Understanding Euler Angles

1. Introduction

Attitude and Heading Sensors from CH Robotics can provide orientation information using both Euler Angles and Quaternions. Compared to quaternions, Euler Angles are simple and intuitive and they lend themselves well to simple analysis and control. On the other hand, Euler Angles are limited by a phenomenon called "Gimbal Lock," which we will investigate in more detail later. In applications where the sensor will never operate near pitch angles of +/- 90 degrees, Euler Angles are a good choice.

Sensors from CH Robotics that can provide Euler Angle outputs include the GP9 GPS-Aided AHRS, and the UM7 Orientation Sensor.

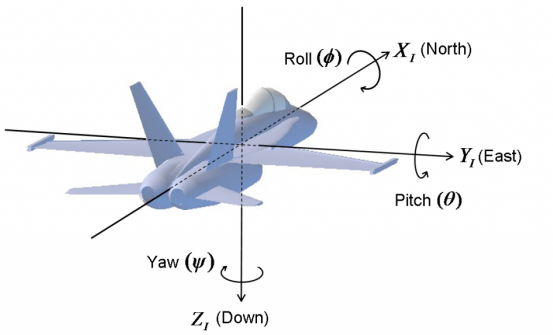


Figure 1 - The Inertial Frame

Euler angles provide a way to represent the 3D orientation of an object using a combination of three rotations about different axes. For convenience, we use multiple coordinate frames to describe the orientation of the sensor, including the "inertial frame," the "vehicle-1 frame," the "vehicle-2 frame," and the "body frame." The inertial frame axes are Earth-fixed, and the body frame axes are aligned with the sensor. The vehicle-1 and vehicle-2 are intermediary frames used for convenience when illustrating the sequence of operations that take us from the inertial frame to the body frame of the sensor.

It may seem unnecessarily complicated to use four different coordinate frames to describe the orientation of the sensor, but the motivation for doing so will become clear as we proceed.

For clarity, this application note assumes that the sensor is mounted to an aircraft. All examples and figures are given showing the changing orientation of the aircraft.

2. The Inertial Frame

The "inertial frame" is an Earth-fixed set of axes that is used as an unmoving reference. CH Robotics' sensors use a common aeronautical inertial frame where the x-axis points north, the y-axis points east, and the z-axis points down as shown below. We will call this a North-East-Down (NED) reference frame. Note that because the z-axis points down, altitude above ground is actually a negative quantity.

The sequence of rotations used to represent a given orientation is first yaw, then pitch, and finally roll.

3. The Vehicle-1 Frame (Yaw Rotation)

As shown in Figure 1, yaw represents rotation about the inertial-frame z-axis by an angle ψ. The yaw rotation produces a new coordinate frame where the z-axis is aligned with the inertial frame and the x and y axes are rotated by the yaw angle ψ. We call this new coordinate frame the vehicle-1 frame. The orientation of the vehicle-1 frame after yaw rotation is show in Figure 2. The vehicle-1 frame axes are colored red, while the inertial frame axes are gray.

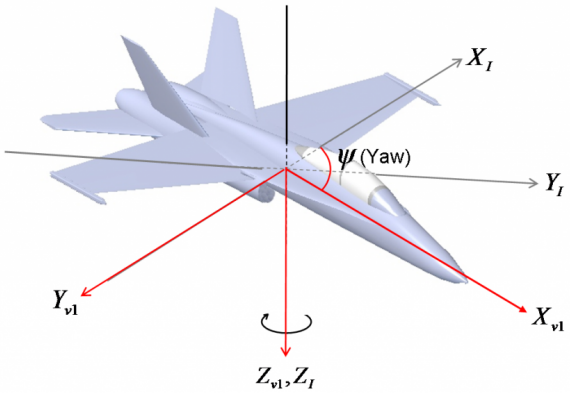


Figure 2 - Yaw rotation into the Vehicle-1 Frame

Rotation of a vector from the Inertial Frame to the Vehicle-1 Frame can be performed by multiplying the vector by the rotation matrix

$$R_I^{v1}(\psi) = \begin{pmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

4. The Vehicle-2 Frame (Yaw and Pitch Rotation)

Pitch represents rotation about the vehicle-1 Y-axis by an angle θ as shown in Figure 3. For clarity, the inertial-frame axes are not shown. The vehicle-1 frame axes are shown in gray, and the vehicle-2 axes are shown in red. It is important to note that pitch is NOT rotation about the inertial-frame Y-axis.

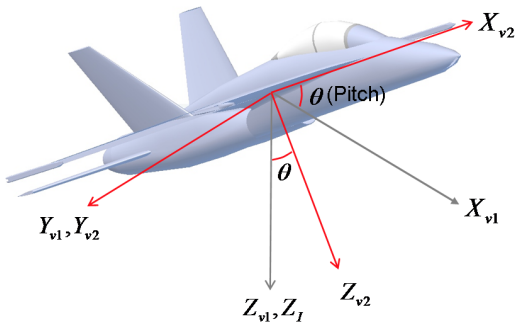


Figure 3 - The Vehicle-3 Frame (Yaw and Pitch Rotation Applied)

The rotation matrix for moving from the vehicle-1 frame to the vehicle-2 frame is given by

$$R_{v1}^{v2}(\theta) = \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{pmatrix}.$$

The rotation matrix for moving from the inertial frame to the vehicle-2 frame consists simply of the yaw matrix multiplied by the pitch matrix:

$$R_I^{v2}(\theta, \psi) = R_{v1}^{v2}(\theta) R_I^{v1}(\psi).$$

5. The Body Frame (Yaw, Pitch, and Roll Rotation)

The body frame is the coordinate system that is aligned with the body of the sensor. On an aircraft, the body frame x-axis typically points out the nose, the y-axis points out the right side of the fuselage, and the z-axis points out the bottom of the fuselage.

The body frame is obtained by performing a rotation by the angle ϕ around the vehicle-2 frame x-axis as shown in Figure 4. For clarity, the inertial frame and vehicle-1 frame axes are not shown. The vehicle-2 frame axes are shown in gray, while the body-frame axes are shown in red.

