Calculation of Orientation Matrices from Sensor Data

Technical Note

Document XXX

Version: Draft

Authors: Mark Pedley and Michael Stanley

Date: xx Aug 2014

Table of Contents

1 Introduction **4**
1.1 Summary 4
1.2 Functions 4

2 General Rotation Matrix **6**
2.1 Introduction 6
2.2 Eigenvector and Eigenvalue 6
2.3 Vector Multiplication 6

3 Accelerometer Tilt Orientation Matrix **7**
3.1 Introduction 7
3.2 Aerospace / NED 7
3.3 Android 8
3.4 Windows 8 10

4 Magnetometer Compass Orientation Matrix **12**
4.1 Introduction 12
4.2 Aerospace / NED 12
4.3 Android 13
4.4 Windows 8 14

5 Accelerometer Magnetometer Tilt eCompass Orientation Matrix **16**
5.1 Introduction 16
5.2 Aerospace / NED 16
5.3 Android 17
5.4 Windows 8 17

6 Converting Between Rotation Matrix and Rotation Quaternion **19**
6.1 Introduction 19
6.2 Rotation Matrix from Quaternion 19
6.3 Quaternion from Rotation Matrix 19

7 Converting Between Rotation Matrix and Rotation Vector **21**
7.1 Introduction 21
7.2 Rotation Matrix from Rotation Vector 21
7.3 Rotation Vector from Rotation Matrix 21
7.4 Virtual Gyro From Rotation Vector 22

8 Low Pass Filtering of Inclination Angle **24**

Glossary

B	Geomagnetic field strength
$\boldsymbol{B}_c$	Calibrated magnetometer measurements in the sensor PCB frame
B_{cx}, B_{cy}, B_{cz}	x, y and z components of the calibrated magnetometer reading
ENU	x=East, y=North, z=Up coordinate system (Android and Windows 8)
$\boldsymbol{G}_p$	Accelerometer reading in the rotated sensor PCB frame
G_{px}, G_{py}, G_{pz}	x, y and z components of the accelerometer reading
NED	x=North, y=East, z=Down coordinate system (Aerospace)
R_{ij}	Element in row i and column j of orientation matrix $\boldsymbol{R}$
$\boldsymbol{R}_{Android}$	Orientation matrix in the Android coordinate system
$\boldsymbol{R}_{NED}$	Orientation matrix in the NED/Aerospace coordinate system
$\boldsymbol{R}_{Win8}$	Orientation matrix in the Windows 8 coordinate system
R_{xy}	Element in row x and column y of orientation matrix $\boldsymbol{R}$
δ	Geomagnetic inclination angle
θ	Pitch angle
ϕ	Roll angle
ψ	Yaw angle

1 Introduction

1.1 Summary

This Application Note introduces the rotation or orientation matrix and documents the functions in file orientation.c used to compute the orientation matrix from sensor measurements.

1.2 Functions

void f3DOFTiltNED(float fR[][3], float fGp[]);

Computes the roll, pitch, tilt orientation matrix from accelerometer measurements in the Aerospace/NED coordinate system.

See section 2.2.

void f3DOFTiltAndroid(float fR[][3], float fGp[]);

Computes the roll, pitch, tilt orientation matrix from accelerometer measurements in the Android coordinate system.

See section 2.3.

void f3DOFTiltWin8(float fR[][3], float fGp[]);

Computes the roll, pitch, tilt orientation matrix from accelerometer measurements in the Windows 8 coordinate system

See section 2.4.

void f3DOFMagnetometerMatrixNED(float fR[][3], float fBc[]);

Computes the 2D eCompass orientation matrix from calibrated magnetometer measurements in the Aerospace/NED coordinate system.

See section 3.2.

void f3DOFMagnetometerMatrixAndroid(float fR[][3], float fBc[]);

Computes the 2D eCompass orientation matrix from calibrated magnetometer measurements in the Android coordinate system.

See section 3.3.

void f3DOFMagnetometerMatrixWin8(float fR[][3], float fBc[]);

Computes the 2D eCompass orientation matrix from calibrated magnetometer measurements in the Windows 8 coordinate system.

See section 3.4.

void feCompassNED(float fR[][3], float *pfDelta, float fBc[], float fGp[]);

Computes the full orientation matrix from accelerometer and calibrated magnetometer measurements in the Aerospace/NED coordinate system plus the geomagnetic inclination angle.

See section 4.2.

void feCompassAndroid(float fR[][3], float *pfDelta, float fBc[], float fGp[]);

Computes the full orientation matrix from accelerometer and calibrated magnetometer measurements in the Android coordinate system plus the geomagnetic inclination angle.

See section 4.3.
void feCompassWin8(float fR[][3], float *pfDelta, float fBc[], float fGp[]); Computes the full orientation matrix from accelerometer and calibrated magnetometer measurements in the Windows 8 coordinate system plus the geomagnetic inclination angle. See section 4.4.
void fRotationMatrixFromQuaternion(float R[][3], const struct fquaternion *pq); Computes a rotation matrix from a rotation quaternion. See section 5.2.
void fQuaternionFromRotationMatrix(float R[][3], struct fquaternion *pq); Computes a rotation quaternion from a rotation matrix. See section 5.3.
void fRotationVectorDegFromRotationMatrix(float R[][3], float rvecdeg[]); Computes the rotation vector (rotation axis and angle) from a rotation matrix. See section 6.
void fLPFScalar(float *pfS, float *pfLPS, float flpf, int32 loopcounter) Low pass filters a scalar time series. Used for low pass filtering the inclination angle. See section 7.

2 General Rotation Matrix

2.1 Introduction

The general rotation matrix $\boldsymbol{R}$ which transforms a vector as a result of a rotation of the coordinate system around the axis $\hat{\boldsymbol{n}}$ by angle η is:

$$\boldsymbol{R} = \begin{pmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{pmatrix} = \begin{pmatrix} \hat{n}_x{}^2 + \left(1-\hat{n}_x{}^2\right)cos\eta & \hat{n}_x\hat{n}_y(1-cos\eta)+\hat{n}_z sin\eta & \hat{n}_x\hat{n}_z(1-cos\eta)-\hat{n}_y sin\eta \\ \hat{n}_x\hat{n}_y(1-cos\eta)-\hat{n}_z sin\eta & \hat{n}_y{}^2 + \left(1-\hat{n}_y{}^2\right)cos\eta & \hat{n}_y\hat{n}_z(1-cos\eta)+\hat{n}_x sin\eta \\ \hat{n}_x\hat{n}_z(1-cos\eta)+\hat{n}_y sin\eta & \hat{n}_y\hat{n}_z(1-cos\eta)-\hat{n}_x sin\eta & \hat{n}_z{}^2 + \left(1-\hat{n}_z{}^2\right)cos\eta \end{pmatrix}$$

Eq 2.1.1

Equation 2.1.1 is known as the Rodrigues rotation matrix. The combination of rotation axis $\hat{\boldsymbol{n}}$ and rotation angle η is termed a rotation vector.

If the coordinate system rotates by angle η about the z axis then a vector rotates by angle $-\eta$ about the z axis relative to the newly rotated coordinate system. When Freescale refers to a rotation vector defining a rotation matrix then it always refers to a coordinate system rotation.

In addition, when a rotation matrix $\boldsymbol{R}$ models the orientation of the sensors relative to the earth (or global) reference frame, then the Freescale standard is that the product $\boldsymbol{Rv}$ transforms the vector $\boldsymbol{v}$ from the global to the sensor frame. The inverse rotation $\boldsymbol{R}^{-1}\boldsymbol{v} = \boldsymbol{R}^T\boldsymbol{v}$ transforms the vector $\boldsymbol{v}$ from the sensor to the global frame. The orientation matrix is always defined relative to an initial orientation in the global frame with the product flat and pointed to magnetic north.

