# SENSOR FUSION USING COMPLEMENTARY FILTER IN MATLAB

V ADARSH (21307R001)
PRADEEP CHUDASAMA (213070111)

## 1 Introduction

In this experiment sensor fusion was achieved in MATLAB using complementary filter. An inertial measurement unit(IMU) was used, which was connected to Arduino for interfacing with PC. IMU's accelerometer and gyroscope readings were fused to get an accurate reading of pitch and roll angles to estimate orientation of IMU. The pitch action is a rotation of IMU about it's y-axis and roll action is rotation of IMU about it's x-axis.

## 2 Need of sensor fusion

IMU contains 3 sensors: accelerometer, gyroscope and magnetometer. In this experiment only accelerometer and gyroscope were used.
An accelerometer, as the name suggests, gives the acceleration acting on the IMU along x,y,z axes of it's Body frame. If the IMU is lying flat on a table with it's x-y plane parallel to ground and z-axis downwards, then the only acceleration acting on it is the acceleration due to gravity, i.e $9.8m/s^2$. Hence the reading will be [0,0,9.8]. But if the IMU sensor is tilted at some angle, then still the only acceleration acting on it would be 'g' but this 'g' would not be along it's z-axis due to tilt and the output of acceleration is just the vector coordinates of this gravity vector in it's tilted body frame. This tilt is just a series of yaw-pitch-roll maneuvers of the frame lying flat on table. The output of accelerometer in tilted frame is just the vector one would get after the vector [0,0,g] goes through a series of yaw-pitch-roll rotations. Since we have the 2 vectors we can determine the rotations needed which are nothing but the pitch and roll angles. And hence by this way one can determine roll/pitch angles from accelerometer.
A gyroscope outputs the angular velocity with which IMU is rotating about it's 3 axes. One can integrate this with respect to time to estimate angular position of the 3 axes over time.
The problem comes from the observation that both the sensor on it's own would not give a reliable estimate of orientation of IMU. For example, in case of gyroscope, it's readings have low noise but HIGH BIAS. The bias here means, if say

there is no rotation about x axis of IMU but gyroscope still outputs some small random value, then this reading is an offset/bias from the true reading of $0°/s$, and since we are integrating the readings, over a longer period of time this bias starts accumulating and hence causing an error.
In case of accelerometer, to find orientation from it's readings we assume the only acceleration acting on it is 'g' but in practice that's not the case since there always exists some small movements and hence the readings of accelerometer is a resultant of all the accelerations acting on it. This results in a jittery behaviour. But it does not suffer the bias problem since we always use gravity vector to estimate orientation and gravity always points downward, even in long duration it does not deviate from true orientation but just shows jittery behaviour about this orientation.
This is why we need to fuse both gyroscope and accelerometer readings to use the best of both. This fusion is achieved using Complementary Filter in this experiment.

# 3 Complementary Filter

We refer to Complementary Filter as CF from here on-wards. As discussed earlier, gyro readings have low noise and large drift/bias and accelerometer readings have low drift but large noise, which here can be modelled as high frequency noise. CF combines both these readings using a high pass and low pass filter and hence the name Complementary Filter.
In s-domain, let

$w_y(s)$ = angular velocity of IMU with respect to it's y-axis obtained from gyroscope

Then the pitch angle can be estimated by integrating the above angular velocity,

$\frac{w_y(s)}{s} = \theta(s) + \frac{c}{s}$

Here in RHS $\theta(s)$ is the true pitch angle and 'c' is the bias/offset/drift from it. The integration acts as an Low pass filter and we can see the term $\frac{c}{s}$ has high magnitude at lower frequencies and hence to attenuate this effect we pass the gyroscope readings through a High pass filter.
On the other hand, let

$\theta_a(s)$ = pitch angle obtained from accelerometer. then,

$\theta_a(s) = \theta(s) + n_\theta(s)$

where $n_\theta(s)$ is the high frequency noise present in accelerometer readings. We pass accelerometer readings through low pass filter to attenuate this noise.