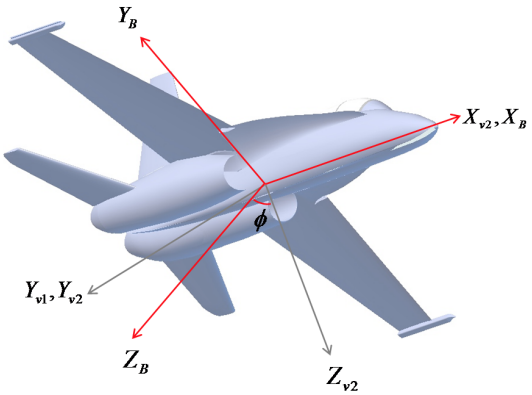


Figure 4 - The Body Frame (Yaw, Pitch, and Roll applied)

The rotation matrix for moving from the vehicle-2 frame to the body frame is given by

$$R^B_{v2}(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{pmatrix}.$$

The complete rotation matrix for moving from the inertial frame to the body frame is given by

$$R^B_I(\phi, \theta, \psi) = R^B_{v2}(\phi) R^{v2}_{v1}(\theta) R^{v1}_I(\psi).$$

Performing the multiplication, and letting *c* represent cos and *s* represent sin, the complete rotation from the inertial frame to the body frame is given by

$$R^B_I(\phi, \theta, \psi) = \begin{pmatrix} c(\psi)c(\theta) & c(\theta)s(\psi) & -s(\theta) \\ c(\psi)s(\phi)s(\theta) - c(\phi)s(\psi) & c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta) & c(\theta)s(\phi) \\ s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta) & c(\phi)s(\psi)s(\theta) - c(\psi)s(\phi) & c(\phi)c(\theta) \end{pmatrix}.$$

The rotation matrix for moving the opposite direction - from the body frame to the inertial frame - is given by

$$R^I_B(\phi, \theta, \psi) = R^{v1}_I(-\psi) R^{v2}_{v1}(-\theta) R^B_{v2}(-\phi).$$

Performing the multiplication, the complete rotation from the body frame to the inertial frame is given by

$$R^B_I(\phi, \theta, \psi) = \begin{pmatrix} c(\psi)c(\theta) & c(\psi)s(\phi)s(\theta) - c(\phi)s(\psi) & s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta) \\ c(\theta)s(\psi) & c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta) & c(\phi)s(\psi)s(\theta) - c(\psi)s(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\phi)c(\theta) \end{pmatrix}.$$

Note that all this does is reverse the order of operations and reverse the direction of rotation.

6. Gimbal Lock

Gimbal lock occurs when the orientation of the sensor cannot be uniquely represented using Euler Angles. The exact orientation at which gimbal lock occurs depends on the order of rotations used. On CH Robotics' sensors, the order of operations results in gimbal lock when the pitch angle is 90 degrees.

Intuitively, the cause of gimbal lock is that when the pitch angle is 90 degrees, yaw and roll cause the sensor to move in exactly the same fashion. Consider Figure 5 for an illustration of the gimbal lock condition. By following the sequence of rotations discussed in this paper, it should be easy to see that the orientation in Figure 5 can be obtained by yawing and then pitching, OR by pitching and then rolling.

An orientation sensor or AHRS that uses Euler Angles will always fail to produce reliable estimates when the pitch angle approaches 90 degrees. This is a fundamental problem of Euler Angles and can only be solved by switching to a different representation method. All CH Robotics attitude sensors use quaternions so that the output is always valid even when Euler Angles are not.

For details about quaternions, please refer to the chapter Understanding Quaternions.

7. Using The Euler Angle Outputs of the Sensor

The rate gyros, accelerometers, and magnetometers on CH Robotics orientation sensors are aligned with the body frame of the sensor, so that if inertial frame data is needed, the sensor outputs must be converted from the body frame to the inertial frame. This can be accomplished by performing a simple matrix multiplication using the matrices described in Section 5.

For example, suppose that we want to obtain inertial frame accelerometer data so that we can integrate acceleration to obtain velocity estimates in the north, east, and down directions. Let $\mathbf{v}_B$ be the measured body-frame acceleration vector reported by the sensor. Then the inertial frame acceleration is given by

$$\mathbf{v}_I = R_B^I(\phi, \theta, \psi)\mathbf{v}_B.$$

The vector $\mathbf{v}_I$ gives us the measured acceleration with respect to the inertial frame. Note that this gives us the *measured* inertial-frame acceleration, not the actual acceleration. A little more work is required before we can extract the physical acceleration of the sensor, and even then, the obtainable velocity accuracy using low-cost sensors is extremely poor. For more details, see Using Accelerometers to Estimate Velocity and Position.

Magnetometer data can also be converted to the inertial frame in exactly the same fashion as the accelerometers if desired.

Converting rate gyro data to the inertial frame is a little more complicated. Like the accelerometer and magnetometer data, the rate gyro data is reported with respect to the body frame of the sensor. This means that the derivative of your Euler Angles is NOT what is being reported by the rate gyros. If you want Euler Angle rates, the rate gyro data must be converted to the proper coordinate frames. This is a little more complicated than it was for the accelerometers and magnetic sensors because each gyro angular rate must be converted to a different coordinate frame. Recall that yaw represents rotation about the inertial frame z-axis, pitch represents rotation about the vehicle-1 frame y-axis, and roll represents rotation about the vehicle-2 frame x-axis. Then to get the angular rates in the proper frames, the z-axis gyro output must be rotated into the inertial frame, the y-axis gyro output must be rotated into the vehicle-1 frame, and the x-axis gyro output must be rotated into the vehicle-2 frame.

The resulting transformation matrix for converting body-frame angular rates to Euler angular rates is given by

$$D(\phi, \theta, \psi) = \begin{pmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)/\cos(\theta) & \cos(\phi)/\cos(\theta) \end{pmatrix}.$$

Let *p* represent the body-frame x-axis gyro output, *q* represent the body-frame y-axis gyro output, and *r* represent the body-frame z-axis output. Then it follows that the Euler Angle rates are computed as

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} p + q\sin(\phi)\tan(\theta) + r\cos(\phi)\tan(\theta) \\ q\cos(\phi) - r\sin(\phi) \\ q\sin(\phi)/\cos(\theta) + r\cos(\phi)/\cos(\theta) \end{pmatrix}.$$

This operation illustrates mathematically why gimbal lock becomes a problem when using Euler Angles. To estimate yaw, pitch, and roll rates, gyro data must be converted to their proper coordinate frames using the matrix *D*. But notice that there is a division by $\cos(\theta)$

in two places on the last row of the matrix. When the pitch angle approaches +/- 90 degrees, the denominator goes to zero and the matrix elements diverge to infinity, causing the filter to fail.

