EE 615 - Control & Computing Lab

Experiment - 3A
Complementry Filter for Attitude Estimation

Submitted by

Mayank Soni & Ankit Singh
213070028 & 213070029

Contents

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 COMPLEMENTARY FILTER

The complementary filter is a computationally inexpensive sensor fusion technique that consists of a low-pass and a high-pass filter. Idea behind complementary filter is to take slow moving signals from accelerometer, magnetometer and fast moving signals from a gyroscope and combine them. When there are multiple sensors such as accelerometer and gyroscope, a complementary filter can perform a low-pass filtering on one sensor value and a highpass filtering on the other to integrate and produce a better output than the raw sensor values.

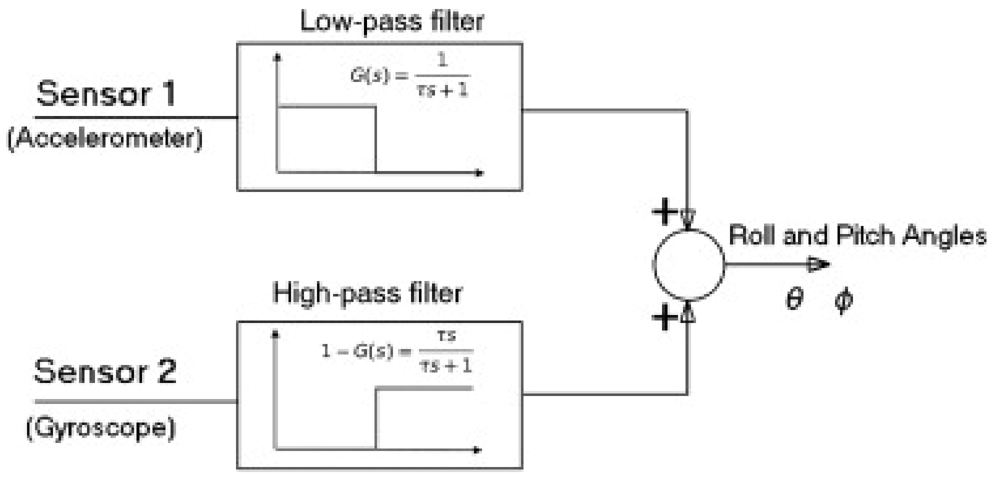


Figure 1: **Complimentary Filter**

3 IMPLEMENTATION

3.1 MATLAB CODE

```
clear all;
% a = arduino(); %Update the name of communication port
a = arduino('COM3', '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 = 5; % seconds
fs = 20;          % Hz
N = duration*fs;
i=0;

accelR=[];
gyroR=[];

openExample('shared_fusion_arduinoio/EstimateOrientationUsingInertialSensorFus
viewer = HelperOrientationViewer('Title',{'Visualization of Orientation'})

while i<N

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

    fuse = imufilter('SampleRate',fs,'DecimationFactor',decim);
    orientation = fuse(accelR,gyroR);

    % 3D figure or Sensor
    for j = numel(orientation)
       viewer(orientation(j));
    end

    orientationEuler = eulerd(orientation,'ZYX','frame');
    % Complementary filter

    P_acc_low = lowpass(accelR(:,1),5,fs);
    R_acc_low = lowpass(accelR(:,2),5,fs);
    Ym = lowpass(accelR(:,3),5,fs);

    Pg_high = highpass(gyroR(:,1),5,fs);
    Rg_high = highpass(gyroR(:,2),5,fs);
    Yg_high = highpass(gyroR(:,3),5,fs);

    P_total = P_acc_low + Pg_high;
    R_total = R_acc_low + Rg_high;
    Y_total = Ym + Yg_high;

    %P_total = P_t * 52.9;
    %R_total = R_t * 52.9;
    %Y_total = Y_t * 52.9;

end

N=N*10;
timeVector = (0:(N-1))/fs;
figure
subplot(2,1,1)
plot(timeVector,accelR)
legend('X-axis','Y-axis','Z-axis')
ylabel('Acceleration (m/s^2)')
title('Accelerometer Readings')

subplot(2,1,2)
plot(timeVector,gyroR)
legend('X-axis','Y-axis','Z-axis')
ylabel('Angular Velocity (rad/s')
xlabel('Time (s)')
title('Gyroscope Readings')


%3D curve
figure
```

```
plot3(orientationEuler(:,1),orientationEuler(:,2),orientationEuler(:,3))
legend('Z-axis','Y-axis','X-axis')
xlabel('Time (s)')
ylabel('Rotation (degrees)')
title('Estimated Orientation')



figure
plot(timeVector,orientationEuler(:,1), ...
    timeVector, orientationEuler(:,2), ...
   timeVector, orientationEuler(:,3));

legend('Yaw', 'Pitch', 'Roll')
xlabel('Time')
ylabel('Rotation')
title('Estimated')


figure
plot(timeVector, Y_total, ...
    timeVector, P_total, ...
    timeVector, R_total);

legend('Yaw', 'Pitch', 'Roll')
xlabel('Time')
ylabel('Rotation')
title('Complementary Filter')
```

