Warsaw University of Technology

FACULTY OF
POWER AND AERONAUTICAL ENGINEERING

Institute of Automation and Robotics

Semester Project

in the subject of Attitude and navigation system

IMU Integration

Munir Fati Haji

student record book number 323834

project supervisor

Prof. Gajda Janusz

consultation

Prof. Gajda Janusz

WARSAW 2021

List of symbols and derivations

Latin

a linear acceleration

ak 3-by-1 acceleration error vector measured in the sensor frame, in g, at time k

b sensor bias

bk 3-by-1 gyroscope zero angular rate bias vector, in deg/s, at time k

Fk state transition model

K_k Kalman gain

L_a Acceleration Vector in the local frame

L_m Magnetic field vector in the local frame

L_ω Angular velocity in the sensor frame

P_k^- predicted (*a priori*) estimate covariance

P_k^+ updated (*a posteriori*) estimate covariance

S_k innovation covariance

t time

v sensor noise

wk 9-by-1 additive noise vector

x_k^- predicted (*a priori*) state estimate; the error process

x_k^+ updated (*a posteriori*) state estimate

y_k innovation

Greek

θ pitch

θk 3-by-1 orientation error vector, in degrees, at time k

ϕ roll

ψ yaw

ω angular velocity

Contents

List of figures

List of Tables

1 Introduction

IMU is the combination of sensors that measure orientation and motion with respect to an inertial reference frame. The inertial reference frame is located at the center of the earth. Earth is rotating about it. IMU is composed of accelerometers, gyroscopes, and sometimes magnetometers. IMU is used in fusion with other navigation sensors like GPS to increase the accuracy of the system.

1.1 Sensor in IMU

IMU consists of the following sensors

- **Gyroscope** which measures angular rate
- **Accelerometer** which measures specific force/acceleration
- **Magnetometer**(optional) which measures the magnetic field around the system

A gyroscope is used to measure the orientation angle or angular rate. They are angular rotational rate sensors. Thus the integration of these rotational rates is used to get angles. Unfortunately, the direct integration is not practical due to gyro bias and gyro drift. When our sensor is stationary our gyro will be reading a non-zero value this is called gyro bias. And if we integrate this reading will gyro bias we will get a gyro drift, that is why even though our sensor is stationary the computed angle will be increasing. There are ways to correct these gyro errors by using filters.

An accelerometer measures the acceleration of motion. The double integration of the measured results from the accelerometer gives the position. Unfortunately just like gyroscopes the direct integration of the acceleration is not possible because the accelerometer also has drifted. Since we are going to integrate twice the error(drift) is quadratic. So accelerometer is difficult compared to gyroscopes

An accelerometer is the primary sensor responsible for measuring inertial acceleration, or the change in velocity over time, and can be found in a variety of different types, including mechanical accelerometers, quartz accelerometers, and MEMS accelerometers. [1]

1.2 Applications of IMU

IMU is mainly used for motion and orientation tracking in electronic devices such as cell phones and video game remotes, AR (augmented reality), and VR (virtual reality) systems. And are also used in drones for balance. IMUs are also applicable in motion detection where GPS is not applicable. And it may also be used with GPS to improve the accuracy of positioning. IMUs are also used to help maneuver aircraft, with or without a manned pilot.

1.3 Error in IMU

As tried to discuss above there are several errors that occur during the IMU integration. These errors include

Bias

If we put our gyro sensor and set it on a completely still surface then we will get a plot that is as follow. ω

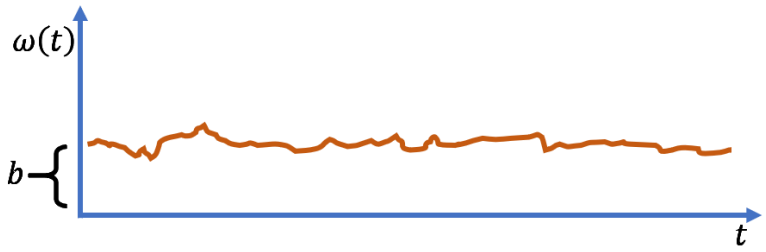

Figure 1 Bias[2]

The gyro will be reading a non-zero angular velocity even though it is completely stationary this is called a Gyro Bias or Gyro offset. This is the quality that is found on every type of sensor even though the magnitude may vary.

Drift

As discussed earlier to compute the angles from the gyroscope we have to integrate the angular rate. So if we integrate measure the angular rates at a completely still surface and then we integrate the values we will get a plot that is as follows

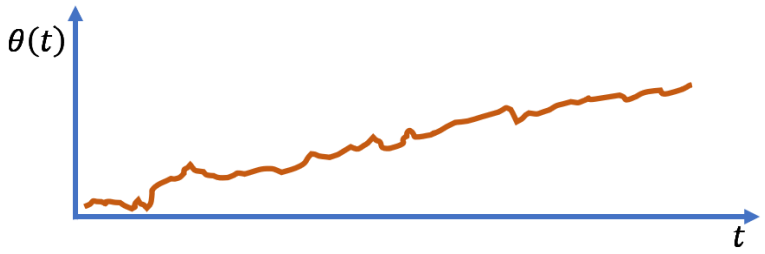

Figure 2 Drift[2]

This drift is caused by since our gyro has measurements has non-zero bias our gyro measurements begin to drift. Thus gyro drift is created by gyro bias.