The conversion from body-frame gyro data to Euler Angle rates happens internally on all CH Robotics sensors, but the converted data is not made available on all CH Robotics products. Refer to specific device datasheets for details on what data is available. For devices where Euler Angle rates are not reported, the body-frame angular rate data can be converted as described above.

Understanding Quaternions

1. Introduction

Attitude and Heading Sensors from CH Robotics can provide orientation information using both Euler Angles and Quaternions. Compared to quaternions, Euler Angles are simple and intuitive and they lend themselves well to simple analysis and control. On the other hand, Euler Angles are limited by a phenomenon called "gimbal lock," which prevents them from measuring orientation when the pitch angle approaches +/- 90 degrees.

Quaternions provide an alternative measurement technique that does not suffer from gimbal lock. Quaternions are less intuitive than Euler Angles and the math can be a little more complicated. This application note covers the basic mathematical concepts needed to understand and use the quaternion outputs of CH Robotics orientation sensors.

Sensors from CH Robotics that can provide quaternion orientation outputs include the UM6 Orientation Sensor and the UM6-LT Orientation Sensor.

2. Quaternion Basics

A quaternion is a four-element vector that can be used to encode any rotation in a 3D coordinate system. Technically, a quaternion is composed of one real element and three complex elements, and it can be used for much more than rotations. In this application note we'll be ignoring the theoretical details about quaternions and providing only the information that is needed to use them for representing the attitude of an orientation sensor.

The attitude quaternion estimated by CH Robotics orientation sensors encodes rotation from the "inertial frame" to the sensor "body frame." The inertial frame is an Earth-fixed coordinate frame defined so that the x-axis points north, the y-axis points east, and the z-axis points down as shown in Figure 1. The sensor body-frame is a coordinate frame that remains aligned with the sensor at all times. Unlike Euler Angle estimation, only the body frame and the inertial frame are needed when quaternions are used for estimation (Understanding Euler Angles provides more details about using Euler Angles for attitude estimation).

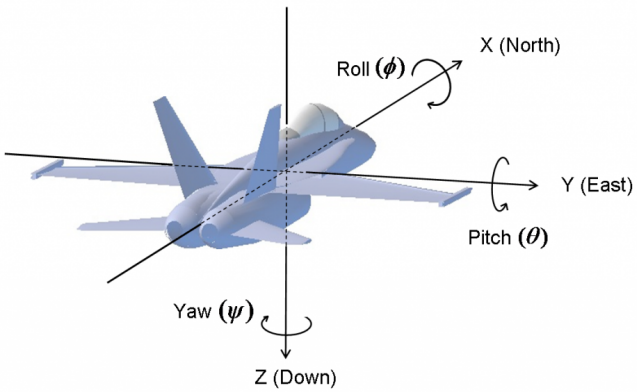


Figure 1 - The Inertial Frame

Let the vector $\mathbf{q}_i^b$ be defined as the unit-vector quaternion encoding rotation from the inertial frame to the body frame of the sensor:

$$\mathbf{q}_i^b = \begin{pmatrix} a & b & c & d \end{pmatrix}^T.$$

where T is the vector transpose operator. The elements *b*, *c*, and *d* are the "vector part" of the quaternion, and can be thought of as a vector about which rotation should be performed. The element *a* is the "scalar part" that specifies the amount of rotation that should be performed about the vector part. Specifically, if θ is the angle of rotation and the vector $\begin{pmatrix} v_x & v_y & v_z \end{pmatrix}^T$ is a unit vector representing the axis of rotation, then the quaternion elements are defined as

$$\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} \cos(0.5\theta) \\ v_x \sin(0.5\theta) \\ v_y \sin(0.5\theta) \\ v_z \sin(0.5\theta) \end{pmatrix}.$$

In practice, this definition needn't be used explicitly, but it is included here because it provides an intuitive description of what the quaternion represents. CH Robotics sensors output the quaternion $\mathbf{q}_i^b$ when quaternions are used for attitude estimation.

3. Rotating Vectors Using Quaternions

The attitude quaternion $\mathbf{q}_i^b$ can be used to rotate an arbitrary 3-element vector from the inertial frame to the body frame using the operation

$$\mathbf{v}_B = \mathbf{q}_i^b \begin{pmatrix} 0 \\ \mathbf{v}_I \end{pmatrix} (\mathbf{q}_i^b)^{-1}.$$

That is, a vector can rotated by treating it like a quaternion with zero real-part and multiplying it by the attitude quaternion and its inverse. The inverse of a quaternion is equivalent to its conjugate, which means that all the vector elements (the last three elements in the vector) are negated. The rotation also uses quaternion multiplication, which has its own definition.

Define quaternions $\mathbf{q}_1 = \begin{pmatrix} a_1 & b_1 & c_1 & d_1 \end{pmatrix}^T$ and $\mathbf{q}_2 = \begin{pmatrix} a_2 & b_2 & c_2 & d_2 \end{pmatrix}^T$. Then the quaternion product $\mathbf{q}_1 \mathbf{q}_2$ is given by

$$\mathbf{q}_1 \mathbf{q}_2 = \begin{pmatrix} a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2 \\ a_1 b_2 + b_1 a_2 + c_1 d_2 - d_1 c_2 \\ a_1 c_2 - b_1 d_2 + c_1 a_2 + d_1 b_2 \\ a_1 d_2 + b_1 c_2 - c_1 b_2 + d_1 a_2 \end{pmatrix}.$$

To rotate a vector from the body frame to the inertial frame, two quaternion multiplies as defined above are required. Alternatively, the attitude quaternion can be used to construct a 3x3 rotation matrix to perform the rotation in a single matrix multiply operation. The rotation matrix from the inertial frame to the body frame using quaternion elements is defined as

$$R_i^b(\mathbf{q}_i^b) = \begin{pmatrix} a^2 + b^2 - c^2 - d^2 & 2bc - 2ad & 2bd + 2ac \\ 2bc + 2ad & a^2 - b^2 + c^2 - d^2 & 2cd - 2ab \\ 2bd - 2ac & 2cd + 2ab & a^2 - b^2 - c^2 + d^2 \end{pmatrix}.$$

Then the rotation from the inertial frame to the body frame can be performed using the matrix multiplication

$$\mathbf{v}_B = R_i^b(\mathbf{q}_i^b)\mathbf{v}_I.$$

Regardless of whether quaternion multiplication or matrix multiplication is used to perform the rotation, the rotation can be reversed by simply inverting the attitude quaternion before performing the rotation. By negating the vector part of the quaternion vector, the operation is reversed.