let $\theta_{CF}(s)$ = pitch angle obtained using CF.

$$\theta_{CF}(s) = \frac{\theta_s(s)}{\tau s+1} + \frac{\tau s}{\tau s+1}\frac{w_y(s)}{s} \qquad (1)$$

using previous 2 equations:

$$\theta_{CF}(s) = \theta(s) + (\frac{n_\theta(s)}{\tau s+1} + \frac{\tau s}{\tau s+1}(\frac{c}{s}))$$

The above equation indicates that high frequency noise $n_\theta(s)$ can be attenuated to nearly 0 by passing through LPF and low frequency drift $\frac{c}{s}$ can be eliminated by passing thorugh HPF and as a result $\theta_{CF}(s) \approx \theta(s)$.
We can represent this process in block diagram as: To implement this is MAT-

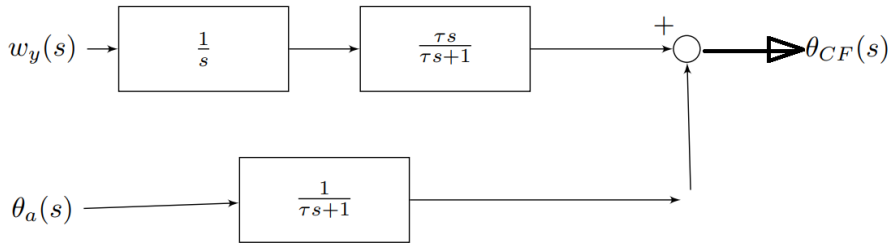


Figure 1: Block diagram of CF

LAB we need the discrete version of this, and hence we can consider z-domanin. Replacing s in equation (1) with $\frac{1-z^{-1}}{T_s}$, where $T_s$ is the sampling period, we get:

$$\theta_{CF}(z) = \frac{\theta_s(z)}{\tau\frac{1-z^{-1}}{T_s}+1} + \frac{\tau}{\tau\frac{1-z^{-1}}{T_s}+1}w_y(z)$$

$$\Rightarrow \theta_{CF}(z) = \frac{T_s\theta_a(z)}{\tau(1-z^{-1})+T_s} + \frac{\tau T_s w_y(z)}{\tau(1-z^{-1})+T_s}$$

$$\Rightarrow \theta_{CF}(z)[\tau - \frac{\tau}{z} + T_s] = T_s\theta_a(z) + \tau T_s w_y(z)$$

$$\Rightarrow \theta_{CF}(z)[\tau + T_s] = T_s\theta_a(z) + \tau T_s w_y(z) + \frac{\tau\theta_{CF}(z)}{z}$$

Now we convert this to discrete domain, and observing $\frac{1}{z}$ as a unit delay, we get:

$$\theta_{CF}(k) = \frac{T_s}{\tau+T_s}\theta_a(k) + \frac{\tau}{\tau+T_s}(\theta_{CF}(k-1) + T_s w_y(k))$$

let $\frac{T_s}{\tau+T_s} = \alpha$, then $\frac{\tau}{\tau+T_s} = 1-\alpha = \beta$. Then we get the final formula we

can use while implementing in MATLAB as:

$$\theta_{CF}(k) = \alpha\, \theta_a(k) + \beta\, (\theta_{CF}(k-1) + T_s w_y(k)) \qquad (2)$$

# 4 Estimating Roll and Pitch angles from Accelerometer and Gyroscope

The roll is rotation of IMU about it's x-axis and pitch is rotation of IMU about it's y-axis.

