EE615- Controls and Computation Lab

Experiment 3b :
Sensor Fusion with complementary 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 this experiment is to calculate roll and pitch values from accelerometer and yaw values from magnetometer along with the gyroscope values and use complimentary filter to estimate final values of roll , pitch and yaw.

2 Estimating roll and Pitch values from accelerometer

First step that we do is go to NED frame from ENU frame.This involves exchanging first column with second column and first row with second row.Then we balance the acceleration equation that we are getting from accelerometer readings.
First two equation will result in

$$b_{a_x} \approx g sin(p)$$

$$b_{a_y} \approx -g cos(p)\, sin(r)$$

Solving these we get roll and pitch from accelerometer , i.e,

$$P_a = sin^{-1}(b_{a_x}/g)$$

$$R_a = -sin^{-1}(b_{a_y}/\, g cos(p))$$

3 Estimating yaw values from magnetometer

In first step what we do is take b_m (magnetometer reading vector) and take its projection on local level horizontal plane , i.e. , to put y=0 in R_b^l matrix and solve for:-

$$L_{p_m} = R_b^l b_m \;\; at \;\; y=0.$$

Solving this we'll get yaw to be:-

$$y_m = tan^{-1} \frac{Lp_{m_y}}{Lp_{m_x}}$$

4 Complimentary Filter

We pass roll and pitch values calculated from accelerometer from a low pass filter and roll and pitch (estimated from gyroscope) through high pass filter and add them to get estimated roll and pitch.
Now we pass the yaw value calculated from magnetometer from a low pass filter and yaw value (estimated from gyroscope) through high pass filter and add them to get estimated yaw.
This type of combination to get estimated yaw, roll and pitch is called complimentary filter.

5 Code

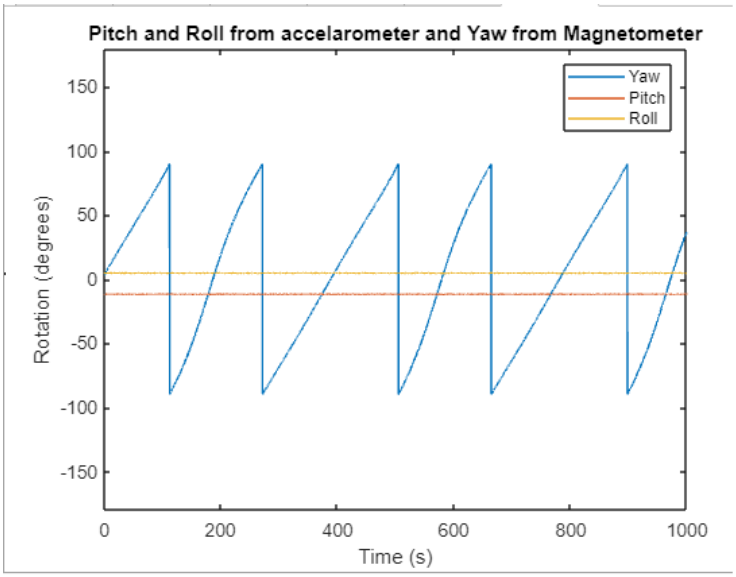
```
%%
orientation=zeros(N,3);

in1=0;                      %roll
in2=pitch*(3.14/180);       %pitch
in3=initialYaw*(3.14/180); %yaw

eul_init=[in1 in2 in3];
%%computation is in quaternion
quat_init = eul2quat(eul_init,'ZYX');

% orient_measu is vector of euler angle
orientation(1,1)=in1*(180/3.14);
orientation(1,2)=in2*(180/3.14);
orientation(1,3)=in3*(180/3.14);

i1=quat_init(1);
i2=quat_init(2);
i3=quat_init(3);
i4=quat_init(4);


for i=1:N

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

vdp2 = @(t,x)[0.5*(wz*x(2)-wy*x(3)+wx*x(4));0.5*(-wz*x(1)+wx*x(3)+wy*x(4));0.5*(wy*x(1)-wx*x

[t,x]=ode45(vdp2,[0,time],[i1,i2,i3,i4]);
```