4. Converting Quaternions to Euler Angles

CH Robotics sensors automatically convert the quaternion attitude estimate to Euler Angles even when in quaternion estimation mode. This means that the convenience of Euler Angle estimation is made available even when more robust quaternion estimation is being used.

If the user doesn't want to have the sensor transmit both Euler Angle and Quaternion data (for example, to reduce communication bandwidth requirements), then the quaternion data can be converted to Euler Angles on the receiving end.

The exact equations for converting from quaternions to Euler Angles depends on the order of rotations. CH Robotics sensors move from the inertial frame to the body frame using first yaw, then pitch, and finally roll. This results in the following conversion equations:

$$\phi = \arctan\left(\frac{2(ab+cd)}{a^2-b^2-c^2+d^2}\right),$$

$$\theta = -\arcsin(2(bd-ac)), \text{ and}$$

$$\psi = \arctan\left(\frac{2(ad+bc)}{a^2+b^2-c^2-d^2}\right).$$

See the chapter on Understanding Euler Angles for more details about the meaning and application of Euler Angles. When converting from quaternions to Euler Angles, the atan2 function should be used instead of atan so that the output range is correct. Note that when converting from quaternions to Euler Angles, the gimbal lock problem still manifests itself. The difference is that since the estimator is not using Euler Angles, it will continue running without problems even though the Euler Angle output is temporarily unavailable. When the estimator runs on Euler Angles instead of quaternions, gimbal lock can cause the filter to fail entirely if special precautions aren't taken.

Sensors for Orientation Estimation

1. Introduction

CH Robotics AHRS and Orientation Sensors combine information from accelerometers, rate gyros, and (in some cases) GPS to produce reliable attitude and heading measurements that are resistant to vibration and immune to long-term angular drift.

Part of what makes reliable orientation estimates possible is the complementary nature of the different kinds of sensors used for estimation. Rate gyros provide good short-term stability and resistance to vibration, accelerometers can provide attitude information that does not become less reliable over time, and magnetic sensors provide heading information in addition to limited attitude information (pitch and roll). By combining data from each type of sensor using a filter such as an Extended Kalman Filter, accurate orientation estimates can be computed.

To fully understand the capabilities and limitations of CH Robotics devices, it helps to have a clear understanding of the operation of the different types of sensors that used to compute attitude and heading. In this application note we describe in detail the operation of accelerometers, rate gyros, and magnetic sensors. We present high level descriptions of how the different types of sensors work, in addition to mathematical models used to quantify the performance characteristics of each type of sensor.

2. Accelerometers

The output of a three-axis accelerometer mounted to a rigid body can be modeled as

$$\mathbf{a}_m = \frac{1}{m}(\mathbf{F} - \mathbf{F}_g),$$

where $\mathbf{a}_m$ is the measured acceleration, *m* is the mass of the body, $\mathbf{F}$ is the sum of all forces on the body (including gravity) expressed in the sensor body frame, and $\mathbf{F}_g$ is the force due to gravity, also expressed in the body frame (See the chapter on Understanding Euler Angles for details about different coordinate frames).

The accelerometer model can be intuitively explained using the highly simplified one-axis accelerometer diagram shown in Figure 1. As shown, an accelerometer can be constructed by attaching a proof-mass to a lever-arm. Any upward or downward acceleration of the sensor in Figure 1 causes deflection of the proof mass, which can be measured to determine acceleration (below the natural frequency of the lever/mass system, deflection is proportional to acceleration). This accounts for the first force term, $\mathbf{F}$, in the accelerometer model: the sum of all forces on the body, including gravity, produces physical acceleration of the sensor, which in turn causes deflection of the proof mass.

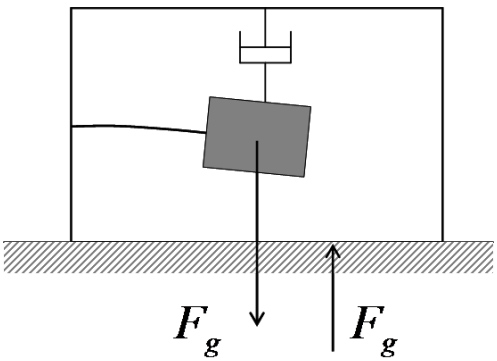


Figure 1 - Simplified One-Axis Accelerometer Diagram

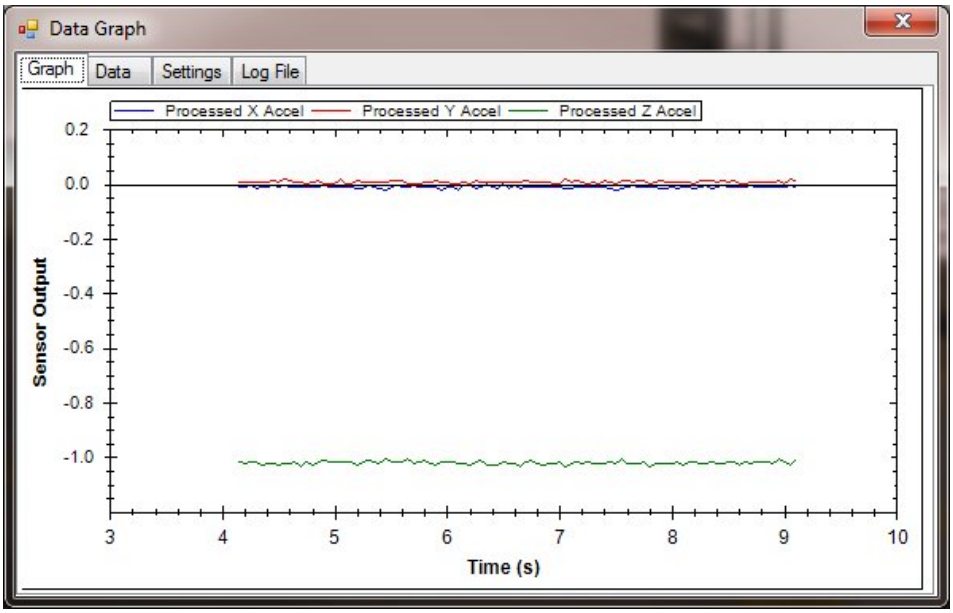
The second force term, $\mathbf{F}_g$, is present in the accelerometer model because the force of gravity not only accelerates the sensor body, it also causes deflection of the proof-mass itself. If the accelerometer is not accelerating ($\mathbf{F} = 0$), then gravity produces a downward deflection of the proof mass that appears equivalent to upward acceleration of the sensor at the acceleration of gravity. Similarly, if the accelerometer is in free-fall ($\mathbf{F} = \mathbf{F}_g$) there is no deflection and hence no measured acceleration despite the fact that the accelerometer is accelerating at 9.8 m/s/s toward the center of the earth.