## 4.1 Gyroscope

We initialize initial orientation vector as $[0, 0, 0]^T$ which means initially IMU is lying flat on table, and pitch,roll, yaw are all 0. Now, after starting the MATLAB program and as we tilt IMU gyroscope gives readings which correspond to angular velocity with which IMU is rotating about it's 3 axes. To get the current pitch,roll,yaw we can simply multiply those readings with sampling time and add that value to the previous pitch,roll,yaw angles which we must have found using the CF. Let the readings be $[w_x, w_y, w_z]$ at $k^{th}$ time instant, and $f_s$ be sampling frequency. To find pitch angle at $k^{th}$ time instant($\theta_{gyro}(k)$),:

$$\theta_{gyro}(k) = \theta_{gyro}(k-1) + \frac{w_y}{f_s}$$

## 4.2 Accelerometer

Any rotation of a rigid body can be expressed as composition of 3 elemental rotations:

- YAW: rotation of coordinate frame about z-axis by an angle $\psi$
- PITCH: rotation of coordinate frame about y-axis by an angle $\theta$
- ROLL: rotation of coordinate frame about x-axis by an angle $\phi$

So to reach any orientation, a sequence YAW-PITCH-ROLL maneuver is enough. If we have the initial and final orientations, these 3 Euler angles can be determined.
The 3 rotation matrices for the 3 Euler angles are given by:

- $R_z(\psi) = \begin{bmatrix} cos\psi & sin\psi & 0 \\ -sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- $R_y(\theta) = \begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix}$
- $R_z(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{bmatrix}$

So assuming NED frame (North-east-down) for x-y-z axes of IMU when it is lying flat, and letting Body frame (BF) be the current tilted frame of IMU, if one wants to go from NED to BF we can multiply the above 3 matrices to get the final rotation matrix as:

$$R_{NED}^{BF} = R_z(\phi)R_y(\theta)R_z(\psi) \tag{3}$$

$$R_{NED}^{BF} = \begin{bmatrix} cos\theta cos\psi & cos\theta sin\psi & -sin\theta \\ cos\psi sin\theta sin\phi - sin\psi cos\phi & sin\psi sin\theta sin\phi + cos\psi cos\phi & sin\phi cos\theta \\ cos\psi sin\theta cos\phi + sin\psi sin\phi & sin\psi sin\theta cos\phi - cos\psi sin\phi & cos\phi cos\theta \end{bmatrix}$$

In the beginning, when accelerometer is lying flat aligned to NED frame the readings will be $[0, 0, -g]^T$. At some time instant k, let the accelerometer readings in BF be $[a_x\ a_y\ a_z]^T$. Then the following holds:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = R_{NED}^{BF} \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}$$

Using the expression of $R_{NED}^{BF}$ in (3) we get pitch and roll angles from accelerometer as:

$\theta_a = sin^{-1}[\frac{a_x}{g}]$

$\phi_a = -sin^{-1}[\frac{a_y}{gcos\theta_a}]$

### 4.3 Euler angles to quaternion

We can represent the rotation matrix $R_{NED}^{BF}$ in quaternion form. To represent a rotation by an angle $a$ about a vector $v$, the following unit quaternion can be used:

$$q = \begin{bmatrix} cos(\frac{a}{2}) \\ vsin(\frac{a}{2}) \end{bmatrix}$$

Then to represent roll,pitch,yaw by quaternions:

$$q_x(\phi) = \begin{bmatrix} cos\frac{\phi}{2} \\ sin\frac{\phi}{2} \\ 0 \\ 0 \end{bmatrix}$$

$$q_y(\theta) = \begin{bmatrix} cos\frac{\theta}{2} \\ 0 \\ \sin\frac{\theta}{2} \\ 0 \end{bmatrix}$$

$$q_z(\psi) = \begin{bmatrix} cos\frac{\psi}{2} \\ 0 \\ 0 \\ sin\frac{\psi}{2} \end{bmatrix}$$

$$\mathrm{q}_{NED}^{BF} = q_z(\psi) \otimes q_y(\theta) \otimes q_x(\phi)$$

# 5 MATLAB CODE

```
%———— section – 1: Interfacing IMU6050 sensor with Arduino Mega250 ————
clc;
clear all;
g=-9.8;
% a = arduino(); %Update the name of communication port
a = arduino('COM5', 'Mega2560', 'Libraries', 'I2C');
fs = 20; % Sample Rate in Hz
imu = mpu6050(a,'SampleRate',fs,'OutputFormat','matrix');
%%
%———— section – 2: Reading the realtime data from IMU6050 sensor ————
decim = 1;
duration = 1; % seconds
fs = 20;          % Hz
N = duration*fs;
i1=0;
t=(0:((N*10)-1))./fs;
accelR=[];
gyroR=[];

