Johnson Le

03/17/2019

CMPE 167

Lab 4: Attitude Estimation

Introduction:

This lab builds upon the previous lab to calibrate the gyroscope using all 9 data that the MPU-9250 can provide; 3 each from accelerometer, magnetometer, and gyroscope. Lab 3 had us calibrate the accelerometer and magnetometer values separately using a scale factor and null offset. These calibrations will be needed. Calibration of the gyroscope is done by utilizing accelerometer and magnetometer to estimate the gyroscope bias which can be used to estimate the attitude.

Part 1: Familiarize With Matlab Code

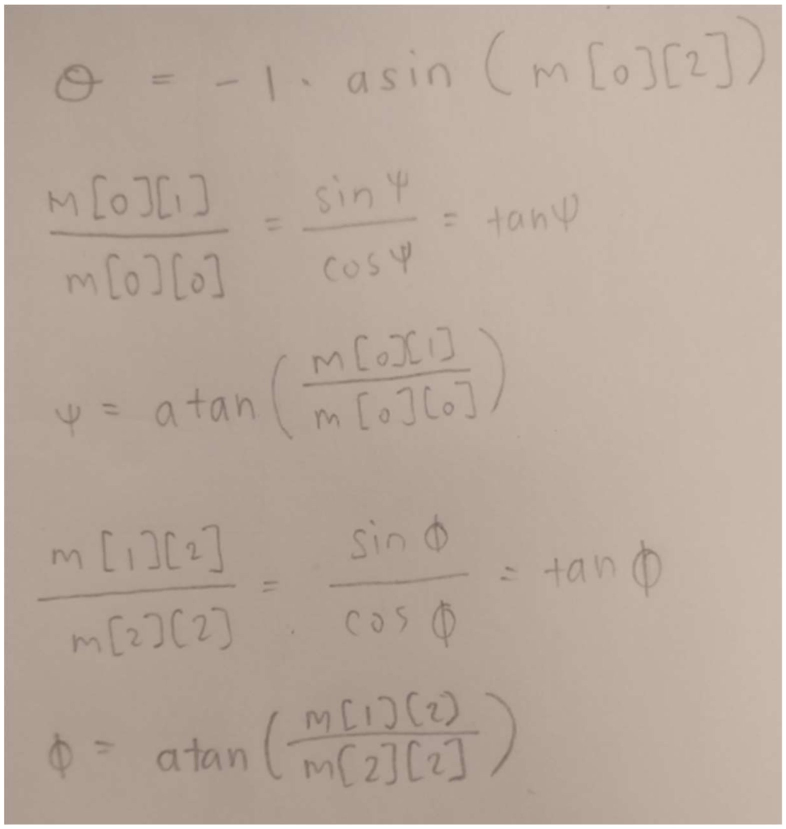
Method:

Nothing to go over here.

Part2: Conversion from DCM to Euler Angles

Method:

Provided the DCM matrix from the lab manual, we can extract the yaw, pitch, and roll angles by doing some trig.



