ESC-320: Add Motion Sensing to Your Device

Matt Liberty
60 minute session
Wednesday, 4 May 2011 - 3:15 PM to 4:15 PM
Embedded Systems Conference Silicon Valley 2011

Products that include motion sensing are becoming ubiquitous due to the introduction of small, inexpensive sensors. A wide range of products from cars to washing machines measure their own motion. The motion can be used for a number of purposes including image stabilization, navigation and user interface control. Selecting the right sensor or sensors and integrating them into an application can still be challenging. This paper describes motion, the available sensors, how to use the sensor output, and how to select the correct motion sensors for your application.

What is Motion?

We move through our environment every day. We walk, run and drive. We turn our heads. We use our hands. We throw and catch Frisbees. We all have an intuitive grasp of motion.

When we move forwards & backwards, left & right and up & down, we are moving among three dimensions of freedom known as linear motion. We intuitively understand that if we walk one meter forward followed by another meter, we will have walked two meters.

We can also look in different directions. We swing our head to face to the left and right. We can swing our head to look up and down. We can tilt our head side to side. These three dimensions of freedom are known as angular motion. The simple intuition of linear motion does not apply to angular motion. If we rotate our head by 180° (π radians) followed by another 180° in the same direction, the difference from the starting location is 0°.

Motion is a general term describing the change among these six degrees of freedom over time. The motion can be defined by the position, velocity (the first derivative of position with respect to time) and acceleration (the first derivative of velocity with respect to time). Any movement of a rigid body can be decomposed into the linear (translational) motion of a point and a rotation about that point. Objects are put into motion by imparting a force, which is determined by acceleration and mass.

All motion is relative. When an object moves, that movement is defined relative to some frame of reference against which the motion is measured. If the same motion is observed from another frame of reference, different motion will be observed. Since no absolute reference frame exists, all motion must be defined according to a frame of reference.

Frames of Reference

The frame of reference is a coordinate system used to measure motion of an object. Let's consider an example of two trains. Imagine sitting next to railroad tracks and watching a passing East-bound train. The train appears to be moving to the East. To someone on the train, you would appear to be moving to the West.

For most practical motion measurement problems, two interesting frames of reference exist. The first frame of reference, the body frame, is attached to the body of interest and moves along with that body. The second frame of reference is fixed to the external object. For the present discussion, we will assume that the second frame of reference is defined relative to a fixed point on Earth's surface, and we will call this the Earth frame. Figure 1 shows the definition of the body frame with the object represented by an aircraft.

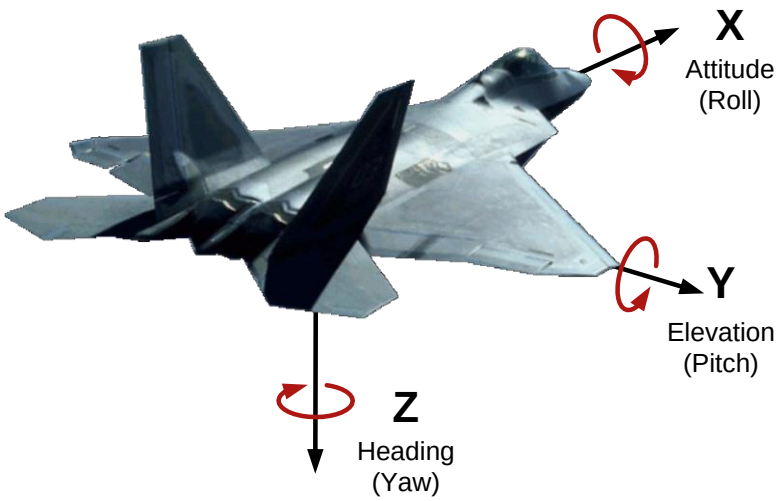

Figure 1: Body-frame Definition

The body frame is connected to the Earth frame by a linear position **p** and angular position **Q**. Figure 2 shows the relationship between the two frames.

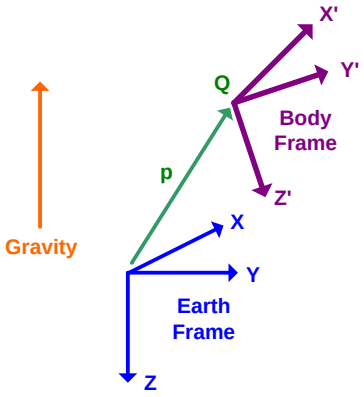

Figure 2: Frame of Reference

Frames of reference can be presumed to not move (inertial) or move (non-inertial). If motion is measured relative to a non-inertial frame, then objects moving relative to this frame will experience additional fictitious forces, such as centrifugal force and Coriolis force.

