EE615- Controls and Computation Lab

Experiment 3c :
Sensor Fusion with Kalman filter

Submitted by

Sayali Ravindra Duragkar
203070025
Abhishek Verma
213070024

Under the guidance of
Prof. Debraj Chakraborty

Department of Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

1 Objective

The objective of the experiment is to fuse accelerometer , magnetometer and gyroscope reading using 12 state Kalman Filter to get the attitudes and then to compare the results achieved with complimentary filter (in Exp 3b) and with ahrs filter also.

2 Error State Model

$$x_e(k) = A(k)x_e(k-1) + W(k)$$

$$\begin{bmatrix} \theta_e(k) \\ b_e(k) \\ S_{a_e(k)} \\ S_{d_e(k)} \end{bmatrix} = A(k) \begin{bmatrix} \theta_e(k-1) \\ b_e(k-1) \\ S_{a_e\,(k-1)} \\ S_{d_e\,(k-1)} \end{bmatrix} + w(k)$$

Where:-
$\theta_e(k)$ - Orientation error (deg.)
$b_e(k)$ - Gyro offset (deg/sec)
$S_{a_e(k)}$ - Acceleration error in sensor frame (units - g)
$S_{d_e(k)}$ - Magnetic disturbance error vector in sensor frame (micro - T)

$$A(k) = 0 \Rightarrow \hat{x_e}^-(k) = E(w(k)) = 0$$

3 Measurement Model

1. Gravity vector is estimated from both accerelometer and gyroscope.
2. The Geomagnetic vector is estimated from magnetometer and gyroscope.

$$S_{ze}(k) = \begin{bmatrix} S_{g_A(k) - S_{g_G(k)}} \\ S_{m_M(k) - S_{m_G(k)}} \end{bmatrix} = C(k)x_e(k) + v(k)$$

$$= C(k) \begin{bmatrix} \theta_e(k) \\ b_e(k) \\ S_{a_e(k)} \\ S_{d_e(k)} \end{bmatrix} + v(k)$$

The covariance matrices of the noise processes are defined as:-

$$Q_w(k) = E(w(k)w(k)^T)$$
$$Q_v(k) = E(v(k)v(k)^T)$$

4 Kalman Filter Equations

Since A(K) = 0 , $P(k)^-$ can be calculated as :-

$$P(k)^- = A(k)P_k - 1^+ + A(k)^T + Q_w(k)$$

$$P(k)^- = Q_w(k)$$

and the kalman gain is given by:-

$$K(k) = Q_w(k)\, C(k)^T [C(k)\, Q_w(k)\, C(k)^T + Q_v(k)]^{-1}$$

4.1 Update equations

$$\hat{x_e^+}(k) = \hat{x_e^-}(k) + K(k)[S_{ze}(k) - C(k)\hat{x_e^-}(k)]$$

Now $\hat{x_e^-}(k) = 0 \Rightarrow \hat{x_e^+}(k) = K(k)S_{z_e(k)}$

Posterior Error Covariance:-

$$P^+(k) = [I - K(k)\, C(k)]\, P^-(k) = [I - K(k)\, C(k)]\, Q_w(k)$$

Where C(k) is given by:-

$$\boldsymbol{C_k} = \begin{pmatrix} 0 & \alpha^S \hat{g}^-_{Gz,k} & -\alpha^S \hat{g}^-_{Gy,k} & 0 & -\alpha\delta t^S \hat{g}^-_{Gz,k} 0 & \alpha\delta t^S \hat{g}^-_{Gy,k} & 1 & 0 & 0 & 0 & 0 & 0 \\ -\alpha^S \hat{g}^-_{Gz,k} & 0 & \alpha^S \hat{g}^-_{Gx,k} & \alpha\delta t^S \hat{g}^-_{Gz,k} & 0 & -\alpha\delta t^S \hat{g}^-_{Gx,k} & 0 & 1 & 0 & 0 & 0 & 0 \\ \alpha^S \hat{g}^-_{Gy,k} & -\alpha^S \hat{g}^-_{Gx,k} & 0 & -\alpha\delta t^S \hat{g}^-_{Gy,k} & \alpha\delta t^S \hat{g}^-_{Gx,k} & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & \alpha^S \hat{m}^-_{Gz,k} & -\alpha^S \hat{m}^-_{Gy,k} & 0 & -\alpha\delta t^S \hat{m}^-_{Gz,k} & \alpha\delta t^S \hat{m}^-_{Gy,k} & 0 & 0 & 0 & -1 & 0 & 0 \\ -\alpha^S \hat{m}^-_{Gz,k} & 0 & \alpha^S \hat{m}^-_{Gx,k} & \alpha\delta t^S \hat{m}^-_{Gz,k} & 0 & -\alpha\delta t^S \hat{m}^-_{Gx,k} & 0 & 0 & 0 & 0 & -1 & 0 \\ \alpha^S \hat{m}^-_{Gy,k} & -\alpha^S \hat{m}^-_{Gx,k} & 0 & -\alpha\delta t^S \hat{m}^-_{Gy,k} & \alpha\delta t^S \hat{m}^-_{Gx,k} & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

