Coordinate Transformations

Dr. John Stryjewski

08/19/20

Abstract

This paper presents some of the more important coordinate transformation required for modeling laser communications systems. This is not meant to be a tutorial, but a reference document for researchers and engineers.

Contents

1 Time

Absolute time is always expressed in Modified Julian Date (MJD).

1.1 Modified Julian Date

The Modified Julian Date (MJD) is derived from the Julian Date (JD). The Julian day is the continuous count of days since the beginning of the Julian Period used primarily by astronomers. The Julian Day Number (JDN) is the integer assigned to a whole solar day in the Julian day count starting from noon Greenwich Mean Time, with Julian day number 0 assigned to the day starting at noon on January 1, 4713 BC, proleptic Julian calendar (November 24, 4714 BC, in the proleptic Gregorian calendar),[1] a date at which three multi-year cycles started and which preceded any historical dates. For example, the Julian day number for the day starting at 12:00 UT on January 1, 2000, is 2,451,545. The Julian date (JD) of any instant is the Julian day number for the preceding noon in Greenwich Mean Time plus the fraction of the day since that instant. Julian dates are expressed as a Julian day number with a decimal fraction added. For example, the Julian Date for 00:30:00.0 UT January 1, 2013, is 2,456,293.520833.

The Modified Julian Date (MJD) was introduced by the Smithsonian Astrophysical Observatory in 1957. The MJD has a starting point of midnight on November 17, 1858 and is computed from Standard Julian date by:

$$MJD = JD - 2400000.5$$

Conversion to and from UTC time is covered in numerous textbooks on astrodynamics.

1.2 Mean Sidereal Time

In practice we live our life by cycles of the Sun. But, since the rotation rate of the Earth is not constant, the Solar day varies in length from week to week. Also, as we progress through an orbit around the Sun we lose on day over the one orbit (year). But since the orbital rate is not constant, we have yet another variation between the wall clock "day" and the solar "day". This is problematic because our clocks run at a constant linear rate. This is solved by adding leap seconds (actually a lag) as needed to keep the solar and wall clocks aligned to within 1 second.

1.3 Universal Time

Universal time or more correctly Universal Time Coordinated (UTC) defined a standard linear time for clocks. In this time standard, a day is exactly 24hrs long. Unfortunately, after a year, the UTC clock is out of sync with the Sidereal[1] clock by a day. Also, the rotation rate of the earth is not constant, resulting in further errors. To reconcile the two clocks we use two corrections. First, the mean time gain of the UTC clock relative to the Sidereal clock is compensated by adding leap seconds (lags) as needed to keep the UTC clock and the Sidereal clock within one second. Second, a correction factor for the non-constant Earth rotation rate is determined and added to the UTC to get UT1 time. Thus UT1 time runs at a non-constant rate and is essentially Sidereal time.

1.4 GPS Time

Global Positioning System (GPS) time is used by the GPS satellite it is a linear time like UTC, but leap seconds are not used. This means that GPS time is slowly gaining relative to UTC. At the time this was written, GPS time was 18s ahead of UTC.

2 Geodetic Coordinate Systems

Several coordinate systems are defined here: Earth Centered Earth Fixed (ECEF) coordinates, Earth Centered Inertial (ECI) coordinates, and horizon Azimuth, Elevation and Range (AER) coordinates.

[1] The sidereal day is referenced to the rotation of the Earth releative to a fixed point in the constellation Aries as opposed to the solar day, which is refereknced to the Sun. A siderael day is about 4min shorter than a solar day.

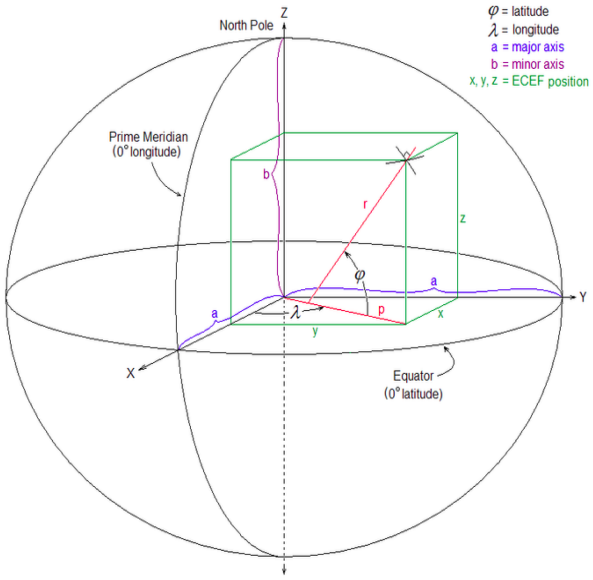

Figure 1: WGS84 geometry

2.1 Earth Centered Earth Fixed (ECEF) Coordinates

ECEF coordinates are defined by the World Geodetic System 1984 (WGS84) standard. Note; this is a geodetic coordinate system not a geocentric coordinate system. The difference between geodetic and geocentric is: the "down" direction in a geodetic coordinate system does not point to the center of the Earth, rather it points normal to the Earth's surface, whereas in a geocentric down always points to the center of the earth. This because the Earth is an oblate spheroid, not a sphere. See the red r direction in Figure 1.

The WGS84 geometry is illustrated in Figure 1. Here λ is the longitude and ϕ is the latitude of a point. X,Y and Z are the Cartesian Earth Centered Earth Fixed geodetic coordinates (ECEF). In this document we will refer to the the ECEF coordinate as E,F and G not X,Y,and Z. Sometimes this the expression Earth Fixed Geodetic (EFG) coordinates to refer to ECEF coordinates, hence E,F and G. Thus, we have the vectors:

$$E = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \; F = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \; G = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{1}$$

The X or E direction passed through the Prime Meridian at the Equator, Z or G points along the spin axis (north pole) of the Earth. The Y, or F, direction is chosen to make the system a right-handed coordinate system. WGS84 defines an ellipsoid (oblate spheroid) as a model of the Earth's geoid, or gravitational equipotential surface. The differences between the model ellipsoid and the geoid can be as much as 100m, but is generally in the range between 10-30m. The geoid (or the model ellipsoid) is not the same as the Mean Sea Level surface because of mountains and other uneven mass distributions. The parameters for the ellipsoid are:

$$\begin{aligned} a &= 6,378,137m \\ 1/f &= 298.257223563 \end{aligned} \tag{2}$$

Here a is the semi-major axis and f is the flattening parameter. Some useful derived parameters are

$$\begin{aligned} b &= a(1-f) \\ e &= \sqrt{2f - f^2} \\ N(\varphi) &= \frac{a}{\sqrt{1-(e \cdot \sin(\varphi))^2}} \end{aligned} \tag{3}$$

