Inertial Management Unit (IMU)
Algorithm Description Document (ADD)

by
Simeon Symeonidis

02/20/2015
revision 1.0

Overview

The purpose of the IMU is to translate Microelectromechanical Systems (MEMS) accelerometer, magnetometer, and gyroscope sensor data into orientation and acceleration measurements.

The algorithm assumes synchronous data. In cases where sensors operate at different rates, it is necessary to interpolate the data into a common reference frame. MI software interpolates lower rate magnetometer and accelerometer data to gyroscope data using timestamps. Other state estimation approaches could eliminate this need to synchronize data, but at the cost of design complexity and measurement bandwidth, i.e. Kalman Filters often reduces near-Nyquist signals. This trade should be re-investigated if there arises a need for a low-latency solution.

The work captured hereinafter is an extension of the paper "An efficient orientation Filter for inertial and inertial/magnetic sensor arrays" written by Sebastian O.H. Madgwick. The algorithm was adapted to meet the needs of a human wearable sensor operating in the presence of Electromagnetic noise.

The report will be organized in three sections. The first section defines variable types and operations to be used in the IMU. This is important because it will establish the conventions to be used within the paper. The second section defines algorithm inputs and outputs. The last section documents the actual algorithm.

Definitions

Sensor inputs will be represented as three dimensional vectors, denoted lowercase letter with an arrow on top. Common vector operations, normalize, dot products and cross products, are captured below.

$$\vec{v}=\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \qquad |\vec{x}|=\sqrt{x_1^{2}+x_2^{2}+x_3^{2}} \qquad \hat{x}=\begin{bmatrix} x_1/|\vec{x}| \\ x_2/|\vec{x}| \\ x_3/|\vec{x}| \end{bmatrix} \qquad \vec{x}\cdot\vec{y}=x_1\cdot y_1+x_2\cdot y_2+x_3\cdot y_3 \qquad \vec{x}\times\vec{y}=\begin{bmatrix} x_2 y_3-x_3 y_2 \\ x_3 y_1-x_1 y_3 \\ x_1 y_2-x_2 y_1 \end{bmatrix}$$

Projection and rejection operations can determine vector components parallel or perpendicular to each other. Their notation/equations are below. Bars denote a scalar value representing the magnitude of the projection.

$$|proj_{\vec{x}}\,\vec{y}|=\frac{\vec{x}\cdot\vec{y}}{|\vec{y}|} \qquad proj_{\vec{x}}\,\vec{y}=\frac{\vec{x}\cdot\vec{y}}{\vec{y}\cdot\vec{y}}\,y \qquad reject_{\vec{x}}\,\vec{y}=x-\frac{\vec{x}\cdot\vec{y}}{\vec{y}\cdot\vec{y}}\,y$$

Unit quaternions were selected to represent the internal state of the IMU. Unlike Euler Angles, which suffer from gimbal lock, they are robust. Also, they are mathematically more efficient and numerically stable than orientation matrices.

$$q=\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \quad \text{where} \quad q_1^{2}+q_2^{2}+q_3^{2}+q_4^{2}=1$$

The best way to understand orientation is through their axis-angle representation, where a unit vector, v, "indicates the direction of an axis of rotation" and an angle, theta, "describes the magnitude of the rotation around the axis." http://en.wikipedia.org/wiki/Axis-angle_representation

$$q=\begin{bmatrix}\cos(\theta/2)\\ v_1\sin(\theta/2)\\ v_2\sin(\theta/2)\\ v_3\sin(\theta/2)\end{bmatrix}$$

The conjugate and multiplication operations are defined below.

$$x^{-1}=[x_{1,}-x_{2,}-x_{3,}-x_4] \qquad x*y=\begin{bmatrix}x_1y_1-x_2y_2-x_3y_3-x_4y_4\\ x_1y_2+x_2y_1+x_3y_4-x_4y_3\\ x_1y_3-x_2y_4+x_3y_1+x_4y_2\\ x_1y_4+x_2y_3-x_3y_2+x_4y_1\end{bmatrix}$$

The derivative of a quaternion, where omega is the rotational rates for each axis, is defined below.

$$\omega=\begin{bmatrix}0\\ \omega_x\\ \omega_y\\ \omega_z\end{bmatrix} \qquad q'=\frac{1}{2}q*\omega \qquad q'=\begin{bmatrix}-\frac{1}{2}q_2\omega_x-\frac{1}{2}q_3\omega_y-\frac{1}{2}q_4\omega_z\\ \frac{1}{2}q1\omega_x+\frac{1}{2}q_3\omega_z-\frac{1}{2}q_4\omega_y\\ \frac{1}{2}q_1\omega_y-\frac{1}{2}q_2\omega_z-\frac{1}{2}q_4\omega_x\\ \frac{1}{2}q_1\omega_z+\frac{1}{2}q_2\omega_y-\frac{1}{2}q_3\omega_x\end{bmatrix}$$