Noise

Noise is classified as any random variation of a measured output when a sensor is subjected to constant input at constant conditions and is usually characterized by either a standard deviation value or a root-mean-square (RMS) value. If a noisy output signal from a sensor is integrated, for example integrating an angular rate signal to determine an angle, the integration will drift over time due to the noise. [3]

Here are our sensors and their outputs with the errors discussed above

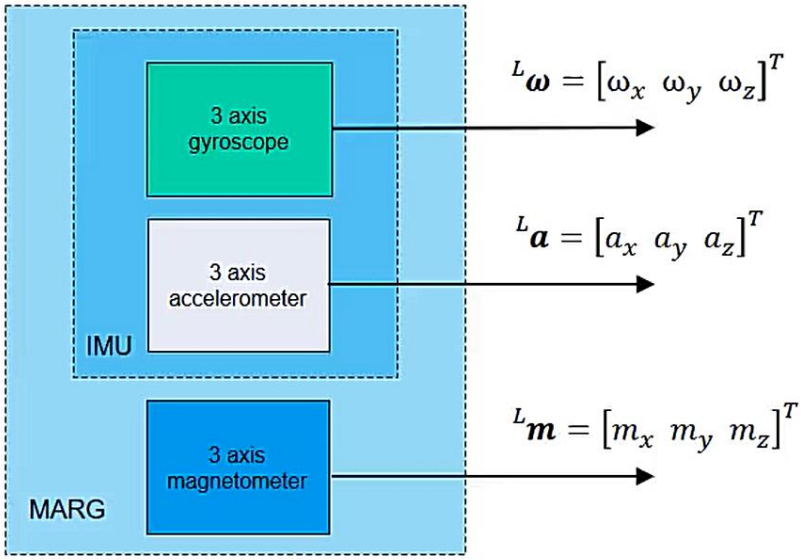

Figure 3 Sensor Fusion[4]

Using Equations we can describe the errors as

$L_\omega = L_\omega + b_g + v_g$

$L_a = L_{a_{body}} + L_g + b_a + v_a$

$L_m = L_{m_{true}} + L_{m_{int}} + b_m + v_m$

Where L_ω = Angular velocity in the sensor frame

L_a = Acceleration Vector in the local frame

L_m= Magnetic field vector in the local frame

b = sensor bias

v = sensor noise

2 Objective

2.1 Main objective

The main objective of the project is to estimate the orientation of smartphones using IMU sensors.

2.2 Specific objectives

The specific objectives of the report are to learn

- What are IMUs
- The different errors that occur in IMU sensors
- Reduce these errors by sensor fusion the use of IMU
- Write a code using MATLAB for orientation estimation.
- Compare the MATLAB results with the actual phone orientation results

3 Problem description

To estimate the orientation of a system we can simply use a gyro. To estimate the orientation simple integration of the angular rates from the reading of the orientation measurement can be used to get the three angles roll(ϕ), pitch(θ), and yaw(ψ). However, we have bias and as we are integrating we will have drift so we are going to have a big error.

We can also get the estimation from the accelerometer by getting the measurement from the accelerometer assuming there is no motion of the body. The only acceleration we measure is gravity. That is not divided into three components. Then using trigonometry we can get the roll(ϕ) and pitch(θ) with respect to the vertical axis because our gravity points down. However, we only have roll and pitch there is no yaw. But accelerometer has a high frequency.

So in short the problem is that gyros have a bias that will lead to drift. And accelerometers have high-frequency noise.

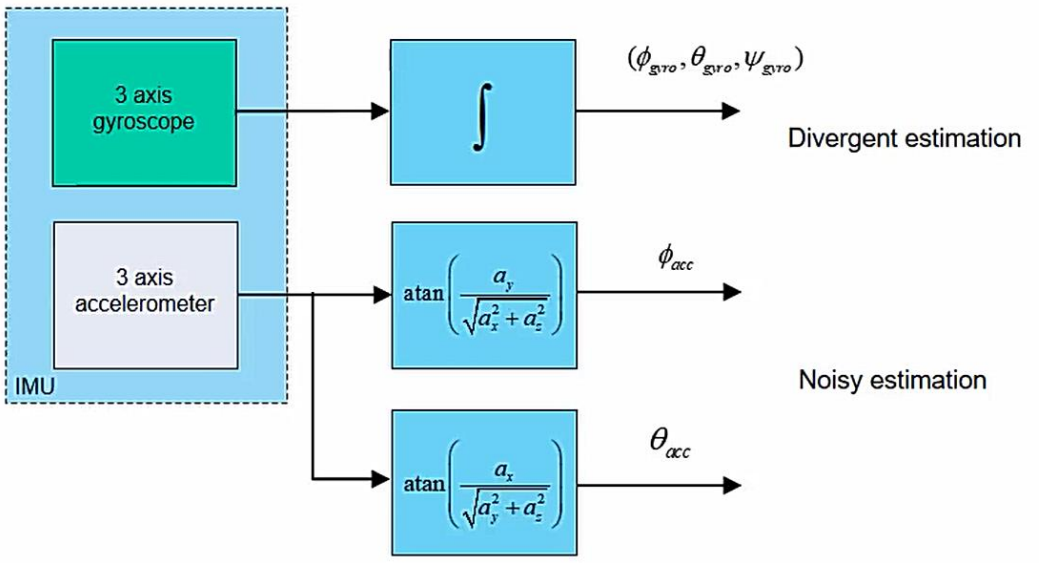

Figure 4 Problem description[4]

4 Limitations

However IMUs are good for the correction of the errors caused. The algorithm used in this project has limitation on correcting the yaw(ψ). The algorithm is able to reduce the bias errors at the initial stages but drifts away with time. To reduce this AHRS filter algorithms or neural networks model may be used. And the other limitation encountered during the project is shortage knowledge of deep learning.