Linear Motion

Given Figure 2, we can define the linear motion of the device using the length 3 vector **p** that contains the position of the device in the x, y and z coordinates. The linear velocity **v** is the rate of change of **p** with respect to time. The linear acceleration **a** is the rate of change of **v** with respect to time. If we define time as t, then calculus can clearly define these relationships as

$$\boldsymbol{v} = \frac{d}{dt}\boldsymbol{p} = \dot{\boldsymbol{p}}$$

$$\boldsymbol{a} = \frac{d}{dt}\boldsymbol{v} = \dot{\boldsymbol{v}} = \ddot{\boldsymbol{p}}$$

Using integration, we can also calculate position from velocity or acceleration.

$$\boldsymbol{p} = \mathbf{p}_0 + \int \boldsymbol{v}\, dt$$

$$\boldsymbol{v} = \mathbf{v}_0 + \int \boldsymbol{a}\, dt$$

Note that the initial condition of $\mathbf{p}_0$ and $\mathbf{v}_0$ are included for clarity. Linear motion forms nice linear equations. For practical applications, the most problematic feature is that is that integration is not numerically stable, and the double integration from acceleration to position is particularly challenging.

Angular Motion

For one dimensional angular motion, the equations are much like linear motion. Given an angular position, θ, the derivative with respect to time is the angular velocity, ω. The derivative of ω with respect to time is the angular acceleration, α. Angular motion in three dimensions is much more complicated. The complication arises from the nature of angular position and its representation, rotations.

Rotations are a mathematical transformation that maintains length and distance. Given two points **a** and **b**, a rotation creates the corresponding points **c** and **d**. For all possible points **a** and **b**, a rotation operator will ensure that

$$\|\boldsymbol{a}\| = \|\boldsymbol{c}\|, \|\boldsymbol{b}\| = \|\boldsymbol{d}\| \text{ and } \|\boldsymbol{a} - \boldsymbol{c}\| = \|\boldsymbol{b} - \boldsymbol{d}\|$$

Unlike linear motion, rotations tend to be confusing. In addition to being non-commutative, rotations depend upon perspective. To understand why, let's examine a case. We would like to rotate the vector [1, 0, 0] about the vector [0, 0, 1] by a small angle θ. Remembering the right-hand thumb rule:

[1, 0, 0] => [cos θ, sin θ, 0]

However, rotations often apply to the coordinate systems instead of vectors. If the coordinate system is transformed by θ, then the original vector viewed from the rotated coordinate system is now

[1, 0, 0] => [cos θ, -sin θ, 0]

Note that the sign of the y-axis component is the opposite of the first case. The rotation of a vector and the coordinate transform are equal but opposite! When applying rotations and especially rotation sequences, appropriate care must be taken to ensure that the distinction is maintained.

Depending upon the application, the designer may choose the angular position to represent either a vector rotation or a frame transformation. Physicists often use frame transformations, but computer graphics usually uses vector rotations. The angular velocity may then be represented in either the body frame or the Earth frame. The angular acceleration is normally represented in the same frame as the angular velocity.

Further complicating rotations is multiple representation forms including Euler angles, direction cosine matrices (DCM) and quaternions. Quantum physics uses Pauli spin matrices which are similar to quaternions. Quaternions are also very similar to axis-angle representations.

Despite these complications, the angular position, velocity and acceleration can still be represented and computed through calculus. Although the application of a rotation is a linear transformation, the calculus is non-linear. The ease of these computations depends upon the rotation representation. The next few sections look at each representation and discuss the advantages and disadvantages of each technique.

Euler Angles

To extend the one dimension angular position case to three dimensions, the natural inclination is to introduce three angular positions, each representing a successive rotation around an axis. This representation of angular position is called Euler angles. Euler's rotation theorem says that two orthonormal coordinate frames are related by no more than three successive rotations about coordinate axes where no two successive rotations are about the same axis. Since rotation operations do not commute and successive rotations are about different axes, 12 such sequences exist.

All 12 possible sequences are valid, but selecting the correct sequence can greatly simplify the representation, similarly to selecting Cartesian or polar coordinates. The aerospace representation of z-y-x is the arguably the most intuitive given z is down, x is north and y is east. The z-y-x rotation sequence consists of a rotation around the z-axis followed by a rotation around the new y-axis followed by a rotation around the new x-axis. The first rotation about the z axis represents the compass heading. The second rotation about the new y axis represents the attitude or angle from the horizontal. The final rotation about the new x axis represents the roll. Each step of the rotation sequence creates a new coordinate system with the final system of X', Y', Z' as shown in Figure 2. A more thorough treatment of rotation sequences can be found in reference [1].

Euler angles, while easier to conceptualize, have a severe numerical stability problem known as gimbal lock. In the case of the z-y-x sequence, when the y rotation is ±90°, then the z rotation and x rotation effectively operate around the same axis. The rotation around z and rotation around x are not separable. Also, when passing through a y rotation of ±90°, the z rotation will have a step function jump that significantly complicates any calculus. Euler angles are good for early investigations into the notions of rotations, but real-world applications tend to use other representations that avoid these problems.

Direction Cosine Matrix

A direction cosine matrix (DCM) is a 3 by 3 matrix with special constraints to ensure that multiplying the DCM by a 3x1 vector performs a rotation. A DCM is a matrix that contains unit length, orthogonal basis vectors. DCMs avoid the gimbal lock problem inherent with Euler angle representations.