Our accelerometer model suggests that the expected accelerometer measurement can be obtained by drawing a free-body diagram that includes all forces except the force of gravity: the first term, $\mathbf{F}$, includes gravity, while the second term, $\mathbf{F}_g$, removes it. From this perspective, accelerometers never measure gravity directly - they measure the normal forces that prevent the sensor from accelerating toward the center of the Earth (in addition to other external forces).

Let us assume that the external forces acting on the sensor are trivial in comparison with the normal forces. Then the accelerometer output can be approximately modeled as

$$\mathbf{a}_m = -\frac{\mathbf{F}_g}{m},$$

from which it is easy to back out the pitch and roll angles of the sensor by simply noting which direction gravity the gravity vector points. For example, if it points straight down, then the sensor is level with pitch = roll = 0. This is illustrated in Figure 1. Notice that the x and y accelerometer outputs are nearly zero, while the z-axis output is negative one gravity.



Accelerometer output with sensor flat and level on a table-top

On the UM7 and the UM7-LT, it is assumed that in general the external forces acting on the accelerometer are close to zero, so that the pitch and roll angles can be measured directly. When this assumption is not valid (i.e. on an airplane, a helicopter, motorcycle, or anything that spends a lot of time accelerating aggressively), the angle estimates of the sensor can degrade. This will be true of *any* orientation sensor or AHRS that doesn't utilize an external aid like GPS.

The simplified accelerometer model given above does not include terms to account for cross-axis alignment, scale factors, and output bias. All of these items can increase the amount of error in the output of the sensor, affecting the accuracy of orientation estimates and (depending on how the accelerometer is being used) on the accuracy of dead-reckoning velocity and position estimates. A more complete model of the accelerometer output is given by

$$\mathbf{a}_m = M_a \left(\tfrac{1}{m} S_a(T)(\mathbf{F} - \mathbf{F}_g) - \beta(T) \right),$$

where M_a is the accelerometer misalignment matrix, $\beta(T)$ is a vector of temperature-varying output biases, and $S_a(T)$ is the diagonal temperature-varying accelerometer sensitivity matrix given by

$$S_a(T) = \begin{pmatrix} S_{a,x} & 0 & 0 \\ 0 & S_{a,y} & 0 \\ 0 & 0 & S_{a,z} \end{pmatrix}.$$

The sensitivity matrix is what encodes the expected accelerometer raw output for a given measured acceleration. The misalignment matrix describes the effect of cross-axis misalignment and, unlike the bias and sensitivity terms, it is not affected be temperature.

When using accelerometer data, we measure $\mathbf{a}_m$ but what we really want to know is the actual acceleration before scale factors, biases, and misalignment distort the measurement. That is, we want to take the measurement and extract the term $(\mathbf{F} - \mathbf{F}_g)/m$. Solving, we get

$$\mathbf{a}_{m,c} = S_a^{-1}(T)(M_a^{-1}\mathbf{a}_m + \beta_a(T)),$$

where $\mathbf{a}_{m,c}$ is defined as the corrected accelerometer measurement vector. In order to obtain the best accuracy, the terms $S_a^{-1}(T)$, M_a^{-1}, and $\beta_a(T)$ should be determined experimentally over the full operating temperature of the sensor. At the time of writing, the UM7 and the UM7-LT are not calibrated in-factory so that costs can be kept low. The GP9 is fully calibrated over temperature.

3. Rate Gyros

Let the vector $\mathbf{p}$ be defined as

$$\mathbf{p} = \begin{pmatrix} p \\ q \\ r \end{pmatrix},$$

where *p* is the actual angular rate about the body-frame x-axis, *q* is the actual angular rate about the body-frame y-axis, and *r* is the actual angular rate about the body-frame z-axis. Then the actual angular rate can be described in terms of the measured angular rate given the expression

$$\mathbf{p} = S_g^{-1}(T)(M_g^{-1}\mathbf{p}_m + \beta_g(T) - C_{g,a}\mathbf{a}_{m,c}),$$

where $\mathbf{p}_m$ is the measured angular rate vector, $C_{g,a}$ is a matrix encoding the sensitivity of the rate gyro axes to acceleration, and $\mathbf{a}_{m,c}$ is the

corrected accelerometer measurement vector. Usually coupling between the acceleration and rate gyro output is small enough that $C_{g,a}$ can be neglected, but in cases where high acceleration is expected (e.g. on a rocket), this term must be measured and included in the model for the best accuracy.

For best accuracy, all calibration terms should be determined experimentally so that compensation can be performed. However, of all the terms affecting the output error of the gyro measurement, the gyro bias and sensitivity are the largest contributors of error, especially over a wide temperature range. If there is insufficient equipment available to perform a complete calibration, the most important calibration procedure is to remove temperature-induced output bias changes, followed by the removal of temperature-induced scale factor changes.

At the time of writing, CH Robotics keeps costs low by not calibrating the gyros on the UM7 and the UM7-LT in-house. The GP9 does include full calibration.

4. Magnetometers

Of all the sensors used by CH Robotics for attitude and heading estimation, magnetometers are perhaps the most tricky to work with for a variety of reasons. First, like accelerometers and rate gyros, magnetic sensors are subject to cross-axis misalignment and temperature-varying scale factors and biases. In addition, the local magnetic field around the magnetometer can be distorted by magnetic and ferrous metal objects. This distortion must be measured and corrected in order for the magnetometer to be useful. The field can also be distorted by time-varying electromagnetic waves from high-power electrical lines and communication lines placed near the sensor. Finally, as the sensor is moved around in practice, the field can be distorted so that, in certain locations (say, next to a concrete wall filled with rebar), the accuracy of the measurement is reduced. All of these factors places some limitations on the use of sensors that rely on magnetometers for heading estimation.

