EE615: Experiment 3
Complementary Filter

Shubham dey, 22M1084
Bhagat Ram Labana, 22M1083

April 4, 2023

Contents

1 Introduction

A complementary filter was proposed by Shane Colton in 2007. For tilt sensing, the filter performs low-pass filtering on low-frequency tilt estimation and the data is from the accelerometer while the high-pass filtering on biased highfrequency tilt estimation is handled by directly integrating with the gyroscope output. The fusion of the two estimations gives an all-pass estimation of the orientation. Obviously, the complementary filter takes the advantage of both accelerometer and gyroscope. On the short term, it uses the data from the gyroscope and the data is precise and not susceptible to external forces; on the long term, it relies on the data from the accelerometer to prevent data drift.

2 Methodology

Our aim is to do the attitude estimation of the to estimate the attitude in the form of Euler angles , rotation matrix or the quarternions. Attitute information is retrieved by the comlementary filter on the data obtained by accelerometer and gyroscope.

- accelerometer
  using accelerometer we want to calculate the angle pitch , roll and yaw. Euler angles are used to know the orientation of the object in fixed coordinate system. from accelerometer we measure the linear accelearation in x, y and z direction as Ax, Ay and Az respectively. roll and pitch is calculated from the the given expression.

$$roll = \tan^{-1}(\frac{Ay}{Az}) \tag{1}$$

$$pitch = \sin^{-1}(\frac{Ax}{g}) \tag{2}$$

- gyroscope
  gyroscope is used to calculate the angular velocity in the direction of x , y and z direction which can be integrated over time to estimate changes in orientation.

- complementary filter
  to combine the benefits of both accelerometers and gyroscope , a complementary filter uses the weighted average of gyroscope and accelerometer measurements to estimate the sensors orientation . The weight given to the gyroscope measurements depends on the estimated error in the gyroscope measurements, while the weight given to the accelerometer measurements. gyroscope values are used in the comlementary filter to estimate the short term orientation of a sensor while the accelerometer values are used to estimate the long term orientation of the sensor.the complementary filter combines these measurement to provide a more accurate estimate of the sensors orientation. Here is the following diagram to implement the complementary filter.