The DCM is a matrix, and the algebra of matrix operations is well understood, which is an advantage relative to the less common algebra of the quaternion. However, the DCM requires 9 values to represent the rotation as compared to 3 for Euler angles and 4 for the quaternion. While Euler angles always perform rotations and quaternions can be easily normalized, the DCM is more difficult to normalize. A 3x3 matrix can also perform scale and skew transformations, so keeping the DCM normalized is important.

The remainder of this section gives a brief introduction to the math behind the DCM. A more expansive introduction can be found in [1].

Definition

$\boldsymbol{Q}$ is a 3x3 matrix, $\boldsymbol{v}$ is a 3x1 vector

Multiplying a vector by a DCM maintains the length of the vector: $\overline{\boldsymbol{v}} = \boldsymbol{Q}\,\boldsymbol{v},\ \|\overline{\boldsymbol{v}}\| = \boldsymbol{v}$

The inverse of a DCM is the same as its inverse: $\boldsymbol{Q}^{-1} = \boldsymbol{Q}^{T}$

The axis of rotation is the eigenvector corresponding to the real eigenvalue of 1:

$$\boldsymbol{Q} = \cos\theta\ \boldsymbol{I} + (1 - \cos\theta)\,\boldsymbol{a}\,\boldsymbol{a}^{T} - \sin\theta\ \boldsymbol{a}^{\times}$$

The determinant of a DCM is 1

Calculus

Given **Q**, determine the angular velocity, **Ω**, in the body frame:

$$\boldsymbol{\Omega} = \boldsymbol{Q}^{T}\,\dot{\boldsymbol{Q}}$$
$$\dot{\boldsymbol{\Omega}} = \dot{\boldsymbol{Q}}^{T}\,\dot{\boldsymbol{Q}} + \boldsymbol{Q}^{T}\,\ddot{\boldsymbol{Q}}$$

Given the above definitions, **Ω** relates to the angular velocity.

$$\boldsymbol{\Omega} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}$$

$$\dot{\boldsymbol{\Omega}} = \begin{pmatrix} 0 & -\dot{\omega}_z & \dot{\omega}_y \\ \dot{\omega}_z & 0 & -\dot{\omega}_x \\ -\dot{\omega}_y & \dot{\omega}_x & 0 \end{pmatrix}$$

Euler Integration

Using the above equations, we can solve for the next time step using integration. Euler integration is the first order approximation:

$$\boldsymbol{Q}(n) = \boldsymbol{Q}(n-1) + \dot{\boldsymbol{Q}}(n)\,\Delta t$$

$$\boldsymbol{Q}(n) = \boldsymbol{Q}(n-1) + \boldsymbol{\Omega}\,\Delta t\,\boldsymbol{Q}(n-1)$$

$$\boldsymbol{Q}(n) = (\boldsymbol{I} + \boldsymbol{\Omega}\,\Delta t)\boldsymbol{Q}(n-1)$$

However, numerical integration introduces error that breaks the rules for a DCM. Higher order terms could be used to reduce, but not eliminate, the error. In practice, the DCM must be renormalized.

DCM Normalization

Several normalization methods exist. The primary method involves decomposing the matrix into the axis of rotation and angle of rotation and then reconstructing the matrix. The axis of rotation is found through eigenvector decomposition and the axis of rotation is found using the trace equation. The DCM is then reconstructed using the axis of rotation and the angle of rotation. The computational complexity of DCM normalization makes quaternions more attractive.

Quaternions

A quaternion is a length four vector with a special algebra. The extra value compared to the length 3 Euler angles enables numerical stability. Unlike the DCM, quaternions can be easily normalized to ensure that they represent rotations. Performing calculus on quaternions is also simpler than performing calculus on DCMs and Euler angles. The biggest challenge of quaternions is that the algebra is unfamiliar to most engineers. This section helps to dispel the mystery and encourage the broader use of quaternions.

A quaternion is analogous to a complex number, but with a higher order. Like complex numbers, a quaternion has a scalar part, but the "complex" part is a length 3 vector. A quaternion is normally written in one of two formats:

$$\boldsymbol{q} = \langle q_0, \quad \boldsymbol{v} \rangle = (q_0, \quad q_x, \quad q_y, \quad q_z)$$

Although all length four vectors could be manipulated as quaternions, only quaternions with unit length can represent rotations.

Definition

A quaternion that rotates around an axis **a** by an angle α is given by:

$$\boldsymbol{q} = \langle \cos\frac{\alpha}{2}, \quad \hat{\boldsymbol{a}}\,\sin\frac{\alpha}{2} \rangle$$

A valid rotation quaternion must have a magnitude of 1: $\|\boldsymbol{q}\| = 1$

The quaternion conjugate is defined as $\boldsymbol{q}^* = \langle q_0, \quad \boldsymbol{v} \rangle^* = \langle q_0, \quad -\boldsymbol{v} \rangle$ where $(\boldsymbol{p}\,\boldsymbol{q})^* = \boldsymbol{q}^*\,\boldsymbol{p}^*$

Given $\boldsymbol{p} = \langle a, \quad \boldsymbol{v} \rangle$ and $\boldsymbol{q} = \langle b, \quad \boldsymbol{w} \rangle$, multiplication is defined as

$$\boldsymbol{p}\,\boldsymbol{q} = \langle a\,b - \boldsymbol{v} \cdot \boldsymbol{w}, \quad \boldsymbol{v} \times \boldsymbol{w} + a\,\boldsymbol{w} + b\,\boldsymbol{v} \rangle$$

Since the cross product is not commutative, neither is quaternion multiplication.

