Attitude Estimation Using Complimentary Feedback Filter

Kyle Jeffrey

Abstract—Estimation of Rotation about the Earth is a common problem and aviation and other fields. The advent of the smart phone pushed the field of estimating orientation into the commercial realm, requiring cheaper solutions to a usually expenisve problem. This paper deeply investigates using cheap sensors in fusion to solve the problem of attitude estimation.

1 Introduction

This article explores the various techniques and concepts circling the formation of an AHRS or Attitude Heading Reference System. In short an AHRS is the combination of several sensors to measure the rotation of an object to a global coordinate frame.

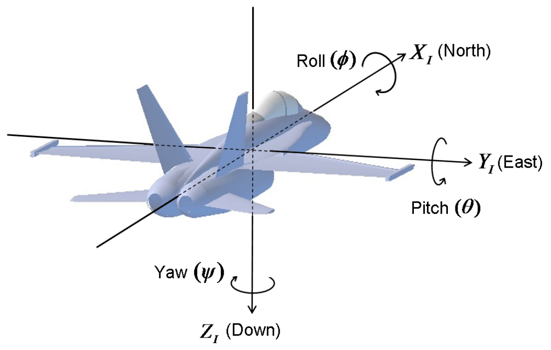


Fig. 1. AHRS Measurments

The AHRS describes the rotation of this plane to Earth. If one considers an aircraft, it becomes necessary to describe the heading of it in some terms. Is it pointing straight down? Is it level with the surface of the Earth? How to describe the rotation of an object, also referred to as the ”attitude” of an object. This attitude needs to be in respect to some frame. When it is colloquially stated that a plane is ”nosediving” it means that the plane is facing down towards the earth. We need systematic way to track and describe this problem.

1.1 Referencing Earth

A magnetometer, accelerometer, and gyroscope sensor will be used to create this sensor. The gyroscope is the primary sensor while the other two will be used to give feedback to the gyro. The accelerometer measures Earth’s gravity and will always have the information for down, and the magnetometer will always be pointing at North because of Earth’s Magnetic Field.

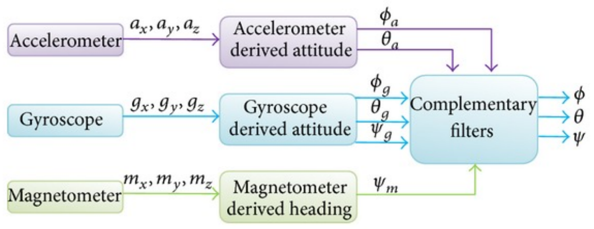


Fig. 2. Fusion Sensor

1.2 MPU9250 Gyro, Mag, Accelerometer Sensor

In the paper, the MPU9250 breakout board by sparkfun was used. The model of the sensors doesn't matter, but this specific board had very simple communication protocools, which won't be discussed in the paper.

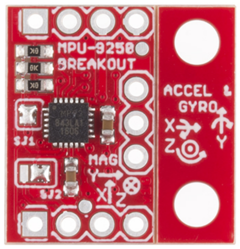


Fig. 3. Fusion Sensor

2 Storing The Rotation Angles

We are familiar with the idea that the aiplane has some form of rotation. This rotation is tracked within it's own coordinate frame, but, we need some way of storing this information and finding it's global coordinate. The infor-

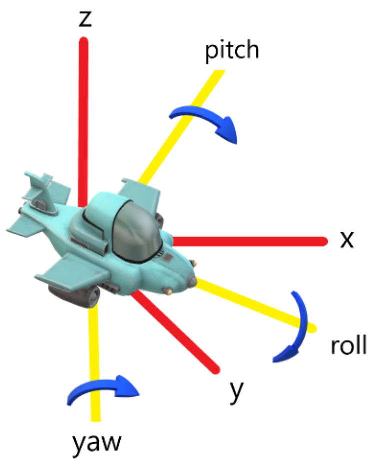


Fig. 4. Fusion Sensor

mation for the angles is stored in a rotation matrix. Using linear algebra basics, this transformation matrix contains the relationship between the airplanes coordinate frame and the global coordinate frame. Where R stands for the rotation matrix, the rotation matrix is the following:

$$\begin{aligned} &R = R_{yaw} * R_{pitch} * R_{roll} \\ &R : \vec{V_{plane}} \rightarrow \vec{V_{global}} \end{aligned} \tag{1}$$

It isn't necessary to be familiar with the exacts of what the matrix does, but from here on, it will be referred to as the Direct Cosine Matrix or DCM for short. This is standard nomenclature. We will investigate this DCM further.