The formula for rotating a vector by a unit quaternion is captured below. To rotate in opposite direction the conjugate of the quaternion is used.

$$rotate_{\vec{x}}q=q*\begin{bmatrix}0\\ x\end{bmatrix}*q^{-1} \qquad rotate_{\vec{x}}q=\begin{bmatrix}-q_2x_x-q_3x_y-q_4x_z\\ q_1x_x+q_3x_z-q_4x_y\\ q_1x_y-q_2x_z+q_4x_x\\ q_1x_z+q_2x_y-q_3x_x\end{bmatrix}*\begin{bmatrix}q_1\\ -q_2\\ -q_3\\ -q_4\end{bmatrix}$$

$$rotate^{-1}{}_{\vec{x}}q=q^{-1}*\begin{bmatrix}0\\ x\end{bmatrix}*q \qquad rotate^{-1}{}_{\vec{x}}q=\begin{bmatrix}q_2x_x+q_3x_y+q_4x_z\\ q_1x_x-q_3x_z+q_4x_y\\ q_1x_y+q_2x_z-q_4x_x\\ q_1x_z-q_2x_y+q_3x_x\end{bmatrix}*\begin{bmatrix}q_1\\ q_2\\ q_3\\ q_4\end{bmatrix}$$

The rotation that provides the shortest arc between two vectors is captured below. The cross product does not produce valid results when the vectors are parallel. This condition has to be checked prior as part of the algorithm.

$$rotate_{\vec{x}}\vec{y} = \begin{bmatrix} \frac{s}{2} \\ \frac{x \times v}{s} \end{bmatrix} \quad \text{where} \quad s = \sqrt{2(1+\vec{x}\cdot\vec{y})}$$

$$rotate_{\vec{x}}\vec{y} = \begin{bmatrix} s/2 \\ (x_2 y_3 - x_3 y_2)/s \\ (x_3 y_1 - x_1 y_3)/s \\ (x_1 y_2 - x_2 y_1)/s \end{bmatrix} \quad \text{where} \quad s = \sqrt{2(1 + x_1 y_1 + x_2 x_2 + x_3 y_3)}$$

Another method of estimating the quaternion rotation between two vectors is through an optimization algorithm, which minimizes the error for the objective function below. This was used by Sebastian Madgwick and served as a starting point for the MI IMU algorithm.

$$\min_q f(rotate_{\vec{x}}\, q - \vec{y})$$

The optimization method used by Sebastian Madgwick was the gradient descent algorithm (see below). It was selected due to its efficiency and simplicity. J is the Jacobian matrix of function, f().

$$\vec{y} = \vec{x} - \mu\left(\frac{\nabla f}{|\nabla f|}\right) \quad \text{Where} \quad \nabla f = J_f f()$$

Full expanded, the function to be minimized is depicted below. This formula was originally derived in Sebastian's paper.

$$f() = \begin{bmatrix} 2x_x(\frac{1}{2} - q_3^2 - q_4^2) + 2x_y(q_1 q_4 + q_2 q_3) + 2x_z(q_2 q_4 - q_1 q_3) - y_x \\ 2x_x(q_2 q_3 - q_1 q_4) + 2x_y(\frac{1}{2} - q_2^2 - q_4^2) + 2x_z(q_1 q_2 - q_3 q_4) - y_y \\ 2x_x(q_1 q_3 - q_2 q_4) + 2x_y(q_3 q_4 - q_1 q_2) + 2x_z(\frac{1}{2} - q_2^2 - q_3^2) - y_z \end{bmatrix}$$

The Jacobian for objective function is:

$$J_f = \begin{bmatrix} 2x_y q_4 - 2x_z q_3 & 2x_y q_3 + 2x_z q_4 & -4x_x q_3 + 2x_y q_2 - 2x_z q_1 & -4x_x q_4 + 2x_y q_1 + 2x_z q_2 \\ -2x_x q_4 + 2x_z q_2 & 2x_x q_3 - 4x_y q_2 + 2x_z q_1 & 2x_x q_2 + 2x_z q_4 & -2x_x q_1 - 4x_y q_4 + 2x_z q_3 \\ 2x_x q_3 - 2x_y q_2 & 2x_x q_4 - 2x_y q_1 - 4x_z q_2 & 2x_x q_1 + 2x_y q_4 - 4x_z q_3 & 2x_x q_2 2 d_y q - 3 \end{bmatrix}$$