5 The method of problem solution

The method proposed for solving the problem is the use of IMU. The main idea of IMU orientation estimation is to get the best out of the two kinds of estimation. So that we can have an estimation that does not have a lot of noise by the same time doesn't drift. And the magnetometer can be added usually to correct the yaws(ψ).

Assumptions: Total acceleration = gravity

When the total acceleration is different from gravity orientation can be estimated by integrating other sensors, by taking orientation from the gyroscope, or other methods like deep learning may be used to train a neural network model.

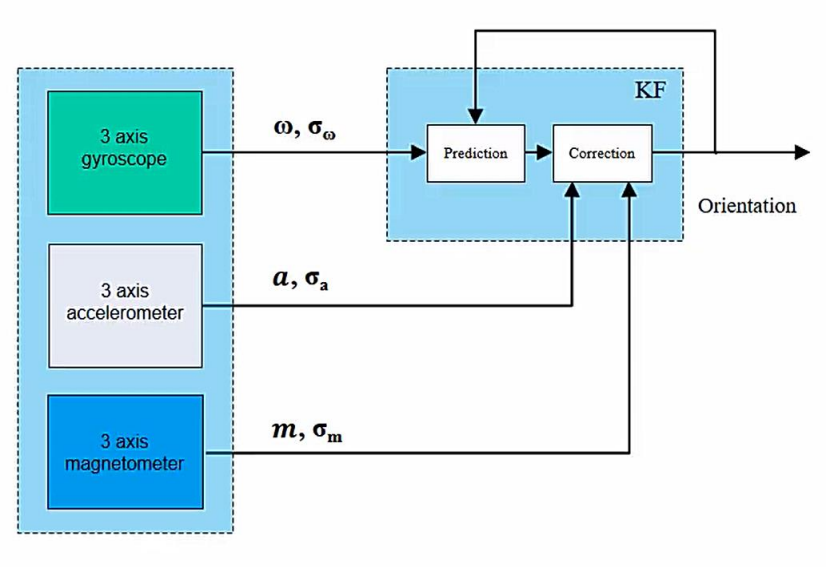

Figure 5 Kalman's filter for sensor fusion[4]

From gyroscopes, we can estimate the orientation by integrating the angular velocity rates

$$\text{roll}(\phi) = \int \omega_x \, dt$$

$$\text{pitch}(\theta) = \int \omega_y \, dt$$

$$\text{yaws}(\psi) = \int \omega_z \, dt$$

Then we the roll and pitch values allow correction of the gyroscope values. Note that the device must be initialized pointing to the Earth-relative X-axis, i.e. the North pole, for orientation detection to function properly. The pitch and roll are defined respectively as

$$\text{roll}(\phi) = \arctan\left(\frac{a_y}{\sqrt{{a_x}^2 + {a_z}^2}}\right)$$

$$\text{pitch}(\theta) = \arctan\left(\frac{a_x}{\sqrt{{a_y}^2 + {a_z}^2}}\right)$$

These equations yield roll and pitch angles from accelerometer readings. These are passed through a low-pass filter to remove high-frequency noise components. The roll and pitch are also obtained from the gyroscope calculations and then compared to these measurements.

There are various types of filters used to correct the gyroscope readings.

1. Complementary filter [5]

The complementary filter combines accelerometer and gyroscope measurements to obtain the unified, gyroscope-drift free roll and pitch angles. We can consider a preliminary model for the filter.

$$\theta_{net} = W_g.\theta_g + W_a.\theta_a$$

where θ_{net} is the true angular displacement, $W_{g,a}$ are gyroscope and accelerometer weighting factors ($W_a = 1 - W_g$) and $\theta_{g,a}$ are the angular displacement obtained from the gyroscope and accelerometer, respectively.

2. Kalman Filter [6]

Kalman filtering is an algorithm that provides estimates of some unknown variables given the measurements observed over time. Kalman filters have been demonstrating its usefulness in various applications. Kalman filters have relatively simple form and require small computational power. Kalman filter algorithm consists of two stages: prediction and update. The process model defines the evolution of the state from time $k-1$ to time k as:

$$x_k = F_{x_{k-1}} + B_{u_{k-1}} + w_{k-1}$$

where F is the state transition matrix applied to the previous state vector x_{k-1}, B is the control-input matrix applied to the control vector u_{k-1}, and w_{k-1} is the process noise vector that is assumed to be zero-mean Gaussian with the covariance Q, i.e., $w_{k-1} \sim N(0, Q)$.

The process model is paired with the measurement model that describes the relationship between the state and the measurement at the current time step k as:

$$z_k = H_{x_k} + \nu_k$$

where z_k is the measurement(observation) vector, H is the measurement matrix, and $\nu_k$is the measurement noise vector that is assumed to be zero-mean Gaussian with the covariance R, i.e., $\nu_k \sim N(0, R)$.

