InvenSense Inc.
1197 Borregas Ave., Sunnyvale, CA 94089 U.S.A.
Tel: +1 (408) 988-7339 Fax: +1 (408) 988-8104
Website: www.invensense.com

Document Number: AN-MPU-3000A-17
Revision: 1.5
Release Date: 5/23/2011

Application Note: 9-Axis MotionFusion, Calibration algorithms

A printed copy of this document is
NOT UNDER REVISION CONTROL
unless it is dated and stamped in red ink as,
"REVISION CONTROLLED COPY."

This information furnished by InvenSense is believed to be accurate and reliable. However, no responsibility is assumed by InvenSense for its use, or for any infringements of patents or other rights of third parties that may result from its use. Specifications are subject to change without notice. InvenSense reserves the right to make changes to this product, including its circuits and software, in order to improve its design and/or performance, without prior notice. InvenSense makes no warranties, neither expressed nor implied, regarding the information and specifications contained in this document. InvenSense assumes no responsibility for any claims or damages arising from information contained in this document, or from the use of products and services detailed therein. This includes, but is not limited to, claims or damages based on the infringement of patents, copyrights, mask work and/or other intellectual property rights.

Certain intellectual property owned by InvenSense and described in this document is patent protected. No license is granted by implication or otherwise under any patent or patent rights of InvenSense. This publication supersedes and replaces all information previously supplied. Trademarks that are registered trademarks are the property of their respective companies. InvenSense sensors should not be used or sold in the development, storage, production or utilization of any conventional or mass-destructive weapons or for any other weapons or life threatening applications, as well as in any other life critical applications such as medical equipment, transportation, aerospace and nuclear instruments, undersea equipment, power plant equipment, disaster prevention and crime prevention equipment.

Copyright ©2010 InvenSense Corporation.

TABLE OF CONTENTS

1. Revision History

Revision Date	Revision	Description
05/12/10	01	Initial Release
06/14/10	1.1	Updated Section 6.4 Added Section 6.8, 6.9, 6.10
09/10/10	1.2	Updated Section 4.3
11/19/10	1.3	Updated Section 6
1/24/11	1.4	Added more overall context in section 6, changed some of the headings in the sub-sections. Added Progressive no-motion calibration Added relative quaternion
5/23/2011	1.5	Replaced MPL with MotionApps and sensor fusion with MotionFusion to adhere to the new trademarks.

2. Reference

[1] MotionApps Platform™ Specification

[2] MPU Self-test application note

3. Overview

The purpose of this document is to provide a description of the 9-axis MotionFusion algorithm implemented by the InvenSense Digital Motion Processor and Motion Processing Library. MotionProcessing Library will be referred to as MotionApps™ Platform or MotionApps™ through this document.

Please refer to [1] for any further details on references to APIs and datasets through this document.

4. MotionFusion

4.1 Overview

The MotionFusion algorithm combines gyroscopes, accelerometers, and compasses to provide a more accurate description of motion than any one type of sensor could provide. The three types of sensors have different strengths and weaknesses, and the MotionFusion algorithm leverages these differences to allow each type of sensor to augment the other types of sensors.

4.2 Gyroscopes

The gyroscope provides rotational information about the X, Y, and Z axes of the device. If the starting orientation of the device is correct, and the sensors are well calibrated, a 3-axis gyroscope can correctly track orientation by itself, without the help of accelerometers and compasses. However, over time the gyroscope will drift, due to the fact that it is measuring angular velocity and not angle. The gyroscope does not detect gravity or magnetic North, and therefore it doesn't have any fixed reference to compensate for the drift. If the gyroscope biases are calibrated, the gyroscope-based orientation will drift very gradually. However, if the gyroscope biases have large errors, then a more rapid continuous drift may be observed. For example, if a gyroscope axis has a bias of 3dps, then the output of the gyroscope-based orientation will rotate at 3dps when the device is not moving. The gyroscope bias has some dependence on temperature.