2.1 The DCM

Tracking the angle of Inertial frame of a robot is a very important problem. This orientation is accomplished using IMU's or Inertial Measurement Units. These IMU's come in a wide variety, but the one we will focus on uses an accelerometer, magnetometer and a gyroscope. How do we store this information? The article "Navigation of robotic platform with using inertial measurement unit and Direct Cosine Matrix" by Beran Ladislav, Chmelar Pavel, and Dobrovolny Martin [1] discusses the use of a Direct Cosine Matrix a.k.a the DCM. The DCM is a transformation matrix based on the pitch(ϕ), roll(θ), and yaw(ψ) provided by the sensors.

The article answers the importance to this specific method. Simply, why is it necessary to use this DCM. The DCM contains all important information of the body frame and inertial frame, and is in fact the transformation between the two. The inertial frame is the world that the IMU sits in. In this case, it is based on the cardinal directions i.e. North, East, South, and West. The body frame is the perspective of the sensors. As an visual, picture that that the robot is facing a wall that an object is 1 meter directly in front of itself, i.e. 1 m north in the body frame. Well, dependent on how the robot is oriented, this isn't necessarily cardinal North, i.e. isn't necessarily 1m North in the inertial frame.

The robot could be pointed pointed straight down. The DCM tracks this orientation, and it can give the position of any object sensed by the robot and transcribe it to global coordinates. The DCM connects the spatial coordinates to angular ones.

The DCM itself is a 3x3 matrix that transforms a 3x1 vector. It is the amalgam of applying a rotation about the x-axis, y-axis and z-axis which corresponds to pitch(ϕ), roll(θ), and yaw(ψ). Matrices have the important property that applying one matrix transformation after another can be combined into one transformation matrix.

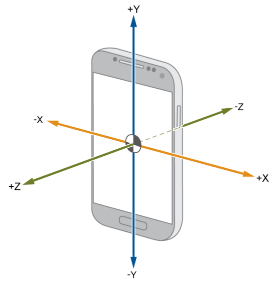
One lastly important point that the article notes is that the transformations are order dependent so $ABC! = CBA$ where A,B,C are matrices. Any equations derived need to be considered with specific order. This article applied the transformations in pitch(ϕ), roll(θ), and yaw(ψ) order i.e. rotation around the x, y, and then z axis.

3 Getting the Rotation Angles

Now that the fashion of storing the angles of rotation is clear, it's necessary to understand how using the sensors can generate these angles. First, let's understand what the individual sensors give us:

3.1 Accelerometer

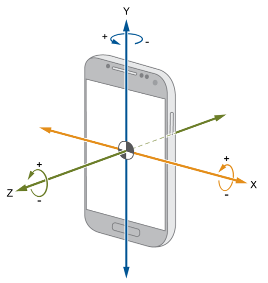
The accelerometer measures acceleration in x, y, z direction of the sensor. It will always measure gravity.



- Has no information on rotation around z-axis i.e. yaw
- Gravity vector points down gives Pitch and Roll information

3.2 Gyroscope

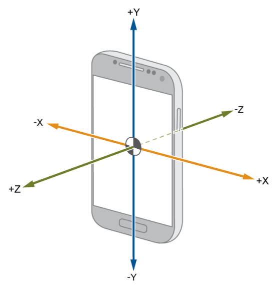
Tracks the rate of change of rotation in x, y, and z-axis. Integrate to get angle of rotation.



- Tracks rotation from starting point
- Primary component of the fusion attitude sensor
- The Magnetometer and Accelerometer give feedback to determine North and Down.

3.3 Magnetometer

Measures magnetic field. Always points to magnetic north of Earth's magnetic field.