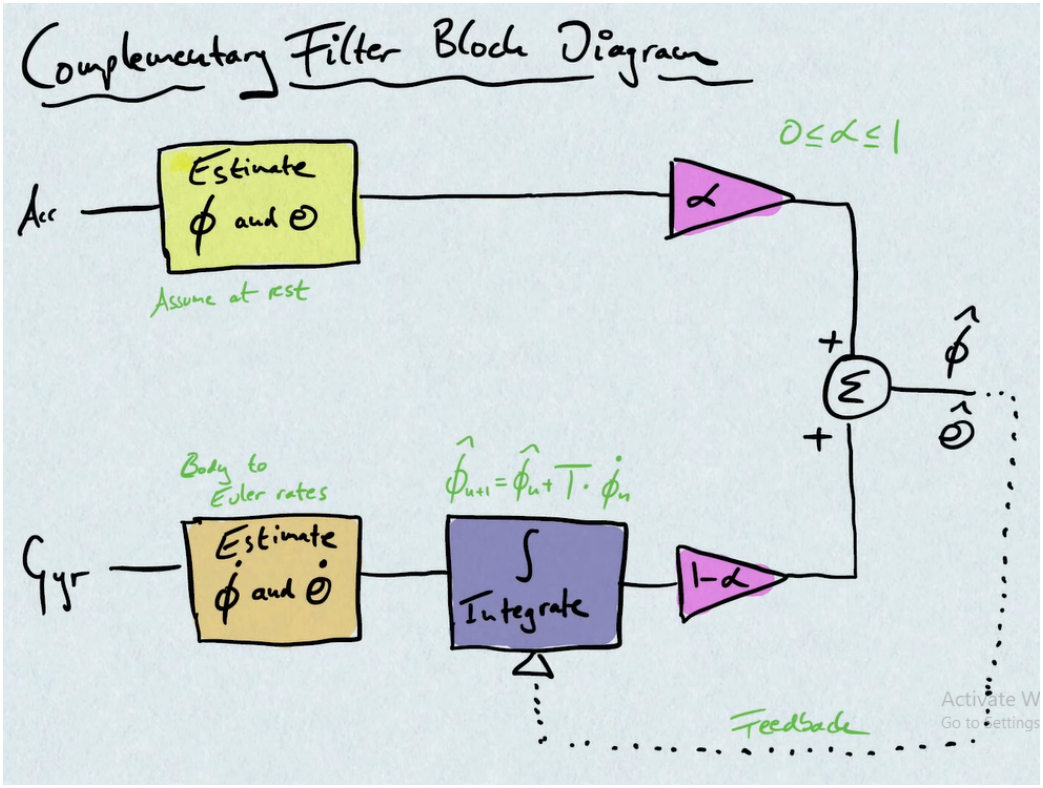

Figure 1: Algorithm for complementary filter

Express to estimate the angle using complementary filter is gieven as

$$\widehat{\Phi}_{n+1} = (\widehat{\Phi}_{acc,n}).\alpha + (1-\alpha).[\widehat{\Phi}_{n+1} + dt.\dot{\Phi}_{gyr,n}]$$

(3)

3 Implementation

Here we have implemented complementary filter using quartenion method. Here in below section matlab code for estimation using the quartenion method is implemented.

3.1 Matlab Code for Quartenion method

```
%------- section - 1: Interfacing IMU6050 sensor with Arduino Mega250 -------
clc;
clear all;
% a = arduino(); %Update the name of communication port
a = arduino('COM4', 'Mega2560', 'Libraries', 'I2C');
fs = 20; % Sample Rate in Hz
imu = mpu6050(a,'SampleRate',fs,'SamplesPerRead',1,'OutputFormat','matrix');
%%
%------- section - 2: Reading the realtime data from IMU6050 sensor -------
decim = 1;
duration = 1; % seconds
fs = 20;          % Hz
N = duration*fs;
i=0;
pitch_gyro=15*(180/pi());
roll_gyro=0;
accelR=[];
gyroR=[];
pitch_final=[];
roll_final=[];
openExample('shared_fusion_arduinoio/EstimateOrientationUsingInertialSensorFu
sionAndMPU9250Example')
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);
%     fuse=complementaryFilter('SampleRate', fs);
%     orientation = fuse(accelR,gyroR,zeros(length(accelR),3));
    orientation = fuse(accelR,gyroR);

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

```
    orientationEuler = eulerd(orientation,'ZYX','frame');
    %computes the Euler angles corresponding to the given orientation
    %which is represented in the 'ZYX' order of rotations.
    pitch_acc = atan2(accelReadings(1,1), sqrt(accelReadings(1,2)^2 +
accelReadings(1,3)^2));
    %computes the pitch angle of the sensor using accelerometer readings.
    roll_acc = atan2(accelReadings(1,2), sqrt(accelReadings(1,1)^2 +
accelReadings(1,3)^2));
    %computes the roll angle of the sensor using accelerometer readings
    pitch_gyro = pitch_gyro + gyroReadings(1,1) * 0.05;
    % updates the pitch angle using gyroscope readings
    roll_gyro = roll_gyro + gyroReadings(1,2) * 0.05;
    % updates the roll angle using gyroscope readings
    pitch = 0.96 * pitch_acc + 0.04* pitch_gyro;
    %computes the final pitch angle by combining the accelerometer and
gyroscope readings
```

```
    roll = 0.96 * roll_acc + 0.04* roll_gyro;
    %computes the final roll angle by combining the accelerometer and
gyroscope readings
    pitch_final=[pitch_final;pitch];
    %appends the current pitch angle value to pitch_final that contains the
previous data of pitch angles
    roll_final=[roll_final;roll];
    %appends the current roll angle value to roll_final that contains the
previous data roll angles.

