Magnetometer Offset Cancellation: Theory and Implementation

William Premerlani, April 11, 2010

Problem: Magnetometers have non-zero offsets from a combination of effects, including local magnetic fields in the body frame, and offsets in the electronics. It is desired to have a simple way to determine the offsets, and subtract them out.

Solution: Enlist the aid of the direction cosine matrix to determine the offsets.

Theory:

It is assumed that the magnetometer's gains are calibrated, it is just the offsets we are worried about. For example, the HMC5843 3-Axis magnetometer has a self-test feature that determines calibration, but it does not have any way to determine the offsets.

It is possible to determine the offsets in flight. As the aircraft (plane or heli) rotates, the direction cosine matrix is good indication of its changes in orientation. The offsets are constant in the body frame of reference. The earth's magnetic field is constant in the earth frame of reference. The total measured magnetic field is the sum of the two fields. Using matrix algebra, it is possible to determine the offsets of the magnetometer.

The starting point is the direction cosine matrix, R, which relates magnetic fields in earth and body frame of reference:

$$\mathbf{b_E} = \mathbf{R} \cdot \mathbf{b_B}$$
$$\mathbf{b_B} = \mathbf{R^T} \cdot \mathbf{b_E}$$
$$\mathbf{b_E} = \text{magnetic field measurement in earth frame}$$
$$\mathbf{b_B} = \text{magnetic field measurement in body frame}$$
$$\mathbf{R} = \text{ the direction cosine matrix}$$
$$\mathbf{R^T} = \text{ the transpose of the direction cosine matrix}$$

equation 1

There are two components of the measurement, the earth's magnetic field, and the magnetometer offsets:

$$\mathbf{b_C} = \text{earth's magnetic field}$$
$$\mathbf{b_0} = \text{magnetometer offset}$$

equation 2

In the body frame of reference, the measured magnetic field vector is related to the offsets and the earth's magnetic field as follows:

$$\mathbf{b_B} = \mathbf{b_0} + \mathbf{R^T} \cdot \mathbf{b_C}$$

equation 3

By itself, equation 3 does not help us determine the offsets from a single measurement report from the magnetometer. However, we have a stream of measurements coming out of the magnetometer as the aircraft changes orientation. Similar to the technique used in wind estimation, although we cannot rely on the direction-cosine-matrix to give us absolute orientation, it is rather accurate in giving us changes in orientation. We can use

those changes, and a stream of measurements, to continually update and null out the offsets.

Start with a pair of measurements, taken at time 1 and time 2. Suppose the orientation of the aircraft has changed. (If it does not change, then the offsets do not matter!) Then we have two instances of equation 3:

$$\begin{aligned}\mathbf{b}_{\mathbf{B}1} &= \mathbf{b}_{\mathbf{0}} + \mathbf{R}_{\mathbf{1}}^{\mathbf{T}} \cdot \mathbf{b}_{\mathbf{C}} \\ \mathbf{b}_{\mathbf{B}2} &= \mathbf{b}_{\mathbf{0}} + \mathbf{R}_{\mathbf{2}}^{\mathbf{T}} \cdot \mathbf{b}_{\mathbf{C}}\end{aligned} \qquad \text{equation 4}$$

Multiplying through by the direction cosine matrix from the left, we find:

$$\begin{aligned}\mathbf{R}_{\mathbf{1}} \cdot \mathbf{b}_{\mathbf{B}1} &= \mathbf{R}_{\mathbf{1}} \cdot \mathbf{b}_{\mathbf{0}} + \mathbf{b}_{\mathbf{C}} \\ \mathbf{R}_{\mathbf{2}} \cdot \mathbf{b}_{\mathbf{B}2} &= \mathbf{R}_{\mathbf{2}} \cdot \mathbf{b}_{\mathbf{0}} + \mathbf{b}_{\mathbf{C}}\end{aligned} \qquad \text{equation 5}$$