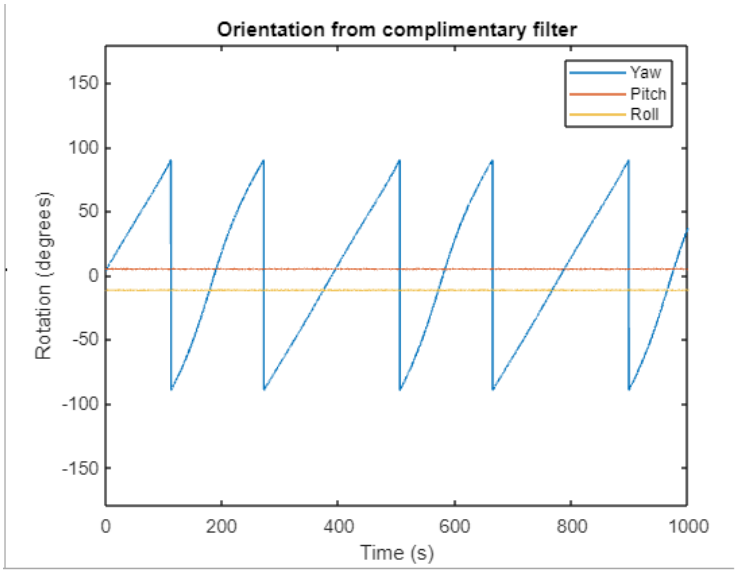
```
i1=x(length(t),1);
i2=x(length(t),2);
i3=x(length(t),3);
i4=x(length(t),4);
quat_variable=[i1 i2 i3 i4];
%eul_variable = quat2eul(quat_variable,'ZYX');
[yaw, pitch, roll] = quat2angle(quat_variable);
% orient_measu(i+1,3)=i1*(180/3.14);%roll
% orient_measu(i+1,2)=i2*(180/3.14);
% orient_measu(i+1,1)=i3*(180/3.14);%orient_measu(i,1):yaw
%

orientation(i+1,1)=roll*(-180/3.14);
orientation(i+1,2)=pitch*(180/3.14);
orientation(i+1,3)=yaw*(180/3.14);
end
%
%
length(eulerAngles(:,2))
length(timeVector)
figure
plot(timeVector,orientation(1:10000,1),...
     timeVector,orientation(1:10000,2),...
     timeVector,orientation(1:10000,3))
 axis([0,duration,-180,180])
 legend('Est_Yaw (Rotation Around Down)','Est_Pitch (Rotation Around East)','Est_Roll (Rotat
 xlabel('Time (s)')
 ylabel('Rotation (degrees)')
 title('Orientatio due to Quaternion Computation')
 pause
%%
pitch_roll = zeros(N,2);
g = 9.81;
ym = zeros(N,1);
time1 = (0:(totalNumSamples-1))/fs;
for i = 1:N
    bax = accelReadings(i,1);
    bay = accelReadings(i,2);
    pa = asin(bax/g);
    ra = -(asin(bay/(g*cos(pa))));
    pitch_roll(i,1) = pa*180/pi;
    pitch_roll(i,2) = ra*180/pi;
end
for j = 1:N
    bmx = magReadings(j,1);
    bmy = magReadings(j,2);
```

```
    bmz = magReadings(j,3);
    p = pitch_roll(j,1);
    r = pitch_roll(j,2);
    Lpmx = ((bmx*cosd(p))+(bmy*sind(r)*sind(p))+(bmz*sind(p)*cosd(r)));
    Lpmy = ((bmy*cosd(r))-(bmz*sind(r)));
    ym1 = atan(Lpmy/Lpmx);
    ym(j,1) = (ym1)*(-180/pi);
end
figure
plot(time1, ym(:,1),...
     time1, pitch_roll(:,1),...
     time1, pitch_roll(:,2))
axis([0,duration,-180,180])
legend('Yaw','Pitch','Roll');
xlabel('Time (s)');
ylabel('Rotation (degrees)');
title('Pitch and Roll from accelarometer and Yaw from Magnetometer');
pause
%%
FinalPitch = zeros(N,1);
FinalRoll = zeros(N,1);
FinalYaw = zeros(N,1);

for i = 1:N

    lp = lowpass(pitch_roll(i,1),15,fs);
    lr = lowpass(pitch_roll(i,2),15,fs);
    ly = lowpass(ym(i),15,fs);

    hp = highpass(orientation(i,2),15,fs);
    hr = highpass(orientation(i,3),15,fs);
    hy = highpass(orientation(i,1),15,fs);

    fp = lp + hp;
    fr = lr + hr;
    fy = ly + hy;

    FinalPitch(i) = fp;
    FinalRoll(i) = fr;
    FinalYaw(i) = fy;

end

figure
plot(timeVector,FinalYaw(:),...
     timeVector,FinalRoll(:),...
```

```
    timeVector,FinalPitch(:))
axis([0,duration,-180,180])
legend('Yaw','Pitch','Roll');
xlabel('Time (s)');
ylabel('Rotation (degrees)');
title('Orientation from complimentary filter');
```

6 Result


Pitch and Roll from accelarometer and Yaw from Magnetometer
Yaw
Pitch
Roll
150
100
50
0
-50
-100
-150
Rotation (degrees)
0
200
400
600
800
1000
Time (s)


Orientation from complimentary filter
Yaw
Pitch
Roll
150
100
50
0
-50
-100
-150
Rotation (degrees)
0
200
400
600
800
1000
Time (s)