4.3 Accelerometers

Accelerometers detect gravity, and can therefore measure pitch and roll. They cannot measure yaw, because during a yaw rotation there is no change in the direction of gravity relative to the accelerometer. Because accelerometers directly measure pitch and roll, rather than measuring angular velocity like gyroscopes, accelerometer-based tilt does not drift over time. However, this measurement is very poor quality during movement due to the fact that accelerometers also measure linear acceleration and add it to the gravitational measurement. The MotionFusion algorithm uses accelerometers and gyroscopes to balance each other out; gyroscopes provide high quality instantaneous motion tracking, and accelerometers ensure that there is no long term drift in pitch or roll, and provide a fixed reference from gravity. When the device is not moving, and the gyroscope biases are calibrated, the final pitch and roll angles delivered by the MotionFusion algorithm are provided by the accelerometer. This means that any bias or scale factor error in the accelerometer will be translated into an error in pitch or roll in the output of the MotionFusion algorithm. In addition, since compasses require accurate pitch and roll measurements in order to provide an accurate yaw measurement, accelerometer bias and scale factor errors will generate heading errors when compasses are used.

4.4 Compasses

Compasses provide a fixed yaw reference by measuring magnetic North. This fixed reference allows the compass to compensate for gyroscope drift in yaw, which cannot be done with an accelerometer. It also allows the MotionFusion algorithm to be referenced to magnetic North, which is useful for navigation and augmented reality applications. Unlike accelerometers and gyroscopes, compasses are extremely sensitive to the environment, and calibration is more complex. Gyroscopes and accelerometers do not typically

depend on environmental factors except for temperature; however, compasses respond to many different types of magnetic interference.

Devices with compasses in them often contain magnets for other purposes, such as speakers and motors. Other metal objects within the device may be magnetized at some times and not at other times. These magnetic errors are known as "hard iron" errors and induce offsets in the compass measurements. Metal objects that are not magnetized may still be capable of bending the magnetic field in the vicinity; these errors are known as "soft iron" errors and may induce cross axis effects in the compass measurements. Users may unknowingly make these problems worse by placing the device on top of surfaces with magnetic fields, or putting the device in a pocket that also contains headphones with magnets in them.

The compass will also detect RF magnetic interference from sources like cell phone transmissions, Bluetooth, Wi-Fi, and so on. They will also detect magnetic field from strong currents in power traces. This makes compasses difficult to place in PCB designs. The RF interference makes the compass output very noisy, and difficult to use by itself as a motion sensor; however, the gyroscope provides an instantaneous measurement of rotation that compensates for the compass noise. In addition to all this, compass sensors themselves will show internal offset variation.

4.5 Motion Artifacts

If all the sensors are perfectly calibrated, there should not be any drifting or settling artifacts. However, realistically, the sensors will all have some errors, causing the MotionFusion algorithm to occasionally accumulate errors. Motion artifacts may appear whenever any of the sensors disagree with other sensors; if the gyroscope measures a rotation of 100 degrees, but the accelerometer measures a rotation of 105 degrees, there may be some motion artifact as the MotionFusion algorithm determines which value to trust. Typically the MotionFusion algorithm will first travel to the orientation determined by the gyroscope data, and then settle to the orientation determined by the accelerometer and compass.

The MotionFusion algorithm has logic for minimizing the size of these settling artifacts. Small artifacts will be removed so subtly that there will not be any observable effect. Sometimes larger errors accumulate, typically during periods with a lot of movement or external magnetic stimuli; after movement or magnetic stimuli stop, the MotionFusion algorithm will remove these errors quickly, causing a noticeable settling artifact. This may be undesirable, but it is better than the alternative of a slow settling that is interpreted as drifting, or no settling at all, resulting in a fixed error.

Magnetic interference from external metal or magnetic objects will distort the output of the MotionFusion algorithm. A small magnetic interference may cause the compass value to deviate, creating a yaw error in the MotionFusion algorithm. A larger magnetic interference will trigger logic that detects such magnetic disturbances by comparing the compass data to the gyroscope data, and temporarily stops using the compass. The gyroscope can be relied on for short periods of time; however, if the magnetic disturbance remains for long periods of time, reliance on the gyroscope data will result in an accumulated error; when the magnetic disturbance disappears, there will be a noticeable settling artifact as the system recalibrates to the compass values.

5. Sensor Mounting Orientation

5.1 Mounting Matrices

Mounting matrices are used by the MotionApps to allow PCB designers the flexibility to pick any mounting orientation for the sensor. The mounting matrices, when passed into the functions MLSetAccelCalibration, MLSetGyroCalibration, MLSetMagCalibration, swap the sensor axes and apply minus signs where necessary in order to rotate all the sensor data into a reference frame useful for the application. In this way, the same application code can run on systems with different PCB layouts; the only software change is in mounting matrix. The mounting matrix is applied to the sensor data, as in the following example:

$$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -y \\ x \\ z \end{bmatrix}$$

