Exploiting the gyroscope to update tilt measurement and eCompass

By Andrea Vitali

### Purpose and benefits

This design tip explains how to exploit gyroscope data to update tilt measurements (roll and pitch angles) and eCompass (yaw angle). A quaternion implementation is also shown, which does not suffer from singularity problem, also known as gimbal-lock.

Benefits:

- Enhanced functionality with respect to simple 6-axis Acc+Mag data fusion which cannot be performed when high-g motion or magnetic anomalies are present
- Reduction of the firmware footprint with respect to using the full-blown data fusion provided by the MotionFX library embedded in the X-Cube-MEMS1 software package, see references in design Support Material paragraph
- Short essential implementation, which enables easy customization and enhancement by the end-user (MotionFX is available only in binary format, not as source code)
- Easy to use solution, applicable to all microcontrollers, although the MotionFX library can only be run on the STM32 MCU family

### Description of Euler angle implementation

Step 1: Compute angle derivatives Phi' / Theta' / Psi' based on current angles Phi / Theta / Psi and on gyroscope data Wx / Wy / Wz (see Figure 1 for reference):

Roll derivative: \( \Phi' = Wx + Wy \times \sin(\Phi) \times \tan(\Theta) + Wz \times \cos(\Phi) \times \tan(\Theta) \)

Pitch derivative: \( \Theta' = Wy \times \cos(\Phi) - Wz \times \sin(\Phi) \)
Yaw derivative: \( \Psi' = W_y \sin(\Phi) / \cos(\Theta) + W_z \cos(\Phi) / \cos(\Theta) \)

Note: If \( \Theta = \pm 90 \text{ deg} \), then \( \cos(\Theta) \) is zero and \( \tan(\Theta) = \sin(\Theta)/\cos(\Theta) \) is \( \pm \infty \). These singularities make it impossible to compute derivatives for Roll and Yaw. This is also known as gimbal-lock: when \( \Theta = \pm 90 \text{ deg} \), \( \Phi \) and \( \Psi \) will describe a rotation around the same vertical axis and one degree of freedom is lost. Because of these singularities, the quaternion implementation should be preferred.

Step 2: Compute updated angles \( \Phi / \Theta / \Psi \) based on angle derivatives:

\[
\begin{align*}
\text{Roll: } & \quad \Phi(t+Ts) = \Phi(t) + \Phi' \times Ts \\
\text{Pitch: } & \quad \Theta(t+Ts) = \Theta(t) + \Theta' \times Ts \\
\text{Yaw: } & \quad \Psi(t+Ts) = \Psi(t) + \Psi' \times Ts
\end{align*}
\]

Euler angles should be reduced so that Roll and Yaw are in the range \([-\pi, +\pi]\) rad or \([-180, +180]\) deg, while Pitch is in the range \([-\pi/2, +\pi/2]\) rad or \([-90, +90]\) deg.

\[
\begin{align*}
\text{Pitch} = \text{mod}(\text{Pitch}, 360), & \quad \text{if } (\text{Pitch}>180) \quad \text{Pitch} = \text{Pitch} - 360 \\
\text{if Abs(Pitch)} > 90, \quad \text{Pitch} = \text{sign(Pitch)} \times 180 - \text{Pitch}, & \quad \text{Roll} = \text{Roll} + 180, \quad \text{Yaw} = \text{Yaw} + 180 \\
\text{Roll} = \text{mod}(\text{Roll}, 360), & \quad \text{if } (\text{Roll}>180) \quad \text{Roll} = \text{Roll} - 360 \\
\text{Yaw} = \text{mod}(\text{Yaw}, 360), & \quad \text{if } (\text{Yaw}>180) \quad \text{Yaw} = \text{Yaw} - 360
\end{align*}
\]

Step 3: Mix with other angles, e.g., computed by Acc+Mag data fusion (optional step)