2.2 Eigenvector and Eigenvalue

The product $\boldsymbol{R}\hat{\boldsymbol{n}}$ evaluates to $\hat{\boldsymbol{n}}$ confirming that $\hat{\boldsymbol{n}}$ is the eigenvector of $\boldsymbol{R}$ with eigenvalue 1. Since $\boldsymbol{R}$ is a rotation matrix, $\hat{\boldsymbol{n}}$ is therefore the rotation axis.

$$\boldsymbol{R}\hat{\boldsymbol{n}} = \begin{pmatrix} \hat{n}_x{}^2 + \left(1-\hat{n}_x{}^2\right)cos\eta & \hat{n}_x\hat{n}_y(1-cos\eta)+\hat{n}_z sin\eta & \hat{n}_x\hat{n}_z(1-cos\eta)-\hat{n}_y sin\eta \\ \hat{n}_x\hat{n}_y(1-cos\eta)-\hat{n}_z sin\eta & \hat{n}_y{}^2 + \left(1-\hat{n}_y{}^2\right)cos\eta & \hat{n}_y\hat{n}_z(1-cos\eta)+\hat{n}_x sin\eta \\ \hat{n}_x\hat{n}_z(1-cos\eta)+\hat{n}_y sin\eta & \hat{n}_y\hat{n}_z(1-cos\eta)-\hat{n}_x sin\eta & \hat{n}_z{}^2 + \left(1-\hat{n}_z{}^2\right)cos\eta \end{pmatrix}\begin{pmatrix} \hat{n}_x \\ \hat{n}_y \\ \hat{n}_z \end{pmatrix}$$

Eq 2.2.1

$$= \begin{pmatrix} \hat{n}_x\left(\hat{n}_x{}^2+\hat{n}_y{}^2+\hat{n}_z{}^2\right) - \left(\hat{n}_x{}^2+\hat{n}_y{}^2+\hat{n}_z{}^2-1\right) \\ \hat{n}_y\left(\hat{n}_x{}^2+\hat{n}_y{}^2+\hat{n}_z{}^2\right) - \left(\hat{n}_x{}^2+\hat{n}_y{}^2+\hat{n}_z{}^2-1\right) \\ \hat{n}_z\left(\hat{n}_x{}^2+\hat{n}_y{}^2+\hat{n}_z{}^2\right) - \left(\hat{n}_x{}^2+\hat{n}_y{}^2+\hat{n}_z{}^2-1\right) \end{pmatrix} = \begin{pmatrix} \hat{n}_x \\ \hat{n}_y \\ \hat{n}_z \end{pmatrix}$$

Eq 2.2.2

2.3 Vector Multiplication

If the rotation matrix $\boldsymbol{R}$ is applied to an arbitrary vector $\boldsymbol{v}$, the resulting vector is:

$$\boldsymbol{Rv} = \begin{pmatrix} \hat{n}_x{}^2 + \left(1-\hat{n}_x{}^2\right)cos\eta & \hat{n}_x\hat{n}_y(1-cos\eta)+\hat{n}_z sin\eta & \hat{n}_x\hat{n}_z(1-cos\eta)-\hat{n}_y sin\eta \\ \hat{n}_x\hat{n}_y(1-cos\eta)-\hat{n}_z sin\eta & \hat{n}_y{}^2 + \left(1-\hat{n}_y{}^2\right)cos\eta & \hat{n}_y\hat{n}_z(1-cos\eta)+\hat{n}_x sin\eta \\ \hat{n}_x\hat{n}_z(1-cos\eta)+\hat{n}_y sin\eta & \hat{n}_y\hat{n}_z(1-cos\eta)-\hat{n}_x sin\eta & \hat{n}_z{}^2 + \left(1-\hat{n}_z{}^2\right)cos\eta \end{pmatrix}\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}$$

Eq 2.3.1

$$= \begin{pmatrix} \left\{\hat{n}_x{}^2 + \left(1-\hat{n}_x{}^2\right)cos\eta\right\}v_x + \left\{\hat{n}_x\hat{n}_y(1-cos\eta)+\hat{n}_z sin\eta\right\}v_y + \left\{\hat{n}_x\hat{n}_z(1-cos\eta)-\hat{n}_y sin\eta\right\}v_z \\ \left\{\hat{n}_x\hat{n}_y(1-cos\eta)-\hat{n}_z sin\eta\right\}v_x + \left\{\hat{n}_y{}^2 + \left(1-\hat{n}_y{}^2\right)cos\eta\right\}v_y + \left\{\hat{n}_y\hat{n}_z(1-cos\eta)+\hat{n}_x sin\eta\right\}v_z \\ \left\{\hat{n}_x\hat{n}_z(1-cos\eta)+\hat{n}_y sin\eta\right\}v_x + \left\{\hat{n}_y\hat{n}_z(1-cos\eta)-\hat{n}_x sin\eta\right\}v_y + \left\{\hat{n}_z{}^2 + \left(1-\hat{n}_z{}^2\right)cos\eta\right\}v_z \end{pmatrix}$$

Eq 2.3.2

3 Accelerometer Tilt Orientation Matrix

3.1 Introduction

Accelerometer sensors are sensitive to both linear acceleration and gravity and are insensitive to rotation about the vertical gravity axis. An accelerometer sensor operating without other sensors can therefore detect roll, pitch and tilt angles from horizontal in the absence of linear acceleration but cannot detect compass heading.

The mathematics below therefore assumes that the yaw or compass angle is always zero degrees. One consequence is that the orientation undergoes a twist of 180 degrees at ±90^o pitch (Aerospace / NED coordinate system) and at ±180^o roll (Android and Windows 8 coordinate systems) to ensure that the orientation remains pointing northwards.

Another way of understanding the same phenomenon is (in the Aerospace / NED coordinate system) to apply a 180^o roll (resulting in the circuit board pointed northwards but inverted) followed by a 180^o yaw rotation (which has no effect since the accelerometer is insensitive to rotation about the vertical gravity axis). The same orientation can also be reached by a 180^o pitch rotation. Without the 180^o orientation twist at 90^o pitch, the same final accelerometer reading would lead to two different orientations dependent on the path taken.

3.2 Aerospace / NED

This section documents the mathematics underlying the function f3DOFTiltNED in file orientation.c.

The Aerospace / NED orientation matrix for zero yaw angle ψ after rotation from the horizontal by pitch angle θ followed by roll angle ϕ is:

$$\boldsymbol{R}_{NED}(\psi = 0) = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ \sin\theta\sin\phi & \cos\phi & \cos\theta\sin\phi \\ \cos\phi\sin\theta & -\sin\phi & \cos\theta\cos\phi \end{pmatrix} \qquad \text{Eq 3.2.1}$$

The Aerospace / NED accelerometer reading at this orientation is:

$$\begin{pmatrix} G_{px} \\ G_{py} \\ G_{pz} \end{pmatrix} = |\boldsymbol{G}_p| \begin{pmatrix} -\sin\theta \\ \sin\phi\cos\theta \\ \cos\phi\cos\theta \end{pmatrix} = |\boldsymbol{G}_p| \begin{pmatrix} -\sin\theta \\ \sin\phi\sqrt{1-\left(\frac{G_{px}}{|\boldsymbol{G}_p|}\right)^2} \\ \cos\phi\sqrt{1-\left(\frac{G_{px}}{|\boldsymbol{G}_p|}\right)^2} \end{pmatrix}$$

Eq 3.2.2

$$\sin\theta = \frac{-G_{px}}{|\boldsymbol{G}_p|}$$

Eq 3.2.3

$$\cos\theta = \sqrt{1-\left(\frac{G_{px}}{|\boldsymbol{G}_p|}\right)^2}$$

Eq 3.2.4

$$\sin\phi = \frac{G_{py}}{|\boldsymbol{G}_p|\sqrt{1-\left(\frac{G_{px}}{|\boldsymbol{G}_p|}\right)^2}}$$

Eq 3.2.5

$$cos\phi = \frac{G_{pz}}{|\boldsymbol{G}_p|\sqrt{1-\left(\frac{G_{px}}{|\boldsymbol{G}_p|}\right)^2}}$$

Eq 3.2.6

The positive square root for $cos\theta$ can always be taken in equation 3.2.4 since the pitch angle θ in the Aerospace / NED coordinate system varies between -90° and +90° giving a non-negative value of $cos\theta$.