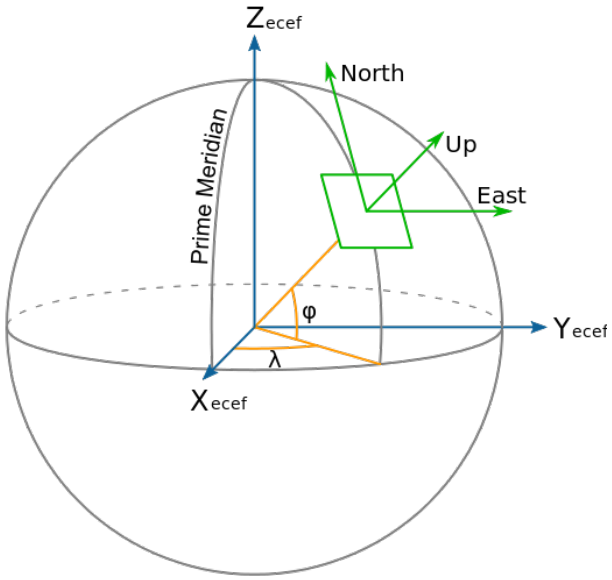

Figure 2: Geometry of the local horizon coordinates

Here b is the semi-minor axis, e is the eccentricity, and $N(\varphi)$ is called the Normal Distance and is the distance from the surface to the G-axis in the downward direction. Thus the complete transformations from Latitude, Longitude and Altitude (LLA) to EFG (or XYZ) are found using:

$$\begin{aligned} E(\lambda,\varphi,h) &= [N(\varphi)+h]\cdot\cos(\lambda)\cdot\cos(\varphi) \\ F(\lambda,\varphi,h) &= [N(\varphi)+h]\cdot\sin(\lambda)\cdot\cos(\varphi) \\ G(\lambda,\varphi,h) &= \left[N(\varphi)\cdot(1-e^2)+h\right]\sin(\varphi) \end{aligned} \tag{4}$$

This transformation is exact. The inverse transform is not as simple, as it requires the solution of a quartic equation to find latitude. There are direct solutions of this problem, such as using Ferrari's approach to solving the quartic equation. However, these solutions are prone to round-off error and fail in some special cases near the poles and equator. The more general approach is to use an iterative approach such as Borkowski's. The approach we have chosen is base on a Regula Falsi approach that converges consistently, and rapidly, for all tested cases. Our method used the following approximation for the transformation:

$$\begin{aligned} \varphi &= \arctan\left(\frac{G}{\sqrt{E^2+F^2}}\cdot\frac{1}{(1-f)^2}\right) \\ \lambda &= \arctan(F/E) \\ h &= \sqrt{E^2+F^2+G^2} - a\cdot\sqrt{\frac{1-e^2}{1-e^2\cdot\cos\left(\arctan\left(\frac{G}{\sqrt{E^2+F^2}}\right)\right)^2}} \end{aligned} \tag{5}$$

After the approximate values of λ,φ and h are found then Equation 4 is used to transform back to EFG coordinates. The error is calculated and the starting EFG is adjusted and then a new estimate is found using Equation 5. Sample C++ code for this method is shown below:

```
1  LLA WGS84::llaFromEFG(EFG& pos) {
2      EFG fakePos = pos;
3      LLA result = LLAFromEFG_APPROX(fakePos);
4      EFG error = efgFromLLA(result) - pos;
5      while ( error.length() > 0.01 ) { // 0.01 meters
6          fakePos -= error;
```



```
7           result = LLAFromEFG_APPROX(fakePos);
8           error = efgFromLLA(result) - pos;
9       }
10      return result;
11  }
12
```

Listing 1: Regula-Falsi method to calculate LLA from EFG.

2.2 Earth Fixed Inertial (ECI) Coordinates

ECI coordinates are essentially the same as geodetic coordinates or ECEF coordinates, but they do not rotate with the Earth. ECI coordinates are fixed with respect to the local celestial frame. In that frame the X-coordinate always points from the center of the Earth to the constellation Ares. These two systems, ECI and ECEF coincide at a time known as the Epoch time. In the paper we use Julian date for January 1, 2000 and the Epoch, also referred to simply as J2000. The transformation to and from ECEF is done by rotating about the G or Z coordinates by and amount proportional to the time since Epoch.

$$\begin{pmatrix} G \\ F \\ G \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \cdot \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{6}$$

Where θ is the angular rotation that has occurred since Epoch.

$$\theta = \omega \cdot (GMST - (UT1 - UTC)) \tag{7}$$

Here $GMST$ is the Greenwich Mead Sidereal Time, $UT1$ is Universal Time 1, and UTC is Universal Time Coordinated. ω is the angular rotation velocity of the Earth:

$$\omega = 0.261799387799149 \cdot radians/hour \tag{8}$$

Note the units. This is done because $GMST$ is given in hours since Epoch.

2.3 Local Horizon Coordinates

While the above EFG coordinate system is useful for describing where something is relative to the Earth, sometime it is necessary to describe where something is relative to a geodetic position on the earth. The two most common of these relative coordinate systems are: azimuth-elevation-range (AER) and east-north-up (ENU). Both are related by the transformations:

$$\begin{pmatrix} E \\ N \\ U \end{pmatrix} = \begin{pmatrix} \sin(az) \cdot \cos(el) \\ \cos(az) \cdot \cos(el) \\ \sin(el) \end{pmatrix} \cdot range \tag{9}$$

and

$$\begin{aligned} az &= \arctan\left(\frac{E}{N}\right) \text{ (Domain is 0 to } 2\pi) \\ el &= \arcsin(U) \\ range &= \sqrt{E^2 + N^2 + U^2} \end{aligned} \tag{10}$$

Both of these coordinate systems are defined in terms of the observer's local horizon (in the WGS84 sense), using local definitions of "north" and "up."

3 Platform Local Coordinates

In general a tracking gimbal vertical axis is not aligned exactly to the up direction and zero azimuth is not aligned to the north direction. For these type "orientation errors" we describe the misalignment in terms of Roll, Pitch , and Yaw.

3.1 Roll Pitch and Yaw (Heading) Conventions

To transform from GIMBAL to TRUE horizon, or TRUE geodetic, we must describe the orientation of the gimbal relative to the TRUE horizon. This transformation will be in the form a rotation in 3-dimensions. The non-commutative nature of rotations in 3D makes it necessary to define.

3.2 Direction Cosine Matrix

The Direction Cosine representation of a unit length vector is by the angle a,b and c that the vector makes with the coordinate axes, Figure 3.

The direction cosines α, β, γ can now be defined:

$$\begin{aligned} \alpha &= \cos(a) = \frac{e_x \cdot v}{|v|} \\ \beta &= \cos(b) = \frac{e_y \cdot v}{|v|} \\ \gamma &= \cos(c) = \frac{e_z \cdot v}{|v|} \end{aligned} \tag{11}$$

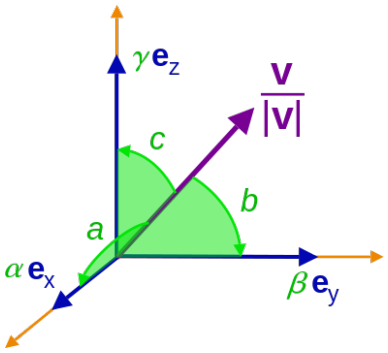

Figure 3: Direction Cosines

Thus, to specify a unit vector in terms of an existing coordinate system requires only 3 parameters, α, β, γ. Using this approach we can represent a transformed coordinate system in terms of the original coordinate system using 9 parameters, 3 each for the new principle axes. In matrix form, we describe the transformation as:

$$\begin{aligned} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{new} &= M \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{old} \\ &= \begin{bmatrix} x_{new} \cdot x_{old} & x_{new} \cdot y_{old} & x_{new} \cdot z_{old} \\ y_{new} \cdot x_{old} & y_{new} \cdot y_{old} & y_{new} \cdot z_{old} \\ z_{new} \cdot x_{old} & z_{new} \cdot y_{old} & z_{new} \cdot z_{old} \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{old} \qquad (12) \\ &= \begin{bmatrix} \alpha_x & \beta_x & \gamma_x \\ \alpha_y & \beta_y & \gamma_y \\ \alpha_z & \beta_z & \gamma_z \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{old} \qquad (13) \end{aligned}$$

The matrix, M, of dot products is called the direction cosine matrix. We can see from inspection that the matrix M is orthogonal (in fact orthonormal) :

$$M^{-1} = M^T \tag{14}$$

This is to be expected since this matrix describes a rotation in 3 dimensions, and the rotation group is homomorphic with the orthonormal group O(3).

