Freescale Semiconductor
Application Note

Document Number:AN4685
Rev. 1.0, 5/2013

Implementing a Virtual Gyro Using Accelerometer and Magnetometer Sensors

by: Mark Pedley

Contents

1 Introduction

This Application Note is an addendum to Version 3 of the Xtrinsic eCompass software release provided under license at www.freescale.com/ecompass. It has been written to describe how to implement a Virtual Gyro to determine rotation rate using accelerometer and magnetometer sensors as an alternative to a physical gyro sensor. This Application Note is part of the technical documentation for the Version 3 Xtrinsic software and its use and distribution are controlled by that license agreement.

1.1 Virtual Gyro

In the absence of acceleration, the combination of an accelerometer and magnetometer sensor can be used to determine orientation in Euler angle (roll, pitch and yaw), rotation matrix or quaternion formats. The rate of change of orientation computed from accelerometer and magnetometer measurements is termed a Virtual Gyro since it provides an estimate of angular velocity without using a physical gyro sensor.

The primary limitation of the Virtual Gyro is that the accelerometer sensor is sensitive to any linear acceleration present as well as orientation of the accelerometer in the earth's gravitational field. The Virtual Gyro algorithm is based on the Xtrinsic eCompass algorithm which, like all eCompass

algorithms, assumes that the linear acceleration component is negligible. This is a valid assumption for some applications but is not valid for applications such as gaming controls, where strong linear and angular acceleration forces act on the accelerometer.

The Xtrinsic Virtual Gyro algorithm described in this document is a simple extension to the Xtrinsic eCompass, but it should not be used until the Xtrinsic eCompass and magnetic calibration algorithms are running smoothly.

1.2 Advantages of Virtual Gyro

- The power consumption of a physical gyro sensor is significantly greater than accelerometer and magnetometer sensors. A Virtual Gyro solution can therefore offer power savings as well as cost savings over the use of a physical gyro sensor.
- The Virtual Gyro does not suffer the zero rate offset error present in a physical gyro sensor which requires real-time calibration. If the accelerometer and magnetometer sensors of the Virtual Gyro are not rotating, then the Virtual Gyro will simply report a zero rotation rate plus noise.

1.3 Disadvantages of Virtual Gyro

- The orientation matrix computed from an accelerometer and magnetometer will be erroneous in the presence of linear acceleration from handshake or other sources. A physical gyro sensor has low sensitivity to linear acceleration. The Virtual Gyro is therefore only suited for benign dynamical environments and should not be considered for applications such as gaming handsets.
- The Virtual Gyro does not suffer the gain calibration error present in a physical gyro sensor. Assuming accurate clocking, the angular rotation rate reported by a Virtual Gyro will be the true rotation rate plus noise.
- The Virtual Gyro uses a magnetometer sensor and therefore requires accurate magnetic calibration of the magnetometer sensor and hard and soft iron magnetic interference from the circuit board. Any magnetic calibration error will manifest as an orientation-dependent rotation rate error. A gyro sensor is a mechanical MEMS device with no sensitivity to magnetic fields.

2 Virtual Gyro Algorithm

2.1 Orientation Matrix

Application Note AN4685 defines the instantaneous 3x3 orientation matrix `thisSV6DOF.fR6DOFn` in the Aerospace/NED, Android and Windows 8 coordinate systems and shows the matrix is computed by the functions `feCompassDirectNED`, `feCompassDirectAndroid` and `feCompassDirectWin8` respectively.

The orientation matrix defines the orientation of the circuit board and can be used to determine its rotation rate by differentiation with respect to time. Differentiation of the Euler angles or the orientation quaternion is not as well suited for calculating rotation rates since these representations have discontinuities at certain orientations which lead to spurious discontinuities in the calculated rotation rates. The rotation matrix representation of orientation has no such discontinuities, varies smoothly as the circuit board orientation is changed and can be differentiated to obtain the Virtual Gyro angular rotation rate.

2.2 Differencing the Rotation Matrix

Differentiation in a discrete time-sampled system is computed from the differences of successive values. The instantaneous (not low pass filtered) orientation matrix $\boldsymbol{R}[k]$ at sample k can be represented as the product of i) the rotation matrix $\Delta\boldsymbol{R}[k]$ defining a rotation during the current sampling period and ii) the matrix $\boldsymbol{R}[k-1]$ representing the orientation at the previous sample $k-1$:

$$\boldsymbol{R}[k] = \Delta\boldsymbol{R}[k]\,\boldsymbol{R}[k-1] \Rightarrow \Delta\boldsymbol{R}[k] = \boldsymbol{R}[k]\boldsymbol{R}[k-1]^{-1} = \boldsymbol{R}[k]\boldsymbol{R}[k-1]^{T} \quad (1)$$

No assumptions are made in equation (1) about the coordinate system and it is therefore valid for NED/Aerospace, Android, Windows 8 and, in fact, any coordinate system.

Although the terminology $\Delta\boldsymbol{R}[k]$ is used for the rotation matrix storing the sample to sample difference in orientation, equation (1) and the rest of this Application Note do not make any small angle assumptions and the mathematics are accurate irrespective of the magnitude of rotation from one sample to the next within the limits discussed in Maximum Theoretical Virtual Gyro Angular Velocity.