The Roll and Yaw range may have spurious discontinuities: e.g. 0 and 360 deg represent the same angle, when averaged the output is 180 which is clearly wrong (should be 0 or 360); as another example, -180 and +180 deg represent the same angle, when averaged the output is 0 which is clearly wrong (should be -180 or +180). The correct weighted average is computed as follows:

\[
\text{While } (\text{Abs}(\text{Angle1-Angle2}) > 180) \quad \text{Angle1} = \text{Angle1} - 360 \times \text{sign(Angle1-Angle2)}
\]

\[
\text{MixedAngle} = \text{Angle1} \times \alpha + \text{Angle2} \times (1-\alpha), \text{ where } 0<\alpha<1
\]

Pitch cannot have spurious discontinuities as it goes from -90 to +90 deg.

After mixing angles, angles can be reduced to their target range. As an example, this is the processing needed to reduce angles to range from -180 to +180 deg (Roll, Pitch and Yaw):

\[
\text{MixedAngle} = \text{mod(MixedAngle, 360 deg)}
\]

\[
\text{If } (\text{MixedAngle} > 180) \quad \text{MixedAngle} = \text{MixedAngle} - 360
\]

\[
\text{If } (\text{MixedAngle} < -180) \quad \text{MixedAngle} = \text{MixedAngle} + 360
\]

And this is the additional processing needed to reduce range to -90 to +90 (Pitch):

\[
\text{If } (\text{MixedAngle} > 90) \quad \text{MixedAngle} = 180 - \text{MixedAngle}
\]
If (MixedAngle<-90) MixedAngle=-180-MixedAngle

Description of Quaternion implementation

Step 1: Compute quaternion derivative based on current quaternion and on gyroscope data Wx / Wy / Wz (see Figure 1 for reference):

\[
\begin{align*}
Q_w' &= (-Q_x \cdot W_x - Q_y \cdot W_y - Q_z \cdot W_z) / 2 \\
Q_x' &= (+Q_w \cdot W_x - Q_z \cdot W_y + Q_y \cdot W_z) / 2 \\
Q_y' &= (+Q_z \cdot W_x + Q_w \cdot W_y - Q_x \cdot W_z) / 2 \\
Q_z' &= (-Q_y \cdot W_x + Q_x \cdot W_y + Q_w \cdot W_z) / 2
\end{align*}
\]

The updated quaternion can then be approximated by addition \( Q(t+Ts) = Q(t) + Q' \cdot Ts \). Normalization is required after the addition, by dividing each component by the norm \( n = \sqrt{Q_w^2 + Q_x^2 + Q_y^2 + Q_z^2} \).

Step 1 (first alternative): A better approximation (in the geometric sense) can be computed by quaternion exponentation and multiplication:

\[
Q(t+Ts) = Q(t) \ast \exp(\begin{bmatrix}0, W_x, W_y, W_z\end{bmatrix} \ast Ts/2) = Q(t) \ast E:
\]

\[
\begin{align*}
W &= \sqrt{W_x^2 + W_y^2 + W_z^2}, \quad E_w = \cos(W \ast Ts/2), \quad S = \sin(W \ast Ts/2) \\
E_x &= S \cdot Wy/W, \quad E_y = S \cdot Wz/W, \quad E_z = S \cdot Wx/W \\
Q_w(t+Ts) &= Q_w \ast E_w - Q_x \ast E_x - Q_y \ast E_y - Q_z \ast E_z \\
Q_x(t+Ts) &= Q_w \ast E_x + Q_x \ast E_w + Q_y \ast E_z - Q_z \ast E_y \\
Q_y(t+Ts) &= Q_w \ast E_y - Q_x \ast E_z + Q_y \ast E_w + Q_z \ast E_x \\
Q_z(t+Ts) &= Q_w \ast E_z + Q_x \ast E_y - Q_y \ast E_x + Q_z \ast E_w
\end{align*}
\]