Figure 1: Mounting matrix

When picking the mounting matrix, there are three important considerations:

1. It is best to start by defining an application level coordinate system that makes intuitive sense. The device containing the sensors should be assigned X, Y, and Z axes, and then mounting matrices should be created that rotate the sensor data into the application coordinate system. As an example, the following coordinate system makes intuitive sense to application developers because it is identical to the standard OpenGL coordinate system:

Figure 2: Example application coordinate system

While the target coordinate system does not have to be the one shown above, it does have to be a right-handed coordinate system. In a right handed coordinate system, if you point your right-hand pointer finger along the X axis, and curl your other fingers such that they follow the Y axis, your thumb will point toward the Z axis. The MotionFusion algorithm will not operate correctly if this rule is violated.

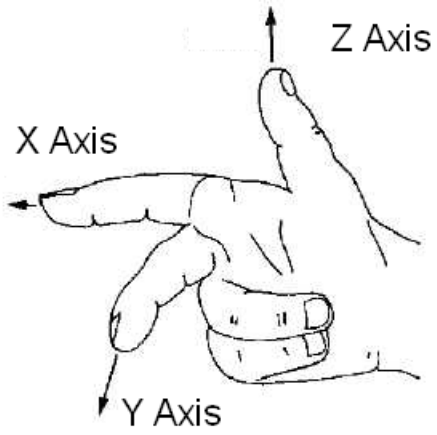

Figure 3: Right-hand rule for coordinate systems

2. Mounting matrices must now be chosen that rotate the sensor data into the application level coordinate system chosen in step 1. Assuming that the raw 3-axis sensor data is already in a right-handed coordinate system, as it usually is, there are restrictions on what mounting matrices are allowable. A good rule of thumb is that when starting with a mounting matrix known to be right-handed, it is valid to switch two axes if a sign is also changed somewhere. Switching two axes without changing a sign is invalid; changing a sign without switching two axes is also invalid. Changing two signs is valid, however. Note that these mounting matrices only support defining orientation in terms of 90 degree increments; they do not support adjusting for other cross axis errors that may be present due to PCB mounting inaccuracies. Because of this, only the numbers 0, 1, and -1 are valid within these matrices.

a) $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ b) $\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ c) $\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ d) $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ e) $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Figure 4: a) Valid matrix b) Valid matrix c) Valid matrix d) Invalid matrix e) Invalid matrix

3. Mounting matrices should rotate all the sensor data into the same coordinate system. If the accelerometer data is not in the same coordinate system as the gyroscope, the MotionFusion algorithm will not operate correctly. For example, when rotating by 45 degrees, the gyroscope may sense a 45 degree rotation, but the accelerometer may measure a -45 degree angle change if the mounting matrices have not been chosen correctly. In this case, the MotionFusion output will show an initial movement toward 45 degrees and than a very large 90 degree settling artifact as the system transitions to -45 degrees.

5.2 Checking Sensor Mounting Orientation

When mounting matrices have been chosen, the sensor data can be tested using the MLGetFloatArray API to get the arrays ML_ACCELS, ML_GYROS, and ML_MAGNETOMETER. This data is not exactly raw sensor data; it has been scaled, swapped, and signs may have changed due to the mounting matrices. When testing this data initially, it is best to disable any auto calibration by setting

MLSetBiasUpdateFunc(ML_NONE). Then the data can be tested by using the diagram in Figure 52 as a reference. Note that this test only needs to be done once when a sensor mounting orientation is changed; this section does not describe a production test.

When testing the accelerometers data with ML_ACCELS, when any axis is pointed upwards, the accelerometer data corresponding to that axis should read +1g. When that axis is horizontal, the data should read 0g; when the axis points downwards, the data should read -1g. If bias errors can be detected when measuring this data, the bias errors can be stored in a file and later set into the array ML_ACCEL_BIAS. This will make the final MotionFusion pitch and roll angles more accurate.

When testing the gyroscope data, the device should be rotated rapidly around each axis. A positive rotation around an axis is defined by a right-hand rule. If you point your right thumb in the direction of an axis, and bend your fingers, your fingers will curl in the direction defined as a positive direction.

Figure 5: Right hand rule for angular velocity

For each axis, if a positive rotation is applied around that axis, the gyroscope data in ML_GYROS array element corresponding to that axis should become large and positive, while the other two elements remain small. When a rotation is applied in the opposite of the direction given by the right-hand rule, the gyroscope data assigned to that axis should become negative.