Substituting into equation 3.2.1 gives the 3DOF orientation matrix directly in terms of the accelerometer reading as:

$$\boldsymbol{R}_{NED}(\psi = 0) = \begin{pmatrix} \frac{\sqrt{G_{py}{}^2 + G_{pz}{}^2}}{|\boldsymbol{G}_p|} & 0 & \frac{G_{px}}{|\boldsymbol{G}_p|} \\ \frac{-G_{px}G_{py}}{|\boldsymbol{G}_p|\sqrt{G_{py}{}^2 + G_{pz}{}^2}} & \frac{G_{pz}}{\sqrt{G_{py}{}^2 + G_{pz}{}^2}} & \frac{G_{py}}{|\boldsymbol{G}_p|} \\ \frac{-G_{px}G_{pz}}{|\boldsymbol{G}_p|\sqrt{G_{py}{}^2 + G_{pz}{}^2}} & \frac{-G_{py}}{\sqrt{G_{py}{}^2 + G_{pz}{}^2}} & \frac{G_{pz}}{|\boldsymbol{G}_p|} \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{G_{py}{}^2 + G_{pz}{}^2}}{|\boldsymbol{G}_p|} & 0 & \frac{G_{px}}{|\boldsymbol{G}_p|} \\ \frac{-R_{xz}R_{yz}|\boldsymbol{G}_p|}{\sqrt{G_{py}{}^2 + G_{pz}{}^2}} & \frac{R_{zz}|\boldsymbol{G}_p|}{\sqrt{G_{py}{}^2 + G_{pz}{}^2}} & \frac{G_{py}}{|\boldsymbol{G}_p|} \\ \frac{-R_{xz}R_{zz}|\boldsymbol{G}_p|}{\sqrt{G_{py}{}^2 + G_{pz}{}^2}} & \frac{-R_{yz}|\boldsymbol{G}_p|}{\sqrt{G_{py}{}^2 + G_{pz}{}^2}} & \frac{G_{pz}}{|\boldsymbol{G}_p|} \end{pmatrix}$$

Eq 3.2.7

In the case of gimbal lock after ±90 degree pitch rotation when $G_{py} = G_{pz} = 0$, the roll angle ϕ is indeterminate, since it is aligned with the axis of the yaw angle ψ, and ϕ can be set to zero:

$$\boldsymbol{R}_{NED}(\phi = \psi = 0) = \begin{pmatrix} 0 & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & \frac{G_{px}}{|\boldsymbol{G}_p|} \\ 0 & 1 & 0 \\ \frac{-G_{px}}{|\boldsymbol{G}_p|} & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & sign(G_{px}) \\ 0 & 1 & 0 \\ -sign(G_{px}) & 0 & 0 \end{pmatrix}$$

Eq 3.2.8

3.3 Android

This section documents the mathematics underlying the function f3DOFTiltAndroid in file orientation.c.

The Android rotation matrix for zero yaw angle ψ after rotation from the horizontal by roll angle ϕ followed by pitch angle θ is:

$$\boldsymbol{R}_{Android}\ (\psi = 0) = \begin{pmatrix} \cos\phi & 0 & sin\phi \\ sin\phi sin\theta & \cos\theta & -\cos\phi\, sin\theta \\ -\cos\theta\, sin\phi & sin\theta & \cos\phi\cos\theta \end{pmatrix}$$ Eq 3.3.1

The Android accelerometer reading at this orientation is:

$$\begin{pmatrix} G_{px} \\ G_{py} \\ G_{pz} \end{pmatrix} = \left|\boldsymbol{G}_p\right| \begin{pmatrix} sin\phi \\ -cos\phi \sin\theta \\ cos\phi \cos\theta \end{pmatrix} = \left|\boldsymbol{G}_p\right| \begin{pmatrix} sin\phi \\ -sin\,\theta \sqrt{1-\left(\frac{G_{px}}{\left|\boldsymbol{G}_p\right|}\right)^2} \\ cos\,\theta \sqrt{1-\left(\frac{G_{px}}{\left|\boldsymbol{G}_p\right|}\right)^2} \end{pmatrix}$$

Eq 3.3.2

$$sin\phi = \frac{G_{px}}{\left|\boldsymbol{G}_p\right|}$$

Eq 3.3.3

$$cos\phi = \sqrt{1-\left(\frac{G_{px}}{\left|\boldsymbol{G}_p\right|}\right)^2}$$

Eq 3.3.4

$$sin\,\theta = \frac{-G_{py}}{\left|\boldsymbol{G}_p\right| \sqrt{1-\left(\frac{G_{px}}{\left|\boldsymbol{G}_p\right|}\right)^2}}$$

Eq 3.3.5

$$cos\,\theta = \frac{G_{pz}}{\left|\boldsymbol{G}_p\right| \sqrt{1-\left(\frac{G_{px}}{\left|\boldsymbol{G}_p\right|}\right)^2}}$$

Eq 3.3.6

The positive square root for $cos\phi$ can always be taken in equation 3.3.4 since the roll angle ϕ in the Android coordinate system varies between -90^o and +90^o giving a non-negative value of $cos\phi$.

Substituting into equation 3.3.1 gives the 3DOF Android orientation matrix directly in terms of the accelerometer reading as:

$$\boldsymbol{R}_{Android}\ (\psi = 0) = \frac{1}{\left|\boldsymbol{G}_p\right|} \begin{pmatrix} \sqrt{{G_{py}}^2 + {G_{pz}}^2} & 0 & G_{px} \\ \frac{-G_{px} G_{py}}{\sqrt{{G_{py}}^2 + {G_{pz}}^2}} & \frac{G_{pz}\left|\boldsymbol{G}_p\right|}{\sqrt{{G_{py}}^2 + {G_{pz}}^2}} & G_{py} \\ \frac{-G_{px} G_{pz}}{\sqrt{{G_{py}}^2 + {G_{pz}}^2}} & \frac{-G_{py}\left|\boldsymbol{G}_p\right|}{\sqrt{{G_{py}}^2 + {G_{pz}}^2}} & G_{pz} \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{{G_{py}}^2 + {G_{pz}}^2}}{\left|\boldsymbol{G}_p\right|} & 0 & \frac{G_{px}}{\left|\boldsymbol{G}_p\right|} \\ \frac{-R_{xz} R_{yz}\left|\boldsymbol{G}_p\right|}{\sqrt{{G_{py}}^2 + {G_{pz}}^2}} & \frac{R_{zz}\left|\boldsymbol{G}_p\right|}{\sqrt{{G_{py}}^2 + {G_{pz}}^2}} & \frac{G_{py}}{\left|\boldsymbol{G}_p\right|} \\ \frac{-R_{xz} R_{zz}\left|\boldsymbol{G}_p\right|}{\sqrt{{G_{py}}^2 + {G_{pz}}^2}} & \frac{-R_{yz}\left|\boldsymbol{G}_p\right|}{\sqrt{{G_{py}}^2 + {G_{pz}}^2}} & \frac{G_{pz}}{\left|\boldsymbol{G}_p\right|} \end{pmatrix}$$

Eq 3.3.7

In the case of gimbal lock after ±90 deg roll rotation when $G_{py} = G_{pz} = 0$, the pitch angle θ is undefined and can be set to zero:

$$\boldsymbol{R}_{Android}\ (\theta = \psi = 0) = \begin{pmatrix} 0 & 0 & sin\phi \\ 0 & 1 & 0 \\ -sin\phi & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & \frac{G_{px}}{|\boldsymbol{G}_p|} \\ 0 & 1 & 0 \\ \frac{-G_{px}}{|\boldsymbol{G}_p|} & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & sign(G_{px}) \\ 0 & 1 & 0 \\ -sign(G_{px}) & 0 & 0 \end{pmatrix}$$

Eq 3.3.8

These equations are coincidentally identical to the equations for the Aerospace / NED coordinate system.