3.3 The Roll Pitch and Yaw (heading) Transformation

While the previous section discussed how to mathematically represent rotations in 3 dimensions, it does not describe how to specify the mechanical (physical) actions to implement this rotation. To describe rotations in a fashion similar to translation in Euclidean space we can choose to represent a physical rotation as three successive rotations about 3 orthogonal axis. The problem then is which three axes to use: the three axes attached to the rotating body (intrinsic) or three original unrotated axes (extrinsic). Furthermore, since rotations operations do not commute, the order of operations is important. This leads to 24 possible rotation conventions! We will use a the convention common in navigation: roll-pitch-yaw (RPY). Strictly speaking, the RPY rotation convention uses Tait-Bryan rotations, not Euler rotations. In Euler rotations the first and last rotations are about the same body axis (z then x then z), in Tait-Bryan rotations each rotation is about a unique axis (x then y then z). Roll, Pitch and Yaw are defined [2] as (See Figure 4):

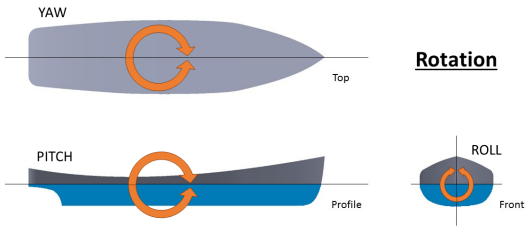

Figure 4: Roll Pitch and Yaw definition

- Roll - Rotation around the front-to-back axis.
- Pitch - Rotation around the side-to-side axis.
- Yaw - Rotation around the vertical axis is called yaw.

Since the order of operations is important, the standard convention is to apply yaw first, then pitch, then roll. Thus,

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{new} = R(\psi)_{roll} \cdot R(\phi)_{pitch} \cdot R(\theta)_{yaw} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{old} \tag{15}$$

Where, the rotation matrices are defines as:

$$\begin{aligned} R(\psi)_{roll} &= \begin{bmatrix} \cos(\psi) & 0 & \sin(\psi) \\ 0 & 1 & 0 \\ -\sin(\psi) & 0 & \cos(\psi) \end{bmatrix} = R(\psi)_y \\ R(\phi)_{pitch} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} = R(\phi)_x \\ R(\theta)_{yaw} &= \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} = R(-\theta)_z \end{aligned} \tag{16}$$

These rotations are about the body centered axes as shown in Figure 4. The net rotation for matrix for the operations, $R(\psi, \phi, \theta) = R(\psi)_{roll} R(\phi)_{pitch} R(\theta)_{yaw}$,is given by:

$$R(\phi)_{pitch} R(\theta)_{yaw} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta)\cos(\phi) & \cos(\theta)\cos(\phi) & -\sin(\phi) \\ -\sin(\theta)\sin(\phi) & \cos(\theta)\sin(\phi) & \cos(\phi) \end{bmatrix}$$

and,

$$R(\psi, \phi, \theta) = R(\psi)_{roll} R(\phi)_{pitch} R(\theta)_{yaw} = \begin{bmatrix} \cos(\psi) & 0 & \sin(\psi) \\ 0 & 1 & 0 \\ -\sin(\psi) & 0 & \cos(\psi) \end{bmatrix} \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta)\cos(\phi) & \cos(\theta)\cos(\phi) & -\sin(\phi) \\ -\sin(\theta)\sin(\phi) & \cos(\theta)\sin(\phi) & \cos(\phi) \end{bmatrix}$$

Thus,

[2] From the Smithsonian National Air and Space Museum website, https://howthingsfly.si.edu/flight-dynamics/roll-pitch-and-yaw

$$R(\psi,\phi,\theta)=\begin{bmatrix} \cos(\theta)\cos(\psi)-\sin(\theta)\sin(\phi)\sin(\psi) & \sin(\theta)\cos(\psi)+\cos(\theta)\sin(\phi)\sin(\psi) & \cos(\phi)\sin(\psi) \\ -\sin(\theta)\cos(\phi) & \cos(\theta)\cos(\phi) & -\sin(\phi) \\ -\cos(\theta)\sin(\psi)-\sin(\theta)\sin(\phi)\cos(\psi) & -\sin(\theta)\sin(\psi)+\cos(\theta)\sin(\phi)\cos(\psi) & \cos(\phi)\cos(\psi) \end{bmatrix} \tag{17}$$

with

$$R(\psi,\phi,\theta)^{-1}=R(\psi,\phi,\theta)^{T}=\begin{bmatrix} \cos(\theta)\cos(\psi)-\sin(\theta)\sin(\phi)\sin(\psi) & -\sin(\theta)\cos(\phi) & -\cos(\theta)\sin(\psi)-\sin(\theta)\sin(\phi)\cos(\psi) \\ \sin(\theta)\cos(\psi)+\cos(\theta)\sin(\phi)\sin(\psi) & \cos(\theta)\cos(\phi) & -\sin(\theta)\sin(\psi)+\cos(\theta)\sin(\phi)\cos(\psi) \\ \cos(\phi)\sin(\psi) & -\sin(\phi) & \cos(\phi)\cos(\psi) \end{bmatrix} \tag{18}$$

If before the rotation, the body directions (x,y,z) coincide with the local horizon coordinates (east, north, up), then Yaw is the same as geodetic heading and the transform may be referred to as the roll-pitch-heading transform.

3.4 Quaternions

Recall that any complex series of rotations can be replaced by a single rotation about a single axis (Euler's rotation theorem). The rotation can thus be represented by a unit vector giving the direction of the rotation axis and a scalar representing the magnitude of the rotation. It turns out that a mathematical construct known as quaternions can be used to encode this information. Consider Euler's formula for the complex representation of a number:

$$z = e^{i\cdot\theta} = cos(\theta) + i\cdot\sin(\theta) \tag{19}$$

This phasor rotation about one axis can be generalized to represent rotations about 3 axes:

$$q = e^{\frac{1}{2}\cdot(u_x\cdot i+u_y\cdot j+u_z\cdot k)\cdot\theta} = \cos(\frac{\theta}{2}) + (u_x\cdot i+u_y\cdot j+u_z\cdot k)\cdot\sin(\frac{\theta}{2}) \tag{20}$$

and

$$q^{-1} = e^{-\frac{1}{2}\cdot(u_x\cdot i+u_y\cdot j+u_z\cdot k)\cdot\theta} = \cos(\frac{\theta}{2}) - (u_x\cdot i+u_y\cdot j+u_z\cdot k)\cdot\sin(\frac{\theta}{2}) \tag{21}$$

This "versor" can be related to the rotation transformation by:

$$\begin{aligned} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{new} &= q\cdot\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{old}\cdot q^{-1} \\ &= \left(\cos(\frac{\theta}{2})+\begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}\cdot\sin(\frac{\theta}{2})\right)\cdot\left(0+\begin{pmatrix} x \\ y \\ z \end{pmatrix}\cdot 1\right)\cdot\left(\cos(\frac{\theta}{2})-\begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}\cdot\sin(\frac{\theta}{2})\right) \end{aligned} \tag{22}$$

Here we have used the fact that a vector can be represented by a quaternion versor with zero rotation (no scalar part). When evaluating the above expression we must use the quaternion math rules NOT the usual vector math rules! Namely,

$$pq = (p_s q_s - q_v\cdot p_v) + (p_s q_v + q_s p_v + p_v\times q_v) \tag{23}$$

Here, the subscripts s and v denote the scalar and vector parts of the quaternion. The normal cross and dot products are used on vector parts.