The quaternion can be used to compute the vector as viewed from the other frame or to rotate the vector within the same frame. These operations are equal and opposite, and no single standard exists. This document assumes that:

$\langle 0, \quad \boldsymbol{w} \rangle \; = \; \boldsymbol{q}^* \, \langle 0, \quad \boldsymbol{v} \rangle \, \boldsymbol{q}$ is the transformation to the new frame

$\langle 0, \quad \boldsymbol{x} \rangle \; = \; \boldsymbol{q} \, \langle 0, \quad \boldsymbol{v} \rangle \, \boldsymbol{q}^*$ is the vector rotation within the frame

As an example, consider the quaternion that rotates the body frame by 30° about the user frame z-axis. When the user frame vector [1 0 0] is viewed from the body frame, it should be $\left(\frac{\sqrt{3}}{2}, \quad -\frac{1}{2}, \quad 0\right)$.

Note that the -½ instead of ½ is critical as this determines the important direction of rotation. The quaternion that performs this rotation is

$$\boldsymbol{q} \; = \; \langle \cos\frac{30°}{2}, \quad \left(0, 0, \sin\frac{30°}{2}\right) \rangle$$

Calculus Derivative

Quaternions are well-suited for calculus and differential equations. Quaternion rotation operators and their derivatives can be used to express the standard rigid-body dynamics equations.

First derivative

Given the quaternion **q** and the angular velocity $\boldsymbol{\omega}$ in the body frame, what is the derivative of the quaternion? First, define angular velocity with respect to an axis of rotation in the user frame and an incremental angle.

$$\lim_{\Delta t \to 0} \left(\langle 0, \quad \frac{\boldsymbol{v} \, \alpha}{2 \, \Delta t} \rangle \right) = \frac{1}{2} \boldsymbol{q} \, \langle 0, \quad \boldsymbol{\omega} \rangle \, \boldsymbol{q}^*$$

Now, calculate the derivate.

$$\boldsymbol{q}(t + \Delta t) \; = \; \boldsymbol{p}(t) \, \boldsymbol{q}(t) \;\; \text{where } \boldsymbol{p}(t) = \langle \cos\frac{\alpha(t)}{2}, \;\; \boldsymbol{v}(t) \;\; \sin\frac{\alpha(t)}{2} \rangle$$

$$\boldsymbol{q}(t + \Delta t) \; = \; \langle \cos\frac{\alpha(t)}{2}, \boldsymbol{v}(t) \;\; \sin\frac{\alpha(t)}{2} \rangle \;\; \boldsymbol{q}(t)$$

$$\boldsymbol{q}(t + \Delta t) \; = \; \boldsymbol{q}(t) + \langle 0, \frac{\boldsymbol{v}(t)\alpha(t)}{2} \rangle \;\; \boldsymbol{q}(t)$$

$$\dot{\boldsymbol{q}} = \frac{d}{dt}\boldsymbol{q} = \lim_{\Delta t \to 0} \left(\frac{\boldsymbol{q}(t + \Delta t) - \boldsymbol{q}(t)}{\Delta t} \right) \; = \; \lim_{\Delta t \to 0} \left(\langle 0, \frac{\boldsymbol{v}(t)\alpha(t)}{2 \, \Delta t} \rangle \;\; \boldsymbol{q}(t) \right) = \frac{1}{2} \boldsymbol{q} \, \langle 0, \quad \boldsymbol{\omega} \rangle$$

$$\langle 0, \quad \boldsymbol{\omega} \rangle \; = \; 2 \, \boldsymbol{q}^* \, \dot{\boldsymbol{q}}$$

See Appendix B of [5] for an alternative approach.

Second derivative

The second derivative may be computed from the first derivative using the chain rule

$$\ddot{\boldsymbol{q}} = \frac{d}{dt}\dot{\boldsymbol{q}} = \frac{d}{dt} \, \frac{1}{2} \boldsymbol{q} \langle 0, \boldsymbol{\omega} \rangle$$

$$\ddot{\boldsymbol{q}} = \frac{\dot{\boldsymbol{q}} \langle 0, \boldsymbol{\omega} \rangle + \boldsymbol{q} \langle 0, \dot{\boldsymbol{\omega}} \rangle}{2} = \frac{\frac{\boldsymbol{q} \langle 0, \boldsymbol{\omega} \rangle \langle 0, \boldsymbol{\omega} \rangle}{2} + \boldsymbol{q} \langle 0, \dot{\boldsymbol{\omega}} \rangle}{2}$$

$$\ddot{\boldsymbol{q}} = \frac{1}{2} \boldsymbol{q} \, \langle \frac{\boldsymbol{\omega} \cdot \boldsymbol{\omega}}{2}, \dot{\boldsymbol{\omega}} \rangle$$