3.4 Windows 8

This section documents the mathematics underlying the function f3DOFTiltWin8 in file orientation.c.

The Windows 8 rotation matrix for zero yaw angle ψ after rotation from the horizontal by pitch angle θ followed by roll angle ϕ is:

$$\boldsymbol{R}_{Win8}(\psi = 0) = \begin{pmatrix} cos\,\phi & sin\phi sin\theta & -sin\phi\,cos\,\theta \\ 0 & cos\,\theta & sin\theta \\ sin\phi & -cos\,\phi\,sin\theta & cos\,\phi\,cos\,\theta \end{pmatrix}$$ Eq 3.4.1

The Windows 8 accelerometer reading at this orientation is:

$$\begin{pmatrix} G_{px} \\ G_{py} \\ G_{pz} \end{pmatrix} = |\boldsymbol{G}_p| \begin{pmatrix} cos\,\theta\,sin\phi \\ -sin\,\theta \\ -cos\,\theta\,cos\phi \end{pmatrix}$$ Eq 3.4.2

$$tan\phi = \frac{-G_{px}}{G_{pz}} \Rightarrow cos\phi = \frac{1}{\sqrt{1 + \left(\frac{G_{px}}{G_{pz}}\right)^2}}$$

Eq 3.4.3

$$\Rightarrow sin\phi = tan\phi cos\phi = \frac{-G_{px}}{G_{pz}\sqrt{1 + \left(\frac{G_{px}}{G_{pz}}\right)^2}}$$

Eq 3.4.4

$$sin\,\theta = \frac{-G_{py}}{|\boldsymbol{G}_p|}$$

Eq 3.4.5

$$cos\,\theta = \frac{-G_{pz}}{|\boldsymbol{G}_p| cos\phi} = \frac{-G_{pz}}{|\boldsymbol{G}_p|}\sqrt{1 + \left(\frac{G_{px}}{G_{pz}}\right)^2}$$

Eq 3.4.6

The positive square root for $cos\phi$ can always be taken in equation 3.4.3 since the roll angle ϕ in the Android coordinate system varies between -90^o and +90^o giving a non-negative value of $cos\phi$.

Substituting into equation 3.4.1 gives the 3DOF orientation matrix directly in terms of the accelerometer reading as:

$$\boldsymbol{R}_{Win8}(\psi = 0) = \frac{1}{|\boldsymbol{G}_p|}\begin{pmatrix} \dfrac{|\boldsymbol{G}_p|}{\sqrt{1+\left(\frac{G_{px}}{G_{pz}}\right)^2}} & \dfrac{G_{px}G_{py}}{G_{pz}\sqrt{1+\left(\frac{G_{px}}{G_{pz}}\right)^2}} & -G_{px} \\ 0 & -G_{pz}\sqrt{1+\left(\frac{G_{px}}{G_{pz}}\right)^2} & -G_{py} \\ \dfrac{-G_{px}|\boldsymbol{G}_p|}{G_{pz}\sqrt{1+\left(\frac{G_{px}}{G_{pz}}\right)^2}} & \dfrac{G_{py}}{\sqrt{1+\left(\frac{G_{px}}{G_{pz}}\right)^2}} & -G_{pz} \end{pmatrix}$$

Eq 3.4.7

$$= \begin{pmatrix} \dfrac{G_{pz}}{sign(G_{pz})\sqrt{G_{px}{}^2+G_{pz}{}^2}} & \dfrac{-G_{px}R_{yz}}{sign(G_{pz})\sqrt{G_{px}{}^2+G_{pz}{}^2}} & \dfrac{-G_{px}}{|\boldsymbol{G}_p|} \\ 0 & \dfrac{-sign(G_{pz})\sqrt{G_{px}{}^2+G_{pz}{}^2}}{|\boldsymbol{G}_p|} & \dfrac{-G_{py}}{|\boldsymbol{G}_p|} \\ \dfrac{-G_{px}}{sign(G_{pz})\sqrt{G_{px}{}^2+G_{pz}{}^2}} & \dfrac{-G_{py}R_{zz}}{sign(G_{pz})\sqrt{G_{px}{}^2+G_{pz}{}^2}} & \dfrac{-G_{pz}}{|\boldsymbol{G}_p|} \end{pmatrix}$$

Eq 3.4.8

$$= \begin{pmatrix} \dfrac{-R_{zz}|\boldsymbol{G}_p|}{sign(G_{pz})\sqrt{G_{px}{}^2+G_{pz}{}^2}} & \dfrac{R_{xz}R_{yz}|\boldsymbol{G}_p|}{sign(G_{pz})\sqrt{G_{px}{}^2+G_{pz}{}^2}} & \dfrac{-G_{px}}{|\boldsymbol{G}_p|} \\ 0 & \dfrac{-1}{\left\{\dfrac{|\boldsymbol{G}_p|}{sign(G_{pz})\sqrt{G_{px}{}^2+G_{pz}{}^2}}\right\}} & \dfrac{-G_{py}}{|\boldsymbol{G}_p|} \\ \dfrac{R_{xz}|\boldsymbol{G}_p|}{sign(G_{pz})\sqrt{G_{px}{}^2+G_{pz}{}^2}} & \dfrac{R_{yz}R_{zz}|\boldsymbol{G}_p|}{sign(G_{pz})\sqrt{G_{px}{}^2+G_{pz}{}^2}} & \dfrac{-G_{pz}}{|\boldsymbol{G}_p|} \end{pmatrix}$$

Eq 3.4.9

In the case of gimbal lock after ±90 degree pitch rotation when $G_{px} = G_{pz} = 0$, the roll angle ϕ is undefined and can be set to zero:

$$\boldsymbol{R}_{Win8}(\phi = \psi = 0) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & sin\theta \\ 0 & -sin\theta & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & \dfrac{-G_{py}}{|\boldsymbol{G}_p|} \\ 0 & \dfrac{G_{py}}{|\boldsymbol{G}_p|} & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -sign(G_{py}) \\ 0 & sign(G_{py}) & 0 \end{pmatrix}$$

Eq 3.4.10

4 Magnetometer Compass Orientation Matrix

4.1 Introduction

This section documents the 2D magnetometer-only eCompass functions which calculate the orientation matrix from calibrated magnetometer measurements on with the assumption that the magnetometer is always flat and has zero tilt away from the horizontal. The market application is an automotive compass where the normal accelerometer plus magnetometer tilt-compensated eCompass functions cannot be used because of vehicle acceleration, braking and cornering forces.

Use these functions with caution. In the automotive application, the 2D eCompass will be robust against the effects of acceleration, braking and cornering forces (since the magnetometer sensor is unaffected by acceleration) but the compass heading will change when the vehicle tilt angle (whether roll or pitch) deviates from horizontal violating the assumption that the magnetometer sensor is always flat.

4.2 Aerospace / NED

This section documents the mathematics underlying the function f3DOFMagnetometerMatrixNED in file orientation.c.