When testing the compass data mounting, it is important to know the approximate direction of North initially. When testing the compass mounting, for each axis hold the axis horizontally and turning in a circle, so that it points North, West, South, and East. If the mounting has been chosen correctly, the data in ML_MAGNETOMETER corresponding to that axis should show a maximum when pointed North, and a minimum when pointed South. Note that while known the direction of North and holding the axis horizontal makes this test easier, the actual maximum value that a compass axis will deliver is not generally when the axis is horizontal or when it points exactly North. At any given location, there is a deviation from true North known as magnetic declination; in Sunnyvale, the magnetic declination is about 15 degrees. The magnetic field is only horizontal near the equator; at other locations, there is an angle from the horizontal known as the inclination. In Sunnyvale, the magnetic inclination is 61 degrees. This means that the magnetic field is actually more vertical than horizontal; however, there is a large enough horizontal component to allow compassing to work. The units of the data in ML_MAGNETOMETER are in µT; the intensity of the Earth's magnetic field varies from about 30µT to about 60µT. the intensity of the Earth's magnetic field in Sunnyvale is about 50µT, but the horizontal component of this is 20µT.

Once all nine sensor axes have been shown to be aligned with the correct application axis, the MotionFusion algorithm should operate correctly.

6. Calibration Algorithms

6.1 Overview

The InvenSense MotionApps provides a variety of methods of tracking sensor parameters that may change over time. It is crucial to ensure that gyro biases are as accurate as possible at all times. For this purpose, MotionApps runs a suite of calibration algorithms a.k.a. bias trackers which, depending upon the algorithm calculate the biases of the gyros at run-time. These biases are constantly written to non-volatile host memory by the MotionApps using Load / Store Calibration APIs (reference [1]).

These calibration algorithms may be enabled or disabled individually using the API MLSetBiasUpdateFunc with a bitwise OR of the required algorithms. For example, calling MLSetBiasUpdateFunc (ML_BIAS_FROM_NO_MOTION) will enable only that algorithm. Calling MLSetBiasUpdateFunc (ML_NONE) will disable all calibration algorithms. Calling MLSetBiasUpdateFunc (ML_BIAS_FROM_NO_MOTION | ML_BIAS_FROM_LPF) will enable only those two algorithms. Calling MLSetBiasUpdateFunc (ML_ALL) will enable all calibration algorithms; enabling all bias trackers at all times is highly recommended.

These bias trackers are continuously run in the background and automatically update the biases entirely transparent to the user.

6.2 Gyro bias from no-motion (ML_BIAS_FROM_NO_MOTION)

The MotionApps will reset the gyroscope biases when it is observed that the sensor is not moving for some period of time (typically 8 – 10seconds). This is the most accurate way to control the gyroscope bias; however, as it requires the user putting down the device temporarily, it is not practical for many applications. For certain applications that demand the most precise motion sensing, the application can notify the user to put the device down temporarily in order to calibrate the gyroscope biases. For other applications, this bias tracker can still be enabled, but it will act more rarely, updating the biases whenever an opportunity presents itself. This algorithm will only update the biases when a period without motion is detected. Unlike the other calibration algorithms, it writes the calculated values directly to the gyroscope offset trim registers.

Bias values from no-motion can also be easily obtained if the InvenSense self test for MPU family of devices is used on the production line while manufacturing the handset. The self test is 2 seconds in duration and it collects an initial reading of gyro biases vs. temperature. This is considered very stable and assumes that the test conditions are as mentioned in reference [2].

6.3 Gyro bias from Low Pass Filter (ML_BIAS_FROM_LPF)

This tracks the user's movement and determines the gyroscope bias based on the movement. It does not require the user to put down the device. However, it is more limited in the range of offset compensation, being limited to only a few degrees per second of compensation. The algorithm uses temperature in order to determine what this range of compensation should be. Within the range of compensation, certain artifacts may appear, including a lack of sensitivity to motion below a threshold, and a lack of sensitivity to constant angular velocities that may be confused with the gyroscope bias. In extreme cases, a user may experience a brief drifting phenomenon that occurs after an extended rotation in one direction at a very slow rate. Due to these artifacts, the output of the library with this algorithm enabled will not be as accurate as the output with this algorithm disabled, if the gyroscope bias has been measured recently. However, it will provide more stable outputs for applications in which the main requirement is a minimum amount of drift. This algorithm updates continuously.