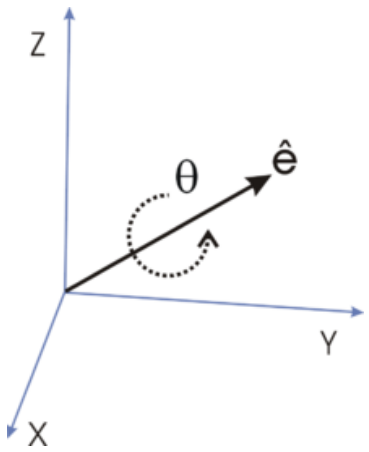

Figure 5: Euler axis-angle representation of a quaternion rotation

Conversion to and from the angle-cosine rotation matrix formalism is simple, consider the quaternion:

$$q = a + bi + cj + dk \tag{24}$$

Then the corresponding rotation matrix is:

$$R = \begin{bmatrix} a^2+b^2-c^2-d^2 & 2bc-2ad & 2bd+2ac \\ 2bc+2ad & a^2+-b+c^2-d^2 & 2cd-2ab \\ 2bd-2ac & 2cd+2ab & a^2-b^2-c^2+d^2 \end{bmatrix} \tag{25}$$

Transforms from a rotation matrix to a quaternion can be found on-line or in numerous books describing quaternions.

4 System Models and Calibration

This section introduces a basic mathematical model for an elevation over azimuth gimbal tracker. It uses the coordinate systems defined in Section 2. The discussion is divided into sections covering: the tracker, the sensors and the platform (ground, ship, aircraft...).

In our model, the errors will be applied (or modeled) in the following order:

1. Sensor Errors
2. Tracker Errors
3. Platform Errors
4. Gravity induced Droop Errors

The order of application is important because some of the errors may be large (degrees) and are modeled as rotations in 3D, which do not commute. The Goal is to find the TRUE telescope viewing direction as a function of the azimuth-elevation orientation of the gimbal, the roll-pitch-yaw(or heading) of the platform, the sensor errors and the platform alignment errors.

4.1 The Modeling Approach

The general idea is to start with the tracking mount model "looking" true north as defined by WGS84. This is the initial look direction or look vector, in ENU coordinates the initial state is:

$$Look\ Direction = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \tag{26}$$

This is defined as zero azimuth and zero elevation. Next, a series of rotation transformations are applied to the line of sight vector. In the ENU coordinate system, we define the local horizon rotations as:

$$
\begin{aligned}
R_{east}(\theta) &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix} \\
R_{north}(\theta) &= \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \\
R_{up}(\theta) &= \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned} \tag{27}
$$

For an azimuth over elevation tracker we can find the general expression for the gimbal look direction as follows: first, starting with the mount looking north rotate the gimbal about the East direction by an amount equal to the elevation; next, rotate the gimbal about the Up direction an amount equal to the azimuth. Thus, if we apply this ideal elevation over azimuth transform, we find the mount pointing direction is given by.

$$
\begin{aligned}
\begin{pmatrix} E \\ N \\ U \end{pmatrix} &= R_{up}(-az) \cdot R_{east}(el) \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} \cos(-az) & -\sin(-az) & 0 \\ \sin(-az) & \cos(-az) & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(el) & -\sin(el) \\ 0 & \sin(el) & \cos(el) \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} \sin(az) \cdot \cos(el) \\ \cos(az) \cdot \cos(el) \\ \sin(el) \end{pmatrix}
\end{aligned} \tag{28}
$$

The minus sign in front of (az)imuth, takes into account that the azimuth rotation direction is opposite of normal right-handed coordinate rotation about the up direction. Note that we applied the elevation transformation first, if we did it the other way around we would have:

$$
\begin{aligned}
\begin{pmatrix} E \\ N \\ U \end{pmatrix} &= R_{up}(-az) \cdot R_{east}(el) \cdot R_{up}(az) \cdot R_{up}(-az) \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \\
&= R_{up}(-az) \cdot R_{east}(el) \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} \sin(az) \cdot \cos(el) \\ \cos(az) \cdot \cos(el) \\ \sin(el) \end{pmatrix}
\end{aligned} \tag{29}
$$

Which is the same result, as Equation (28).

4.2 Gimbal Lock

At this point, it is useful to discuss the concept of "Gimbal Lock." In an elevation over azimuth gimbal, "lock" occurs when the tracker is pointing straight up (elevation = 90degrees). When the mount is in this position, movements in azimuth have no effect; in fact, for the zenith orientation mathematically azimuth is not defined. We can observe the behavior of the tracker in the vicinity of $el = \pi/2$ by using the following approximations (Taylor series expansion about $el = \pi/2$):

$$\begin{aligned}\sin(el)|_{el\to\pi/2} &= \sin(\frac{\pi}{2}) + \cos(\frac{\pi}{2})(el-\frac{\pi}{2}) - \frac{1}{2}\sin(\frac{\pi}{2})(el-\frac{\pi}{2})^2 + \cdots = 1 - \frac{1}{2}\Delta el^2 + = \cos(\Delta el)\\ \cos(el)_{el\to\pi/2}| &= \cos(\frac{\pi}{2}) - \sin(\frac{\pi}{2})(el-\frac{\pi}{2}) + = \Delta el + \cdots = \sin(\Delta el)\end{aligned}$$

Substituting into Equation 28, we find:

$$\begin{pmatrix} E \\ N \\ U \end{pmatrix} = \begin{pmatrix} \sin(az)\cdot\sin(\Delta el) \\ \cos(az)\cdot\sin(\Delta el) \\ \cos(\Delta el) \end{pmatrix} \tag{30}$$

As $\Delta el \to \pi/2$, we find:

$$\begin{pmatrix} E \\ N \\ U \end{pmatrix} = \begin{pmatrix} \sin(az)\cdot 0 \\ \cos(az)\cdot 0 \\ 1 \end{pmatrix} \tag{31}$$

$$= \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{32}$$

Thus azimuth motion is undefined for zenith.

Gimbal lock is not just a mathematical effect, it has a real manifestation for an elevation over azimuth gimbal. This is observed by the fact when the tracker is at zenith it has only 1 degree of freedom (elevation) because azimuth has no effect, it is "locked." Another way of visualizing it is that as the gimbal moves through zenith going beyond 90degrees, the azimuth direction must instantly switch 180degrees. Gimbal lock has important consequences for LEO satellite tracking for high elevation passes.

5 Sensor Errors

There are three primary sensor errors: Sensor Elevation Bias, Sensor Azimuth Bias (also known as sensor skew), and Droop. The two biases are due to the mounting of the sensor to the gimbal and modeled mathematically. Droop, however, is very complex and depends on the mechanical response of the sensor system to gravity as elevation changes.