Step 1 (second alternative): Yet another approximation can be computed by converting the scaled angular velocity vector to a quaternion and performing the multiplication:

\[
Q(t+Ts) = Q(t) \ast \text{euler2quat}(W \ast Ts) = Q(t) \ast P:
\]

\[
\begin{align*}
P_w &= \cos(Wx \ast Ts/2) \ast \cos(Wy \ast Ts/2) \ast \cos(Wz \ast Ts/2) + \sin(Wx \ast Ts/2) \ast \sin(Wy \ast Ts/2) \ast \sin(Wz \ast Ts/2) \\
P_x &= \sin(Wx \ast Ts/2) \ast \cos(Wy \ast Ts/2) \ast \cos(Wz \ast Ts/2) - \cos(Wx \ast Ts/2) \ast \sin(Wy \ast Ts/2) \ast \sin(Wz \ast Ts/2) \\
P_y &= \cos(Wx \ast Ts/2) \ast \sin(Wy \ast Ts/2) \ast \cos(Wz \ast Ts/2) + \sin(Wx \ast Ts/2) \ast \cos(Wy \ast Ts/2) \ast \sin(Wz \ast Ts/2) \\
P_z &= \cos(Wx \ast Ts/2) \ast \cos(Wy \ast Ts/2) \ast \sin(Wz \ast Ts/2) - \sin(Wx \ast Ts/2) \ast \sin(Wy \ast Ts/2) \ast \cos(Wz \ast Ts/2)
\end{align*}
\]

\[
\begin{align*}
Q_w(t+Ts) &= Q_w \ast P_w - Q_x \ast P_x - Q_y \ast P_y - Q_z \ast P_z \\
Q_x(t+Ts) &= Q_w \ast P_x + Q_x \ast P_w + Q_y \ast P_z - Q_z \ast P_y \\
Q_y(t+Ts) &= Q_w \ast P_y - Q_x \ast P_z + Q_y \ast P_w + Q_z \ast P_x \\
Q_z(t+Ts) &= Q_w \ast P_z + Q_x \ast P_y - Q_y \ast P_x + Q_z \ast P_w
\end{align*}
\]
Step 2: Mixing with other quaternions, e.g. computed by Acc+Mag data fusion (optional)

Quaternions have no spurious discontinuities, but they have redundant representations: +Q and –Q do represent the same orientation, the same set of angles. Simply averaging would give an incorrect result. The correct weighted average is computed as follows:

\[
\text{If } (Q1w*Q2w+Q1x*Q2x+Q1y*Q2y+Q1z*Q2z)<0, \ Q2=-Q2
\]

\[
Q = Q1 * \text{alpha} + Q2 * (1-\text{alpha}), \text{ where } 0<\text{alpha}<1, \ Q \text{ should be normalized}
\]

In literature, this is Linear Interpolation. The averaged quaternion should be normalized by dividing each component by \(\sqrt{Qw^2+Qx^2+Qy^2+Qz^2}\). Spherical linear interpolation (SLERP) may be preferred if increments in alpha must give equal rotation increments.

Step 3: Conversion from Quaternion to Euler angles (optional)

\[
Q\text{mod} = Qw^2 + Qx^2 + Qy^2 + Qz^2
\]

\[
Qt = Qw*Qy – Qx*Qz, \text{ to check for singularities}
\]

If \(Qt>+Q\text{mod}/2\), Roll \(\Phi = 0\), Pitch \(\Theta = +90\), Yaw \(\Psi = -2*\text{Atan2}(Qx,Qw)\)

If \(Qt<-Q\text{mod}/2\), Roll \(\Phi = 0\), Pitch \(\Theta = -90\), Yaw \(\Psi = +2*\text{Atan2}(Qx,Qw)\)

Roll: \(\Phi = \text{Atan2}( 2*(Qw*Qx+Qy*Qz) , Qw^2 –Qx^2 –Qy^2 +Qz^2 )\)

Pitch: \(\Theta = \text{Asin}( 2*Qt / Q\text{mod})\), when argument is between -1 and +1

Yaw: \(\Psi = \text{Atan2}( 2*(Qw*Qz+Qx*Qy) , Qw^2 +Qx^2 –Qy^2 –Qz^2 )\)

Notes

Gyroscope data usually has a non-zero output even if the angular rate is zero. This is known as gyroscope bias and must be subtracted before the data is used. As an example, the bias can be estimated by averaging the gyroscope output when the system is standing still. The system is standing still when the data from the accelerometer and magnetometer is constant and their respective modulus is near 1g and local Earth magnetic field.