The Aerospace / NED rotation matrix for rotation in yaw angle ψ only while remaining horizontal is:

$$\boldsymbol{R}_{NED}(\psi) = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{Eq 4.2.1}$$

The Aerospace / NED calibrated magnetometer reading at this orientation is:

$$\begin{pmatrix} B_{cx} \\ B_{cy} \\ B_{cz} \end{pmatrix} = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} B \begin{pmatrix} \cos\delta \\ 0 \\ \sin\delta \end{pmatrix} = B \begin{pmatrix} \cos\psi\cos\delta \\ -\sin\psi\cos\delta \\ \sin\delta \end{pmatrix} \quad \text{Eq 4.2.2}$$

The horizontal component of the geomagnetic field has value:

$$B\cos\delta = \sqrt{{B_{cx}}^2 + {B_{cy}}^2} \quad \text{Eq 4.2.3}$$

The yaw ψ (and compass heading angle) is given by:

$$\tan\psi = \frac{\sin\psi}{\cos\psi} = \frac{-B_{cy}}{B_{cx}} \quad \text{Eq 4.2.4}$$

$$\sin\psi = \frac{-B_{cy}}{\sqrt{{B_{cx}}^2 + {B_{cy}}^2}} \quad \text{Eq 4.2.5}$$

$$cos\psi = \frac{B_{cx}}{\sqrt{{B_{cx}}^2 + {B_{cy}}^2}}$$

Eq 4.2.6

The orientation matrix is independent of the unknown magnetic inclination angle δ and has value:

$$\boldsymbol{R}_{NED}(\psi) = \frac{1}{\sqrt{{B_{cx}}^2 + {B_{cy}}^2}} \begin{pmatrix} B_{cx} & -B_{cy} & 0 \\ B_{cy} & B_{cx} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Eq 4.2.7

4.3 Android

This section documents the mathematics underlying the function f3DOFMagnetometerMatrixAndroid in file orientation.c.

The Android rotation matrix for rotation in yaw angle ψ only while remaining horizontal is:

$$\boldsymbol{R}_{Android}(\psi) = \begin{pmatrix} cos\,\psi & -sin\psi & 0 \\ sin\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$ Eq 4.3.1

The Android calibrated magnetometer reading at this orientation is:

$$\begin{pmatrix} B_{cx} \\ B_{cy} \\ B_{cz} \end{pmatrix} = \begin{pmatrix} cos\,\psi & -sin\psi & 0 \\ sin\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} B \begin{pmatrix} 0 \\ cos\delta \\ -sin\delta \end{pmatrix} = B \begin{pmatrix} -sin\psi cos\delta \\ cos\psi cos\delta \\ -sin\delta \end{pmatrix}$$ Eq 4.3.2

The horizontal component of the geomagnetic field has value:

$$Bcos\delta = \sqrt{{B_{cx}}^2 + {B_{cy}}^2}$$

Eq 4.3.3

The yaw ψ (and compass heading) angle is given by:

$$tan\psi = \frac{sin\psi}{cos\psi} = \frac{-B_{cx}}{B_{cy}}$$

Eq 4.3.4

$$sin\psi = \frac{-B_{cx}}{\sqrt{{B_{cx}}^2 + {B_{cy}}^2}}$$

Eq 4.3.5

$$cos\psi = \frac{B_{cy}}{\sqrt{{B_{cx}}^2 + {B_{cy}}^2}}$$

Eq 4.3.6

The orientation matrix is independent of the unknown magnetic inclination angle δ and has value:

$$\boldsymbol{R}_{Android}\ (\psi) = \frac{1}{\sqrt{{B_{cx}}^2 + {B_{cy}}^2}} \begin{pmatrix} B_{cy} & B_{cx} & 0 \\ -B_{cx} & B_{cy} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Eq 4.3.7

4.4 Windows 8

This section documents the mathematics underlying the function f3DOFMagnetometerMatrixWin8 in file orientation.c.

The Windows 8 rotation matrix for rotation in yaw angle ψ only while remaining horizontal is:

$$\boldsymbol{R}_{Win8}(\psi) = \begin{pmatrix} cos\,\psi & sin\psi & 0 \\ -\,sin\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$ Eq 4.4.1

The Windows 8 calibrated magnetometer reading at this orientation is:

$$\begin{pmatrix} B_{cx} \\ B_{cy} \\ B_{cz} \end{pmatrix} = \begin{pmatrix} cos\,\psi & sin\psi & 0 \\ -\,sin\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} B \begin{pmatrix} 0 \\ cos\delta \\ -sin\delta \end{pmatrix} = B \begin{pmatrix} sin\psi cos\delta \\ cos\psi cos\delta \\ -sin\delta \end{pmatrix}$$ Eq 4.4.2

The horizontal component of the geomagnetic field has value:

$$Bcos\delta = \sqrt{{B_{cx}}^2 + {B_{cy}}^2}$$

Eq 4.4.3

The yaw ψ (equal to minus the compass heading in the Windows 8 coordinate system) angle is given by:

$$tan\psi = \frac{sin\psi}{cos\psi} = \frac{B_{cx}}{B_{cy}}$$

Eq 4.4.4

$$sin\psi = \frac{B_{cx}}{\sqrt{{B_{cx}}^2 + {B_{cy}}^2}}$$

Eq 4.4.5

$$cos\psi = \frac{B_{cy}}{\sqrt{{B_{cx}}^2 + {B_{cy}}^2}}$$

Eq 4.4.6

The orientation matrix is independent of the unknown magnetic inclination angle δ and has value:

$$\boldsymbol{R}_{Win8}(\psi) = \frac{1}{\sqrt{{B_{cx}}^2 + {B_{cy}}^2}} \begin{pmatrix} B_{cy} & B_{cx} & 0 \\ -B_{cx} & B_{cy} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Eq 4.4.7

5 Accelerometer Magnetometer Tilt eCompass Orientation Matrix

5.1 Introduction

This section documents the calculation of the orientation matrix from accelerometer and magnetometer measurements. This is the classic tilt-compensated eCompass which provides a complete determination of orientation with the assumption that there is no linear acceleration so that the accelerometer reading is a function of orientation in the earth's gravitational field only.

The orientation matrix can be calculated with an elegant algorithm which is most easily understood from the trivial mathematical identity of equation 5.1.1.

$$\begin{pmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{pmatrix} = \begin{pmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{Eq 5.1.1}$$

The three columns of the identity matrix are the three unit vectors in the *x*, *y* and *z* directions aligned with the initial orientation in the global reference frame. The accelerometer measures the gravity vector rotated into its coordinate system and therefore provides the right hand column of the orientation matrix. The geomagnetic vector in the global frame points northwards and downwards (upwards) in the northern (southern) hemisphere. The vector product between the rotated gravity and rotated geomagnetic vector is therefore the rotated easterly pointing vector from the global frame which forms the *y* (Aerospace / NED coordinate system) or *x* (Android and Windows 8 coordinate systems) column vector in the orientation matrix. Since the orientation matrix is orthonormal, a final vector product between the two vectors determined so far gives the third column vector.

5.2 Aerospace / NED

This section documents the mathematics underlying the function feCompassNED in file orientation.c.

The *z* axis in the initial NED orientation is aligned parallel to the *z* axis accelerometer reading for the NED The right hand column vector of the rotation matrix is then given by:

$$\begin{pmatrix} R_{xz} \\ R_{yz} \\ R_{zz} \end{pmatrix} = \frac{1}{\sqrt{{G_{px}}^2 + {G_{py}}^2 + {G_{pz}}^2}} \begin{pmatrix} G_{px} \\ G_{py} \\ G_{pz} \end{pmatrix}$$

Eq 5.2.1

The geomagnetic vector, or calibrated magnetometer reading, in the default orientation points northwards (in the *x* direction for the NED coordinate system) and downwards (upwards) in the northern (southern) hemisphere. The vector product of the NED *z* and *x* vectors gives a vector orthogonal to both and therefore aligned along the *y* axis:

$$\begin{pmatrix} R_{xy} \\ R_{yy} \\ R_{zy} \end{pmatrix} = \begin{pmatrix} R_{xz} \\ R_{yz} \\ R_{zz} \end{pmatrix} \times \left\{ \frac{1}{\sqrt{{B_{cx}}^2 + {B_{cy}}^2 + {B_{cz}}^2}} \begin{pmatrix} B_{cx} \\ B_{cy} \\ B_{cz} \end{pmatrix} \right\}$$