```

```
openExample('shared_fusion_arduinoio/
    EstimateOrientationUsingInertialSensorFusionAndMPU9250Example
    ')
viewer_imuf = HelperOrientationViewer('Title',{'
    Visualization of imuf_Orientation'})
viewer_CF = HelperOrientationViewer('Title',{'
    Visualization of CF_Orientation'})

orientation_CF = zeros(N*10,1,'quaternion');
angles_prev=zeros(10,3);
while i1<N

    [accelReadings, gyroReadings] = read(imu);
    i1=i1+1;
    accelR = [accelR;accelReadings];
    gyroR = [gyroR;gyroReadings];

    accel_pitch=(asin(accelReadings(:,1)/g));
    %accel_roll=atan2(accelReadings(:,2),accelReadings
        (:,3)); %% use this for full 180 deg roll
    accel_roll=-(asin(accelReadings(:,2)./(g*cos(
        accel_pitch)))); %%%[-90,90]

    gyro_pitch=gyroReadings(:,2)*(1/fs)+angles_prev(1,1);
    gyro_roll=gyroReadings(:,1)*(1/fs)+angles_prev(1,2);
    gyro_yaw=gyroReadings(:,3)*(1/fs)+angles_prev(1,3);

    accel_pitch_roll=0.98*[accel_pitch,accel_roll];
    gyro_pitch_roll=0.02*[gyro_pitch,gyro_roll];
    res_ag=accel_pitch_roll+gyro_pitch_roll;

    res=[res_ag,res_ag,zeros(10,1)];
    angle=res;
    angle_prev=angle;
    A=zeros(size(res,1),1);
    B=zeros(size(res,1),1);
    C=zeros(size(res,1),1);
    D=zeros(size(res,1),1);

    for i=1:size(res,1)
        v=res(i,:);
        q1=[cos(v(1,3)/2),0,0,sin(v(1,3)/2)];
        q2=[cos(v(1,1)/2),0,sin(v(1,1)/2),0];
        q3=[cos(v(1,2)/2),sin(v(1,2)/2),0,0];
        qmul=quatmultiply(q1,quatmultiply(q2,q3));
        A(i,1)=qmul(1,1);
```

```
60          B(i,1)=qmul(1,2);
61          C(i,1)=qmul(1,3);
62          D(i,1)=qmul(1,4);
63
64      end
65
66      q=quaternion(A,B,C,D); % orientation found by CF
67
68      % euler 321: yaw(z)-pitch(y)-roll(x)
69
70      orientation_CF(1+10*(i1-1):i1*10)=(q);
71      fuse = imufilter('SampleRate',fs,'DecimationFactor',
            decim);
72      orientation = fuse(accelR,gyroR);
73
74      % 3D figure or Sensor
75      for j =1: 10%numel(orientation)
76         %viewer_imuf(orientation(j));
77         viewer_CF(q(j));
78      end
79
80      orientationEuler = eulerd(orientation,'ZYX','frame');
81  end
82
83  N=N*10;
84  timeVector = (0:(N-1))/fs;
85  %3D curve
86  figure
87  plot3(orientationEuler(:,1),orientationEuler(:,2),
        orientationEuler(:,3))
88  legend('Z-axis','Y-axis','X-axis')
89  xlabel('Time (s)')
90  ylabel('Rotation (degrees)')
91  title('Estimated Orientation')
92  disp(size(orientationEuler(:,1:2)));
93  disp((eulerd(orientation_CF,'ZYX','frame')));
94  euler_orientation_CF=eulerd(orientation_CF,'ZYX','frame')
        ;
95  figure(5)
96  plot3(euler_orientation_CF(:,1),euler_orientation_CF(:,2)
        ,euler_orientation_CF(:,3))
97  legend('Z-axis','Y-axis','X-axis')
98  xlabel('Time (s)')
99  ylabel('Rotation (degrees)')
100 title('Estimated Orientation')
101 figure(3)
```

```
plot(timeVector,orientationEuler(:,[2 3]));
xlabel('Time(s)')
ylabel('Rotation(deg)')
title('orientation ---IMUFILTER');
legend('Y-axis(PITCH)','X-axis(ROLL)');
timeVector = (0:(N-1))/fs;
figure(4)
plot(timeVector,euler_orientation_CF(:,[1 3]));
xlabel('Time(s)')
ylabel('Rotation(deg)')
title('orientation ---complementary filter');
legend('Y-axis(PITCH)','X-axis(ROLL)');
```