- Gives the information for vertical direction of the sensor i.e. yaw.
- Can be affected by electronic devices or permanent magnets in the area

4 Integration of Constant Body-fixed Rates

To become familiar with the gyro sensor and DCM, some simulated analysis and experiments analysis was done. In a simulated approach, a static value for rotation rates of the gyro were set to gain some familiarity with finding the DCM using matrix math in C. After that, the Gyro was directly used to determine real-time DCM values. This section will also explore the two techniques for integrating Gyro velocity data into spatial values. There are two ways(there could be others) of converting the gyro deg/second velocity data into deg of rotation position data. Those are *"(1) using the simple forward integration ; and (2) using the matrix exponential form."* . Forward integration in matrix form is: $R^{+} = R^{-} + [\vec{\omega}\times]R^{-}\Delta t$

4.1 Forward Integration

The first method of finding the rotation matrix is to do basic integration of the gyro rotation rates. In matrix form this looks somewhat cryptic, but essentially the skew symmetric "$\vec{\omega}$" matrix is used to find $\int_0^t V_k dx$ where V_k is replaced with a gyro reading for each axis.

4.1.1 Simulated Forward Integration

A matrix math library was used to do all of the float matrix multiplication between the skew symmetricc matrix and the rotation matrix. With a rotation rate of 1 deg/sec set statically and a simulated polling rate of 1 Hz, below is a graph of the rotation of the body frame for 45 seconds, i.e. 45 degrees of rotation.

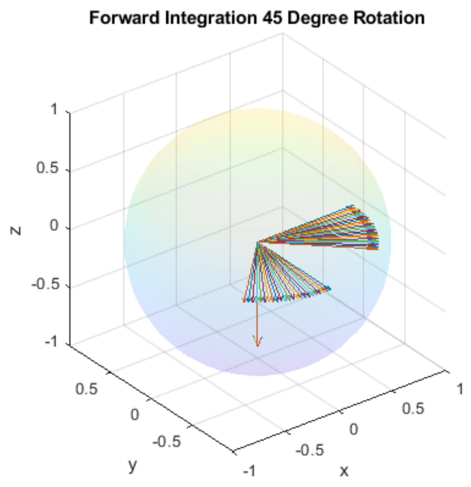
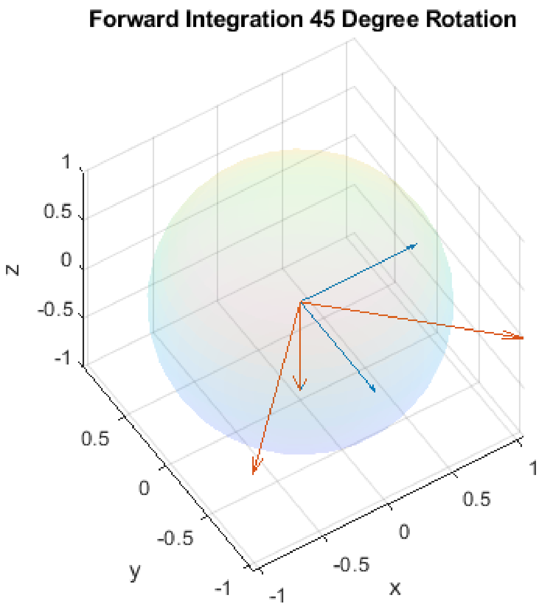


Fig. 5. Simulated Gyro Rotation

The rotation appears fairly stable from the forward integration but when left to this forward integration, the DCM doesn't maintain orthornormality. Here is a graph of the body frame vectors after nine revolutions. Notice that x and y vectors are no longer unit vectors:



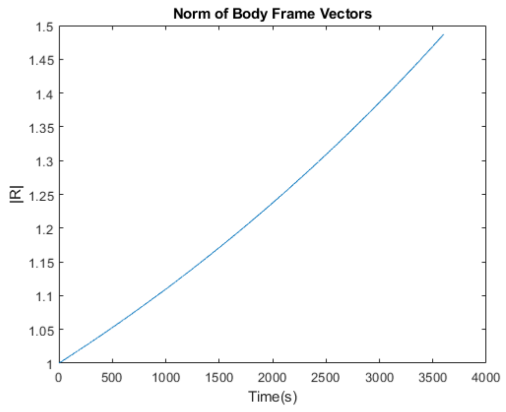
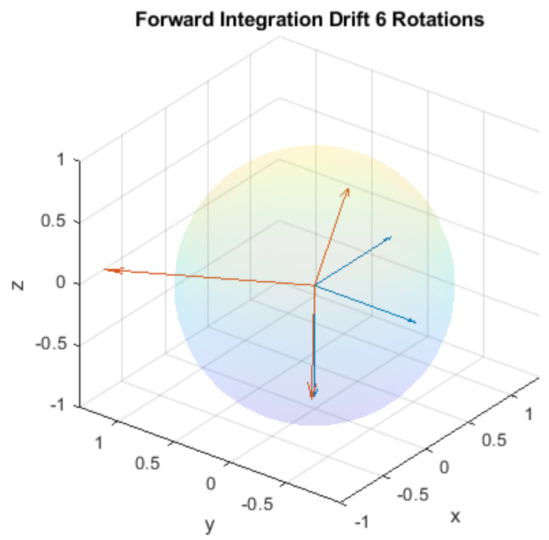


Fig. 6. Simulated Gyro Rotation

4.1.2 *Experimental Forward Integration*

Using the same technique now, the Gyro data with previously determined biases, and scale factor was used to find the DCM and then extract Euler Angles to print to the OLED screen. Here is the drift of the DCM out of orthornormality after 6 rotations:



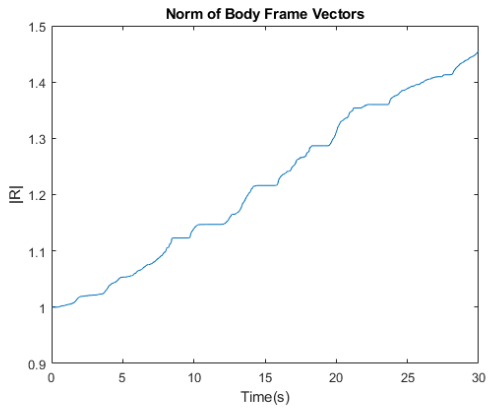


Fig. 7. Experimental Gyro Rotation

4.1.2.1 Notice how this drift only took 30 seconds. This is a huge issue to deal with and shows that simple forward integration can create huge drifting errors in a short period of time. :

4.2 Rodrigues Rotation Matrix

Now, the matrix exponential form building the DCM will be tested with gyro data experimentally and it's drift will be analyzed. It should be such that the orthornormality of the DCM is maintained but, there will be errors in the DCM in regards to the gyro sensor in real space. The Rodrigues Exponential is $Rexp(\omega\Delta t) = I + sinc(\omega\Delta t_{norm}) * cos(\omega\Delta t_{norm}) * \omega\Delta t + sinc(\omega\Delta t_{norm})^2/2 * (\omega\Delta t)^2$

5 Closed Loop Integration

The gyro sensor based off the previous parts has no reference for the inertial frame. Ideally, it should know where geographic north is and where down is in respect to some absolute frame. As far as the current setup, the gyro only measures rotation away from it's starting position when turned on. So now, the magnetometer and accelerometer will be used to compare what down is, in respect to gravity measures from the accelerometer, and where north is, in respect to the magnetometers reading of magnetic north.

The closed loop uses to corrections for the feedback loop. The first is the error correction, in this case with the z-axis, and the second is a bias estimate. The error correction is what corrects the absolute angle of rotation for the pitch and roll of the system. This correction is investigated here.

5.1 Z-Axis Error Correction

This error correction is calculated as $wmeas_a = \vec{a}_b \times R'\vec{a}_i$ where i stands for inertial frame and b stands for body frame. In essence, the difference between the DCM and the accelerometer are measured, as the accelerometer should always be pointing down towards earth. Before the gyro values are integrated the error is added as $\omega + = Kp_a * wmeas_a$ Without the bias terms introduced yet, notice that the pitch and roll axis converge on to the correct absolute rotations which have been set to 0 here. Now here's an investigation with bias on the gyro terms.