Although quaternions were selected to represent the IMU state, some operations are easier to perform using other methods. Rotation matrices are used to specify a reference axis-set. Rotation matrices are orthogonal square matrices, whose determine is zero. Rotation matrices will be denoted by a capital letter.

$$R=\begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} \\ R_{2,1} & R_{2,2} & R_{2,3} \\ R_{3,1} & R_{3,2} & R_{3,3} \end{bmatrix}$$

The best way to interpret rotation matrices is through three direction vectors, each specifying an axis of a local coordinate system.

$$R=[\vec{x} \quad \vec{y} \quad \vec{z}] \qquad R=\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix}$$

To rotate a vector using a rotation matrix, the vector is multiplied by the matrix as shown below

$$rotate_{\vec{x}} R = R * x \qquad rotate_{\vec{x}} R = \begin{bmatrix} x_1 R_{1,1} + x_2 R_{1,2} + x_3 R_{1,3} \\ x_1 R_{2,1} + x_2 R_{2,2} + x_3 R_{2,3} \\ x_1 R_{3,1} + x_2 R_{3,2} + x_3 R_{3,3} \end{bmatrix}$$

To convert from a rotation matrix into a quaternion, the following equation should be used:

$$q_R = \begin{bmatrix} \frac{n}{4} \\ \frac{R_{3,2}-R_{2,3}}{n} \\ \frac{R_{1,3}-R_{3,1}}{n} \\ \frac{R_{2,1}-R_{1,2}}{n} \end{bmatrix} \quad \text{where} \quad n=2\sqrt{(1+R_{1,1}+R_{2,2}+R_{3,3})} \quad \text{if} \quad R_{1,1}+R_{2,2}+R_{3,3}>0$$

$$q_R = \begin{bmatrix} \frac{R_{3,2}-R_{2,3}}{n} \\ \frac{n}{4} \\ \frac{R_{1,2}-R_{2,1}}{n} \\ \frac{R_{1,3}-R_{3,1}}{n} \end{bmatrix} \quad \text{where} \quad n=2\sqrt{(1+R_{1,1}-R_{2,2}-R_{3,3})} \quad \text{else if} \quad R_{1,1}>R_{2,2} \wedge R_{1,1}>R_{2,2}$$

$$q_R = \begin{bmatrix} \frac{R_{1,3}-R_{3,1}}{n} \\ \frac{R_{1,2}-R_{2,1}}{n} \\ \frac{n}{4} \\ \frac{R_{2,3}-R_{3,2}}{n} \end{bmatrix} \quad \text{where} \quad n = 2\sqrt{(1+R_{2,2}-R_{1,1}-R_{3,3})} \quad \text{else if} \quad R_{2,2} > R_{3,3}$$

$$q_R = \begin{bmatrix} \frac{R_{2,1}-R_{1,2}}{n} \\ \frac{R_{1,3}-R_{3,1}}{n} \\ \frac{R_{2,3}-R_{3,2}}{n} \\ \frac{n}{4} \end{bmatrix} \quad \text{where} \quad n = 2\sqrt{(1+R_{3,3}-R_{1,1}-R_{2,2})} \quad \text{else}$$

Inputs and Outputs

There are three IMU sensor inputs, each represented by a three-dimensional (3D) vector: gyroscope (g), accelerometer (a), and magnetometer (m). The accelerometer measure acceleration and can be broken into two components, the gravity component, which opposes the force of gravity (points upward), and the motion component, which represents device acceleration relative to the earth. The magnetometer points to the magnetic north pole. Since the direction of the magnetic pole is not orthogonal to the grown plane, i.e. does not form a 90 degree angle with the accelerometer gravity component, it can also be decomposed into two components, the forward component which is the projection of the vector onto the ground plane, and the downward component, which is perpendicular to the ground plane. The gyroscope measure rates relative to three axis of the device, an input for calculating the quaternion derivative (see previous section). This is summarized in the equations below.

$$\vec{a} = \vec{a}_g + \vec{a}_m \qquad \vec{m} = \vec{m}_f + \vec{m}_d \qquad \vec{g} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

The primary responsibility of the IMU is to estimate the orientation, which is internally represented as a quaternion, and acceleration, relative to the ground plane. The gyroscope rates is used to update the orientation while the accelerometer gravity component and the magnetometer forward component is used to dead-reckon the estimate. Acceleration, relative to the ground plane, is derived from removing orientation information from the accelerometer input. This estimate is significantly more noisy than the orientation estimate due to lack of sensor redundancy, and noise from the orientation is included in this calculating.