Figure 1: 6*12 C(k) Matrix

5 Code

```
 %Kalman Filter
duration = 1000; % seconds
fs = 10;         % Hz
N = duration*fs;
timeVector = (0:(totalNumSamples-1))/fs;
alpha = 3.14/180;
t = 0.0185;
Pk = eye(12,12);
Qv = 0.64*ones(6,6);
vk = 0.1*[1;1;1;1;1;1];
state = [0;0;0;0;0;0;0;0;0;0;0;0];
pitch_hat_kalman = zeros(1, N);
roll_hat_kalman = zeros(1, N);
yaw_hat_kalman = zeros(1, N);

for i =1:N

        wx=gyroReadings(i,1);
        wy=gyroReadings(i,2);
        wz=gyroReadings(i,3);

        bax = accelReadings(i,1);
        bay = accelReadings(i,2);
        baz = accelReadings(i,3);

        bmx = magReadings(i,1);
        bmy = magReadings(i,2);
        bmz = magReadings(i,3);


        Ck = [0 alpha*wz -alpha*wy 0 -alpha*t*wz alpha*t*wy 1 0 0 0 0 0; -alpha*wz 0 alpha*w
            alpha*bmy -alpha*bmx 0 -alpha*t*bmy alpha*t*bmx 0 0 0 0 0 0 -1];

        Kk1 = Pk*(Ck')/(Ck*Pk*(Ck')+ Qv);
        state = state + Kk1*([bax-wx;bay-wy;baz-wz;bmx-wx;bmy-wy;bmz-wz]-Ck*state);
        %state = state + Kk1*vk;
        Pk = (eye(12) - Kk1*Ck)* Pk;

        pitch_hat_kalman(i) = state(2);
        roll_hat_kalman(i) = state(1);
        yaw_hat_kalman(i) = state(3);
end
decim = 1;
fuse = ahrsfilter('SampleRate',fs,'DecimationFactor',decim);
```

```
orientation = fuse(accelReadings,gyroReadings,magReadings);

orientationEuler = eulerd(orientation,'ZYX','frame');

figure
plot(timeVector,orientationEuler,...
     timeVector,yaw_hat_kalman(:),...
     timeVector,roll_hat_kalman(:),...
     timeVector,pitch_hat_kalman(:))
 axis([0,duration,-180,180])
 legend('Yaw','Pitch','Roll')
 xlabel('Time (s)')
 ylabel('Rotation (degrees)')
 title('Kalman filter')
```

6 Result

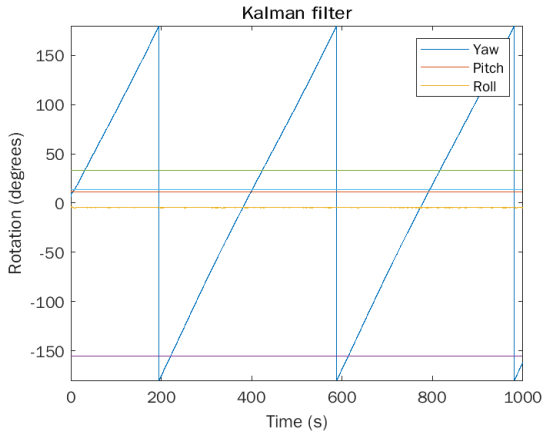

Figure 2: Output of Kalman and AHRS Filter