2.3 Computing the Instantaneous Virtual Gyro Rotation Rate

Equation 70 of AN4676 defines the rotation matrix representing an arbitrary rotation about a given rotation axis. With the notation that the differenced rotation matrix $\boldsymbol{R}[k]$ encodes the instantaneous rotation angle $\Delta\eta$ about the instantaneous rotation axis vector $\hat{\boldsymbol{n}} = \{\hat{n}_x, \hat{n}_y, \hat{n}_z\}$, then that equation can be written in the form:

$$\Delta\boldsymbol{R}[k] = \begin{pmatrix} \hat{n}_x{}^2 + \left(1-\hat{n}_x{}^2\right)cos\Delta\eta & \hat{n}_x\hat{n}_y(1-cos\Delta\eta) + \hat{n}_z sin\Delta\eta & \hat{n}_x\hat{n}_z(1-cos\Delta\eta) - \hat{n}_y sin\Delta\eta \\ \hat{n}_x\hat{n}_y(1-cos\Delta\eta) - \hat{n}_z sin\Delta\eta & \hat{n}_y{}^2 + \left(1-\hat{n}_y{}^2\right)cos\Delta\eta & \hat{n}_y\hat{n}_z(1-cos\Delta\eta) + \hat{n}_x sin\Delta\eta \\ \hat{n}_x\hat{n}_z(1-cos\Delta\eta) + \hat{n}_y sin\Delta\eta & \hat{n}_y\hat{n}_z(1-cos\Delta\eta) - \hat{n}_x sin\Delta\eta & \hat{n}_z{}^2 + \left(1-\hat{n}_z{}^2\right)cos\Delta\eta \end{pmatrix} \quad (2)$$

The differenced rotation matrix $\Delta\boldsymbol{R}[k]$ can now be passed to function `fAxisAngleDegFromRotationMatrix`, documented in section 8.3 of AN4676, to compute the axis-angle vector (also known as a rotation vector) equal to the product $\hat{\boldsymbol{n}} = \{\hat{n}_x, \hat{n}_y, \hat{n}_z\}$. The instantaneous Virtual Gyro angular velocity vector $\boldsymbol{\omega}[k]$, given a sampling interval Δt seconds from sample $k-1$ to sample k, is (in units of degrees per second):

$$\boldsymbol{\omega}[k] = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = f_s\Delta\eta\hat{\boldsymbol{n}} = f_s\Delta\eta\begin{pmatrix} \hat{n}_x \\ \hat{n}_y \\ \hat{n}_z \end{pmatrix} \quad (3)$$

f_s is the sampling frequency in Hz defined as:

$$f_s = 1/\Delta t \quad (4)$$

2.4 Maximum Theoretical Virtual Gyro Angular Velocity

The maximum theoretical angular velocity ω_{max}, detectable by the Virtual Gyro algorithm without aliasing of the rotation rate, corresponds to a rotation of 180° per sample interval giving:

$$\omega_{max} = 180 f_s \; deg/s \tag{5}$$

Higher rotation rates are aliased into rotations about a negated rotation axis. This can easily be understood by considering a rotation rate of 181° per sample interval, which will be interpreted as a rotation –179° about the same axis or +179° about the negated rotation axis.

In practice, this limit is never achieved. At a typical sample frequency of 50 Hz, the maximum theoretical Virtual Gyro rotation rate is 9000°/s or 25 complete 360° rotations per second. The fundamental assumption in the Virtual Gyro algorithm of no linear or centripetal accelerations will have broken down long before this point is reached.

2.5 Low Pass Filtering the Virtual Gyro Angular Velocity

The instantaneous, meaning unfiltered, Virtual Gyro angular velocity vector will be noisy i) as a result of sensor noise and ii) through being computed from the difference of two noisy orientation estimates separated in time by just Δt seconds. Low pass filtering of the three components of the Virtual Gyro angular velocity vector $\boldsymbol{\omega}[k]$ will be required.

Application Note AN4697 documents a suitable class of low pass Butterworth filters. The function `fLPFOrientationMatrix` in that document is, however, constructed specifically to low pass filter the six degrees of freedom orientation matrix. The function `fLPFScalar`, listed below, implements the same Butterworth filter, but on a single floating point variable. Three calls are needed in succession, one for each member of the Virtual Gyro angular velocity vector. The filter coefficients should be computed prior to the first call using the function `fInitLPFOrientationMatrix`, also described in AN4697.

```
void fLPFScalar(float *pfSn, float *pfSnm1, float *pfSnm2, float *pfLPSn, float *pfLPSnm1,
    float *pfLPSnm2, int32 loopcounter, float fb0, float fa1, float fa2)
{
    // initialize delay lines on first pass
    if (loopcounter == 0)
    {
        // set S[LP,n-2]=S[LP,n-1]=S[n-2]=S[n-1]=S[n]
        *pfLPSnm2 = *pfLPSnm1 = *pfSnm2 = *pfSnm1 = *pfSn;
    }

    // low pass filter the scalar
    if (ANGLE_LPF_FPU == 1.0F)
    {
        // all pass case
        *pfLPSn = *pfSn;
    }
    else
    {
        // apply the low pass Butterworth filter
        *pfLPSn = fb0 * (*pfSn + 2.0F * *pfSnm1 + *pfSnm2) + fa1 * *pfLPSnm1 + fa2 *
*pfLPSnm2;
    }

    // update the delay line for this iteration
    *pfLPSnm2 = *pfLPSnm1;
    *pfLPSnm1 = *pfLPSn;
    *pfSnm2 = *pfSnm1;
    *pfSnm1 = *pfSn;

    return;
}
```

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

Document Number: AN4685
Rev. 1.0, 5/2013