To perform analytics, it is preferable to report orientation using an intuitive angle-space representation. The most common representation is Euler angles, i.e. yaw, pitch, and roll. These angles represent three rotations (order is important) to constitute one transform. To convert quaternions into Euler Angles, the following formulas can be used. These three values will constitute the orientation output.

$$yaw = atan2(2(q_2 q_3 + q_4 q_1), q_2 - q_3 - q_4 + q_1)$$
$$pitch = asin(-2(q_2 q_4 - q_3 q_1))$$
$$roll = atan2(2(q_3 q_4 + q_2 q_1), -q_2 - q_3 + q_4 + q_1)$$

Definitions

Normalize Quaternions

[1, 0, 0, 0]	no rotation
[0, 1, 0, 0]	180deg turn around X axis
[0, 0, 1, 0]	180deg turn around Y axis
[0, 0, 0, 1]	180deg turn around Z axis
[sqrt(0.5), sqrt(0.5), 0, 0]	90deg turn around X axis
[sqrt(0.5), 0, sqrt(0.5), 0]	90deg turn around Y axis
[sqrt(0.5), 0, 0, sqrt(0.5)]	90deg turn around Z axis
[sqrt(0.5), -sqrt(0.5), 0, 0]	-90deg turn around X axis
[sqrt(0.5), 0, -sqrt(0.5), 0]	-90deg turn around Y axis
[sqrt(0.5), 0, 0, -sqrt(0.5)]	-90deg turn around Z axis

Gyroscope Equations

$${}^{S}_{E}(\dot{q}) <- current\ orientation$$
$${}^{S}_{E}(\dot{q}) = [\dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4]$$
$${}^{S}_{E}(\hat{q}) <- previous\ orientation$$
$${}^{S}_{E}(\hat{q}) = [q_1, q_2, q_3, q_4]$$
$$s_\omega <- gyroscope\ rates$$
$$s_\omega = [0, \omega_x, \omega_y, \omega_z]$$
$${}^{S}_{E}(\dot{q}) = \frac{1}{2} \cdot {}^{S}_{E}(\hat{q}) \times s_\omega$$
$${}^{S}_{E}(\dot{q})_{\omega,t} = \frac{1}{2} \cdot {}^{S}_{E}(\hat{q})_{est,t-1} \times s_{\omega_t}$$
$${}^{S}_{E}(\dot{q})_{\omega,t} = \begin{bmatrix} -\frac{1}{2} q_2 \omega_x - \frac{1}{2} q_3 \omega_y - \frac{1}{2} q_4 \omega_z \\ \frac{1}{2} q1 \omega_x + \frac{1}{2} q_3 \omega_z - \frac{1}{2} q_4 \omega_y \\ \frac{1}{2} q_1 \omega_y - \frac{1}{2} q_2 \omega_z - \frac{1}{2} q_4 \omega_x \\ \frac{1}{2} q_1 \omega_z + \frac{1}{2} q_2 \omega_y - \frac{1}{2} q_3 \omega_x \end{bmatrix}$$
$${}^{S}_{E}(q)_{\omega,t} = {}^{S}_{E}(\hat{q})_{\omega,t-1} + {}^{S}_{E}(\dot{q})_{\omega,t} \Delta t$$

General Optimization Algorithm

$J^T <- Jacobian$

$E_{\hat{d}} <- reference\ direction$

$E_{\hat{d}} = [0, d_x, d_y, d_z]$

$s_{\hat{s}} <- measured\ direction$

$s_{\hat{s}} = [0, s_x, s_y, s_z]$

$$\min_{{}^S_E(\hat{q})_{\omega,t}} f({}^S_E(\hat{q})_{\omega,t}, E_{\hat{d}}, s_{\hat{s}})$$

$$f({}^S_E(\hat{q})_{\omega,t}, E_{\hat{d}}, s_{\hat{s}}) = {}^S_E(\bar{\hat{q}})_{\omega,t} \times E_d \times {}^S_E(\hat{q})_{\omega,t} - s_{\hat{s}}$$

$${}^S_E(\dot{q}) = {}^S_E(\hat{q}) - \mu \frac{\nabla f({}^S_E(\hat{q})_k, E_{\hat{d}}, s_{\hat{s}})}{\|\nabla f({}^S_E(\hat{q})_k, E_{\hat{d}}, s_{\hat{s}})\|}$$

$$\nabla f({}^S_E(\hat{q})_k, E_{\hat{d}}, s_{\hat{s}}) = J^T({}^S_E(\hat{q})_k, E_{\hat{d}}) f({}^S_E(\hat{q})_k, E_{\hat{d}}, s_{\hat{s}})$$