5.1 Elevation Bias

The sensor elevation bias (*seb*) is modeled as an extra rotation about the initial elevation axis:

$$\begin{aligned}\begin{pmatrix} E \\ N \\ U \end{pmatrix} &= R_{up}(-az)\cdot R_{east}(el)\cdot R_{east}(seb)\cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\\ &= R_{up}(-az)\cdot R_{east}(el+seb)\cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \qquad (33)\\ &= \begin{pmatrix} \sin(az)\cdot\cos(el+seb) \\ \cos(az)\cdot\cos(el+seb) \\ \sin(el+seb) \end{pmatrix}\end{aligned}$$

It is clear from Equation (33) that Sensor Elevation Bias and Elevation are indistinguishable. Therefore, *Sensor Elevation Bias will not be modeled.*

5.2 Skew (Sensor Azimuth Bias)

Sensor skew is just a rotation about the "up" axis by amount *skew*:

$$\begin{aligned}\begin{pmatrix} E \\ N \\ U \end{pmatrix} &= R_{up}(-az) \cdot \mathrm{R}_{\mathrm{east}}(\mathrm{el}) \cdot R_{up}(-skew) \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} \cos(skew) \cdot \sin(az) \cdot \cos(el) - \cos(az) \cdot \sin(skew) \\ \cos(skew) \cdot \cos(az) \cdot \cos(el) + \sin(az) \cdot \sin(skew) \\ \cos(skew) \cdot \sin(el) \end{pmatrix} \end{aligned} \tag{34}$$

If the skew is small ($el < 0.1 radians$), then we can use the linearized approximations:

$$\begin{aligned}\sin(\theta) &\approx \theta \\ \cos(\theta) &\approx 1\end{aligned}$$

substituting into Equation 34, we find:

$$\begin{aligned}\begin{pmatrix} E \\ N \\ U \end{pmatrix}_{skew} &\approx \begin{pmatrix} 1 \cdot \sin(az) \cdot \cos(el) - \cos(az) \cdot (skew) \\ 1 \cdot \cos(az) \cdot \cos(el) + \sin(az) \cdot (skew) \\ 1 \cdot \sin(el) \end{pmatrix} \\ &\approx \begin{pmatrix} \sin(az) \cdot \cos(el) \\ \cos(az) \cdot \cos(el) \\ \sin(el) \end{pmatrix} + \begin{pmatrix} -\cos(az) \cdot (skew) \\ +\sin(az) \cdot (skew) \\ 0 \end{pmatrix} \\ &\approx \begin{pmatrix} E \\ N \\ U \end{pmatrix} + \begin{pmatrix} -\cos(az) \\ \sin(az) \\ 0 \end{pmatrix} \cdot skew \\ &\approx \begin{pmatrix} E \\ N \\ U \end{pmatrix} + \frac{\mathrm{d}}{\mathrm{d}az} \begin{pmatrix} E \\ N \\ U \end{pmatrix} \cdot skew \end{aligned}$$

The first term on the right is the "un-skewed" pointing model and the term on the right is the skew error. Upon inspection it is clear that, for small angles, the skew is in the azimuth direction.

5.3 Sensor Droop

Droop is the different from the other sensor errors; it is a result of gravity, not mounting error. As such, it can not be applied in the tracker reference frame like skew or elevation bias, it must be applied in the absolute (WGS84) frame. If we assume that droop is approximately proportional to the component of gravity perpendicular to the optical axis, we get the following approximation for the droop error:

$$\begin{aligned}\triangle el &= droop \cdot \cos(el) \\ &= droop \cdot \cos(\arcsin(U)) \\ &= droop \cdot \sqrt{1 - U^2}\end{aligned} \tag{35}$$

This elevation correction must be applied as the last correction after sensor, mount and platform corrections have been applied.

6 Tracker Errors

This sections discusses errors that are unique to the tracker hardware and are not due to the platform the tracker sits on or the sensors themselves.

6.1 Non-Orthogonality

Non-orthogonality is when the rotation axes for azimuth and elevation are not perpendicular; we model this by tilting the elevation axis about the North direction. The new model is now:

$$\begin{aligned}
\begin{pmatrix} E \\ N \\ U \end{pmatrix} &= R_{up}(-az) \cdot R_{north}(\text{non}) \cdot \text{R}_{\text{east}}(\text{el}) \cdot R_{north}(\text{-non}) \cdot N \\
&= \begin{pmatrix} \cos(-az) & -\sin(-az) & 0 \\ \sin(-az) & \cos(-az) & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos(non) & 0 & \sin(non) \\ 0 & 1 & 0 \\ -\sin(non) & 0 & \cos(non) \end{pmatrix} \cdot \\
&\quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(el) & -\sin(el) \\ 0 & \sin(el) & \cos(el) \end{pmatrix} \cdot \begin{pmatrix} \cos(-non) & 0 & \sin(-non) \\ 0 & 1 & 0 \\ -\sin(-non) & 0 & \cos(-non) \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} \cos(-az) & -\sin(-az) & 0 \\ \sin(-az) & \cos(-az) & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos(non) & 0 & \sin(non) \\ 0 & 1 & 0 \\ -\sin(non) & 0 & \cos(non) \end{pmatrix} \cdot \\
&\quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(el) & -\sin(el) \\ 0 & \sin(el) & \cos(el) \end{pmatrix} \cdot \begin{pmatrix} \cos(non) & 0 & -\sin(non) \\ 0 & 1 & 0 \\ \sin(non) & 0 & \cos(non) \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} \sin(az) \cdot \cos(el) + \cos(az) \cdot \sin(el) \cdot \sin(non) \\ \cos(az) \cdot \cos(el) - \sin(az) \cdot \sin(el) \cdot \sin(non) \\ \sin(el) \cdot \cos(non) \end{pmatrix}
\end{aligned} \tag{36}$$

The most important effect of non-orthogonality is to prevent the tracker from pointing near zenith. This is illustrated by setting $azimuth = 0$ and $elevation = \pi/2$.

$$\begin{aligned}
\begin{pmatrix} E \\ N \\ U \end{pmatrix} &= \begin{pmatrix} \sin(0) \cdot \cos(\frac{\pi}{2}) + \cos(0) \cdot \sin(\frac{\pi}{2}) \cdot \sin(non) \\ \cos(0) \cdot \cos(\frac{\pi}{2}) - \sin(0) \cdot \sin(\frac{\pi}{2}) \cdot \sin(non) \\ \sin(\frac{\pi}{2}) \cdot \cos(non) \end{pmatrix} \\
&= \begin{pmatrix} \sin(non) \\ 0 \\ \cos(non) \end{pmatrix}
\end{aligned} \tag{37}$$