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')
%My code


figure
plot3(pitch_final*(180/pi()),roll_final*(180/pi()),orientationEuler(:,3))
legend('Z-axis','Y-axis','X-axis')
xlabel('Time (s)')
ylabel('Rotation (degrees)')
title('Estimated Orientation Result')
```

Figure 2: Matlab Code

3.2 Results

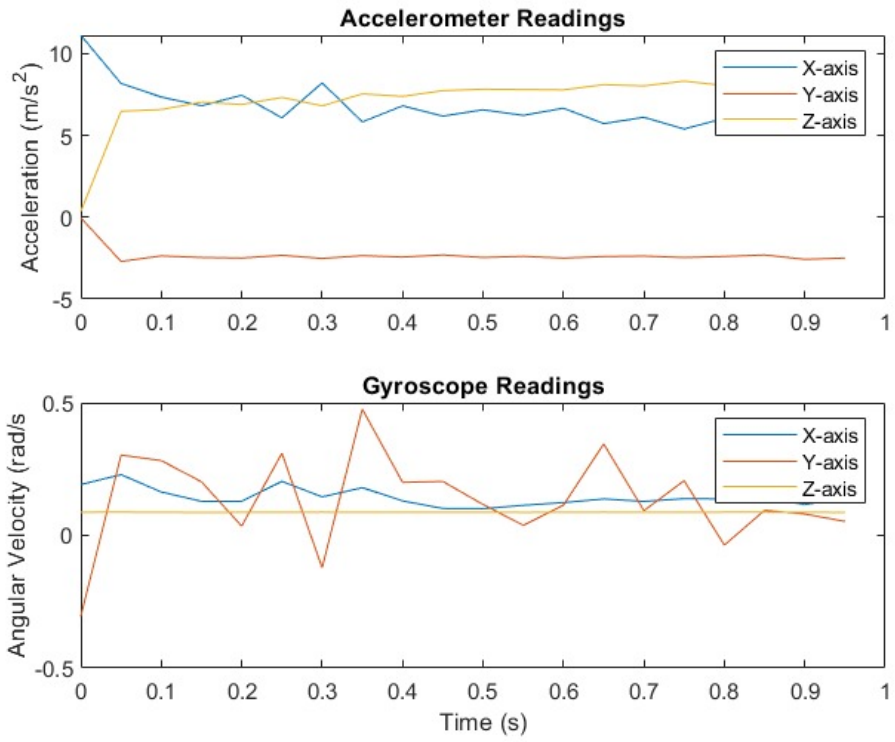

Figure 3: Accelerometer and gyroscope readings

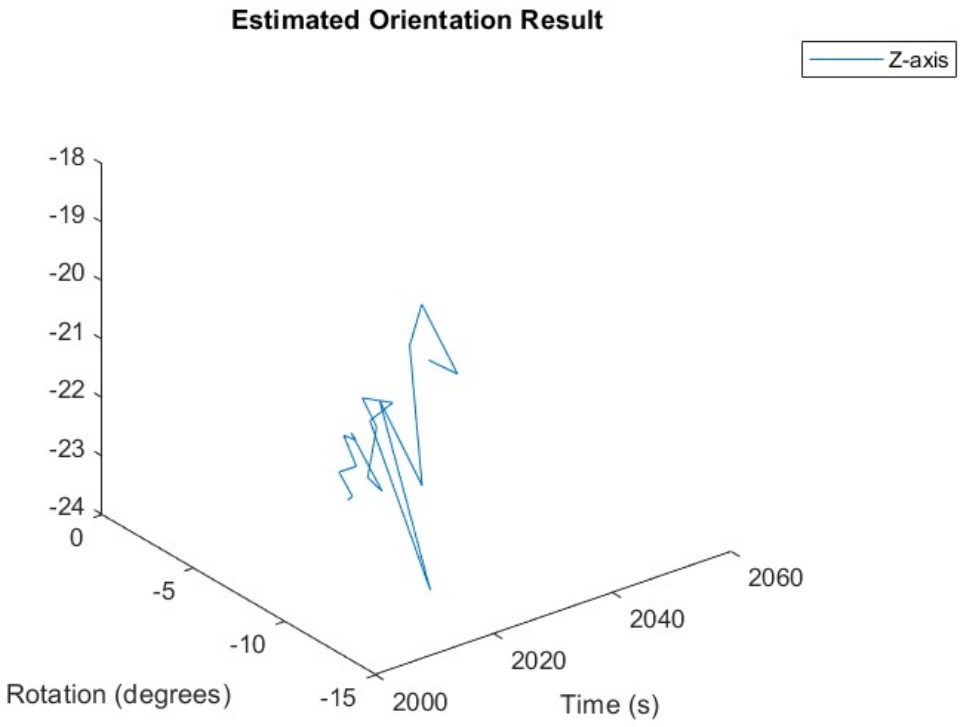

Figure 4: By quartenion Algorithm

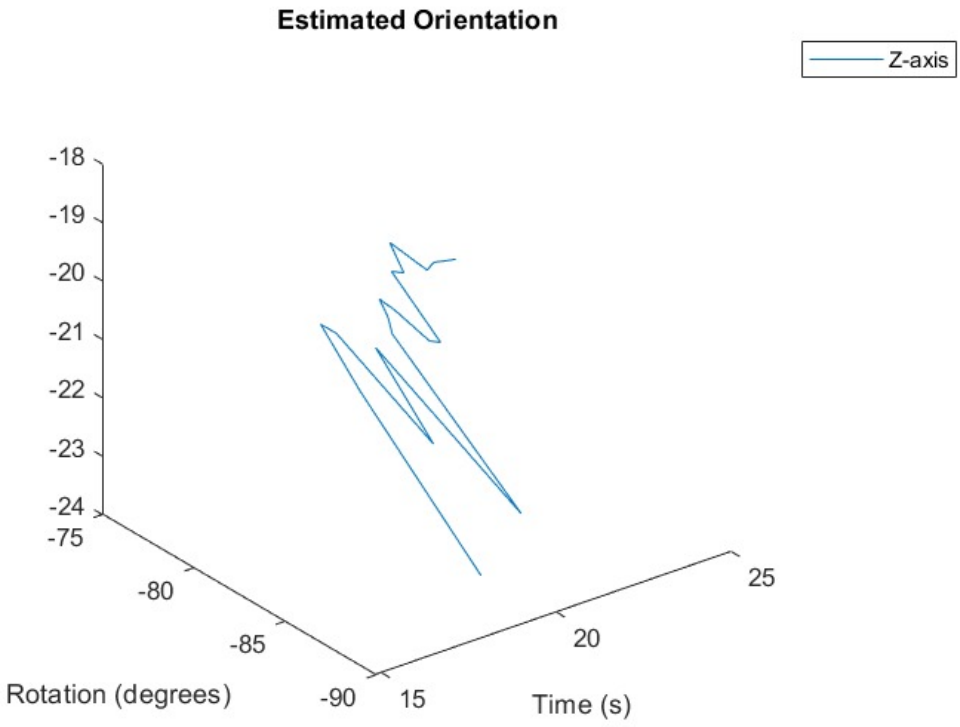

Figure 5: by in built function

4 Conclusion

we have learnt how two implement complementary filter using the quartenion method.

5 References

1. Valenti, R. G., Dryanovski, I., Xiao, J. (2015). Keeping a good attitude: A quaternion-based orientation filter for IMUs and MARGs. Sensors, 15(8), 19302-19330.

2. P. Gui, L. Tang and S. Mukhopadhyay, "MEMS based IMU for tilting measurement: Comparison of complementary and kalman filter based data fusion," 2015 IEEE 10th Conference on Industrial Electronics and

Applications (ICIEA), Auckland, New Zealand, 2015, pp. 2004-2009, doi: 10.1109/ICIEA.2015.7334442.

3. https://youtu.be/RZd6XDx5VXo

4. https://youtu.be/BUW2OdAtzBw