DT0060
Design tip

Exploiting the gyroscope to update tilt measurement and eCompass

By Andrea Vitali

Main components	
LSM6DS33	iNEMO inertial module: always-on 3D accelerometer and 3D gyroscope
LSM6DSO	iNEMO inertial module: always-on 3D accelerometer and 3D gyroscope
LSM6DSOX	iNEMO inertial module with embedded Machine Learning Core: always-on 3D accelerometer and 3D gyroscope
LSM6DSR	iNEMO inertial module: always-on 3D accelerometer and 3D gyroscope
LSM6DSRX	iNEMO inertial module with embedded Machine Learning Core: always-on 3D accelerometer and 3D gyroscope

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 * Sin(Phi) * Tan(Theta) + Wz * Cos(Phi) * Tan(Theta)

Pitch derivative: Theta' = Wy * Cos(Phi) – Wz * Sin(Phi)

Yaw derivative: Psi’ = Wy * Sin(Phi) / Cos(Theta) + Wz * Cos(Phi) / Cos(Theta)

Note: If Theta = ±90 deg, then Cos(Theta) is zero and Tan(Theta)=Sin(Theta)/Cos(Theta) is ±Infinity. These singularities make it impossible to compute derivatives for Roll and Yaw. This is also known as gimbal-lock: when Theta = ±90 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:

Roll: Phi(t+Ts) = Phi(t) + Phi’ * Ts

Pitch: Theta(t+Ts) = Theta(t) + Theta’ * Ts

Yaw: Psi(t+Ts) = Psi(t) + Psi’ * Ts

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].

Pitch = mod(Pitch, 360), if (Pitch>180) Pitch = Pitch - 360

If Abs(Pitch)>90, Pitch = sign(Pitch)*180-Pitch, Roll=Roll+180, Yaw=Yaw+180

Roll = mod(Roll,360), if (Roll>180) Roll = Roll - 360

Yaw = mod(Yaw,360), if (Yaw>180) Yaw = Yaw - 360

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:

While (Abs(Angle1-Angle2)>180) Angle1=Angle1 – 360*Sign(Angle1-Angle2)

MixedAngle = Angle1 * alpha + Angle2 * (1-alpha), 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):

MixedAngle = mod(MixedAngle,360 deg)

If (MixedAngle>180) MixedAngle=MixedAngle-360

If (MixedAngle<-180) MixedAngle=MixedAngle+360

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

If (MixedAngle>90) MixedAngle=180-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):

Qw' = (– Qx*Wx – Qy*Wy – Qz*Wz) / 2

Qx' = (+ Qw*Wx – Qz*Wy +Qy*Wz) / 2

Qy' = (+ Qz*Wx + Qw*Wy – Qx*Wz) / 2

Qz' = (– Qy*Wx + Qx*Wy +Qw*Wz) / 2

The updated quaternion can then be approximated by addition **Q(t+Ts) = Q(t) + Q' * Ts**. Normalization is required after the addition, by dividing each component by the norm **n = sqrt($Qw^2 + Qx^2 + Qy^2 + Qz^2$)**.

Step 1 (first alternative): A better approximation (in the geometric sense) can be computed by quaternion exponentation and multiplication:
Q(t+Ts) = Q(t) * exp([0, Wx, Wy, Wz] * Ts/2) = Q(t) * E:

W = sqrt($Wx^2 + Wy^2 + Wz^2$), C = cos(W*Ts/2), S = sin(W*Ts/2)

Ew = C, Ex = S*Wx/W, Ey = S*Wy/W, Ez = S*Wz/W

Qw(t+Ts) = Qw*Ew – Qx*Ex – Qy*Ey – Qz*Ez

Qx(t+Ts) = Qw*Ex + Qx*Ew + Qy*Ez – Qz*Ey

Qy(t+Ts) = Qw*Ey – Qx*Ez + Qy*Ew + Qz*Ex

Qz(t+Ts) = Qw*Ez + Qx*Ey – Qy*Ex + Qz*Ew

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) * euler2quat(W * Ts) = Q(t) * P:

Pw=cos(Wx*Ts/2)*cos(Wy*Ts/2)*cos(Wz*Ts/2) + sin(Wx*Ts/2)*sin(Wy*Ts/2)*sin(Wz*Ts/2)

Px=sin(Wx*Ts/2)*cos(Wy*Ts/2)*cos(Wz*Ts/2) – cos(Wx*Ts/2)*sin(Wy*Ts/2)*sin(Wz*Ts/2)

Py=cos(Wx*Ts/2)*sin(Wy*Ts/2)*cos(Wz*Ts/2) + sin(Wx*Ts/2)*cos(Wy*Ts/2)*sin(Wz*Ts/2)