# 6 Results

In this section, the plots of roll and pitch angles over a time of 10 seconds are presented for different values of $\alpha$ and $\beta$ (refer eq(2)). For comparison, we use the MATLAB's inbuilt IMUFILTER. The Left plot in each figure corresponds to roll and pitch angles estimated by the CF implemented by us in MATLAB while the right plot corresponds to the same angles estimated by MATLAB's inbuilt IMUFILTER.

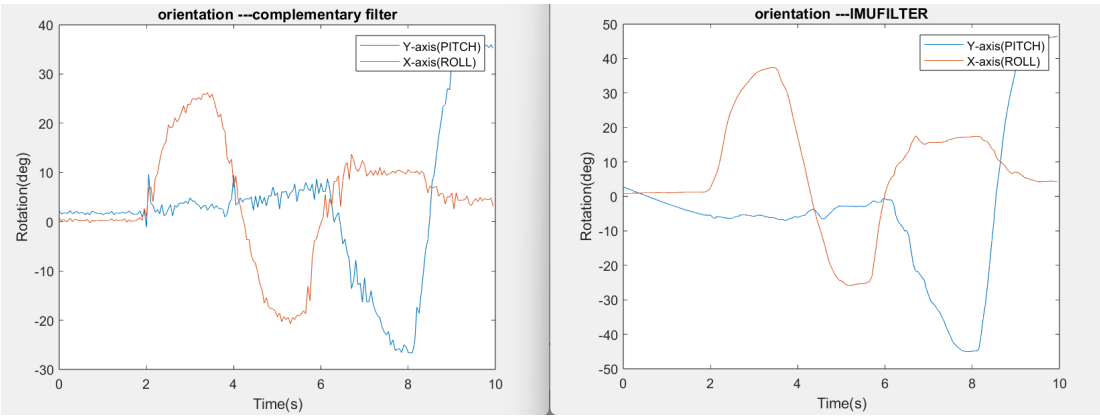


Figure 2: roll,pitch plot: $\alpha = 0.7$ and $\beta = 0.3$

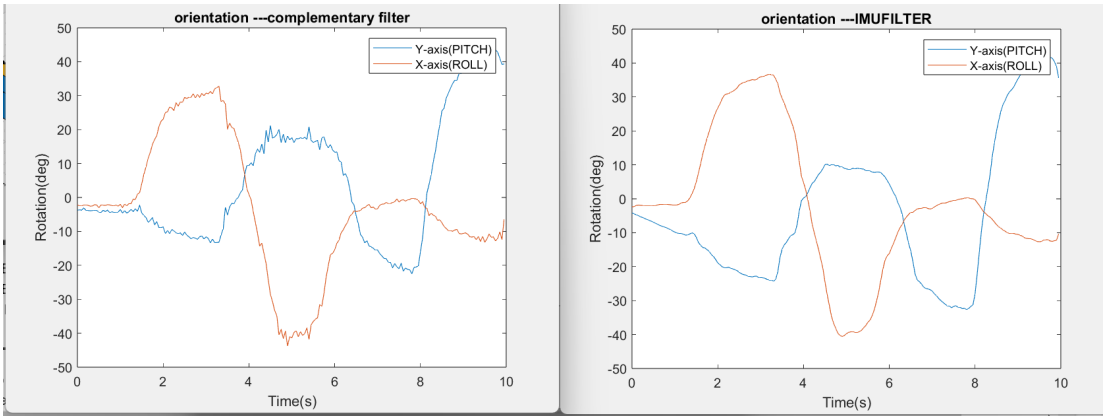


Figure 3: roll,pitch plot: $\alpha = 0.95$ and $\beta = 0.05$