Euler approximation of the first derivative

This section applies quaternion math as an exercise to compute the Euler approximation of the first derivative. To reduce derivative error, high precision applications often include higher order terms when calculating the derivate.

$$\dot{\boldsymbol{q}} = \frac{\boldsymbol{q}(n) - \boldsymbol{q}(n-1)}{\Delta t}$$

$$\langle 0, \quad \boldsymbol{\omega} \rangle \; = \; 2\, \boldsymbol{q}^* \, \dot{\boldsymbol{q}}$$

$$\langle 0, \quad \boldsymbol{\omega} \rangle \; = \; 2\, \boldsymbol{q}(n)^* \, \frac{\boldsymbol{q}(n) - \boldsymbol{q}(n-1)}{\Delta t}$$

$$\langle 0, \quad \boldsymbol{\omega} \rangle \; = \; \frac{2}{\Delta t} \langle 1, \quad \mathbf{0} \rangle - \frac{2}{\Delta t}\, \boldsymbol{q}(n)^* \, \boldsymbol{q}(n-1)$$

$$\langle 0, \quad \boldsymbol{\omega} \rangle \; = \; -\frac{2}{\Delta t} \langle 1, \quad \mathbf{0} \rangle + \frac{2}{\Delta t}\, \boldsymbol{q}(n-1)^* \, \boldsymbol{q}(n)$$

Interestingly, this equation shows that the magnitude of the scalar component indicates the amount of integration error due to the first order integration approximation. Rearranging the terms for integration gives

$$\boldsymbol{q}(n) \; = \boldsymbol{q}(n-1) \; \langle 1, \quad \frac{\Delta t}{2} \boldsymbol{\omega} \rangle$$

Numerical Integration

Integrating acceleration to velocity is straightforward and linear.

$$\frac{d}{dt} \boldsymbol{\omega} = \dot{\boldsymbol{\omega}}$$

All standard numerical methods are available including Euler integration, trapezoidal integration and higher order methods. Higher order integrating polynomial methods do better as can the ODE Methods including Runge-Kutta and Predictor-Corrector.

Determining angular position from angular velocity is challenging since the relationship is nonlinear. Numerical methods including Euler integration, Taylor series integration, and successive approximation integration are all possibilities. The integration process must continue to insure that the quaternion represents a rotation by keeping the quaternion magnitude equal to 1. Using the chain rule, we can derive the differential equation constraint that maintains quaternion normalization

$$\|\boldsymbol{q}\| = \sqrt{\boldsymbol{q} \cdot \boldsymbol{q}^*} \; = \; 1 = \boldsymbol{q} \cdot \boldsymbol{q}^*$$

$$\dot{\boldsymbol{q}} \cdot \boldsymbol{q}^* + \boldsymbol{q} \cdot \dot{\boldsymbol{q}}^* \; = \; 0$$

$$\boldsymbol{q} \cdot \dot{\boldsymbol{q}} \; = \; 0$$

In addition, all the higher order derivatives are similarly constrained. By applying the chain rule again, we find that $\dot{\boldsymbol{q}} \cdot \dot{\boldsymbol{q}} + \boldsymbol{q} \cdot \ddot{\boldsymbol{q}} \; = \; 0$ and so forth. The implication is the process can enforce this constraint on the quaternion while solving the ODE.

Available Sensing Methods

Numerous sensing technologies are available for measuring motion. The following list contains the most inexpensive, small and easy to use technologies.

Type	Measures	Advantages	Disadvantages
Accelerometer	Linear acceleration including gravity	Small & inexpensive Low power Long-term stable measurement of 2/3 angular position Can include wake-on-motion and tap detection	Double integration to linear position Difficult to separate gravity from linear acceleration
MEMS gyroscope (rate gyro)	Angular velocity	Responsive	Integration to angular position Higher power Bias instability
Magnetometer	Magnetic field	Small & inexpensive Low power Long-term stable measurement of 2/3 angular position	Magnetic fields are subject to fringing Earth's magnetic field is weak and variable Other magnetic sources
Pressure	Atmospheric pressure	Small & inexpensive Low-power Long-term stable estimate of height	Bias value changes over time
Camera	Angular direction with distance approximation	Long-term stable	Low update rate Large data quantity Requires suitable lighting
Other optical uses IR transmitters and receivers	Linear position and angular position	Long-term stable High update rate	New technology Multi-path
Radar System uses RF transmitters and receivers	Linear position and angular position	Effective over long distances	Expensive RF travels quickly Multi-path
Ultrasonic System uses ultrasonic transducers	Linear position and angular position	Sound travels slowly	Requires constellation of transducers Multi-path
Magnetic System uses coils and magnetometer	Linear position and angular position	Proven technology	Metal distortion Energy proportional to $\frac{1}{distance^3}$
GPS	Linear position	Long-term stable	May not work indoors Power hungry

Table 1 – Available Sensing Methods

This list is not exhaustive. Many other sensors exist and these same sensing technologies may be applied in alternative configurations. Other sensors include Hall effect sensors and capacitive sensing. Wikipedia contains a fairly complete list of sensors [8].