$$\theta = -1 \cdot \mathrm{asin}(m[0][2])$$

$$\frac{m[0][1]}{m[0][0]} = \frac{\sin\psi}{\cos\psi} = \tan\psi$$

$$\psi = \mathrm{atan}\left(\frac{m[0][1]}{m[0][0]}\right)$$

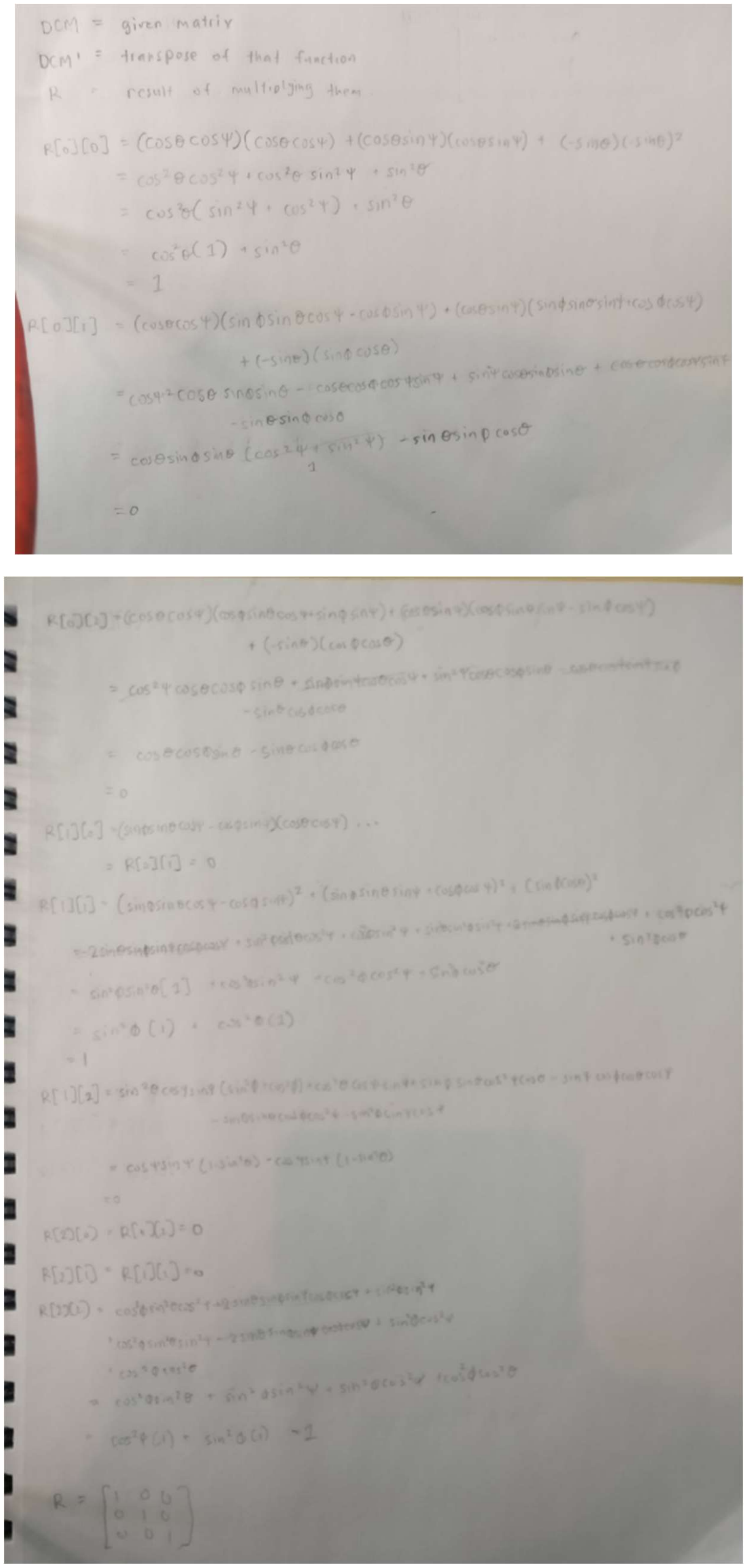
$$\frac{m[1][2]}{m[2][2]} = \frac{\sin\phi}{\cos\phi} = \tan\phi$$

$$\phi = \mathrm{atan}\left(\frac{m[1][2]}{m[2][2]}\right)$$

Figure 1: Euler Angles from DCM

In figure 1, m is the DCM matrix. This is but one of many solutions. The solution will be given in radians and should be converted to degrees which will be used to update the OLED.

To prove that the provided matrix is orthonormal, we can take the original matrix and multiply it with its transpose. The result should be the identity matrix. This can be done on paper or through a program such as Matlab. Using Matlab, we can provide arbitrary values of the angles to fill in the matrix. Multiply this matrix with the transpose and the resulting matrix should be the identity matrix.