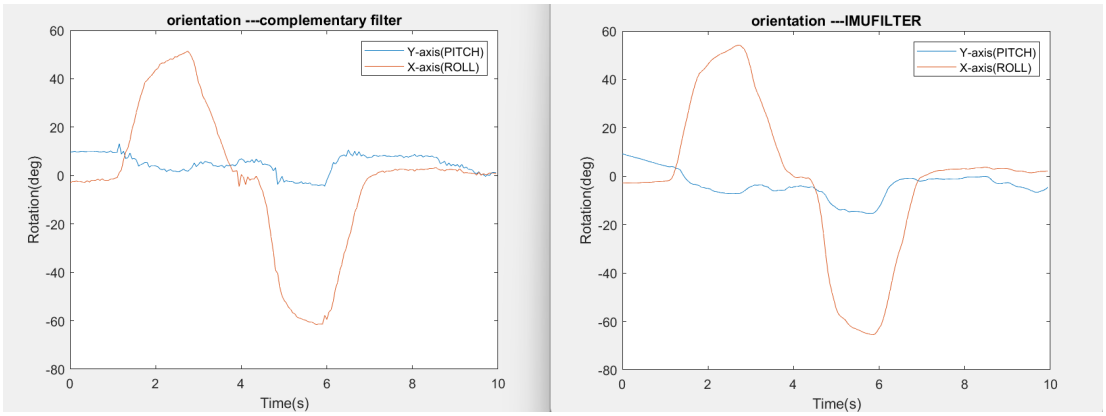


Figure 4: roll,pitch plot: $\alpha = 0.98$ and $\beta = 0.02$

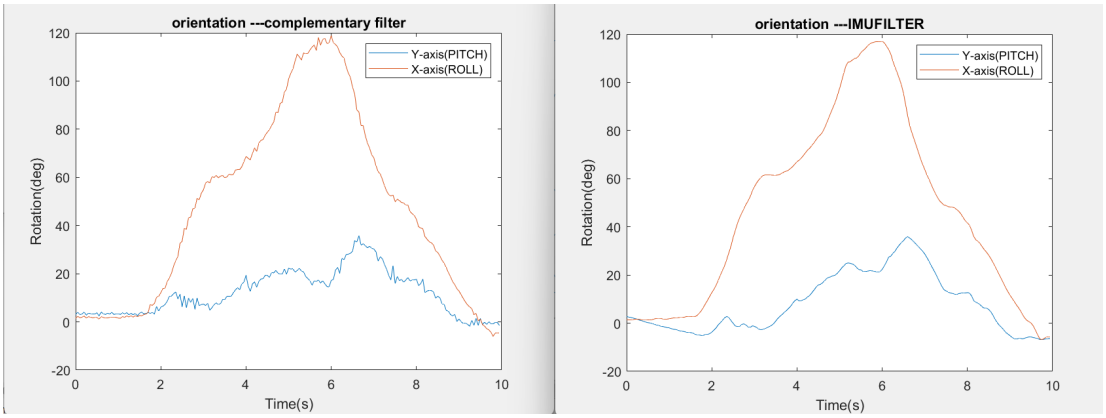


Figure 5: roll,pitch plot: $\alpha = 0.98$ and $\beta = 0.02$

From the above plots it can be seen that we get best results for $\alpha > 0.9$. Equation (2) can be seen as which reading among the accelerometer and gyroscope we believe more and belief is indicated by $\alpha, \beta$.

# 7 References

1. Quan, Quan. Introduction to multicopter design and control. Singapore: Springer, 2017.

2. Brian Doughlas. *Drone Control and the Complementary Filter* .Youtube.2018. URL:"`https://www.youtube.com/watch?v=whSw42XddsU`"