The angle that this direction makes with up is:

$$\begin{aligned}
\text{Angle} &= \arccos\left[\begin{pmatrix} \sin(non) \\ 0 \\ \cos(non) \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right] \\
&= \arccos\left[\cos(non)\right] \\
&= non
\end{aligned} \tag{38}$$

as expected. Now, including *Skew*, we can write:

$$\begin{pmatrix} E \\ N \\ U \end{pmatrix} = R_{up}(-az) \cdot R_{north}(\text{non}) \cdot \text{R}_{\text{east}}(\text{el}) \cdot R_{north}(\text{-non}) \cdot R_{up}(-skew) \cdot N$$

$$= \begin{bmatrix} \cos(skew) \cdot (\sin(az) \cdot \cos(el) + \cos(az) \cdot \sin(el) \cdot \sin(non)) + \\ \sin(skew) \cdot \left(\cos(az) \cdot \cos(non)^2 - \sin(non) \cdot \sin(az) \cdot \sin(el) + \cos(az) \cdot \cos(el) \cdot \sin(non)^2\right) \\ \\ \cos(skew) \cdot (\cos(az) \cdot \cos(el) - \sin(az) \cdot \sin(el) \cdot \sin(non)) - \\ \sin(skew) \cdot \left(\sin(az) \cdot \cos(non)^2 + \sin(non) \cdot \cos(az) \cdot \sin(el) + \sin(az) \cdot \cos(el) \cdot \sin(non)^2\right) \\ \\ \cos(skew) \cdot \sin(el) \cdot \cos(non) - \sin(skew) \cdot (\cos(non) \cdot \sin(non) - \cos(el) \cdot \cos(non) \cdot \sin(non)) \end{bmatrix}$$

Since the non-orthogonality is expected to be small, we can use the following approximations:

$$\begin{aligned} sin(non) &\approx non + O(non^3) \\ cos(non) &\approx 1 + O(non^2) \end{aligned} \tag{39}$$

Thus,

$$\begin{pmatrix} E \\ N \\ U \end{pmatrix} = R_{up}(-az) \cdot R_{north}(non) \cdot \text{R}_{\text{east}}(\text{el}) \cdot R_{north}(-non) \cdot R_{up}(-skew) \cdot N$$

$$\approx \begin{bmatrix} \cos(skew) \cdot (\sin(az) \cdot \cos(el) + \cos(az) \cdot \sin(el) \cdot non) + \sin(skew) \cdot (\cos(az) - non \cdot \sin(az) \cdot \sin(el)) \\ \cos(skew) \cdot (\cos(az) \cdot \cos(el) - \sin(az) \cdot \sin(el) \cdot non) - \sin(skew) \cdot (\sin(az) + non \cdot \cos(az) \cdot \sin(el)) \\ \cos(skew) \cdot \sin(el) - \sin(skew) \cdot (1 - \cos(el)) \cdot non \end{bmatrix}$$

In the limit that *non* goes to zero we have:

$$\begin{aligned} \begin{pmatrix} E \\ N \\ U \end{pmatrix} &= \lim_{non \to 0} [R_{up}(-az) \cdot R_{north}(non) \cdot \text{R}_{\text{east}}(\text{el}) \cdot R_{north}(-non) \cdot R_{up}(-skew) \cdot N] \\ &\approx \begin{bmatrix} \cos(skew) \cdot \sin(az) \cdot \cos(el) + \sin(skew) \cdot \cos(az) \\ \cos(skew) \cdot \cos(az) \cdot \cos(el) - \sin(skew) \cdot \sin(az) \\ \cos(skew) \cdot \sin(el) \end{bmatrix} \end{aligned}$$

And in the limit that *skew* goes to zero:

$$\begin{aligned} \begin{pmatrix} E \\ N \\ U \end{pmatrix} &= \lim_{skew, non \to 0} [R_{up}(-az) \cdot R_{north}(non) \cdot \text{R}_{\text{east}}(\text{el}) \cdot R_{north}(-non) \cdot R_{up}(-skew) \cdot N] \\ &\approx \begin{bmatrix} \sin(az) \cdot \cos(el) \\ \cos(az) \cdot \cos(el) \\ \sin(el) \end{bmatrix} \end{aligned}$$

As expected, this is the ideal az-el model given in Equation (28).

Notice, in if $non = 0$, then:

$$\begin{pmatrix} E \\ N \\ U \end{pmatrix} = \begin{bmatrix} \cos(skew)\cdot\sin(az)\cdot\cos(el)+\sin(skew)\cdot\cos(az) \\ \cos(skew)\cdot\cos(az)\cdot\cos(el)-\sin(skew)\cdot\sin(az) \\ \cos(skew)\cdot\sin(el) \end{bmatrix}$$
$$= \cos(skew)\cdot\begin{bmatrix} \sin(az)\cdot\cos(el)+\tan(skew)\cdot\cos(az) \\ \cos(az)\cdot\cos(el)-\tan(skew)\cdot\sin(az) \\ \cdot\sin(el) \end{bmatrix}$$

Thus, the error in elevation due to skew is:

$$\triangle el = \arcsin\left(\cos(skew)\cdot\sin(el)\right) - el$$

and, the error in azimuth due to skew is:

$$\triangle az = \arctan\left(\frac{\cos(az)\cdot\cos(el)-\tan(skew)\cdot\sin(az)}{\sin(az)\cdot\cos(el)+\tan(skew)\cdot\cos(az)}\right) - az$$

6.2 Azimuth and Elevation Bias

This "error" compensates for the fact that the mount azimuth and elevation encoders do not read $(0,0)$ when the mount is pointing due north. These are not really errors, but reflect the mounting position of the encoders themselves. These biases can be large depending on the tracker. The biased encoders are sometimes referred to as corrected encoders, *corrected encoder* = *raw encoder* − *bias*, we will use this convention. To simplify the notation, we will use:

$$\begin{aligned} caz &= az - azBias \\ cel &= el - elBias \end{aligned} \tag{40}$$

They are expressed simply as:

$$\begin{pmatrix} E \\ N \\ U \end{pmatrix} = R_{up}(-caz)\cdot R_{north}(non)\cdot \mathrm{R}_{\mathrm{east}}(\mathrm{cel})\cdot R_{north}(-non)\cdot R_{up}(-skew)\cdot N$$
$$\approx \begin{bmatrix} \cos(skew)\left(\sin(caz)\cos(cel)+\cos(caz)\sin(cel)(non)+\sin(skew)\left(\cos(caz)-(non)\sin(caz)\sin(cel)\right)\right. \\ \cos(skew)\left(\cos(caz)\cos(cel)-\sin(caz)\sin(cel)(non)-\sin(skew)\left(\sin(caz)+(non)\cos(caz)\sin(cel)\right)\right. \\ \cos(skew)\sin(cel)-\sin(skew)\left(1-\cos(cel)\right)non \end{bmatrix} \tag{41}$$

This model has all of the static corrections for the tracker and sensor. The true azimuth and elevation are found by:

$$\begin{aligned} az &= \arctan\left(\frac{N}{E}\right) \text{ (Domain is 0 to } 2\pi) \\ el &= \arcsin(U) \\ range &= \sqrt{E^2+N^2+U^2} \end{aligned} \tag{42}$$

Now we must add the platform corrections.

7 Platform Errors

The platform error consists of a general rotation of the gimbal relative to the platform. In the case where the mount is on land and the platform is the earth, this is a combination of mislevel and azimuth rotation. In the case when the mount is on a moving platform (ship or aircraft), we call it the IMU mounting errors. In both cases we choose to model platform errors as the general roll-pitch-yaw errors. These are illustrated in Figure 6.