Subtracting the two equations just given, we find:

$$\mathbf{R}_{\mathbf{2}} \cdot \mathbf{b}_{\mathbf{B}2} - \mathbf{R}_{\mathbf{1}} \cdot \mathbf{b}_{\mathbf{B}1} = \left(\mathbf{R}_{\mathbf{2}} - \mathbf{R}_{\mathbf{1}}\right) \cdot \mathbf{b}_{\mathbf{0}} \qquad \text{equation 6}$$

At this point, we might be tempted to invert the matrix on the right hand side of equation 6, but we would be falling into a trap, because its determinant is identically 0. So we cannot do that. Some thought will reveal the reason why the determinant is 0, and will give us an approach to nulling the offset anyway.

Consider the case in which the aircraft rotates around the Z axis. This will give us information only about the X and Y offsets, but tells us nothing about the Z axis offset. Similarly, equation 6 gives us information only about the offset in the plane that is perpendicular to the rotation from position 1 to position 2. Still, the information in that plane is useful, because it is all we need to null the offset in that plane, and we can wait until there is a change in orientation that gives us information in other planes to give us the complete picture. So, we can use equation 6 to obtain the component of the offset that is perpendicular to the rotation axis. Before stating the solution to equation 6, it is useful to first define the relative rotation between the two orientations and its transpose:

$$\begin{aligned}\mathbf{R}_{\mathbf{21}} &= \mathbf{R}_{\mathbf{2}}^{\mathbf{T}} \cdot \mathbf{R}_{\mathbf{1}} \\ \mathbf{R}_{\mathbf{21}}^{\mathbf{T}} &= \mathbf{R}_{\mathbf{1}}^{\mathbf{T}} \cdot \mathbf{R}_{2}\end{aligned} \qquad \text{equation 7}$$

Now, we can solve equation 6 for the component of the offset that is perpendicular to the axis of rotation from orientation 1 to 2. (The interested reader might want to give it a try.)

$$\mathbf{b}_{\mathbf{0}\perp} = \frac{\mathbf{b}_{\mathbf{B1}} + \mathbf{b}_{\mathbf{B2}} - \left[\mathbf{R}_{\mathbf{21}} \cdot \mathbf{b}_{\mathbf{B1}} + \mathbf{R}_{\mathbf{21}}^{\mathbf{T}} \cdot \mathbf{b}_{\mathbf{B2}}\right]}{3 - Trace\left(\mathbf{R}_{\mathbf{21}}\right)} \qquad \text{equation 8}$$

The function “Trace” operating on a matrix returns the sum of the diagonal elements. This is the first time I have ever used this operator. I am delighted.

Equation 8 is an exact solution to equation 6. But we do not need an exact solution. What we need is something that we can feed into an integrator that will gradually drive the offsets to zero. The numerator of equation 8 is a vector that we can use to drive the offset vector in the correct direction. It is convenient to multiply equation 8 by the denominator of the right hand side. After we do that, we will have a vector in the correct direction,

multiplied by a gain that rises and falls with the amount of rotation that the aircraft is experiencing. This is just the thing that we want, more gain when it is rotating, less gain than when it is not. So, now we have a negative feedback recipe for driving the offsets to zero in flight.

We will maintain an up to date estimate of the magnetic offset vector of the magnetometer:

$$\hat{\mathbf{b}}_0 = \text{estimate of the magnetometer offset} \qquad \text{equation 9}$$

Each time we make a magnetometer measurement, we will subtract the offset before we report it:

$$\mathbf{b}_B = \mathbf{b}_{magnetometer} - \hat{\mathbf{b}}_0 \qquad \text{equation 10}$$

We will use the adjusted values in the direction cosine matrix drift correction. We will also use pairs of adjusted values and their associated direction cosine matrices to compute an update for the estimate of the offset vector:

$$\boxed{\hat{\mathbf{b}}_0 = \hat{\mathbf{b}}_0 + \frac{\Delta t}{\tau}\left(\mathbf{b}_{B1} + \mathbf{b}_{B2} - \left[\mathbf{R}_{21} \cdot \mathbf{b}_{B1} + \mathbf{R}_{21}^{T} \cdot \mathbf{b}_{B2}\right]\right)} \qquad \text{equation 11}$$

Delta t is the time step, tau is the desired time constant of the adjustment. The elegance of equation 11 is that we can execute it for every new magnetometer measurement, whether the aircraft is changing orientation or not. If it is not changing orientation, nothing will happen, because the right hand side of equation 11 will be zero. We will not be driving the offset to zero, but it does not matter, because we are not changing direction anyway.