6.4 Gyro bias from Accel & Compass (ML_BIAS_FROM_GRAVITY)

This algorithm correlates the gyroscope data with the accelerometer data, and removes all drift in pitch and roll relative to the Earth. With this algorithm enabled, drift will only occur in yaw relative to the Earth. The process of correlation will cause some additional sensitivity to linear acceleration in applications that are designed to respond only to rotation. This algorithm updates continuously.

If a compass is present, this algorithm will also use the compass data to update the gyroscope bias. In this case, any long term drift should be eliminated in roll, pitch, and yaw, but some additional sensitivity to magnetic interference will be present.

6.5 Gyro bias from Software Temperature Compensation (ML_BIAS_FROM_TEMPERATURE, ML_LEARN_BIAS_FROM_TEMPERATURE)

The software temperature compensation algorithm removes any gyroscope bias error due to temperature by applying a linear fit to the gyroscope bias. Using the flag ML_BIAS_FROM_TEMPERATURE enables the algorithm. To utilize this algorithm, the gyroscope data must first be measured with an initial reading of temperature using the Self-test (reference [2]) on the production line while the device is motionless; the resulting value would be the slope of the line for gyroscope data versus temperature. This slope can be entered into the array ML_GYRO_TEMP_SLOPE; at this point the DMP will begin compensating for any temperature effects in real-time. This algorithm updates continuously.

Moreover, if the flag ML_LEARN_BIAS_FROM_TEMPERATURE is enabled, the algorithm gradually learns the temperature dependence of the gyro bias. Whenever a period of zero motion is detected, the temperature will be polled, and a temperature compensation table will be updated. If enough information is collected in the temperature compensation table, a linear fit will be calculated, and the gyro bias becomes temperature compensated. In this case, it is not necessary to heat the gyroscope in the factory. The temperature compensation information is typically stored in a calibration file such that the data can be updated over the lifetime of the device, as it is exposed to a range of different temperatures.

6.6 Gyro bias from Progressive no-motion (ML_PROGRESSIVE_NO_MOTION)

This bias tracker algorithm uses a combination of user activities (no-motion, small degrees of movement, etc.) as well as cumulative sensor information (accel and compass) to converge to accurate biases much faster than other bias trackers. This is normally useful when there is not much opportunity of using above mentioned more accurate bias trackers.

6.7 Compass bias using figure “8” (ML_MAG_BIAS_FROM_MOTION)

Since compass biases depend on the internal sensor biases and external magnetic fields, these biases may vary considerably from unit to unit after assembly, and may change in the field as well. This calibration algorithm determines when useful data is available by comparing the compass values with the gyro values, and calculates the compass biases when enough data has been collected. The biases will update at this point; this will be apparent if the software is polling the biases stored in ML_MAG_BIAS, as these numbers will change at this point. Before the compass is calibrated, the output of MotionFusion will be constructed from only gyroscope and accelerometer data. When the compass calibrates, any error in yaw will be removed immediately, resulting in a noticeable change in the MotionFusion data.

Giving the algorithm enough data to calibrate can be done by using a “figure eight” movement as is usually suggested in current handsets that contain compasses. However, doing this movement correctly requires a little more understanding. The goal of the “figure eight” movement is really to expose the compass to as many orientations as possible. It is possible to move the device in a figure eight pattern without ever

changing it's orientation; this will not result in any useful compass calibration. The movement does not have to be a figure eight; it is also valid to rotate the device randomly until it has been exposed to enough orientations.

When moving the compass in order to calibrate it, it is best to avoid external magnetic fields. If a figure eight pattern is used such that during part of the movement the compass is close to a magnetic field, and during another part it is further away from a magnetic field, the calibration algorithm may not work well. Placing the compass on a surface while doing this is also dangerous, as many surfaces have magnetic fields.

6.8 Compass bias using gyros (ML_MAG_BIAS_FROM_GYRO)

This algorithm compares the compass data to the gyroscope data and updates the compass bias. It runs continuously in the background. Using gyro data to calibrate compass biases allows the compass heading to converge to something useful with less movement then is required for a typical compass calibration algorithm. For example, instead of doing a figure eight movement to generate enough data to calibrate the compass, a small back-and-forth rotation will cause the compass to calibrate such that the heading is roughly within +/- 20 degrees or so. While this heading is not particularly accurate, it is accurate enough to be useful for a user trying to determine which way to start walking. To get a more accurate compass calibration, more movement is necessary.