When not using the bias estimate feedback correction here, the pitch and roll will attempt to be corrected, but they cannot account for the bias error. As the values get closer to the correct angle, the difference will decrease in the error correction and will be overpowered by the bias

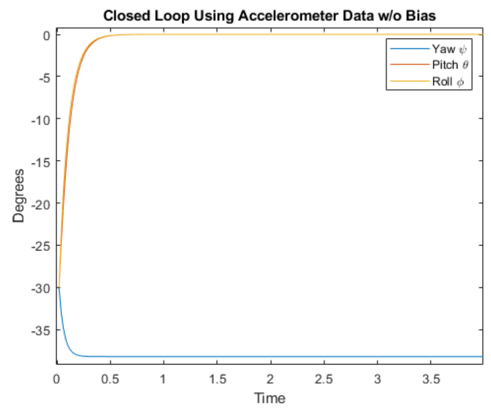


Fig. 8. Closed Loop Accel Correction w/o Bias

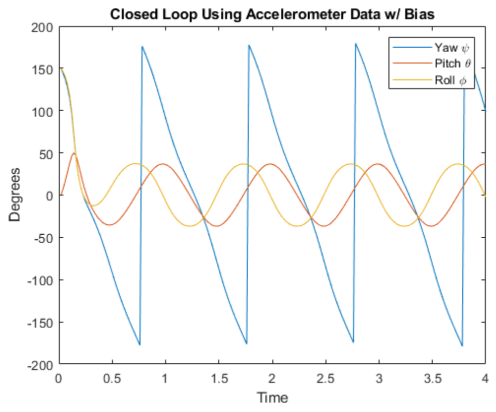


Fig. 9. Closed Loop Accel Correction w/ Bias

in the gyro values. This is why bias estimate is needed. Again, the accelerometer can do nothing to account for the absolute angle of the yaw, as it only contains information about the down inertial vector. When using the bias estimate

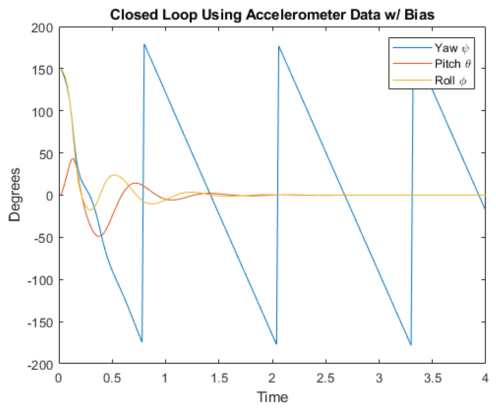


Fig. 10. Closed Loop Accel Correction w/ Bias

which is $B_{estimate}+ = -Ki_a * wmeas_a$ the pitch and roll now level out, even with bias, and hold absolute positions.

5.2 Tuning the Gains(Simulated Data)

The gains are the coefficients for both the error correction and bias estimate correction of the gyro. Here the values of these gains are investigated with the accelerometer error measurement($wmeas_a$) and the accelerometer error measurement($wmeas_m$).

6 Feedback Using only Accelerometer

Now all of these techniques are used on a microcontroller and compared to a matlab simulation. Again, the coefficients will be fiddled with to determine the best noise to correction ratio.

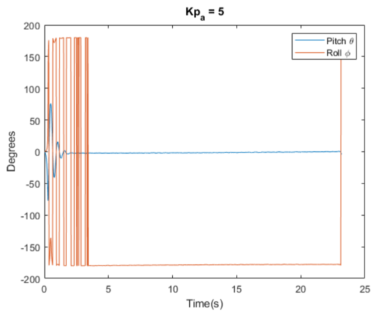


Fig. 11. Kp = .1

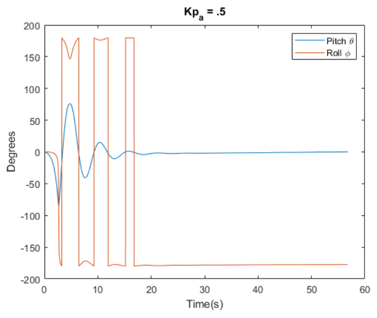


Fig. 12. Kp = .5

The accelerometer appears to be giving very noise free signal so setting the coefficient Kp to 10 gives good feedback response, and notice that it reduces acquiring the initial position