DCM = given matrix

DCM' = transpose of that function

R = result of multiplying them

$R[0][0] = (\cos\theta\cos\psi)(\cos\theta\cos\psi) + (\cos\theta\sin\psi)(\cos\theta\sin\psi) + (-\sin\theta)(-\sin\theta)^2$

$= \cos^2\theta\cos^2\psi + \cos^2\theta\sin^2\psi + \sin^2\theta$

$= \cos^2\theta(\sin^2\psi + \cos^2\psi) + \sin^2\theta$

$= \cos^2\theta(1) + \sin^2\theta$

$= 1$

$R[0][1] = (\cos\theta\cos\psi)(\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi) + (\cos\theta\sin\psi)(\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi)$

$+ (-\sin\theta)(\sin\phi\cos\theta)$

$= \cos\psi^2\cos\theta\sin\phi\sin\theta - \cos\theta\cos\phi\cos\psi\sin\psi + \sin^2\psi\cos\theta\sin\phi\sin\theta + \cos\theta\cos\phi\cos\psi\sin\psi$

$- \sin\theta\sin\phi\cos\theta$

$= \cos\theta\sin\phi\sin\theta\ (\underbrace{\cos^2\psi + \sin^2\psi}_{1}) - \sin\theta\sin\phi\cos\theta$

$= 0$

$R[0][2] = (\cos\theta\cos\psi)(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) + (\cos\theta\sin\psi)(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)$

$+ (-\sin\theta)(\cos\phi\cos\theta)$

$= \cos^2\psi\cos\theta\cos\phi\sin\theta + \sin\phi\sin\psi\cos\theta\cos\psi + \sin^2\psi\cos\theta\cos\phi\sin\theta -$ [illegible]

$- \sin\theta\cos\phi\cos\theta$

$= \cos\theta\cos\phi\sin\theta - \sin\theta\cos\phi\cos\theta$

$= 0$

$R[1][0] = (\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi)(\cos\theta\cos\psi) \ldots$

$= R[0][1] = 0$

$R[1][1] = (\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi)^2 + (\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi)^2 + (\sin\phi\cos\theta)^2$

$= -2\sin\theta\sin\phi\sin\psi\cos\phi\cos\psi + \sin^2\phi\sin^2\theta\cos^2\psi + \cos^2\phi\sin^2\psi + \sin^2\phi\sin^2\theta\sin^2\psi +$ [illegible] $+ \cos^2\phi\cos^2\psi$

$+ \sin^2\phi\cos^2\theta$

$= \sin^2\phi\sin^2\theta[1] + \cos^2\phi\sin^2\psi + \cos^2\phi\cos^2\psi + \sin^2\phi\cos^2\theta$

$= \sin^2\phi(1) + \cos^2\phi(1)$

$= 1$

$R[1][2] = \sin^2\theta\cos\psi\sin\psi(\sin^2\phi + \cos^2\phi) + \cos^2\theta\cos\psi\sin\psi + \sin\phi\sin\theta\cos^2\psi\cos\theta - \sin\psi\cos\phi\cos\theta\cos\psi$

$- \sin\phi\sin\theta\cos\phi\cos^2\psi - \sin^2\theta\sin\psi\cos\psi$

$= \cos\psi\sin\psi(1 - \sin^2\theta) - \cos\psi\sin\psi(1 - \sin^2\theta)$

$= 0$

$R[2][0] = R[0][2] = 0$

$R[2][1] = R[1][2] = 0$

$R[2][2] = \cos^2\phi\sin^2\theta\cos^2\psi + 2\sin\theta\sin\phi\sin\psi\cos\phi\cos\psi + \sin^2\phi\sin^2\psi$

$+ \cos^2\phi\sin^2\theta\sin^2\psi - 2\sin\theta\sin\phi\sin\psi\cos\phi\cos\psi + \sin^2\phi\cos^2\psi$

$+ \cos^2\phi\cos^2\theta$

$= \cos^2\phi\sin^2\theta + \sin^2\phi\sin^2\psi + \sin^2\phi\cos^2\psi + \cos^2\phi\cos^2\theta$

$= \cos^2\phi(1) + \sin^2\phi(1) = 1$

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 2:Orthonormal Calculations

Part3: Integration of Constant Body-Fixed Rates

Method:

We begin by analyzing the provided IntegrateOpenLoop matlab function by understanding what it does and how to use it. We will be implementing this function to C to be used to integrate our IMU data. We can observe that IntegrateOpenLoop calls upon two other functions Rexp and rcross. Begin by converting these functions before implementing IntegrateOpenLoop. To test the converted functions, provide random values and run the matlab code alongside the c code comparing results It is important to test accuracy after multiple iterations.