We begin with a sensor model essentially equivalent to the gyro and accelerometer models already presented, resulting in the following equation for correcting cross-axis misalignment, biases, and scale factors:

$$\mathbf{b} = S_b^{-1}(T)(M_b^{-1}\mathbf{b}_m + \beta_b(T)).$$

Similar to the rate gyro and accelerometer cases already discussed, the vector $\mathbf{b}$ represents the actual magnetic field, $\mathbf{b}_m$ is the measured magnetic field, S_a^{-1} is the inverse of the magnetometer sensitivity matrix, M_b^{-1} is the inverse of the magnetometer misalignment matrix, and $\beta_b(T)$ is the magnetometer bias vector.

After the magnetometer measurement is corrected for axis misalignment, scale factor, and bias errors, it must be corrected for additional local magnetic field distortions. There are two types of distortions that affect the measurement: soft-iron distortion and hard-iron distortions.

A soft-iron field distortion is caused by ferrous metal objects that bend the Earth's magnetic field. For example, the screws in the UM7 enclosure bend the local magnetic field. To obtain reasonable accuracy, this distortion must be calibrated out of the measurement. This is actually done in-factory for the UM7. A hard-iron distortion is caused by an object (like the permanent magnets in a motor that has its own permanent magnetic field.

In the absence of any hard or soft-iron distortions, the outputs of the magnetometer as it is rotated should trace out a perfect sphere centered around zero. Soft-iron distortions distort the sphere so that it appears to be an ellipsoid. Hard-iron distortions shift the mean of the sphere away from zero.

One way to calibrate the magnetometer for hard and soft iron distortions is to fit an ellipsoid to a set of magnetometer data collected over a wide variety of different orientations. The center of the ellipsoid is equivalent to the bias caused by hard iron distortions, while the ellipsoid shape on the x, y, and z axes are caused by the soft-iron distortions. A correction matrix can then be calculated that, when multiplied by the magnetometer measurements, alters the ellipsoid so that it looks like a sphere again. This is essentially the approach taken by the magnetometer calibration algorithm that is included in the CHR Serial Interface software.

There are many other methods for calibrating magnetometers, but in most cases the calibration algorithm produces a matrix and a vector such that the corrected magnetometer measurement is given by

$$\mathbf{b}_c = D_s\mathbf{b} - \beta_h.$$

Combining the soft and hard-iron corrections with the calibrated magnetometer measurement, we get

$$\mathbf{b}_c = D_s S_a^{-1}(T)(M_a^{-1}\mathbf{a}_m + \beta_a(T)) - \beta_h.$$

Terms in the above equation can be combined so that there are fewer calibration terms to estimate during calibration, although the process is complicated by the fact that the bias, scale factor, and cross-axis alignment terms will remain constant, while the hard and soft iron calibration terms will change depending on where the end-user mounts the sensor in the end application.

Fundamentals of Attitude Estimation

1. Introduction

When estimating attitude and heading, the best results are obtained by combining data from multiple types of sensors to take advantage of their relative strengths. For example, rate gyros can be integrated to produce angle estimates that are reliable in the short-term, but that tend to drift in the long-term. Accelerometers, on the other hand, are sensitive to vibration and other non-gravity accelerations in the short-term, but can be trusted in the long-term to provide angle estimates that do no degrade as time progresses. Combining rate gyros and accelerometers can produce the best of both worlds - angle estimates that are resistant to vibration and immune to long-term angular drift.

In this article, we describe a simple method for combining rate gyros and accelerometers to produce reliable pitch and roll angle estimates. The GP9 and the UM7 use methods that are more sophisticated than what is described here, but the basic intuition behind both methods is the same - gyros are used to give good short-term stability, while accelerometers are used for their better long-term stability.

Note that there are some problems with the simple method described here. We discuss these problems in more detail near the end of the article.

To fully understand the contents of this article, you should first read Understanding Euler Angles and Sensors for Orientation Estimation.

2. Estimating angles with accelerometers

Recall that the output of a three-axis accelerometer can be modeled as

$$\mathbf{a}_m = \frac{1}{m}(\mathbf{F} - \mathbf{F}_g),$$

where $\mathbf{a}_m$ is the measured acceleration, m is the mass of the body, $\mathbf{F}$ is the sum of all forces on the body (including gravity) expressed in the sensor body frame, and $\mathbf{F}_g$ is the force due to gravity, also expressed in the body frame.

If we assume that in general $\mathbf{F} = 0$, then the output of the accelerometer is given by

$$\mathbf{a}_m = -\frac{\mathbf{F}_g}{m}.$$

In the inertial frame, the force of gravity is given by the vector $\begin{pmatrix} 0 & 0 & mg \end{pmatrix}^T$. That is, there is no force on the x and y axes, and there is mg of force on the z axis.

Rotating the gravity into the body-frame of the sensor, we get

$$\mathbf{F}_g = R_I^B \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} = \begin{pmatrix} -mg\sin(\theta) \\ mg\cos(\theta)\sin(\phi) \\ mg\cos(\phi)\cos(\theta) \end{pmatrix}.$$

Thus, the expected accelerometer measurement in the body-frame is given by

$$\mathbf{a}_m = \begin{pmatrix} g\sin(\theta) \\ -g\cos(\theta)\sin(\phi) \\ -g\cos(\phi)\cos(\theta) \end{pmatrix}.$$

Let the components of $\mathbf{a}_m$ be given by $a_{m,x}$, $a_{m,y}$, and $a_{m,z}$. Then we can solve for the pitch (θ) and roll (ϕ) angles using

$$\hat{\theta}_{accel} = \arcsin\left(\frac{a_{m,x}}{g}\right), \text{ and}$$

$$\hat{\phi}_{accel} = \arctan\left(\frac{a_{m,y}}{a_{m,z}}\right).$$

The ^ symbol above the angles indicates that they are estimated angles, not actual angles.

This method provides a quick, simple way to estimate pitch and roll using only accelerometers. However, note that we assumed that the only force acting on the accelerometers was gravity. Vibration and other external forces will directly influence our measured pitch and roll angles, making them very noisy and, in many cases, unusable by themselves.

3. Estimating angles with rate gyros

Rate gyros can also be used to measure angles. Unlike accelerometers, rate gyros are nominally unaffected by acceleration, making gyro-based angle estimates immune to external forces that make accelerometers unreliable.

Remember that rate gyros measure angular rates about the *x*, *y*, and *z* axes in the sensor *body-frame*. We would like to take those measurements

and use them to figure out our Euler Angle rates. Since roll, pitch, and yaw each occur in different reference frames, we need to take the rate gyro outputs and rotate them into the appropriate frames (more details about this process can be found here).