Gyroscope sensitivity may be non-unity, i.e. there can be a 3% tolerance. Calibration may improve the output. Calibration can be done by performing a full rotation around a given axis and comparing the final angle estimated by exploiting the gyroscope with the angle measured by other sensors such as an accelerometer and magnetometer.

Time interval \(T_s\) is critical to get accurate results. The actual value should match as closely as possible to the target value; any discrepancy will cause errors similar to non-unity gyroscope sensitivity.

The smaller the time interval \(T_s\) is, the more accurate the output will be, so it is better to use the faster output data rate which is available from the gyroscope (e.g. LSM6DS33 can reach 1.6kHz) and/or use interpolation.
Figure 1. Reference orientation for input data from gyroscope, and reference orientation for output data: roll, pitch and yaw angles

Roll, Pitch, Yaw
degrees

Acc (Gravity vector)
G = 1g = 1000mg

Mag (Earth mag field)
B = 10..90uT = 100..900 mG

<table>
<thead>
<tr>
<th>Roll, Pitch, Yaw</th>
<th>Acc (Gravity vector)</th>
<th>Mag (Earth mag field)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 0, 0</td>
<td>0, 0, +G</td>
<td>+B Cos(i), 0, +B Sin(i)</td>
</tr>
<tr>
<td>+90, 0, 0</td>
<td>0, +G, 0</td>
<td>+B Cos(i), +B Sin(i), 0</td>
</tr>
<tr>
<td>0, +90, 0</td>
<td>-G, 0, 0</td>
<td>-B Sin(i), 0, +B Cos(i)</td>
</tr>
<tr>
<td>0, 0, +90</td>
<td>0, 0, +G</td>
<td>0, -B Cos(i), +B Sin(i)</td>
</tr>
<tr>
<td>+90, +90, 0</td>
<td>-G, 0, 0</td>
<td>-B Sin(i), +B Cos(i), 0</td>
</tr>
</tbody>
</table>

Accelerometer output is positive for axis aligned with gravity and pointing down.
Support material

<table>
<thead>
<tr>
<th>Related design support material</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP-SNS-MOTENV1, STM32Cube function pack for IoT node with BLE connectivity and environmental and motion sensors</td>
</tr>
<tr>
<td>User manual, UM2016, Getting started with the STM32 ODE function pack for IoT node with BLE connectivity and environmental and motion sensors</td>
</tr>
<tr>
<td>Quick Start Guide, STM32Cube function pack for IoT node with BLE connectivity, environmental and motion sensors (FP-SNS-MOTENV1)</td>
</tr>
<tr>
<td>X-CUBE-MEMS1, Sensor and motion algorithm software expansion for STM32Cube</td>
</tr>
<tr>
<td>User manual, UM2220, Getting started with MotionFX sensor fusion library in X-CUBE-MEMS1 expansion for STM32Cube</td>
</tr>
</tbody>
</table>

Documentation

Design Tip, DT0058, Computing tilt measurement and tilt-compensated eCompass

Revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>06-Nov-2015</td>
<td>1</td>
<td>Initial release</td>
</tr>
<tr>
<td>22-Nov-2018</td>
<td>2</td>
<td>Added information on averaging quaternions (NLERP, SLERP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Figure 1 changed to improve clarity</td>
</tr>
<tr>
<td>08-Jan-2021</td>
<td>3</td>
<td>Added formula to reduce Euler angles</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Corrected formula for Quaternion derivative</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added more accurate formula for Quaternion update</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated references to reflect current software packages</td>
</tr>
</tbody>
</table>