Eq 5.2.2

Finally, the vector product of the *y* and *z* axes gives the remaining *x* column of the orientation matrix:

$$\begin{pmatrix} R_{xx} \\ R_{yx} \\ R_{zx} \end{pmatrix} = \begin{pmatrix} R_{xy} \\ R_{yy} \\ R_{zy} \end{pmatrix} \times \begin{pmatrix} R_{xz} \\ R_{yz} \\ R_{zz} \end{pmatrix} \quad \text{Eq 5.2.3}$$

The geomagnetic inclination angle δ is the angle by which the geomagnetic field dips below horizontal. Since the scalar product of two vectors is invariant under rotation, the inclination angle is related to the scalar product of the accelerometer and calibrated magnetic measurements by:

$$\boldsymbol{G}_p \cdot \boldsymbol{B}_c = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot B \begin{pmatrix} cos\delta \\ 0 \\ sin\delta \end{pmatrix} = Bsin\delta \Rightarrow sin\delta = \frac{G_{px} B_{cx} + G_{py} B_{cy} + G_{pz} B_{cz}}{|\boldsymbol{G}_p||\boldsymbol{B}_c|}$$

Eq 5.2.4

5.3 Android

This section documents the mathematics underlying the function feCompassAndroid in file orientation.c.

The accelerometer reading in the Android coordinate system in the initial orientation is also parallel to the *z* axis since the Android coordinate system is ENU with the z axis upwards and acceleration (not gravity) positive:

$$\begin{pmatrix} R_{xz} \\ R_{yz} \\ R_{zz} \end{pmatrix} = \frac{1}{\sqrt{{G_{px}}^2 + {G_{py}}^2 + {G_{pz}}^2}} \begin{pmatrix} G_{px} \\ G_{py} \\ G_{pz} \end{pmatrix}$$

Eq 5.3.1

The geomagnetic vector, or calibrated magnetometer reading, in the default orientation points northwards (in the *y* direction for the Android coordinate system) and downwards (upwards) in the northern (southern) hemisphere. The vector product of the Android *y* and *z* vectors gives a vector orthogonal to both aligned along the *x* axis:

$$\begin{pmatrix} R_{xx} \\ R_{yx} \\ R_{zx} \end{pmatrix} = \left\{ \frac{1}{\sqrt{{B_{cx}}^2 + {B_{cy}}^2 + {B_{cz}}^2}} \begin{pmatrix} B_{cx} \\ B_{cy} \\ B_{cz} \end{pmatrix} \right\} \times \begin{pmatrix} R_{xz} \\ R_{yz} \\ R_{zz} \end{pmatrix}$$

Eq 5.3.2

Finally, the vector product of the *z* and *x* axes gives the remaining *y* component of the orientation matrix:

$$\begin{pmatrix} R_{xy} \\ R_{yy} \\ R_{zy} \end{pmatrix} = \begin{pmatrix} R_{xz} \\ R_{yz} \\ R_{zz} \end{pmatrix} \times \begin{pmatrix} R_{xx} \\ R_{yx} \\ R_{zx} \end{pmatrix}$$

Eq 5.3.3

In the Android coordinate system, the inclination angle is related to the scalar product of the accelerometer and calibrated magnetic measurements by:

$$\boldsymbol{G}_p \cdot \boldsymbol{B}_c = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot B \begin{pmatrix} 0 \\ cos\delta \\ -sin\delta \end{pmatrix} = -Bsin\delta \Rightarrow sin\delta = \frac{-\left(G_{px} B_{cx} + G_{py} B_{cy} + G_{pz} B_{cz}\right)}{|\boldsymbol{G}_p||\boldsymbol{B}_c|}$$

Eq 5.3.4

5.4 Windows 8

This section documents the mathematics underlying the function feCompassWin8 in file orientation.c.

The accelerometer reading in the Windows 8 coordinate system in the initial orientation is anti-parallel to the *z* axis since the Windows 8 coordinate system is ENU with the z axis upwards and gravity (not acceleration) positive:

$$\begin{pmatrix} R_{xz} \\ R_{yz} \\ R_{zz} \end{pmatrix} = \frac{-1}{\sqrt{{G_{px}}^2 + {G_{py}}^2 + {G_{pz}}^2}} \begin{pmatrix} G_{px} \\ G_{py} \\ G_{pz} \end{pmatrix}$$

Eq 5.4.1

The geomagnetic vector, or calibrated magnetometer reading, in the default orientation points northwards (in the *y* direction for the Windows 8 coordinate system) and downwards (upwards) in the northern (southern) hemisphere. The vector product of the Windows 8 *y* and *z* vectors gives a vector orthogonal to both aligned along the *x* axis:

$$\begin{pmatrix} R_{xx} \\ R_{yx} \\ R_{zx} \end{pmatrix} = \left\{ \frac{1}{\sqrt{{B_{cx}}^2 + {B_{cy}}^2 + {B_{cz}}^2}} \begin{pmatrix} B_{cx} \\ B_{cy} \\ B_{cz} \end{pmatrix} \right\} \times \begin{pmatrix} R_{xz} \\ R_{yz} \\ R_{zz} \end{pmatrix}$$

Eq 5.4.2

Finally, the vector product of the *z* and *x* axes gives the remaining *y* component of the orientation matrix:

$$\begin{pmatrix} R_{xy} \\ R_{yy} \\ R_{zy} \end{pmatrix} = \begin{pmatrix} R_{xz} \\ R_{yz} \\ R_{zz} \end{pmatrix} \times \begin{pmatrix} R_{xx} \\ R_{yx} \\ R_{zx} \end{pmatrix}$$

Eq 5.4.3

In the Windows 8 coordinate system, the inclination angle is related to the scalar product of the accelerometer and calibrated magnetic measurements by:

$$\boldsymbol{G}_p \cdot \boldsymbol{B}_c = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \cdot B \begin{pmatrix} 0 \\ cos\delta \\ -sin\delta \end{pmatrix} = Bsin\delta \Rightarrow sin\delta = \frac{G_{px} B_{cx} + G_{py} B_{cy} + G_{pz} B_{cz}}{|\boldsymbol{G}_p||\boldsymbol{B}_c|}$$

Eq 5.4.4

6 Converting Between Rotation Matrix and Rotation Quaternion

6.1 Introduction

The orientation is most efficiently computed from sensor measurements in the form of the rotation matrix. Later algorithms may be more efficiently performed in either quaternion algebra or in matrix algebra. It is therefore necessary to efficiently convert between rotation matrices and rotation quaternions and vice versa.

The derivations start from the equivalence between performing a coordinate system rotation using the rotation matrix $\boldsymbol{R}$ and the rotation quaternion q. For any vector $\boldsymbol{v}$:

$$\boldsymbol{R}\boldsymbol{v} = q^*\boldsymbol{v}q \qquad \text{Eq 6.1.1}$$

Expanding equation 6.1.1 into its Cartesian components gives the identity:

$$\begin{pmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{pmatrix}\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = (q_0 - q_1\boldsymbol{i} - q_2\boldsymbol{j} - q_3\boldsymbol{k})\left(v_x\boldsymbol{i} + v_y\boldsymbol{j} + v_z\boldsymbol{k}\right)(q_0 + q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k}) \qquad \text{Eq 6.1.2}$$

6.2 Rotation Matrix from Quaternion

Expanding the right hand side of equation 6.1.2 and writing in matrix form gives:

$$\begin{pmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{pmatrix}\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} {q_0}^2 + {q_1}^2 - {q_2}^2 - {q_3}^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & {q_0}^2 - {q_1}^2 + {q_2}^2 - {q_3}^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & {q_0}^2 - {q_1}^2 - {q_2}^2 + {q_3}^2 \end{pmatrix}\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}$$