Let the variables *p*, *q* , and *r* represent the rate gyro outputs on the *x*, *y*, and *z* axes, respectively. Then the Euler Angle rates are given by

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} p + q\sin(\phi)\tan(\theta) + r\cos(\phi)\tan(\theta) \\ q\cos(\phi) - r\sin(\phi) \\ q\sin(\phi)/\cos(\theta) + r\cos(\phi)/\cos(\theta) \end{pmatrix}.$$

Since we are only estimating pitch and roll in this example, we will ignore the term for yaw (ψ).

Assuming that the rate gyros are sampled at some fixed interval T, we can estimate pitch and roll using Euler Integration as follows:

$\hat{\phi}^+ = \hat{\phi} + T\dot{\phi}$, and

$\hat{\theta}^+ = \hat{\theta} + T\dot{\theta}$.

In the preceding equations, we use the + symbol to distinguish the new angle estimate from the old angle estimate.

This method allows us to use rate gyros to estimate angles that aren't sensitive to vibration and other external forces. However, because rate gyros are noisy and imperfect, every time we add new gyro measurements, we add errors to our angle estimates. Over time, errors will accumulate, causing our gyro-based angles estimates to drift over time. We will solve this problem in the next section.

4. Combining accelerometer and rate gyro data

As described previously, angle estimates based on rate gyros alone drift over time, making them unreliable in the long-term. In contrast, accelerometers do not cause angle estimates to drift, but they are sensitive to external forces like vibration, making short-term estimates unreliable. In this section, we combine the outputs of both types of sensor to produce angle estimates that are resistant to vibration and immune to long-term drift.

The estimation process will be split into two steps, a "prediction" step where we will use the rate gyros to estimate incremental changes in the angle, and an "update" step where we will use the accelerometers to correct rate gyro drift.

The prediction step consists of angular rate integration as we saw in the rate gyro section:

$\hat{\theta}^+ = \hat{\theta} + T\dot{\theta}$, and

$\hat{\phi}^+ = \hat{\phi} + T\dot{\phi}$.

In the update step, we assume that the accelerometer-based angle is usually something close to the truth. We take the measured angle (from the accelerometers) and the predicted angle (from the rate gyros), compute the difference, and add a portion of the difference to the final angle estimate as shown below:

$\hat{\theta} = \hat{\theta}^+ + L(\hat{\theta}_{accel} - \hat{\theta}^+)$,

$\hat{\phi} = \hat{\phi}^+ + L(\hat{\phi}_{accel} - \hat{\phi}^+)$

The value L is a constant value between 0 and 1. Notice that if L is equal to 0, then only the gyro-based angle is used because no accelerometer-based correction is applied. Similarly, if L is equal to 1, then the estimated angle is equivalent to the accelerometer-based angle.

As L approaches 1, the accelerometers are trusted more and the rate gyros are trusted less. This makes the angle estimates less sensitive to rate gyro noise and biases, but more sensitive to vibration and other external forces.

As L approaches 0, the rate gyros are trusted more and the accelerometers are trusted less. This makes the angle estimates more sensitive to rate gyro noise and biases, but less sensitive to vibration and other external forces.

In practice, L can be adjusted based on the desired behavior of the angle estimates, but having high-quality rate gyros allows a lower gain to be used, preventing unwanted accelerations from affecting the angle estimates. From an intuitive standpoint, the rate gyros estimate the angle in the short-term while the accelerometers gradually pull the angle estimate toward the average angle, whatever that happens to be.

This formulation can be called a "fixed-gain observer," and it bears a lot of similarity to a Kalman Filter. The main difference is that in a Kalman Filter, the observer gain L is selected optimally using known characteristics of the physical system. In addition, a Kalman Filter can exploit knowledge of the physical system so that accelerometer data (and other data) needn't be converted to angles before using it to make corrections to the angle estimates.

The UM7 and the UM7-LT both use Kalman Filters to optimally combine rate gyro and accelerometer data.

5. Limitations

The downside of this method is that accelerometers can't always be expected to provide reliable estimates of the actual angle. In some dynamic

systems (most notably in aircraft), accelerometers can provide incorrect angle information for arbitrarily large amounts of time. This causes the angle estimates to degrade and possibly even become completely unreliable.

More sophisticated filters combine inertial sensors with other types of sensors (like GPS) to allow the angle estimates to maintain their accuracy even during sustained, high-G maneuvers. Our GP9: GPS-Aided AHRS is an example of a sensor that performs better on moving vehicles.

Using Accelerometers to Estimate Position and Velocity

1. Introduction

We are commonly asked whether it is possible to use the accelerometer measurements from CH Robotics orientation sensors to estimate velocity and position. The short answer is "yes and no." It depends entirely on how much accuracy is needed. In general, accelerometer-based position and velocity estimates from low-cost sensors (hundreds of US dollars instead of tens of thousands) are very poor and are simply unusable. This isn't because the accelerometers themselves are poor, but because the orientation of the sensor must be known with a high degree of accuracy so that gravity measurements can be distinguished from the physical acceleration of the sensor. Even small errors in the orientation estimate will produce extremely high errors in the measured acceleration, which translate into even larger errors in the velocity and position estimates. Without nicer rate gyros (FOG or Ring-Laser, for example) or without the addition of an external reference like GPS, accurate dead-reckoning is usually not possible.

Nevertheless, not all application require a great deal of accuracy, and sometimes absolute accuracy is not as important as the ability to measure short-term deviations in velocity and position. For an overview of the kinds of accuracy you can expect to obtain using CH Robotics sensors for dead-reckoning, skip to Section 4 - Accuracy of Velocity and Position Estimates.

In this application note, we discuss what is required to get velocity and position estimates using data from CH Robotics sensors that do not already provide accelerometer-based velocity and position as standard outputs (at the time of writing, this includes the UM7 and the UM7-LT). We also discuss the expected accuracy of the acceleration measurements, and how that accuracy will affect the reliability of the resulting velocity and position estimates.

This application note assumes familiarity with coordinate frames used for attitude estimation on CH Robotics sensors. For simplicity, it also assumes that Euler Angle ouputs are being used instead of the quaternion outputs of the sensor, although the discussion applies equally well to quaternion outputs. For details about coordinate frames and Euler Angles, see the library chapter on Understanding Euler Angles.