6.9 Reset compass biases automatically (ML_AUTO_RESET_MAG_BIAS)

This algorithm determines when the compass calibration is incorrect, and resets all the compass calibration algorithms. This is helpful when using compasses with offset which may change dramatically. The algorithm compares the compass data to the gyroscope data in order to determine when the compass calibration data is incorrect.

6.10 Getting/Setting gyro and compass calibration values after self test and run-time (MLGetArray / MLSetArray)

In some cases, it is useful to be able to explicitly set calibration parameters. For example, as mentioned in sections 6.2 Gyro bias from no-motion, after assembly, the self test writes measures gyroscope biases by measuring the gyro data while the device is stationary and stores this data in a file. When the device is powered up, by default its initial gyro bias estimate is zero for all three axes. In order to start with a more accurate measurement, a call to MLSetArray (ML_GYRO_BIAS, gyroBias) can be made, in which the array "gyroBias" contains gyro bias measurements that were stored in the file.

Similarly, after a successful compass calibration, the estimated compass biases can be retrieved using a call to MLGetArray (ML_MAG_BIAS, magBias). The resulting values in the array "magBias" can then be stored to a file. If the system is powered down and powered back up, this compass bias values can be reloaded into the MotionApps by first loading them from the file into the array magBias, and then calling MLSetArray(ML_MAG_BIAS, magBias). Reference code for implementing of storing into and loading from a file is provided in MotionApps but the final implementation for this is left up to the system integrator.

MLGetArray and MLSetArray can also be used later on during the actual handset use to respectively get and set the most recent set of gyro bias and compass bias values.

6.11 Resetting the compass calibration

Sometimes a user may want to deliberately restart the compass calibration algorithm. This can be done by callig MLResetMagCalibration. After this is called, some additional motion will be required to recalibrate the compass.

6.12 Checking the current calibration accuracy

Retrieving the array ML_MAG_BIAS_ERROR will allow a user to see how accurately the compass is calibrated. ML_MAG_BIAS_ERROR contains three values, one for each axis. These values are unitless and approximate, but values under 100 typically indicate well calibrated axes. As en example, if the Z axis is vertical, it is not necessary for that axis to be well calibrated; only the X and Y axes need to be well calibrated.

7. MotionFusion Output

7.1 Quaternion

The MotionFusion algorithm can output a quaternion, which represents the orientation of the motion sensors in 3D. This is very important in motion-based applications such as pedestrian navigation, gaming, etc.

7.2 Relative Quaternion

This is the same as quaternion mentioned above except for the fact that it does not reset its direction according to true north. This can be used to avoid any settling effects, for example, while playing a first-person shooter game.

7.3 Rotation Matrix

The algorithm can output a rotation matrix which is equivalent to the quaternion, but in a 3x3 matrix format.

7.4 Euler Angles

The MotionFusion algorithm can output the Euler angles roll, pitch, and yaw, which are derived from the internally held quaternion and rotation matrix. Roll, pitch, and yaw are not used internally in the algorithm because of the poor mathematical properties, but are useful as output data for application programs.

7.5 Linear Acceleration in World Coordinates

Using the 3D orientation data, the MotionFusion algorithm removes the gravitational component from the accelerometer data, and rotates the resulting data into world coordinates, providing a measurement of acceleration in world coordinates. Note that this resulting acceleration vector is highly dependent on the accelerometer and gyroscope sensor errors, and typically cannot be double-integrated to provide position. Its primary use is to provide information about transient linear movements relative to the Earth.

7.6 Linear Acceleration in Body Coordinates

Using the 3D orientation data, the MotionFusion algorithm removes the gravitational component from the accelerometer data, and provides the output as acceleration in body coordinates. Note that this resulting

acceleration vector is highly dependent on the accelerometer and gyroscope sensor errors, and typically cannot be double-integrated to provide position. Its primary use is to provide information about transient linear movements relative to device.

7.7 Angular velocity in Body Coordinates

Using the 3D orientation data, the MotionFusion algorithm removes DC components from the gyroscope data in pitch and roll relative to the Earth, and provides angular velocity data that is more stable than pure gyroscope data in pitch and roll. Note that yaw data will still have drift due to the gyroscope, as the MotionFusion algorithm does not have a yaw reference with which to stabilize this degree of freedom.

7.8 Gravity

Using the 3D orientation data, the MotionFusion algorithm removes the linear components from the accelerometer data, and provides the output as a vector indicating the direction of gravity relative to the device. This data can be used as a tilt measurement similar to the raw accelerometer data normally used, but will provide dynamic tilt information in addition to the static tilt information provided by accelerometers alone.