Eq 6.2.1

Since equation 6.2.1 holds for all vectors $\boldsymbol{v}$, it follows that:

$$\begin{pmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{pmatrix} = \begin{pmatrix} {q_0}^2 + {q_1}^2 - {q_2}^2 - {q_3}^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & {q_0}^2 - {q_1}^2 + {q_2}^2 - {q_3}^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & {q_0}^2 - {q_1}^2 - {q_2}^2 + {q_3}^2 \end{pmatrix} \qquad \text{Eq 6.2.2}$$

The normalization constraint for a rotation quaternion is:

$${q_0}^2 + {q_1}^2 + {q_2}^2 + {q_3}^2 = 1 \qquad \text{Eq 6.2.3}$$

Substituting equation 6.2.3 into equation 6.2.2 gives the rotation matrix directly in terms of the components of the rotation quaternion. If the rotation quaternion is correctly normalized then the rotation matrix will be orthonormal.

$$\begin{pmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{pmatrix} = \begin{pmatrix} 2({q_0}^2 + {q_1}^2) - 1 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & 2({q_0}^2 + {q_2}^2) - 1 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & 2({q_0}^2 + {q_3}^2) - 1 \end{pmatrix} \qquad \text{Eq 6.2.4}$$

6.3 Quaternion from Rotation Matrix

Equation 6.2.4 can also be used to determine the rotation quaternion from the rotation matrix. The procedure is straightforward except for rotations close to 180^o where a fallback algorithm is needed to avoid numerical rounding errors.

The sum of 1 plus the trace of the rotation matrix evaluates to:

$$1 + tr(\boldsymbol{R}) = 1 + 2({q_0}^2 + {q_1}^2) - 1 + 2({q_0}^2 + {q_2}^2) - 1 + 2({q_0}^2 + {q_3}^2) - 1 \qquad \text{Eq 6.3.1}$$

Applying the normalization constraint of equation 6.2.3 gives:

$$1 + tr(\boldsymbol{R}) = 4{q_0}^2 \qquad \text{Eq 6.3.2}$$

q_0 is always non-negative since negative q_0 in a rotation quaternion corresponds to a rotation angle greater than 180^o which is equivalent to a negated rotation of less than 180^o about the negated rotation axis.

The positive square root of equation 6.3.2 can therefore always be taken giving:

$$q_0 = \frac{\sqrt{1 + tr(\boldsymbol{R})}}{2} = \frac{\sqrt{1 + R_{xx} + R_{yy} + R_{zz}}}{2}$$

Eq 6.3.3

Differencing elements across the diagonal gives the solution for all remaining elements of the rotation quaternion valid except when $q_0 = 0$ at a 180^o rotation:

$$R_{yz} - R_{zy} = 4q_0q_1 \Rightarrow q_1 = \frac{(R_{yz} - R_{zy})}{4q_0}$$

Eq 6.3.4

$$R_{zx} - R_{xz} = 4q_0q_2 \Rightarrow q_2 = \frac{(R_{zx} - R_{xz})}{4q_0}$$

Eq 6.3.5

$$R_{xy} - R_{yx} = 4q_0q_3 \Rightarrow q_3 = \frac{(R_{xy} - R_{yx})}{4q_0}$$

Eq 6.3.6

Equations 6.3.4 to 6.3.5 fail near 180^o rotation about any axis since the rotation matrix becomes symmetric (giving near zero numerator) and the scalar quaternion component q_0 approaches zero (giving a near zero denominator). The fallback algorithm in this case uses the elements on the leading diagonal to give:

$$R_{xx} = 2({q_0}^2 + {q_1}^2) - 1 \Rightarrow q_1 = \pm\sqrt{\frac{1 + R_{xx}}{2} - {q_0}^2}$$

Eq 6.3.7

$$R_{yy} = 2({q_0}^2 + {q_2}^2) - 1 \Rightarrow q_2 = \pm\sqrt{\frac{1 + R_{yy}}{2} - {q_0}^2}$$

Eq 6.3.8

$$R_{zz} = 2({q_0}^2 + {q_3}^2) - 1 \Rightarrow q_3 = \pm\sqrt{\frac{1 + R_{zz}}{2} - {q_0}^2}$$

Eq 6.3.9

The unknown signs in equations 6.3.7 to 6.3.9 can be resolved by taking the signs of equations 6.3.4 to 6.3.6 and using the fact that q_0 is always non-negative:

$sign(q_1) = sign(R_{yz} - R_{zy})$ Eq 6.3.10

$sign(q_2) = sign(R_{zx} - R_{xz})$ Eq 6.3.11

$sign(q_3) = sign(R_{xy} - R_{yx})$ Eq 6.3.12

7 Converting Between Rotation Matrix and Rotation Vector

7.1 Introduction

Any rotation can be represented by its normalized rotation axis $\hat{\boldsymbol{n}}$ and the rotation angle η about that axis giving a total of three degrees of freedom. The rotation vector $\eta\hat{\boldsymbol{n}}$ is the most efficient encoding of a rotation but has no useful mathematical properties. The rotation quaternion and rotation matrix are therefore used in preference.

The output of the gyroscope sensor (in deg/sec) multiplied by the sampling interval is, however, a rotation vector and is normally immediately converted to a rotation matrix or quaternion.

With the definition of $\hat{\boldsymbol{n}}$ as the normalized rotation axis η as the rotation angle, the rotation vector is defined as:

$$\eta\hat{\boldsymbol{n}} = \eta\begin{pmatrix}\hat{n}_x\\ \hat{n}_y\\ \hat{n}_z\end{pmatrix} \qquad \text{Eq 7.1.1}$$

7.2 Rotation Matrix from Rotation Vector

The Rodrigues rotation matrix defined in equation 2.1.1 defines a rotation matrix directly in terms of the rotation vector which rotates a coordinate system by angle η about axis $\hat{\boldsymbol{n}}$:

$$\boldsymbol{R} = \begin{pmatrix}R_{xx} & R_{xy} & R_{xz}\\ R_{yx} & R_{yy} & R_{yz}\\ R_{zx} & R_{zy} & R_{zz}\end{pmatrix} = \begin{pmatrix}\hat{n}_x{}^2 + (1-\hat{n}_x{}^2)cos\eta & \hat{n}_x\hat{n}_y(1-cos\eta) + \hat{n}_z sin\eta & \hat{n}_x\hat{n}_z(1-cos\eta) - \hat{n}_y sin\eta\\ \hat{n}_x\hat{n}_y(1-cos\eta) - \hat{n}_z sin\eta & \hat{n}_y{}^2 + (1-\hat{n}_y{}^2)cos\eta & \hat{n}_y\hat{n}_z(1-cos\eta) + \hat{n}_x sin\eta\\ \hat{n}_x\hat{n}_z(1-cos\eta) + \hat{n}_y sin\eta & \hat{n}_y\hat{n}_z(1-cos\eta) - \hat{n}_x sin\eta & \hat{n}_z{}^2 + (1-\hat{n}_z{}^2)cos\eta\end{pmatrix}$$

Eq 7.2.1

Equation 7.2.1 allows the direct calculation of the rotation matrix from the rotation vector of equation 7.1.1.

7.3 Rotation Vector from Rotation Matrix

The calculation of the rotation vector $\eta\hat{\boldsymbol{n}}$ from the rotation matrix is similar to the calculation of the quaternion from the rotation matrix.