2. Extracting Inertial-Frame Acceleration

To get inertial frame velocities and positions, it is first necessary to obtain the physical acceleration of the sensor in the inertial frame. To convert the accelerometer measurement into actual physical acceleration of the sensor, it is important to first understand exactly what the accelerometer is measuring. Application note AN1008 describes accelerometer behavior in detail, so the complete discussion won't be included here. To summarize, the accelerometer measures both the physical acceleration of the sensor, AND the contribution of normal forces that prevent the accelerometer from accelerating toward the center of the Earth (i.e. the force exerted on the accelerometer by the table it is sitting on is actually measured by the sensor). To measure only the component of acceleration that is caused by physical acceleration, normal forces must be removed. As described in application note AN1008, the 3-axis accelerometer measurement vector $\mathbf{a}_m$ can be modeled as

$$\mathbf{a}_m = \mathbf{a}_B - R_I^B \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix},$$

where $\mathbf{a}_B$ is the actual body-frame acceleration, g is the acceleration of gravity, and R_I^B is the rotation matrix from the inertial frame to the body frame of the sensor. This model assumes that there are no cross-axis alignment, scale factor, or bias errors in the measurement.

In order to estimate the inertial frame velocity and position of the sensor, we need to remove the normal force component from the acceleration measurement. Solving for the body frame acceleration we get

$$\mathbf{a}_B = \mathbf{a}_m + R_I^B \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix}.$$

The body-frame acceleration must then be rotated into the inertial frame so that it can be integrated to obtain velocity and position. This yields

$$\mathbf{a}_I = R_B^I \mathbf{a}_m + \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix}.$$

This equation can be used directly to measure the inertial frame acceleration of the sensor. Note that CH Robotics sensors typically output acceleration in units of gravities, not m/s/s or ft/s/s. Thus the term should g either be 1 gravity, or $\mathbf{a}_m$ must be converted to the desired units. We recommend using g =9.8 m/s/s and multiplying by the same so that the inertial acceleration obtained is in m/s/s. Then integrating the acceleration measurement will yield a velocity in m/s, and a position in meters.

3. Estimating Velocity and Position

Once the measured inertial-frame acceleration is obtained, it can be integrated to obtain inertial frame velocity $\mathbf{v}_I$ and position $\mathbf{r}_I$:

$$\mathbf{v}_I = \int \mathbf{a}_I,$$

$$\mathbf{r}_I = \iint \mathbf{a}_I.$$

In practice, data is obtained at discrete time intervals so that the estimated velocity and position are estimated using

$$\mathbf{v}_I[k+1] = \mathbf{v}_I[k] + T\mathbf{a}_I[k],$$

$$\mathbf{r}_I[k+1] = \mathbf{r}_I[k] + T\mathbf{v}_I[k].$$

where *T* is the sample period. Not that depending on the hardware used for communicating with the sensor, the sampling period may not be constant, and should therefore be measured when making estimates. This is easier if the sensor data is being read using a microcontroller or a computer with an RTOS. A standard windows PC will introduce unpredictable delays in the actual arrival time of serial data, which will cause timing accuracy issues.

4. Accuracy of Velocity and Position Estimates

In Section 2, we discussed the operations needed to convert measured acceleration to physical (non-gravity) acceleration in the inertial frame. Recall that making the conversion requires that we use the current orientation estimate to rotate the measurement into the inertial frame.

Note that there is a matrix rotation to transform the body-frame measurement to the inertial frame, following by the addition of the expected gravity vector in the inertial frame. This addition removes the measurement of normal forces that don't cause physical acceleration of the sensor.

If the orientation estimate were perfect, this step would introduce no additional error into the acceleration measurement and the only error contribution would come from the inaccuracy of the accelerometer itself.

Since in practice we never know the sensor's orientation perfectly, the addition of the gravity term in will fail to remove measured normal forces, and those forces will be mistaken for physical acceleration. It turns out that on low-cost sensors (hundreds of US dollars instead of tens of thousands), orientation error will cause acceleration measurement errors that may dwarf errors from accelerometer misalignment and miscalibration. The magnitude of these errors will be large enough that accelerometer-based position and velocity estimates will be unusable for most applications.

Table 1 summarizes the acceleration, velocity, and position errors that can be expected given different errors in the orientation estimate of the sensor.

Angle Error (degrees)	Acceleration Error (m/s/s)	Velocity Error (m/s) at 10 seconds	Position Error (m) at 10 seconds	Position Error (m) at 1 minute	Position Error (m) at 10 minutes	Position Error (m) at 1 hour
0.1	0.017	0.17	1.7	61.2	6120	220 e 3
0.5	0.086	0.86	8.6	309.6	30960	1.1 e 6
1.0	0.17	1.7	17	612	61200	2.2 e 6
1.5	0.256	2.56	25.6	921.6	92160	3.3 e 6
2.0	0.342	3.42	34.2	1231.2	123120	4.4 e 6
3.0	0.513	5.13	51.3	1846.8	184680	6.6 e 6
5.0	0.854	8.54	85.4	3074.4	307440	11 e 6

Table 1 - A summary of velocity and position errors caused by attitude estimation error.

As shown, an angle error of even one degree will cause the estimated velocity to be off by 1.7 m/s after 10 seconds, and the position to be off by 17.1 meters in the same amount of time. After a single minute, one degree of angle error will cause the position estimate to be off by almost a full kilometer. After ten minutes, the position will be off by 62 kilometers.

From the factory, the accuracy of the UM7 and the UM7-LT sensors can be expected to be within about 2 degrees of actual during normal operating conditions. Using this as an upper bound, the UM7 or the UM7-LT could be used for dead-reckoning velocity and position estimation as long as 34.2 meters of positioning error at 10 seconds and 3.4 m/s of velocity error at 10 seconds wouldn't be a problem.

There is a silver-lining to the fact that attitude errors makes it hard to estimate position using accelerometers: because acceleration error is tightly coupled to position and velocity error, it is actually possible to use velocity and position measurements to estimate attitude! Our GP9 does this. More details about how we do that can be found here.

It is also worth considering that this analysis only considers the effect of orientation estimate errors. Accelerometer miscalibration will also cause

measurement errors, particularly when uncalibrated sensors are used over wide temperature ranges. The change in the scale factors and biases can be significant over the full operating temperature range, so that calibration becomes critically important.

In summary, it is possible to use the accelerometers to estimate velocity and position, but the accuracy will be very poor so that in most cases, the estimates are not useful.