If the aircraft is rapidly changing orientation, then the right hand side of equation 11 will be large, and will quickly drive the offset to zero, exactly when we need to do that.

Implementation

Implementation of equation 11 is straightforward. The only special case we have to be careful about is when the aircraft is sitting on the ground and not moving at all. In that situation, equation 11 should return identically zero, but because of roundoff, there can be a bias, which would gradually ramp up the offset estimates. The errors would dissipate quickly once the aircraft began moving, but it would be best to prevent them in the first place. That is easy enough to do by simply putting a small deadband in the integration of equation 11.

The following is the implementation of equation 11, as well as the yaw compensation of the direction cosine matrix, in a magnetometer demo written for the UAV DevBoard. Note that the yaw compensation takes only 6 lines, it is a lot easier to do in terms of direction cosines than it is in terms of tilt-compensating the magnetometer in terms of Euler angles.

The bulk of the code is for computing the magnetometer offsets:

```
void mag_drift()
{
  int mag_error ;
  int vector_index ;
  fractional rmatTransposeMagField[3] ;
  fractional offsetSum[3] ;
  fractional deltaMagField[3] ;
  if ( flags._.mag_drift_req )
  {
    setDSPLibInUse(true) ;
    // Compensate the DCM algorithm for drift:
    magFieldEarth[0]=VectorDotProduct(3,&rmat[0],magFieldBody )<<1 ;
    magFieldEarth[1]=VectorDotProduct(3,&rmat[3],magFieldBody )<<1 ;
    magFieldEarth[2]=VectorDotProduct(3,&rmat[6],magFieldBody )<<1 ;
    mag_error=100*VectorDotProduct(2,magFieldEarth,declinationVector) ;
    VectorScale( 3 , errorYawplane , &rmat[6] , mag_error ) ;

    // Compute the offsets in the magnetometer:
    VectorAdd( 3 , offsetSum , magFieldBody , magFieldBodyPrevious ) ;
    for ( vector_index = 0 ; vector_index < 3 ; vector_index++ )
    {
      offsetSum[vector_index] >>= 1 ;
    }
    MatrixMultiply(1,3,3,
              rmatTransposeMagField,magFieldEarthPrevious,rmat) ;
    VectorSubtract(3 ,offsetSum,offsetSum,rmatTransposeMagField ) ;
    MatrixMultiply(1,3,3,
              rmatTransposeMagField,magFieldEarth,rmatPrevious) ;
    VectorSubtract(3,offsetSum,offsetSum,rmatTransposeMagField) ;
    for ( vector_index = 0 ; vector_index < 3 ; vector_index++ )
    {
      int adjustment ;
      adjustment = offsetSum[vector_index] ;
      if ( abs( adjustment ) < 3 )
      {
        offsetSum[vector_index] = 0 ;
        adjustment = 0 ;
      }
      offsetDelta[vector_index] = adjustment ;
    }
    if ( flags._.first_mag_reading == 0 )
    {
      VectorAdd ( 3 , magOffset , magOffset , offsetSum ) ;
    }
    else
    {
      flags._.first_mag_reading = 0 ;
    }
    VectorCopy ( 3 , magFieldEarthPrevious , magFieldEarth ) ;
    VectorCopy ( 3 , magFieldBodyPrevious , magFieldBody ) ;
    VectorCopy ( 9 , rmatPrevious , rmat ) ;
    setDSPLibInUse(false) ;
    flags._.mag_drift_req = 0 ;
  }
  return ;
```