The role of the Kalman filter is to provide estimate of $x_k$at time k, given the initial estimate of x_0, the series of measurement, $z1,z2,...,zk$, and the information of the system described by F, B, H, Q, and R. Q and R are usually used as tuning parameters that the user can adjust to get desired performance.

To produce an accurate orientation estimate, the orientations estimated by angular rate integration and vector observation should be fused to compensate flaws of each other. Kalman filter has become the majority of 3-D orientation fusion.

6 Algorithm

The algorithm that is going to be used is `imufilter`.

The `imufilter` uses the six-axis Kalman filter structure. The algorithm attempts to track the errors in orientation, gyroscope offset, and linear acceleration to output the final orientation and angular velocity. Instead of tracking the orientation directly, the indirect Kalman filter models the error process, *x*, with a recursive update:[7]

$$x_k = \begin{bmatrix} \theta_k \\ b_k \\ a_k \end{bmatrix} = F_k \begin{bmatrix} \theta_{k-1} \\ b_{k-1} \\ a_{k-1} \end{bmatrix} + w_k$$

where x_k is a 9-by-1 vector consisting of:

- θk — 3-by-1 orientation error vector, in degrees, at time *k*
- bk — 3-by-1 gyroscope zero angular rate bias vector, in deg/s, at time *k*
- ak — 3-by-1 acceleration error vector measured in the sensor frame, in g, at time *k*
- wk — 9-by-1 additive noise vector
- Fk — state transition model

Because xk is defined as the error process, the *a priori* estimate is always zero, and therefore the state transition model, Fk, is zero. This insight results in the following reduction of the standard Kalman equations:[7]

Standard Kalman equations:

Prediction:

Predicted state estimate	$x_k{}^- = F_k x_{k-1}{}^+$
Predicted error covariance	$P_k{}^- = F_k P_{k-1}{}^+ F^T{}_k + Q_k$

Update:

Measurement residual	$y_k = z_k - H_k x_k{}^-$
	$S_k = R_k + H_k P_k{}^- H^T{}_k$
Kalman gain	$K_k = P_k{}^- H^T{}_k (S_k)^{-\mathbf{1}}$
Updated state estimate	$x_k{}^+ = x_k{}^- + K_k y_k$
Updated error covariance	$P_k{}^+ = P_k{}^- - K_k H_k P_k{}^-$

Kalman equations used in this algorithm:

Prediction:

Predicted state estimate	$x_k^- = 0$
Predicted error covariance	$P_k^- = Q_k$

Update:

Measurement residual	$y_k = z_k$
	$S_k = R_k + H_k P_k^- H^T{}_k$
Kalman gain	$K_k = P_k^- H^T{}_k (S_k)^{-1}$
Updated state estimate	$x_k^+ = K_k y_k$
Updated error covariance	$P_k^+ = P_k^- - K_k H_k P_k^-$

where

- x_k^- — predicted (*a priori*) state estimate; the error process
- P_k^- — predicted (*a priori*) estimate covariance
- y_k — innovation
- S_k — innovation covariance
- K_k — Kalman gain
- x_k^+ — updated (*a posteriori*) state estimate
- P_k^+ — updated (*a posteriori*) estimate covariance

k represents the iteration, the superscript $^+$ represents an *a posteriori* estimate, and the superscript $^-$ represents an *a priori* estimate.

The graphic and following steps describe a single frame-based iteration through the algorithm.

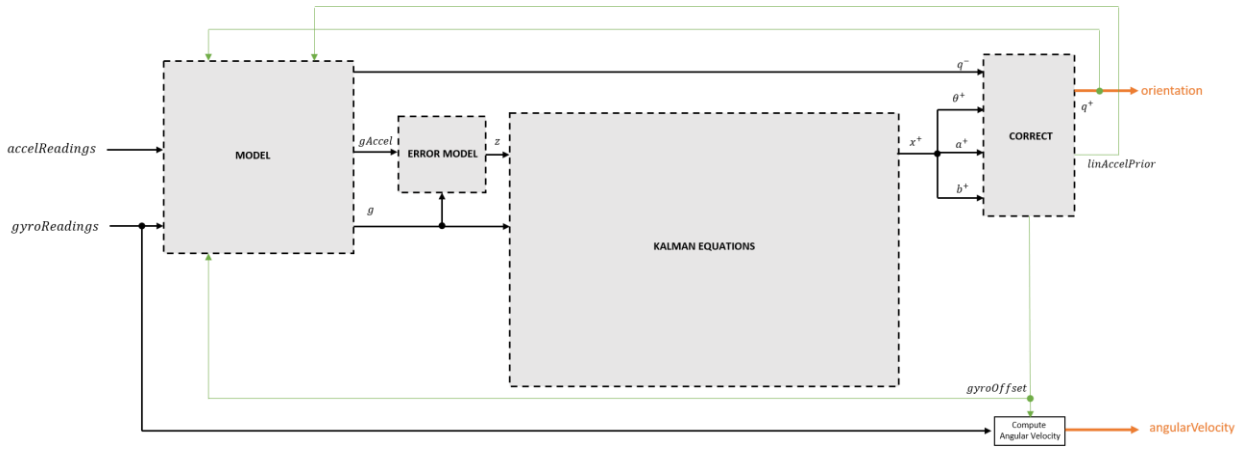

Figure 6 MATLAB Arlgorithm for IMUFilter[7]

Before the first iteration, the accelReadings and gyroReadings inputs are chunked into 1-by-3 frames and DecimationFactor-by-3 frames, respectively. The algorithm uses the most current accelerometer readings corresponding to the chunk of gyroscope readings.

