=2:1:length(t))
P=A*P*A'+B*Q*B'; %predicting P
Xest(:,i)=A*Xest(:,i-1)+B*u(:,i-1); %Predicitng the state
K=P*C'/(C*P*C'+R); %calculating the Kalman gains
Xest(:,i)=Xest(:,i)+K*([xm(i); ym(i); thm(i)]-
C*Xest(:,i)); %Correcting: estimating the state
P=(eye(6)-K*C)*P; %Correcting: estimating P
end
subplot(311)
%plot(t,Xest(2,:),'r',t,vtrue,'b')
%xlabel('time [sec]');
%ylabel('velocity [m/s]');
%title('Velocity');
%legend('estimated velocity','true velocity')
plot(t,Xest(1,:),'r',t,xm,'g',t,xtrue,'b')
xlabel('time [sec]');
ylabel('displacementx [m/s]');
title('displacementx');
```

```
legend('estimated displacementx','measured
displacementx','true displacementx');
subplot(312)
plot(t,Xest(2,:),'r',t,ym,'g',t,ytrue,'b')
xlabel('time [sec]');
ylabel('displacementy [m/s]');
title('displacementy');
legend('estimated displacementy','measured
displacementy','true displacementy');
t=0:0.1:40;
subplot(313)
plot(t,Xest(3,:),'r',t,thm,'g',t,thtrue,'b')
xlabel('time [sec]');
ylabel('angle');
title('angle theta');
legend('estimated angle theta','measured angle
theta','true angle theta');
t=0:0.1:40;
figure
hold on
%simple animation:
for i=1:1:length(t)
axis([min(xtrue)-500 max(xtrue)+500 min(ytrue)-500
max(ytrue)+500]);
%viscircles([xtrue(i) ytrue(i)],20,'color','b')
%viscircles([Xest(1,i) Xest(2,i)],20,'color','r')
plot(xtrue(i),ytrue(i),'bo');
plot(Xest(1,i),Xest(2,i),'rx');
pause(0.1)
end
```

4.2 Results


displacementx
displacementx [m/s]
estimated displacementx
measured displacementx
true displacementx
time [sec]
displacementy
displacementy [m/s]
estimated displacementy
measured displacementy
true displacementy
time [sec]
angle theta
angle
estimated angle theta
measured angle theta
true angle theta
time [sec]