Using Sensors

Any real sensor measures the quantity of interest along with error and noise. Additional concerns are sampling, dynamic range and latency. To effectively use a sensor, the system designer must understand the performance and limitations of the sensor.

Sensors typically have the following errors:

- Bias (Zero value offset)
- Sensitivity accuracy
- Sensitivity non-linearity
- Cross-axis sensitivity (coupling)
- Responsiveness to non-intended signals

All of these errors vary based upon numerous parameters including between devices, over time (aging), over humidity, over supply voltage and over temperature. The industry is still maturing and sensor specifications are often insufficient to determine final performance in an application.

The sensor noise depends upon the raw performance of the sensor, amplification and processing. The sensor output is often converted from analog form to digital form. This analog to digital conversion (ADC) process is subject to the standard signal processing constraints including sampling frequency, aliasing and quantization. The sampling process can introduce additional error through sampling frequency jitter, improper filtering (aliasing) and improper ADC selection (quantization). A majority of new sensors have integrated digital outputs with SPI or I^2C interfaces which greatly simplify the integration effort. The amount of noise is the limiting factor of resolution.

In some cases, the sensor output may be processed to further reduce noise. Although signal processing cannot easily remove noise in the bandwidth of interest, the overall noise may be decreased by reducing the overall bandwidth. For example, the signal could be filtered and downsampled to decrease overall level of noise. This averaging method does not reduce the noise level in the bandwidth of interest, but does reduce total noise. In many cases, increasing the sampling frequency to allow this process may increase noise in the bandwidth of interest which negates this technique.

The dynamic range of a sensor determines the magnitude of the quantity of interest that the sensor can measure. If the sensor experiences input beyond the dynamic range, the sensor saturates and only outputs its maximum value. The sensor must be selected to measure the full dynamic range for the application.

Latency is another concern. Latency is the time between when a device moves and when that motion registers with the system. For many applications, such as user interfaces, increasing latency results in a drastic reduction in performance. A low latency system must be carefully designed, and many different factors contribute to the overall system latency. Latency factors include sensor filtering, sampling synchronization, rate conversion, buffering, communication links, graphical rendering and video frame buffers (some of which may be in the display device).

Once a sensor is selected, the designer must integrate the sensor into the full system. Like any hardware component, the system must provide the appropriate interface, such as SPI, I^2C, UART or analog. The system must also provide a voltage within the device's specified operating range and be located in an environment within the device's operating specification. Software concerns such as data throughput, data acquisition method (interrupt versus polling), interrupt service routines, interrupt rate, and buffering must be considered.

Finally, the application must correctly use the output of the sensor. Although all the sensors above respond to motion, they may not respond in the way desired by the application. The next section takes a common example.

Example: Accelerometer

An accelerometer measures linear acceleration. Given **p** as the linear position, **g** as the gravity vector, **Q** as the DCM representing angular position, and assuming that the accelerometer is located at **p**, then the accelerometer experiences a linear acceleration

$$\boldsymbol{a} = \boldsymbol{Q}\,(\ddot{\boldsymbol{p}} + \boldsymbol{g})$$

where the magnitude of **g** is approximately 9.8 meters / second^2.

When placed into a product, the point of interest, **p**, is often the center of mass. Due to design constraints, the accelerometer may not be located at this point of interest. Let's derive the acceleration measured by the accelerometer when it is located at **r** which is a vector **s** from **p**.

$$\boldsymbol{r} = \boldsymbol{p} + \boldsymbol{Q}^T \boldsymbol{s}$$
$$\dot{\boldsymbol{r}} = \dot{\boldsymbol{p}} + \dot{\boldsymbol{Q}}^T \boldsymbol{s} + \boldsymbol{Q}^T \dot{\boldsymbol{s}}$$
$$\ddot{\boldsymbol{r}} = \ddot{\boldsymbol{p}} + \ddot{\boldsymbol{Q}}^T \boldsymbol{s} + 2\,\dot{\boldsymbol{Q}}^T \dot{\boldsymbol{s}} + \boldsymbol{Q}^T \ddot{\boldsymbol{s}}$$
$$\boldsymbol{a} = \boldsymbol{Q}\left(\ddot{\boldsymbol{p}} + \ddot{\boldsymbol{Q}}^T \boldsymbol{s} + 2\,\dot{\boldsymbol{Q}}^T \dot{\boldsymbol{s}} + \boldsymbol{Q}^T \ddot{\boldsymbol{s}} + \boldsymbol{g}\right)$$

Presuming that **s** is fixed by design, then the derivatives of **s** are 0. The equation then simplifies to

$$\boldsymbol{a} = \boldsymbol{Q}(\ddot{\boldsymbol{p}} + \boldsymbol{g}) + \boldsymbol{Q}\,\ddot{\boldsymbol{Q}}^T \boldsymbol{s}$$
$$\boldsymbol{a} = \boldsymbol{Q}(\ddot{\boldsymbol{p}} + \boldsymbol{g}) + \dot{\boldsymbol{\omega}} \times \boldsymbol{s} + \boldsymbol{\omega} \times \boldsymbol{\omega} \times \boldsymbol{s}$$

where $\boldsymbol{\omega}$ is the angular velocity and $\dot{\boldsymbol{\omega}}$ is the angular acceleration. In order to accurately determine the linear acceleration at the point of interest, the system must now accurately know the angular velocity and angular acceleration!