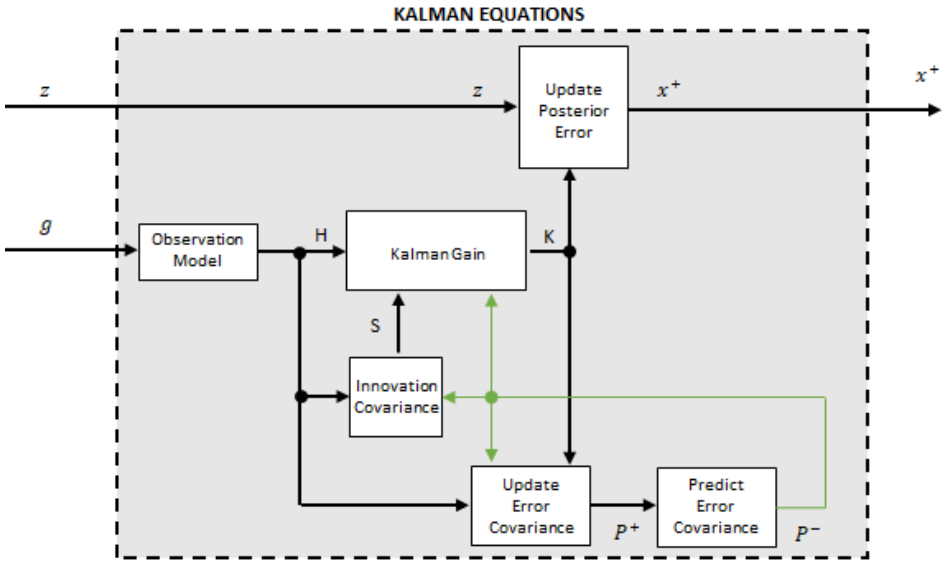

Figure 7 Kalman's Equation diagram for this algorithm[7]

The algorithm that is going to be used has the following steps

1. Measure the accelerometer and gyroscope data from the mobile device
2. Save sensor logfile on mobile MATLAB app
3. Load the sensor data
4. Define the IMU filter and assign sensors to noise covariances

5. Align reference frames
6. Fuse the sensors using the IMU filter
7. 90 deg rotation to match the phone frame
8. Convert quaternion into Euler angles
9. Visualize

7 Program code descriptions

The program developed takes the accelerometer and gyroscope reading of a mobile and estimate the orientation of the phone in the three Euler angles roll(ϕ), pitch(θ), and yaw(ψ). It also plots the estimated 3D visualization of the mobile. The program uses the IMU Kalman filter in the MATLAB.

The program first predicts the orientation based on the gyroscope reading and as mentioned above gyroscopes are biased thus it takes the reading from the accelerometer and corrects the bias using the accelerometer reading. For this, the IMU filter is used which is a Kalman filter. And the output is feedback to the predicting function so that it will have a better prediction for the next prediction. Thus the output will not be biased and it will not be noisy.

During data collection, the actual orientation of the phone is also collected. And finally, after estimation of the orientation, the estimated orientation is compared with the actual orientation collected.

During the running of the code and collecting of data, MATLAB software is used. For running the code MATLAB 2021b is used. Whereas for the collection of the data MATLAB Mobile 5.6.0 is used. The device used to run the code has the following specifications

- ASUS TUF GAMING F15
- Ram 8GB
- Storage 512GB SSD
- Processor intel core i5 10300H
- GPU Nvidia GTX-1650

7.1 Input values

The input data for the program is collected from mobile device. During the data collection MATLAB mobile is used to collect the data from the accelerometer, magnetometer, and gyroscope sensors in the mobile phone. In addition the actual orientation of the phone is also collected from the phone. The data is collected at a sampling rate of 100Hz.

7.2 Output values

The outputs from the code is the three Euler angles roll(ϕ), pitch(θ), and yaw(ψ) with time. In addition the 3D visualization of the orientation of the phone is plotted with time.

7.3 Block diagram

The program works in the following way as shown in the block diagram

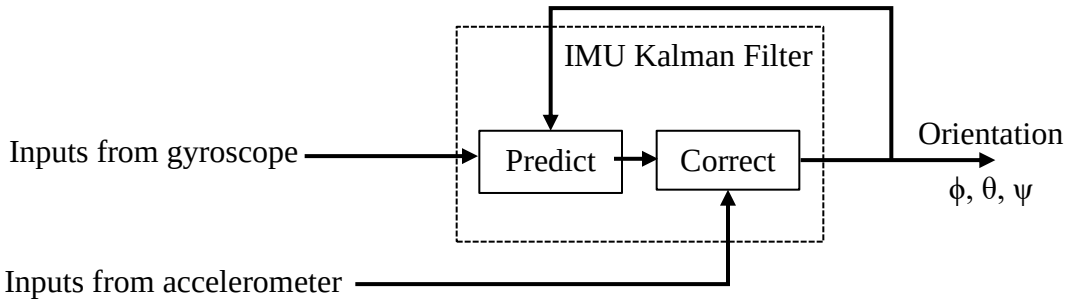

Figure 8 Block diagram for code

8 Program testing

In this section input data and results are discussed

8.1 Input data

As mentioned above the input data is collected from MATLAB at a frequency of 100Hz. After collection of the data, the file is exported as a log file. And this log file is used during the running of the program. During the collection of the input data the mobile is rotated once in x, once in y, and once in z. The sample of collected data is as follows