Part4: Open Loop Gyro Integration

Method:

Similar lab3, we begin by taking 10 seconds of gyro data and averaging to achieve a bias value which we can remove. From part 3, we have an IntegrateOpenLoop that we can feed in corrected gyro data. DeltaT is the period in which data was taken. The function will output the DCM matrix and Euler angles can be extracted from it. Print the Euler angles out to the Oled.

Result:

Upon start up, the angles start at somewhat random values and slowly drift towards zero eventually hitting zero. Once at zero, the values of the angle slightly drift. The drift is quite unnoticeable unless displaying low decimals. The values are smooth and react to turns very quickly. Values for roll and yaw are very snappy and accurate while pitch takes some time to settle.

Part5: Feedback Using Only the Accelerometer

Method:

So far, we have only worked with gyro reads but using accelerometer and magnetometer data to further calibrate gyro data may prove to be worthwhile. For part 5, we will be feeding corrected accelerometer data into a closed loop integration function alongside gyro data to output a DCM and gyro bias estimate that is fed into the next iteration. Like part 3, convert the provided IntegrateClosedLoop Matlab function to c. Feed this function corrected gyro and accelerometer data.

Result:

At first glance, the values are very sporadic but eventually settle down. Rotating the IMU will make the values jump constantly over-shooting then under-shooting until settling down again to the desired value. The yaw attitude always goes back to the same value. That is because the yaw cannot be derived from using accelerometer data. My yaw values settle a lot quicker than my pitch and appears to be more accurate. I observed my pitch values to be always 1 or 2 degrees higher than its supposed to.

Comparing the IntegrateOpenLoop(IOL) to the IntegrateClosedLoop(ICL) function, IOL was a lot smoother but has a slight drift while ICL does not drift but values take a while to settle and slight movements make values go crazy.

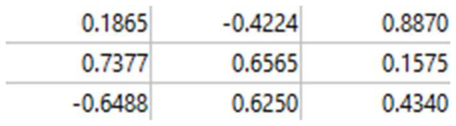
Part6: Misalignment of Accelerometer and Magnetometer

Method:

We will be extracting a misalignment matrix that will be used to manipulate the magnetometer values to be in a consistent axis set as the accelerometer. This matrix can be extracted by utilizing the provided AlignMasterSlave matlab function.

We start by grabbing tumble data like lab 3 where we imaging we are painting the surface of a sphere with the IMU. Print out these values, get them into matlab. Make sure that the accelerometer and magnetometer data arrays are in 3xn form where n is the number of data points for each axis. It is important to run with enough iterations untill the values will stop changing. Input these values into the formula to extract the misalignment matrix. This matrix will be used to multiply the incoming corrected magnetometer data for part 7 and 8.

Result:



0.1865	-0.4224	0.8870
0.7377	0.6565	0.1575
-0.6488	0.6250	0.4340

Figure 3: Misalignment Matrix

I notice at around 150 iterations, the values stopped changing and further iterations were unnecessary.

Part7: Full Complementary Attitude Estimation Using Mag and Accel Feedback

Method:

Part 7 is almost exactly as part 5 but now we add magnetometer data. Now that we have a misalignment matrix, multiply this matrix the for every x-y-z read of the magnetometer after correcting. Along with accelerometer data, initial bias feedback, and initial DCM, throw it into the IntegrateClosedLoop function. Extract the angles and display onto OLED.

Result:

Starting up, the angles try to get to 0. Compared to part 5, rotating will alter roll and pitch values to the desired value much quicker but the range of values are smaller only reaching -70 to 70 before going haywire. The yaw acts similarly.

Part8: Confirmation of Gyro Integration Using Wahba's Problem Solution

Method:

Begin by converting the provided DCMfromTriad matlab function to c. Provide it corrected accelerometer data and corrected aligned magnetometer data. The function will spit out a 3 by 3 DCM matrix that we can extract Euler angles from.

Conclusion:

Lab was very difficult because of strange terminology used in the manual as well as the expected results were hard to get too. This was especially true for part 7 and 8 which took up most of the time.