Pz=cos(Wx*Ts/2)*cos(Wy*Ts/2)*sin(Wz*Ts/2) – sin(Wx*Ts/2)*sin(Wy*Ts/2)*cos(Wz*Ts/2)

Qw(t+Ts) = Qw*Pw – Qx*Px – Qy*Py – Qz*Pz

Qx(t+Ts) = Qw*Px + Qx*Pw + Qy*Pz – Qz*Py

Qy(t+Ts) = Qw*Py – Qx*Pz + Qy*Pw + Qz*Px

Qz(t+Ts) = Qw*Pz + Qx*Py – Qy*Px + Qz*Pw

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:

If (Q1w*Q2w+Q1x*Q2x+Q1y*Q2y+Q1z*Q2z)<0, Q2=-Q2

Q = Q1 * alpha + Q2 * (1-alpha), where 0<alpha<1, Q 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)

Qmod = Qw^2 + Qx^2 + Qy^2 + Qz^2

Qt = Qw*Qy – Qx*Qz, to check for singularities

If (Qt>+Qmod/2), Roll Phi = 0, Pitch Theta = +90, Yaw Psi = -2*Atan2(Qx,Qw)

If (Qt<-Qmod/2), Roll Phi = 0, Pitch Theta = -90, Yaw Psi = +2*Atan2(Qx,Qw)

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

Pitch: Theta = Asin(2*Qt / Qmod), when argument is between -1 and +1

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

Notes

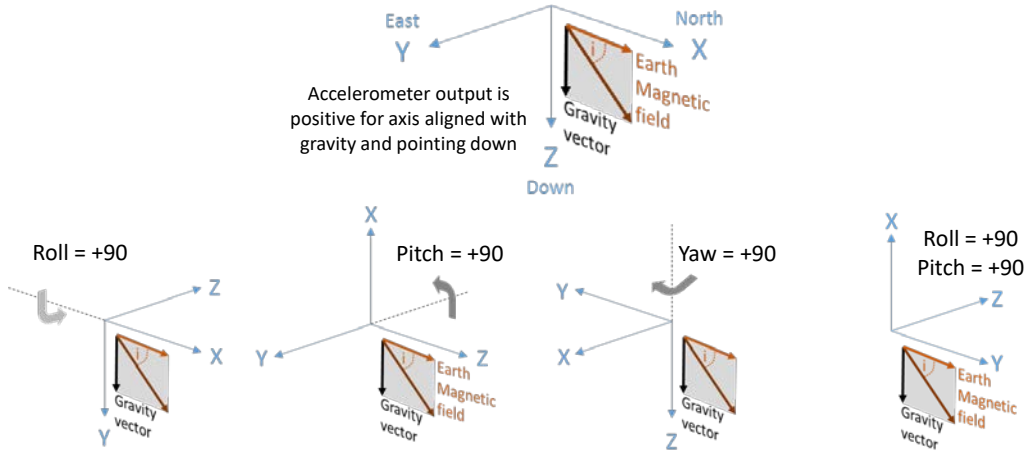
Gyroscope data usually has a non-zero output even if the angular rate is zero. This is known as gyroscope bias and must be subtracted before the data is used. As an example, the bias can be estimated by averaging the gyroscope output when the system is standing still. The system is standing still when the data from the accelerometer and magnetometer is constant and their respective modulus is near 1*g* 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 Ts 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 Ts 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
0, 0, 0	0, 0, **+G**	**+B Cos(i),** 0, +B Sin(i)
+90, 0, 0	0, **+G**, 0	**+B Cos(i),** +B Sin(i), 0
0, **+90**, 0	**-G**, 0, 0	-B Sin(i), 0, **+B Cos(i)**
0, 0, **+90**	0, 0, **+G**	0, **-B Cos(i),** +B Sin(i)
+90, **+90**, 0	**-G**, 0, 0	-B Sin(i), **+B Cos(i),** 0

Support material

Related design support material
FP-SNS-MOTENV1, STM32Cube function pack for IoT node with BLE connectivity and environmental and motion sensors
User manual, UM2016, Getting started with the STM32 ODE function pack for IoT node with BLE connectivity and environmental and motion sensors
Quick Start Guide, STM32Cube function pack for IoT node with BLE connectivity, environmental and motion sensors (FP-SNS-MOTENV1)
X-CUBE-MEMS1, Sensor and motion algorithm software expansion for STM32Cube
User manual, UM2220, Getting started with MotionFX sensor fusion library in X-CUBE-MEMS1 expansion for STM32Cube
Documentation
Design Tip, DT0058, Computing tilt measurement and tilt-compensated eCompass

Revision history

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

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, please refer to *www.st.com/trademarks*. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.