i. Sample accelerometer data

Table 1 Sample accelerometer data

Timestamp	a_x	a_y	a_z
'24-Nov-2021 16:37:39.427'	0.056231	1.116551	9.385847
'24-Nov-2021 16:37:39.436'	0.116052	1.128515	9.419347
'24-Nov-2021 16:37:39.445'	0.151944	1.114158	9.565309
'24-Nov-2021 16:37:39.454'	0.226421	1.051945	9.723235
'24-Nov-2021 16:37:39.463'	0.233599	0.982553	9.809377
'24-Nov-2021 16:37:39.472'	0.279063	0.989731	9.931709

ii. Sample gyroscope data

Table 2 Sample gyroscope data

Timestamp	ω_x	ω_y	ω_z
'24-Nov-2021 16:37:39.410'	-0.11713	-0.04566	-0.07208
'24-Nov-2021 16:37:39.419'	-0.16111	-0.07376	-0.07452
'24-Nov-2021 16:37:39.428'	-0.20021	-0.09819	-0.07819
'24-Nov-2021 16:37:39.437'	-0.2283	-0.10797	-0.08185
'24-Nov-2021 16:37:39.446'	-0.23808	-0.11163	-0.0843
'24-Nov-2021 16:37:39.455'	-0.22953	-0.12263	-0.08674

iii. Sample orientation data

Table 3 Sample orientation data

Timestamp	roll ϕ	pitch θ	yaw ψ
'24-Nov-2021 16:37:39.411'	76.4106	-7.12664	0.423782
'24-Nov-2021 16:37:39.420'	76.48906	-7.05532	0.38822
'24-Nov-2021 16:37:39.429'	76.64991	-6.96433	0.341612
'24-Nov-2021 16:37:39.438'	76.73681	-6.85467	0.27685
'24-Nov-2021 16:37:39.446'	76.82878	-6.72867	0.204433
'24-Nov-2021 16:37:39.456'	77.00907	-6.59169	0.123917

Note: Since the data is long here only the sample data is presented

8.2 Ressults

The sample results of orientation are as follows

Table 4 Sample output data

Timestamp	roll ϕ	pitch θ	yaw ψ
'24-Nov-2021 16:37:39.427'	76.4106	-6.72045	-0.36342
'24-Nov-2021 16:37:39.436'	76.44601	-6.66773	-0.46809
'24-Nov-2021 16:37:39.445'	76.48178	-6.56349	-0.58692
'24-Nov-2021 16:37:39.454'	76.51041	-6.38466	-0.74979
'24-Nov-2021 16:37:39.463'	76.546	-6.17465	-0.87352
'24-Nov-2021 16:37:39.472'	76.57969	-6.00833	-1.01266

Note : Since the data is long here only the sample data is presented

After computing the orientation of the system using the imufilter, the results are compared with actual orientation of the mobile and the results are presented using a table and graphically as follows.

Table 5 Result comparison

actual roll ϕ	predicted roll ϕ	error	actual pitch θ	predicted pitch θ	error	actual yaw ψ	predicted yaw ψ	error
76.4106	76.4106	0	-7.12664	-6.72045	-0.40619	0.423782	-0.36342	0.787202
76.48906	76.44601	0.04305	-7.05532	-6.66773	-0.38759	0.38822	-0.46809	0.85631
76.64991	76.48178	0.16813	-6.96433	-6.56349	-0.40084	0.341612	-0.58692	0.928532
76.73681	76.51041	0.2264	-6.85467	-6.38466	-0.47001	0.27685	-0.74979	1.02664
76.82878	76.546	0.28278	-6.72867	-6.17465	-0.55402	0.204433	-0.87352	1.077953
77.00907	76.57969	0.42938	-6.59169	-6.00833	-0.58336	0.123917	-1.01266	1.136577