Now consider the case of an accelerometer being used to measure tilt relative to gravity. To determine the accuracy of the tilt measurement, each parameter must be considered over the full operating range. Bias performance is often the largest error contributor. Consider the calculation for the 2D error due to bias only:

$$\theta_{error} = \tan^{-1}\frac{1000\,\sin\theta \pm b_x}{1000\,\cos\theta \pm b_z} - \theta = 8°\ when\ \theta = 45°, b = 100\,mg$$

Sensor Fusion

Sensors have strengths and weaknesses in the motions they measure. For example, a linear accelerometer is excellent at measuring linear acceleration (the rate of change of the rate of change of position) if the angular position does not change. Since the linear accelerometer also measures gravity, it is also excellent for measuring roll and tilt absent of linear motion. However, even the best linear accelerometer alone cannot accurately do both roles simultaneously.

Many modern systems use a variety of sensors to measure motion. These systems leverage the strengths of each sensor type to produce a system that estimates motion better than any individual sensor. Inertial navigation systems often combine a XYZ linear accelerometer, an XYZ gyroscope, an XYZ magnetometer, a pressure sensor (barometer) and a temperature sensor. These outputs are combined together using a sensor fusion method.

Sensor fusion uses statistical and probabilistic information about each sensor to determine a single best output. The best output can be defined in many different ways depending upon the application, but the definition often includes short-term accuracy, long-term accuracy (stability) and resolution.

To understand how sensor fusion is possible, consider two equally skilled craftsmen who are going to cut a new countertop for your kitchen. One measures the size to be 256 cm, and the other measures 257 cm. What is the best estimate? Assuming equal accuracy on both measurements, then 266.5 cm (the average) would be the best estimate. What is the general best estimate when the individual measurement accuracies are not equal? Assuming a normal distribution with standard deviation σ, then the best estimate for combining two estimates is:

$$\frac{\sigma_A^2}{\sigma_A^2 + \sigma_B^2} m_B + \frac{\sigma_B^2}{\sigma_A^2 + \sigma_B^2} m_A$$

The variance for the new estimate is

$$\frac{\sigma_A^2\,\sigma_B^2}{\sigma_A^2 + \sigma_B^2}$$

This statistical combination of values is the heart of sensor fusion. The Kalman filter is an iterative, multidimensional solution to this problem. The filter allows for both a process model and a measurement model which are related to the internal state through a matrix. The filter contains state and error covariance. The state can be any length vector. For each new data input, the new state is computed using a predictor/corrector structure based upon the measurement, existing state, error covariance, process model and measurement model. The Kalman filter presumes that the process and measurement models are linear with normally (Gaussian) distributed noise.

If the assumptions hold, then the Kalman filter provably computes the optimal least squares solution. Unfortunately, both assumptions are problematic for angular position, but non-linear alternatives to the Kalman filter exist. The extended Kalman filter applies the Kalman filter to non-linear problems by linearizing about the current operating point. This method is relatively simple, but often results in estimation divergence when the linearization assumption is insufficient. Unscented Kalman filtering, Bayesian filtering and particle filters further reduce the dependency on the linear and normal noise assumptions.

The details of sensor fusion are beyond the scope of this document. An excellent introduction to Kalman filtering can be found in [9].

What Motion Does Your Application Need to Measure?

Determining application requirements is the first step to selecting the appropriate sensor or sensor system. The process to determine the actual requirements can be challenging. The following examples provide case studies into determining requirements.

Example: Joystick

A joystick measures the amount of deviation in the roll (x-axis) and pitch (y-axis) directions. Since gravity is in the negative z-axis direction, an accelerometer can be used to accurately measure the angles using

$$\theta_x = \sin^{-1}\frac{x}{\sqrt{x^2+y^2+z^2}}$$

$$\theta_y = \sin^{-1}\frac{y}{\sqrt{x^2+y^2+z^2}}$$

If the user accidently moves the device linearly, then the joystick angles will change even if the user does not move the device angularly. For some applications this tradeoff may be acceptable. For improved performance, the system could add a XY gyroscope. The output would rely on the gyroscope for short term accuracy and the accelerometer for long-term accuracy.

Example: Cursor control

The simplest form of cursor control just needs to measure the z-axis angular velocity and the y-axis angular velocity. However, if the user does not hold the device properly, then up would not be up. Adding a linear accelerometer allows the output to have orientation compensation. Regardless of how the user holds the device, the device uses the gravity estimate to ensure that moving up would always cause the cursor to more up. Using an accelerometer to determine the direction of up is problematic in the presence of linear acceleration as shown in the previous example. By integrating the angular velocity from the gyroscopes, a more stable output could be determined, and even more accuracy is possible by adding the final x-axis of angular velocity sensing.

