Joystick Project

While the sensorstick is static, there is always a constant gravity vector **g**. Since **g** is always pointing downward, it can be used to estimate the orientation of the sensorstick. By slowly tilting the sensorstick or holding it static at various angles, the orientation of a graphical object on screen (or other hardware) can be shown on the computer. Fast movements and lateral movements (including rotation) on the other hand create accelerations that will be misinterpreted as rotation angles and the usefulness of this joystick is therefore limited to slowly rotational movement.

Extraction of Constant Gravity Vector

The triaxial accelerometer constantly measures the acceleration in the x-, y-, and z-direction which can be combined into the angular orientation of the gravity vector compared to the reference frame of the device as shown in Figure 1. When the sensorstick is static, g_x, g_y, and g_z are measured and $\mathbf{g}=(g_x, g_y, g_z)$ can therefore be reconstructed. There are multiple ways to estimate β and γ and depending on the angle one is numerically more stable than the other.

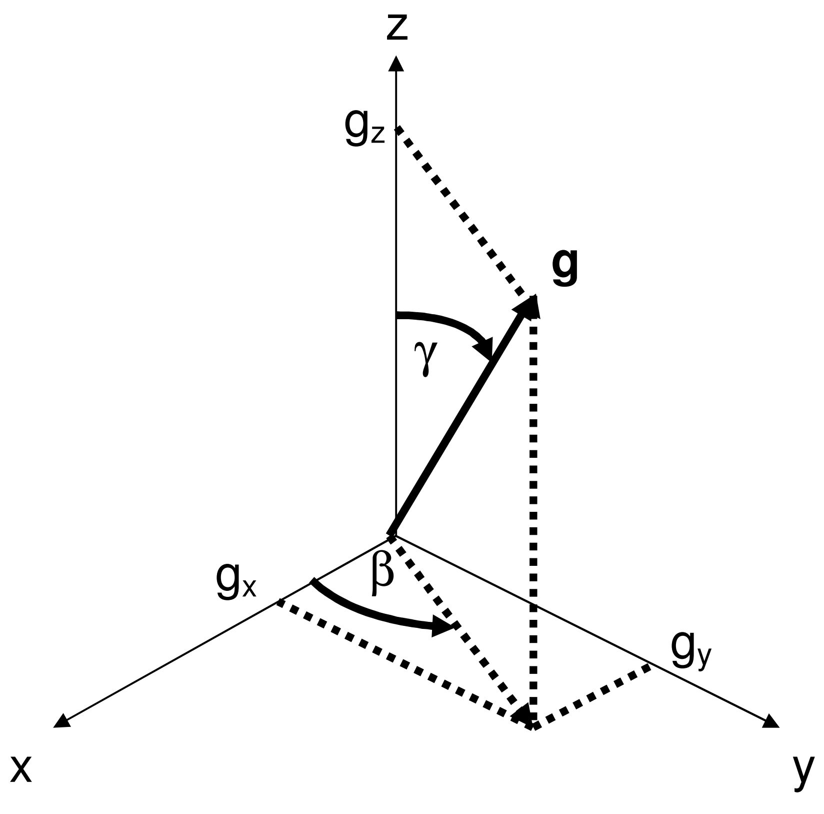

Figure 1. Reference frame of the sensorstick. The sensorstick measures g_x, g_y, and g_z with some angles γ and β. The angle of the sensorstick with respect to the vertically skyward pointing vector g can therefore be determined. Rotating the sensorstick can be used to control the orientation of a target.

The relationship between the angles and the components of **g** are (see Figure 1):
$|\mathbf{g}_{xy}|^2 = g_x^2 + g_y^2$
$\cos(\beta) = g_x/|\mathbf{g}_{xy}|$
$\sin(\beta) = g_y/|\mathbf{g}_{xy}|$
$\tan(\beta) = g_y/g_x$
$|\mathbf{g}|^2 = |\mathbf{g}_{xy}|^2+g_z^2 = g_x^2 + g_y^2 + g_z^2$
$\cos(\gamma) = g_z/|\mathbf{g}|$
$\sin(\gamma) = |\mathbf{g}_{xy}|/|\mathbf{g}|$
$\tan(g) = |\mathbf{g}_{xy}|/g_z$

The angle γ is unique while the range of β can be established from the sign of g_x. The stability of the inverse sine and cosine of these angles depends on the ratios but the angles themselves need not be computed because only the sines and cosines of the angles are needed for the rotation matrices.