7.1 Platform transforms

From Equations 14 and 15 from TN 15-001 Coordinate Systems, the RPY rotation is given by:

$$R(\Delta\psi,\Delta\phi,\Delta\theta) = \begin{bmatrix} \cos(\theta)\cos(\psi) - \sin(\theta)\sin(\phi)\sin(\psi) & \sin(\theta)\cos(\psi) + \cos(\theta)\sin(\phi)\sin(\psi) & \cdots \\ -\sin(\theta)\cos(\phi) & \cos(\theta)\cos(\phi) & \cdots \\ -\cos(\theta)\sin(\psi) - \sin(\theta)\sin(\phi)\cos(\psi) & -\sin(\theta)\sin(\psi) + \cos(\theta)\sin(\phi)\cos(\psi) & \end{bmatrix} \tag{43}$$

$$\begin{matrix} \cos(\phi)\sin(\psi) \\ \cdots \quad -\sin(\phi) \\ \cos(\phi)\cos(\psi) \end{matrix} \Bigg]$$

and the inverse by

$$R(\Delta\psi,\Delta\phi,\Delta\theta)^{-1} = \begin{bmatrix} \cos(\theta)\cos(\psi) - \sin(\theta)\sin(\phi)\sin(\psi) & \\ \sin(\theta)\cos(\psi) + \cos(\theta)\sin(\phi)\sin(\psi) & \cdots \\ \cos(\phi)\sin(\psi) & \end{bmatrix} \tag{44}$$

$$\begin{matrix} -\sin(\theta)\cos(\phi) & -\cos(\theta)\sin(\psi) - \sin(\theta)\sin(\phi)\cos(\psi) \\ \cdots \quad \cos(\theta)\cos(\phi) & -\sin(\theta)\sin(\psi) + \cos(\theta)\sin(\phi)\cos(\psi) \\ -\sin(\phi) & \cos(\phi)\cos(\psi) \end{matrix} \Bigg]$$

7.2 IMU Conventions

By definition we will use the RPY transform to go from the gimbal coordinate system to the platform coordinate system. For example, if the gimbal is pointing due north (in the gimbal reference) then the true pointing direction is found by:

$$\begin{pmatrix} east \\ north \\ up \end{pmatrix}_{true} = R(\psi)_{roll} \cdot R(\phi)_{pitch} \cdot R(\theta)_{yaw} \cdot \begin{pmatrix} east \\ north \\ up \end{pmatrix}_{gimbal} \tag{45}$$

where the rotation matrices are defined in Equation 13 in TN 15-001 Coordinate Systems (shown below).

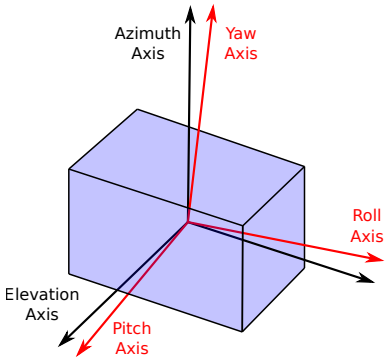

Figure 6: IMU referencing errors

$$
\begin{aligned}
R(\psi)_{roll} &= \begin{bmatrix} \cos(\psi) & 0 & \sin(\psi) \\ 0 & 1 & 0 \\ -\sin(\psi) & 0 & \cos(\psi) \end{bmatrix} = R(\psi)_y \\
R(\phi)_{pitch} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} = R(\phi)_x \\
R(\theta)_{yaw} &= \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} = R(-\theta)_z
\end{aligned}
$$

8 Mount Calibration Approach

Now that we have an error model for the gimbal, we need a methodology to determine the values of the errors. Our general approach is to point the gimbal at a series of targets that we know the location: the difference between the measure and (theoretical) apparent azimuths and elevations is due to the errors. The error model transformations are used to determine the model-based apparent azimuth and elevation.

$$
\begin{aligned}
\begin{pmatrix} E \\ N \\ U \end{pmatrix}_{apparent} = R_{up}(-azBias) \cdot R_{east}(elBias) \cdot R_{up}(-az) \cdot R_{north}(non) \cdot \mathrm{R_{east}(el)} \cdot R_{north}(-non) \cdot \\
R_{up}(-skew) \cdot R(-\theta)_{yaw} \cdot R(-\phi)_{pitch} \cdot R(-\psi)_{roll} \begin{pmatrix} E \\ N \\ U \end{pmatrix}_{true}
\end{aligned}
\tag{46}
$$

Note that the inverse RPY transformation is used to go from TRUE to GIMBAL coordinates.

8.1 Star Calibration

We measure the difference between the observed apparent direction and the measured direction, the designate difference, for a number of targets and then perform a nonlinear least squares fit to determine the designate difference errors. A non-linear least squares minimization the sum, σ, where:

$$
\sigma = \frac{1}{2} \sum (\Delta\theta)^2
$$

In practice we an approximation to this sum, which is easier to calculate. In this simplified approach we minimize the following quantity:

$$
\sigma = \sum \left(1 - \begin{pmatrix} E \\ N \\ U \end{pmatrix}_{apparent} \cdot \begin{pmatrix} E \\ N \\ U \end{pmatrix}_{measured} \right) \tag{47}
$$

Which for small differences (<1mRad) between the measured and apparent direction becomes

$$\begin{aligned} \sigma &= \sum (1 - \cos(\theta)) \\ &\approx \sum \left(1 - \left(1 - \frac{1}{2}(\Delta\theta)^2\right)\right) \\ &\approx \frac{1}{2}\sum (\Delta\theta)^2 \end{aligned} \tag{48}$$

This approach (47) eliminates the need to calculate inverse trigonometric functions.

8.2 Refraction Correction Algorithm

This algorithm is based on the US Naval Observatory code shown below. This is a standard algorithm used by the astronomy community and is based the approach given by the Astronomical Almanac (see reference in code snippet below).

```
double calculateRefraction(double altitude, double elevation, double pressure=-1,
                                            double temperature=10.0,bool fromTrue=true ) {
    /**
     * This function calculates the atmospheric refraction angle for exo-atmospheric objects
     * given the apparent elevation. This formula is accurate to about = 290 microradians
     *
     * @param altitude     This is the geodetic altitude, in meters.
     * @param elevation    This is the angle above the horizontal, in radians.
     * @param pressure     This is the atmospheric pressure, in kPa.
     * @param temperature  This is the ambient temperature, in degrees celsius.
     * @param fromFrue     This is true the angle is measured from the true target elevation,
     *                     otherwise it is measured from the apparent position.
     *
     * For source material on the algorithm see: Bennett, G.G. (1982). "The Calculation of
     * Astronomical Refraction in Marine Navigation". Journal of Navigation 35: 255-259
     */

    const double scaleHeight = 9.1e3;  // the approximate scale height of atmosphere in meters
    double refractionAngle = 0;

    if ((elevation < 0.0) || (elevation > 89.5)) {
        return refractionAngle;
    }

    if (pressure < 0)
        pressure = 101.0 * exp(-altitude / scaleHeight);

    if (fromTrue) {

        const double c1 = 8.232136307e-4;
        const double c2 = 5.879116202e-3;
        const double c3 = 8.918632477e-2;
        refractionAngle  = c1*pressure / (temperature+273.0) / tan(elevation+c2/(elevation+c3));
    }
    else { //from Apparent

        const double c1 = 8.150630006e-4;
        const double c2 = 2.226753338e-3;
        const double c3 = 7.679448708e-2;
        refractionAngle  = c1*pressure / (temperature+273.0) / tan(elevation+c2/(elevation+c3));
    }
    return refractionAngle;
}
```

Listing 2: Atmospheric Refraction Algorithm

Figure 7 shows the refraction error as a function of elevation above the horizon.