Example: Gestures

A gesture is a motion that is recognized and translated into an event. The minimum amount of motion sensing required to implement a gesture depends upon the gesture itself. If the gesture is to recognize when the device has been flipped from right-side up to upside down, then the application could use a single accelerometer to estimate gravity. More complicated gestures may require more information.

Gesture recognition is further complicated by the number of gestures. The system must employ a motion measurement system that can individually distinguish each gesture. Even with optimal motion measurement, the more gestures recognized by the system, the higher the chance of a false detection.

Example: Golf game

The easiest method to control a 3D object onscreen is using a motion sensing system that measures linear position and angular position. However, this full system is not usually required given reasonable constraints. An individual playing a golf game is really not going to walk the full course, so a large range of linear position is not required. In addition, golf games usually assume that the user will contact the golf ball, so fine linear position is not required at all. To control the swing, only the angular position in one dimension is really required. If the game would like to determine hooks, slices, inside-out and outside-in swings, then all three angular dimensions would be required.

The application designer has significant latitude in determining the interaction method. Golf games have been designed using just button presses. Although motion sensing can add realism to the interaction, full reality is often not the objective. The application designer has the challenge of balancing gameplay with the amount of realism. This balance determines the motion sensing requirements.

Example: Navigation

At a minimum, a navigation system attempts to determine linear position. More advanced navigation systems also determine heading or even full angular position. Location based services (LBS) required both linear position and angular position. A navigation system requires both short term accuracy and long-term stability, and unusually combines an array of sensors to achieve an overall solution. The common solution is to fuse GPS with an XYZ accelerometer, XYZ gyroscope, barometer and XYZ magnetometer.

For many practical applications, the surface of the Earth can be considered an inertial frame, which is not strictly true. Depending upon the accuracy of the measurement system, the assumption that Earth is a non-inertial frame breaks down. The Earth spins about its axis once per day (15° per hour) at a radius of 6,378 kilometers. The Earth then rotates about the sun once per year (0.04° per hour) at a radius of 1 AU (149,597,871 kilometers). The sun rotates about the Milky Way once per 225 million years (1.8×10^{-10} ° per hour) at a radius of 25,000 light years [7]. Depending upon the accuracy of your system, you have to take these factors into account, along with the US export restrictions on military grade navigation equipment.

Conclusion

Adding basic motion sensing to your next project can be as simple as selecting an accelerometer, connecting the sensor to your processor using I^2C or SPI, and using the data. The sensor accuracy for motion is improving rapidly, and the prices are dropping just as fast. For more demanding motion sensing applications, the math can be quite tricky. Numerous companies supply turnkey modules with the motion sensors, sensor compensation algorithms and sensor fusion. Modules exist that can connect to your microprocessor using UART, I^2C and SPI or a host PC through USB. Either way, the cost and size of motion sensing is likely within the constraints of your next project.

References

1. Kuipers, J. B. (1999), Quaternions and Rotation Sequences, Princeton University Press.
2. Sweetster, Doug (1997), Web site.: http://world.std.com/%7Esweetser/quaternions/intro/tools/tools.html.
3. Omelyan, Igor P. (1999), On the numerical integration of motion for rigid polyatomics: The modified quaternion approach, arXiv.
4. http://en.wikipedia.org/wiki/Frame_of_reference
5. Baraff, David (1997), An Introduction to Physically Based Modeling. Appendix B http://www.cs.cmu.edu/~baraff/pbm/rigid2.pdf

6. Press, William H (2002). Numerical Recipes in C, Cambridge University Press, New York: http://www.library.cornell.edu/nr/cbookcpdf.html.
7. Leong, Stacy (2002). "Period of the Sun's Orbit around the Galaxy (Cosmic Year)". *The Physics Factbook*. http://hypertextbook.com/facts/2002/StacyLeong.shtml.
8. List of Sensors (As of July 2010), http://en.wikipedia.org/wiki/List_of_sensors
9. Welch, Greg and Bishop, Gary (2006). "An Introduction to the Kalman Filter". http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf
10. Titterton and Weston (2004). Strapdown Inertial Navigation Technology, IEEE.

Notation

Lower case letters represent scalar variables: x, y, z

Lower case bold letters represent vectors: **x**, **y**, **z** (which are presumed to be column vectors)

Upper case bold letters represent matrices: **X**, **Y**, **Z**

$\boldsymbol{x}\,\boldsymbol{y} = \boldsymbol{x} \cdot \boldsymbol{y}$ is the dot product of vectors **x** and **y**

$\|\boldsymbol{v}\| = \sqrt{\boldsymbol{v} \cdot \boldsymbol{v}}$ is the magnitude of vector **v**

x × **y** is the cross product of vectors **x** and **y**

$\boldsymbol{v}^{\times}$ is the cross product matrix $\begin{pmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{pmatrix}$

X y is the matrix multiplication of matrix **X** and vector **y**

$\mathbf{X}^{\mathsf{T}}$ is the matrix transpose

$\hat{\boldsymbol{v}} = \frac{v}{\|v\|}$ is the unit vector in the direction of **v**

Length 2 vector **v** is assumed to have subcomponents named (v_x, v_y)

Length 3 vector **v** is assumed to have subcomponents named (v_x, v_y, v_z)