3.2 Result

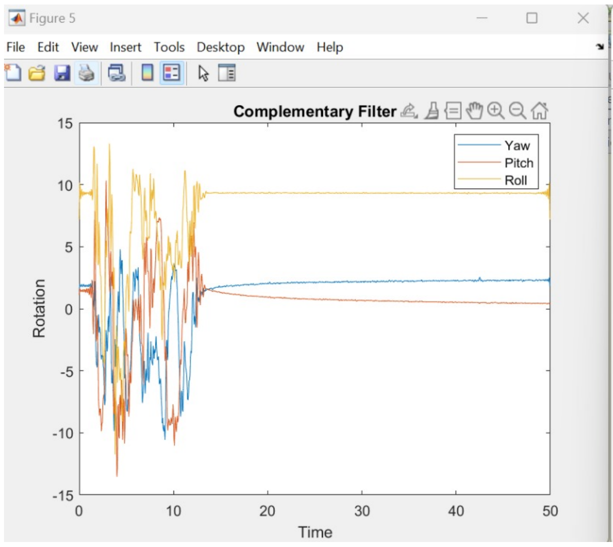


Figure 5
File Edit View Insert Tools Desktop Window Help
Complementary Filter
Yaw
Pitch
Roll
Rotation
Time

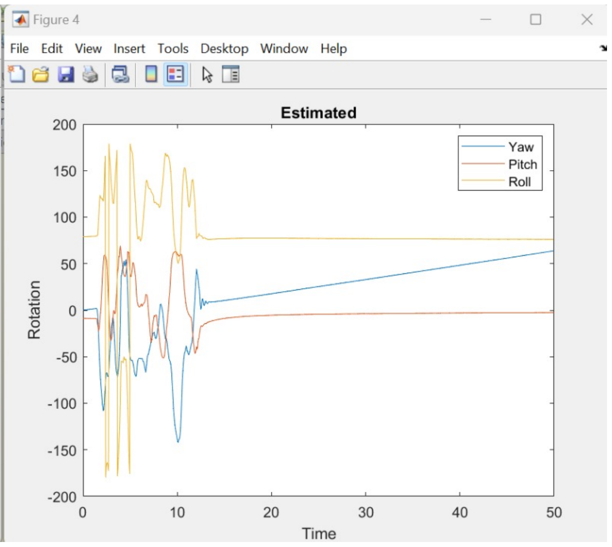
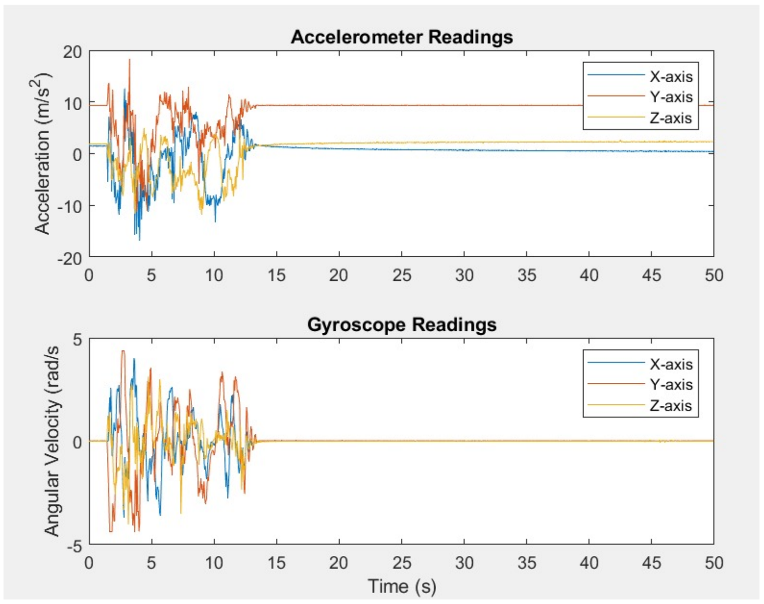
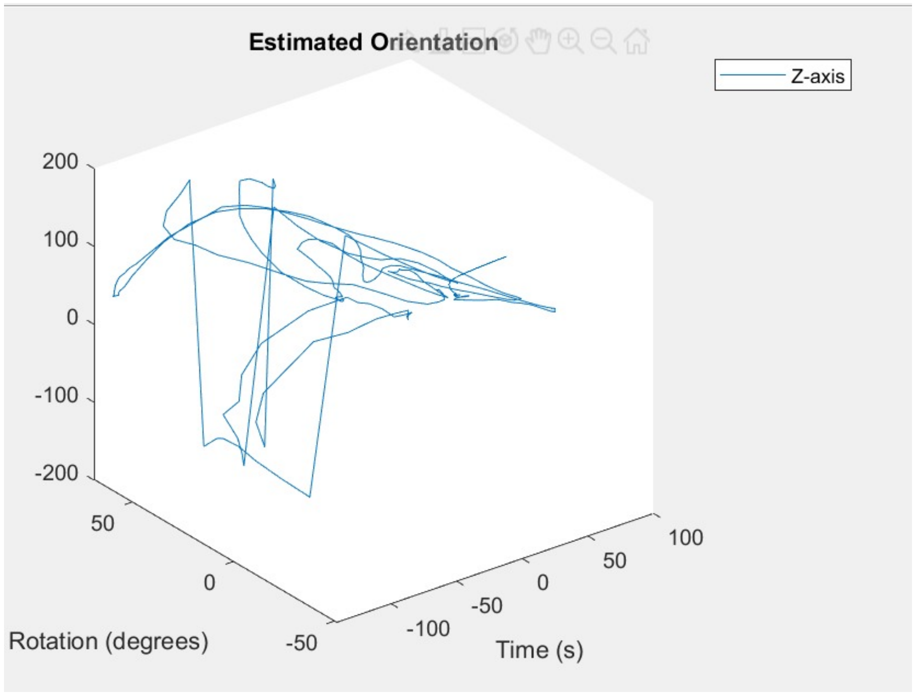


Figure 4
File Edit View Insert Tools Desktop Window Help
Estimated
Yaw
Pitch
Roll
Rotation
Time


Accelerometer Readings
X-axis
Y-axis
Z-axis
Acceleration (m/s^2)
Gyroscope Readings
X-axis
Y-axis
Z-axis
Angular Velocity (rad/s
Time (s)


Estimated Orientation
Z-axis
200
100
0
-100
-200
50
0
-50
Rotation (degrees)
-100
-50
0
50
100
Time (s)