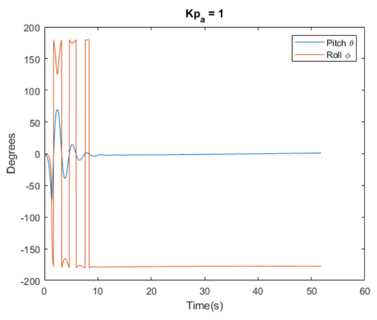


Fig. 13. Kp = 1

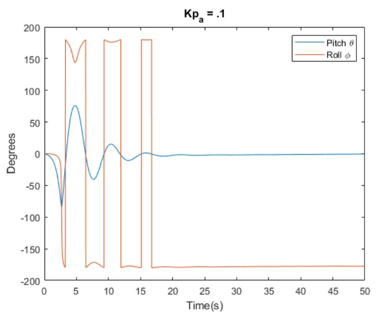


Fig. 14. Kp = 5

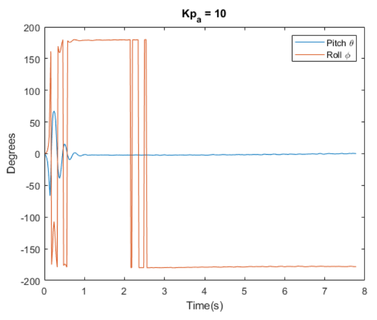


Fig. 15. Kp = 10

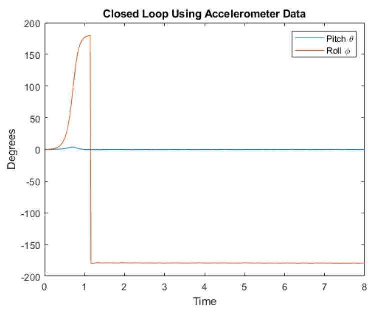


Fig. 16. Expected Closed Loop Accelerometer

from about 15 seconds down to about 1 second.

After exploring the coefficients and how they effect the data, they are now compared here with the MATLAB script for closed loop integration to view expected values and actual values of the DCM and Euler angles. Here is a comparison between actual and experimental with Kp = 10 and Ki = .5. The sensor is placed upside down to see how the feedback converges to a 180 degree rotation.

The responses appear to be comparable, though the axes approach 180 degree rotation from the other side.

7 Conclusion

With the calibration techniques and Familiarization with DCM rotation matrices, attitude sensors are now well understood. There are several ways to accomplish this including using the TRIAD algorithm vs. the closed loop feedback systems. Attitude sensors can be accomplished in many ways and this is just one version of an implementation.

References

[1] B. Ladislav, C. Pavel and D. Martin, *"Navigation of robotic platform with using inertial measurement unit and Direct Cosine Matrix,"* Proceedings ELMAR-2014, Zadar, 2014, pp. 1-4, doi: 10.1109/ELMAR.2014.6923322.

[2] G. Welch, G. Bishop, *"An Introduction to the Kalman Filter"*

[3] H. G. de Marina, F. J. Pereda, J. M. Giron-Sierra and F. Espinosa, *"UAV Attitude Estimation Using Unscented Kalman Filter and TRIAD,"* in IEEE Transactions on Industrial Electronics, vol. 59, no. 11, pp. 4465-4474, Nov. 2012, doi: 10.1109/TIE.2011.2163913.

Kyle Jeffrey is a Senior Robotics Engineering Student at the University of California Santa Cruz. He is the Secretary of the Engineering Fraternity Tau Beta Pi and the lead Hardware Engineer at the on campus startup Yektasonics.

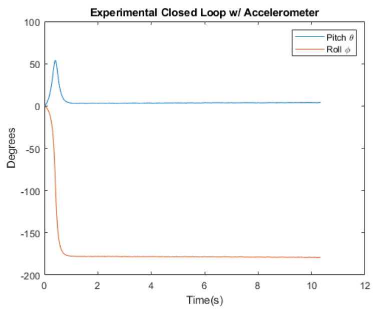


Fig. 17. Actual Closed Loop Accelerometer Feedback