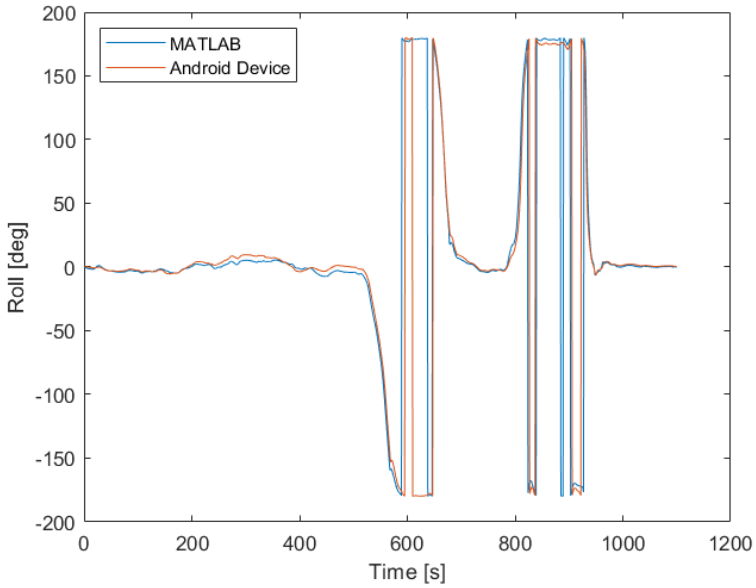

Figure 9 Result of roll

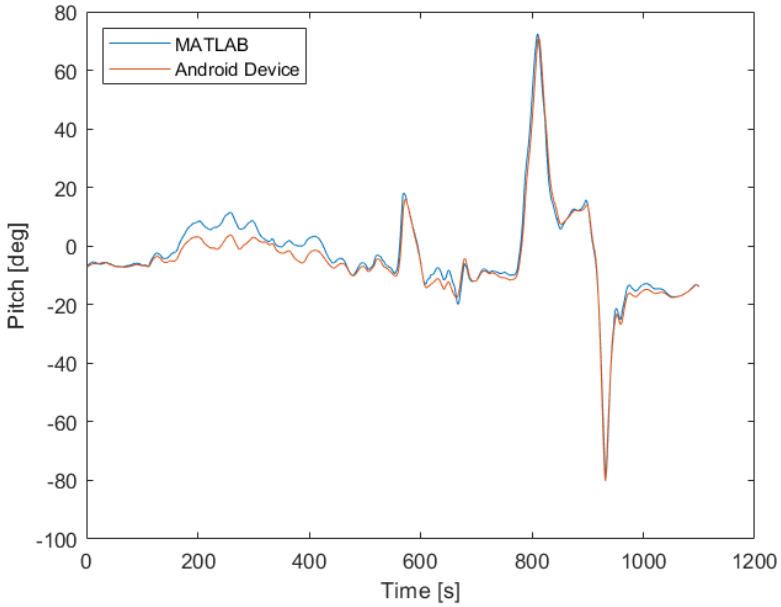

Figure 10 Result of pitch

In imufilter filters the results of roll and pitch. So as seen from the above results the orientation result from the mobile phone is close to the one from the imufilter.

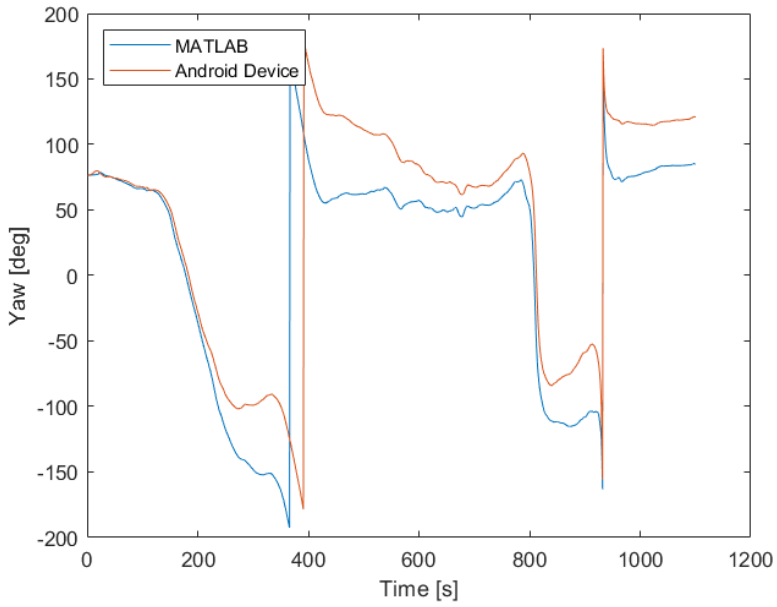

Figure 11 Result of yaw

In order to get better results for the yaw ahrsfilter may be used which combines the accelerometer, gyroscope, and magnetometer.

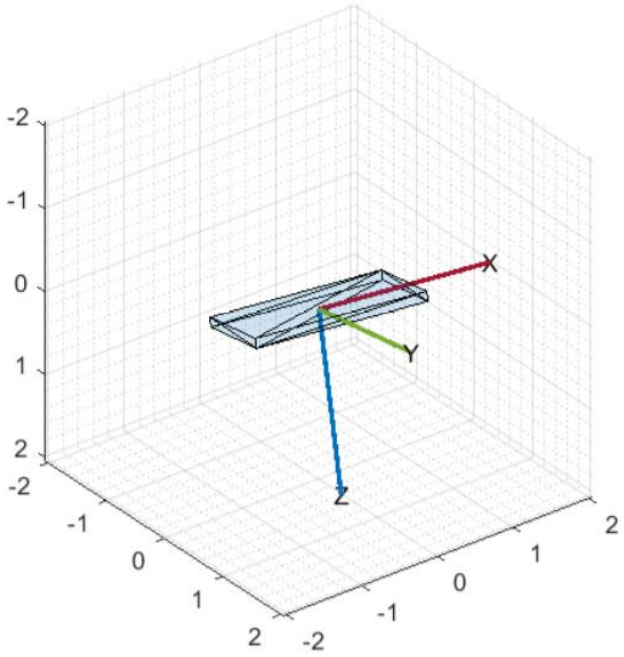

Figure 12 3D Visualization of the result

9 Conclusions

As tried to discuss in the objective of the report. The report explains

- What are IMUs
- The different errors that occur in IMU sensors
- Reduce these errors by sensor fusion the use of IMU
- Write a code using MATLAB for orientation estimation.
- Compare the MATLAB results with the actual phone orientation results

And learned that sensor fusion is a good method to reduce errors in measurements using sensors. And from the orientation estimation code, it is seen that imufilters are good for orientation estimation of the angles roll and pitch. However, it is not accurate in yaw angles. To have a better yaw angles measurement it is better to fuse the readings from the magnetometer and use the ahrs fiter.