In Figure 1 the fixed reference frame is that of the sensorstick. In the real world reference frame g is fixed pointing downward. If $\mathbf{g}=(g_x, g_y, g_z)$ in the sensorstick coordinate system and we define g to point in the negative z-direction in the real world reference frame. This defines the x-y plane in the real world reference plane but not the orientation of the x- and y- axis (this degree of freedom can be further determined by compass). We therefore rotate g into –z around an axis that is perpendicular to g-z plane. We represent the orientation of the sensorstick by unit vectors in the x-, y-, and z-directions of the sensorstick coordinate system. We represent these unit vectors in the gravity reference frame where g points into the positive z-direction. The x-, y-, and z-coordinates of $(1,0,0)^T$, $(0,1,0)^T$, $(0,0,1)^T$ in the sensorstick reference frame are converted into the g-reference frame by applying to the unit vectors the same rotation that is needed to rotate **g** into $(0,0,1)^T$. Without regards for stability (for now) the vectors are first rotated around the z-axis by $-\beta$ until **g** is in the xz-plane. The xz-plane is defined as the span of the x- and z-axis. In this rotation γ remains unchanged. A second rotation around the y-axis by $\pi-\gamma$ followed by the inverse of the initial rotation of β around the z-axis will yield the desired rotation of **g**. The only rotation that remains ill-defined by this definition is the rotation of the vector g.

The right handed rotation around the z-axis by an angle q is given by

$$Rz(q) = \begin{pmatrix} \cos q & -\sin q & 0 \\ \sin q & \cos q & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

and rotations about the x, and y axis are given by

$$Rx(q) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos q & -\sin q \\ 0 & \sin q & \cos q \end{pmatrix}, Ry(q) = \begin{pmatrix} \cos q & 0 & \sin q \\ 0 & 1 & 0 \\ -\sin q & 0 & \cos q \end{pmatrix}$$

The complete rotation can be pre-computed and doesn't have to be performed on the fly. The rotated unit vectors are the first, second, and third columns of the product of 3 rotation matrices. For unstable rotation matrices alternative definitions of the angles may alleviate the problem.

The rotation matrix is given by subsequent rotations:

$$R_{tot} = Rz(\beta)\cdot Ry(\pi-\gamma)\cdot Rz(-\beta)$$
$$= \begin{pmatrix} -\cos(\beta)^2 * \cos(\gamma)+\sin(\beta)^2 & -\cos(\beta)\cos(\gamma)\sin(\beta)-\sin(\beta)\cos(\beta) & -\cos(\beta)\sin(\gamma) \\ -\cos(\beta)\cos(\gamma)\sin(\beta)-\sin(\beta)\cos(\beta) & -\sin(\beta)^2\cos(\gamma)+\cos(\beta)^2 & -\sin(\beta)\sin(\gamma) \\ -\cos(\beta)\sin(\gamma) & \sin(\beta)\sin(\gamma) & -\cos(\gamma) \end{pmatrix}$$

Alternative definition for the axis of rotation is by first defining the axis of rotation, and simply using the cosine and sine of the angle of rotation about that axis. The axis of rotation is given by **a**=$(a_x, a_y, a_z) = (g_x, g_y, g_z)\times(0,0,1)$ and the angle of rotation is given by γ above. The vector **a** describing the axis must be normalized to be of unit length. In this case the rotation matrix is given by

$$R_{tot} = \begin{pmatrix} a_x^2-(1-a_x^2)\cos(\gamma) & a_xa_y(1+\cos(\gamma))-a_z\sin(\gamma) & a_xa_z(1-\cos(\gamma))+a_y\sin(\gamma) \\ a_xa_y(1+\cos(\gamma))+a_z\sin(\gamma) & a_y^2-(1-a_y^2)\cos(\gamma) & a_ya_z(1+\cos(\gamma))-a_x\sin(\gamma) \\ a_xa_z(1+\cos(\gamma))-a_y\sin(\gamma) & a_ya_z(1+\cos(\gamma))+a_x\sin(\gamma) & a_z^2-(1-a_z^2)\cos(\gamma) \end{pmatrix}$$

Matlab Implementation

The file joystick.m is the main program that handles the hardware interface as well as contains the main runtime loop that executes data acquisition and graphing. The rotation described above is separated in the file rotation.m. In rotation.m in the Matlab subfolder the second rotation matrix described above using the rotation about an arbitrary axis commented out and not used. It should be noted that a continuous rotation of 360 degrees around the x- or y-axis is not well defined as a continuous path of the screen object because of the choice for β that was made when g is parallel to the z-axis (positive and negative). The problem is most noticable near γ=180^o. In this case β is not well defined and arbitrarily set to π.