The rotation angle η is computed from the trace of the rotation matrix:

$$tr(\boldsymbol{R}) = R_{xx} + R_{yy} + R_{zz} = \hat{n}_x{}^2 + (1-\hat{n}_x{}^2)cos\eta + \hat{n}_y{}^2 + (1-\hat{n}_y{}^2)cos\eta + \hat{n}_z{}^2 + (1-\hat{n}_z{}^2)cos\eta \qquad \text{Eq 7.3.1}$$

$$= (\hat{n}_x{}^2 + \hat{n}_y{}^2 + \hat{n}_z{}^2) + 3cos\eta - (\hat{n}_x{}^2 + \hat{n}_y{}^2 + \hat{n}_z{}^2)cos\eta = 1 + 2cos\eta \qquad \text{Eq 7.3.2}$$

$$\Rightarrow \eta = cos^{-1}\left(\frac{tr(\boldsymbol{R}) - 1}{2}\right)$$

Eq 7.3.3

There is no angle ambiguity in equation 7.3.3 since the rotation angle η varies between 0^o and 180^o. Rotation angles between -180^o and 0^o are equivalent to rotation by the negated angle about the negated rotation axis. The rotation vector $\eta\hat{\boldsymbol{n}}$ is unaltered by a double sign change.

The trace of the matrix varies between -1 and +3. A trace of -1 corresponds to a rotation of 180^o and a trace of +3 to a rotation of 0^o degrees.

In the general case, the rotation axis $\hat{\boldsymbol{n}}$ and rotation vector $\eta\hat{\boldsymbol{n}}$ are determined by differencing matrix elements across the leading diagonal:

$$\hat{\boldsymbol{n}} = \frac{1}{2sin\eta}\begin{pmatrix}R_{yz} - R_{zy}\\ R_{zx} - R_{xz}\\ R_{xy} - R_{yx}\end{pmatrix}$$

Eq 7.3.4

$$2sin\eta = \sqrt{\left(R_{yz} - R_{zy}\right)^2 + (R_{zx} - R_{xz})^2 + \left(R_{xy} - R_{yx}\right)^2}$$ Eq 7.3.5

The positive square root can be taken in equation 7.3.5 since the sine of a rotation angle between 0^o and 180^o is always non-negative. Substitution of equation 7.3.5 into 7.3.4 gives the rotation vector as:

$$\eta\hat{\boldsymbol{n}} = \frac{\eta}{\sqrt{\left(R_{yz} - R_{zy}\right)^2 + (R_{zx} - R_{xz})^2 + \left(R_{xy} - R_{yx}\right)^2}} \begin{pmatrix} R_{yz} - R_{zy} \\ R_{zx} - R_{xz} \\ R_{xy} - R_{yx} \end{pmatrix}$$

Eq 7.3.6

Equations 7.3.4 to 7.3.6 fail when $sin\eta$ approaches zero which occurs near $\eta = 0^o$ and $\eta = 180^o$ when the rotation matrix becomes symmetric. These two special cases can be detected using i) equation 7.3.5 to determine $sin\eta$ near zero and ii) the matrix trace to distinguish the two cases of 0^o rotation angle (with trace = 3) and 180^o rotation angle (with trace = -1) both of which give $sin\eta = 0$.

For the case of a rotation angle η near zero, $sin\eta \approx \eta$ and the rotation vector becomes:

$$\eta\hat{\boldsymbol{n}} = \frac{1}{2}\begin{pmatrix} R_{yz} - R_{zy} \\ R_{zx} - R_{xz} \\ R_{xy} - R_{yx} \end{pmatrix}$$ Eq 7.3.7

For the case of the rotation angle η near 180^o, the matrix approximates the form below with a trace of -1:

$$\boldsymbol{R} = \begin{pmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{pmatrix} = \begin{pmatrix} 2\hat{n}_x{}^2 - 1 & \hat{n}_x\hat{n}_y(1 - cos\eta) + \hat{n}_z sin\eta & \hat{n}_x\hat{n}_z(1 - cos\eta) - \hat{n}_y sin\eta \\ \hat{n}_x\hat{n}_y(1 - cos\eta) - \hat{n}_z sin\eta & 2\hat{n}_y{}^2 - 1 & \hat{n}_y\hat{n}_z(1 - cos\eta) + \hat{n}_x sin\eta \\ \hat{n}_x\hat{n}_z(1 - cos\eta) + \hat{n}_y sin\eta & \hat{n}_y\hat{n}_z(1 - cos\eta) - \hat{n}_x sin\eta & 2\hat{n}_z{}^2 - 1 \end{pmatrix}$$

Eq 7.3.8

$$\begin{pmatrix} \hat{n}_x \\ \hat{n}_y \\ \hat{n}_z \end{pmatrix} = \begin{pmatrix} \pm\sqrt{\frac{R_{xx} + 1}{2}} \\ \pm\sqrt{\frac{R_{yy} + 1}{2}} \\ \pm\sqrt{\frac{R_{zz} + 1}{2}} \end{pmatrix}$$

Eq 7.3.9

The sign ambiguity can be resolved by differencing across the leading diagonal and using the result that $sin\eta$ is non-negative.

$$R_{yz} - R_{zy} = 2\hat{n}_x sin\eta \Rightarrow sign(\hat{n}_x) = sign\left(R_{yz} - R_{zy}\right)$$ Eq 7.3.10

$$R_{zx} - R_{xz} = 2\hat{n}_y sin\eta \Rightarrow sign\left(\hat{n}_y\right) = sign(R_{zx} - R_{xz})$$ Eq 7.3.11

$$R_{xy} - R_{yx} = 2\hat{n}_z sin\eta \Rightarrow sign(\hat{n}_z) = sign\left(R_{xy} - R_{yx}\right)$$ Eq 7.3.12

7.4 Virtual Gyro From Rotation Vector

In systems with physical gyroscope sensors, the angular velocity is determined primarily from the gyroscope sensor. If a gyroscope sensor is not present, then the angular velocity can be determined from the sequence of orientation matrices (or quaternions) using a technique termed a "virtual gyro".

The incremental rotation matrix $\Delta\boldsymbol{R}[n]$ relating the orientation matrix $\boldsymbol{R}[n]$ at iteration n to that at iteration $n-1$ is given by:

$$\boldsymbol{R}[n] = \Delta\boldsymbol{R}[n]\boldsymbol{R}[n-1] \Rightarrow \Delta\boldsymbol{R}[n] = \boldsymbol{R}[n]\boldsymbol{R}[n-1]^T \qquad \text{Eq 7.4.1}$$

The virtual gyro simply takes uses the algorithm of section 7.3 to compute the incremental rotation vector $(\Delta\eta)\hat{\boldsymbol{n}}$ from $\Delta\boldsymbol{R}[n]$. Multiplying the incremental rotation vector by the sampling rate f_s gives the virtual gyro angular velocity vector $\boldsymbol{\omega}$ as:

$$\boldsymbol{\omega} = f_s(\Delta\eta)\hat{\boldsymbol{n}} \qquad \text{Eq 7.4.2}$$

8 Low Pass Filtering of Inclination Angle

The inclination angle δ is filtered with a single pole low pass filter in function fLPFScalar.

The filter difference equation is:

$$y[n] = (1-\alpha)y[n-1] + \alpha x[n] \qquad \text{Eq 8.1}$$

where:

$$0 < \alpha \leq 1 \qquad \text{Eq 8.2}$$

The transfer function $H(z)$ is:

$$H(z) = \frac{\alpha}{1-(1-\alpha)z^{-1}} \qquad \text{Eq 8.3}$$

with a pole at $z = (1-\alpha)$.

The case $\alpha = 1$ corresponds to the all pass filter:

$$y[n] = x[n] \qquad \text{Eq 8.4}$$

In the general case, an impulse at time zero $x[0] = 1$ gives the exponentially decaying output:

$$y[n] = (1-\alpha)^n \qquad \text{Eq 8.5}$$

The 1/e time constant N is then given by:

$$\frac{1}{e} = (1-\alpha)^N \Rightarrow N = \frac{-1}{ln(1-\alpha)} \qquad \text{Eq 8.6}$$

For small α:

$$N = \frac{1}{\alpha} \qquad \text{Eq 8.7}$$

For $\alpha = 0.125$, equation 8.6 gives the exact time constant as $N = 7.49$ and equation 8.7 gives the approximation $N = 8.00$.