10 List of references

[1] VectorNav. (n.d.). *What is an inertial measurement unit?* VectorNav. Retrieved January 10, 2022, from https://www.vectornav.com/resources/inertial-navigation-articles/what-is-an-inertial-measurement-unit-imu

[2] Wrona, M. (2021, May 9). *Gyro noise and Allan Deviation + imu example*. Gyro Noise and Allan Deviation + IMU Example. Retrieved January 10, 2022, from https://mwrona.com/posts/gyro-noise-analysis/

[3] VectorNav. (n.d.). *3.1 IMU Specifications*. IMU Specifications Explained · VectorNav. Retrieved January 10, 2022, from https://www.vectornav.com/resources/inertial-navigation-primer/specifications--and--error-budgets/specs-imuspecs

[4] MATLAB, M. A. T. L. A. B. (2018, December 12). *Sensor fusion for orientation estimation*. YouTube. Retrieved January 10, 2022, from https://www.youtube.com/watch?v=UZsxFpjmdAs&t=5s&ab_channel=MATLAB

[5] Elarabi, Tarek & Suprem, Abhijit & Deep, Vishal. (2016). Orientation and Displacement Detection for Smartphone Device Based IMUs. IEEE Access. PP. 1-1. 10.1109/ACCESS.2016.2631000.

[6] Kim, Y., & Bang, H. (2018, November 5). *Introduction to kalman filter and its applications*. IntechOpen. Retrieved January 10, 2022, from https://www.intechopen.com/chapters/63164

[7] MATLAB. (n.d.). *Samplerate*. Orientation from accelerometer and gyroscope readings - MATLAB. Retrieved January 10, 2022, from https://www.mathworks.com/help/fusion/ref/imufilter-system-object.html

[8] V. Gupta, S. Brennan, Terrain-based vehicle orientation estimation combining vision and inertial measurements, J. Field Robotics 25 (3) (2008) 181–202.

[9] H. Ahmed, M. Tahir, Improving the accuracy of human body orientation estimation with wearable IMU sensors, IEEE Trans. Instrum. Meas. 66 (3) (2017) 535–542.

[10] S. Yean, B. S. Lee, C. K. Yeo, C. H. Vun and H. L. Oh, "Smartphone Orientation Estimation Algorithm Combining Kalman Filter With Gradient Descent," in IEEE Journal of Biomedical and Health Informatics, vol. 22, no. 5, pp. 1421-1433, Sept. 2018, doi: 10.1109/JBHI.2017.2780879.

Appendix

Load data

```
close all;
clear;
clc;


% Load sensor measurement data
load sensorlog_20211124_163739.mat
samplerate=100;


%Put the measurement data into variables
[a] = Acceleration.Variables;
[w] = AngularVelocity.Variables;
[oin] = Orientation.Variables;


% Make sure that the time vector has the minimum length that will be matched with
the
% orientation data (for plotting purpose only )
tfin = min([length(a),length(w),length(oin)]);
t = (1:tfin);


% Define the IMU filter and assign sensors noise covariances
aFilter = imufilter('SampleRate',samplerate);
aFilter.DecimationFactor=1;
aFilter.GyroscopeNoise          = 0.0002;
aFilter.AccelerometerNoise      = 0.0006;
aFilter.LinearAccelerationNoise = 0.0005;


% Align reference frames
w(:,1) = -w(:,1);
w(:,2) = w(:,2);
w(:,3) = -w(:,3);
a(:,2) = -a(:,2);
a(:,3) = a(:,3);
qyaw = quaternion([sqrt(2)/2 0 0 sqrt(2)/2]);
```

Fusion processing loop

```
for i=1:length(t)
    % This is where the IMU fusion function is called
    orientation(i) = aFilter(a(i,:),w(i,:));

    % 90 deg rotation to match the phone frame
    orientation(i) = orientation(i)*qyaw;

    % Convert quaternion into Euler angles
    eulFilt(i,:)= euler(orientation(i),'ZYX','frame');
end


% Release the system object
release(aFilter)


inityaw = eulFilt(1,1)*180/pi-oin(1,1);
```

Visualization

```
% Plot the estimated orientations in Euler representation along with the
orientation already provided by the phone

%% Plot figure for Roll

figure(1)

plot(t,eulFilt(:,3)*180/pi,t,oin((1:tfin),3));

xlabel('Time [s]');

ylabel('Roll [deg]');

legend('MATLAB', 'Android Device','location', 'northwest');

%% Plot figure for Pitch
figure(2)
plot(t,-eulFilt(:,2)*180/pi,t,oin((1:tfin),2));
xlabel('Time [s]');
ylabel('Pitch [deg]');
legend('MATLAB', 'Android Device','location', 'northwest');

%% Plot figure for Yaw
figure(3)
plot(t,eulFilt(:,1)*180/pi-inityaw,t,oin((1:tfin),1));
xlabel('Time [s]');
ylabel('Yaw [deg]');
legend('MATLAB', 'Android Device', 'location', 'northwest');

% 3D Visualization of the IMU Filter
figure
pp = poseplot("MeshFileName", "phoneMesh.stl");
for ii=1:size(a,1)
    qimu = aFilter(a(ii,:), w(ii,:));
    set(pp, "Orientation", qimu)
    drawnow
end
```