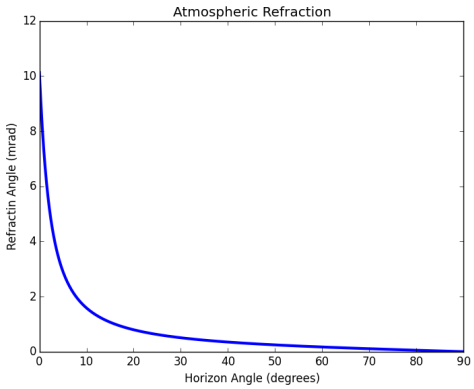

Figure 7: Atmospheric Refraction correction for apparent elevations.

The cited document (Bennett, 1982) also discusses the error in this estimate. That error will be added to the real time satellite buffer calculation.

A Law of Cosines for Azimuth Elevation Space

This appendix describes some useful mathematical relationships for relative: angles, angular velocity, and angular acceleration. These result will look similar to but is not identical to the spherical-polar transform Law of Cosines that is common in many math text books. Here elevation is measured from the equator, while the polar zenith angle is measured from the north pole. Thus, the spherical Law of Cosines (expressed in azimuth elevation space) gives:

$$\theta = \arccos\left(\sin(el_1)\cdot\sin(el_2) + \cos(el_1)\cdot\cos(el_2)\cdot\cos(az_1 - az_2)\right)$$

for clarity, we make the substitution:

$$x = \sin(el_1)\cdot\sin(el_2) + \cos(el_1)\cdot\cos(el_2)\cdot\cos(az_1 - az_2)$$

Thus,

$$\theta = \arccos\left(x\right)$$

and

$$\frac{d\theta}{dt} = \frac{-1}{\sqrt{1-x^2}}\cdot\frac{dx}{dt}$$

where θ is the closure angle and the subscripts b and s denote beam and satellite. Now, the closure rate can be calculated once we know, dx/dt. By the chain rule:

$$\frac{dx}{dt} = \frac{\partial x}{\partial el_1}\frac{del_1}{dt} + \frac{\partial x}{\partial el_2}\frac{del_2}{dt} + \frac{\partial x}{\partial az_1}\frac{daz_1}{dt} + \frac{\partial x}{\partial az_2}\frac{daz_2}{dt}$$

where the elevation and azimuth velocities are measured

$$\begin{aligned}
\frac{\partial x}{\partial el_1} &= \cos(el_1)\cdot\sin(el_2) - \sin(el_1)\cdot\cos(el_2)\cdot\cos(az_1 - az_2)\\
\frac{\partial x}{\partial el_2} &= \sin(el_1)\cdot\cos(el_2) - \cos(el_1)\cdot\sin(el_2)\cdot\cos(az_1 - az_2)\\
\frac{\partial x}{\partial az_1} &= -\cos(el_1)\cdot\cos(el_2)\cdot\sin(az_1 - az_2)\\
\frac{\partial x}{\partial az_2} &= \cos(el_1)\cdot\cos(el_2)\cdot\sin(az_1 - az_2) = -\frac{\partial x}{\partial az_1}
\end{aligned}$$

In a similar fashion, the closure acceleration can be calculated

$$\frac{d^2\theta}{dt^2} = \frac{-x}{(1-x^2)^{3/2}}\cdot\left(\frac{dx}{dt}\right)^2 + \frac{-1}{\sqrt{1-x^2}}\cdot\frac{d^2x}{dt^2}$$

and

$$\begin{aligned}
\frac{d^2x}{dt^2} &= \frac{\partial^2 x}{\partial el_1^2}\left(\frac{del_1}{dt}\right)^2 + \frac{\partial^2 x}{\partial el_2^2}\left(\frac{del_2}{dt}\right)^2 + \frac{\partial^2 x}{\partial az_1^2}\left(\frac{daz_1}{dt}\right)^2 + \frac{\partial^2 x}{\partial az_2^2}\left(\frac{daz_2}{dt}\right)^2 \\
&+ 2\left(\frac{\partial^2 x}{\partial el_1 \partial el_2}\right)\left(\frac{del_1}{dt}\right)\left(\frac{del_2}{dt}\right) + 2\left(\frac{\partial^2 x}{\partial el_1 \partial az_1}\right)\left(\frac{del_1}{dt}\right)\left(\frac{daz_1}{dt}\right) \\
&+ 2\left(\frac{\partial^2 x}{\partial el_1 \partial az_2}\right)\left(\frac{del_1}{dt}\right)\left(\frac{daz_2}{dt}\right) + 2\left(\frac{\partial^2 x}{\partial el_2 \partial az_1}\right)\left(\frac{del_2}{dt}\right)\left(\frac{daz_1}{dt}\right) \\
&+ 2\left(\frac{\partial^2 x}{\partial el_2 \partial az_2}\right)\left(\frac{del_2}{dt}\right)\left(\frac{daz_2}{dt}\right) + 2\left(\frac{\partial^2 x}{\partial az_1 \partial az_2}\right)\left(\frac{daz_1}{dt}\right)\left(\frac{daz_2}{dt}\right) \\
&+ \frac{\partial x}{\partial el_1}\frac{d^2 el_1}{dt^2} + \frac{\partial x}{\partial el_2}\frac{d^2 el_2}{dt^2} + \frac{\partial x}{\partial az_1}\frac{d^2 az_1}{dt^2} + \frac{\partial x}{\partial az_2}\frac{d^2 az_2}{dt^2}
\end{aligned}$$

Where,

$$\begin{aligned}
\frac{\partial^2 x}{\partial el_1^2} &= -\sin(el_1)\cdot\sin(el_2) - \cos(el_1)\cdot\cos(el_2)\cdot\cos(az_1 - az_2) = -x \\
\frac{\partial^2 x}{\partial el_1 \partial el_2} &= \cos(el_1)\cdot\cos(el_2) + \sin(el_1)\cdot\sin(el_2)\cdot\cos(az_1 - az_2) \\
\frac{\partial^2 x}{\partial el_1 \partial az_1} &= \sin(el_1)\cdot\cos(el_2)\cdot\sin(az_1 - az_2) \\
\frac{\partial^2 x}{\partial el_1 \partial az_2} &= -\sin(el_1)\cdot\cos(el_2)\cdot\sin(az_1 - az_2) = -\frac{\partial^2 \theta}{\partial el_1 \partial az_1} \\
\frac{\partial^2 x}{\partial el_2^2} &= -\sin(el_1)\cdot\sin(el_2) - \cos(el_1)\cdot\cos(el_2)\cdot\cos(az_1 - az_2) = -x \\
\frac{\partial^2 x}{\partial el_2 \partial az_1} &= \cos(el_1)\cdot\sin(el_2)\cdot\sin(az_1 - az_2) \\
\frac{\partial^2 x}{\partial el_2 \partial az_2} &= -\cos(el_1)\cdot\sin(el_2)\cdot\sin(az_1 - az_2) = -\frac{\partial^2 \theta}{\partial el_2 \partial az_1} \\
\frac{\partial^2 x}{\partial az_1^2} &= -\cos(el_1)\cdot\cos(el_2)\cdot\cos(az_1 - az_2) \\
\frac{\partial^2 \theta}{\partial az_1 \partial az_2} &= \cos(el_1)\cdot\cos(el_2)\cdot\cos(az_1 - az_2) = -\frac{\partial^2 \theta}{\partial az_1^2} \\
\frac{\partial^2 \theta}{\partial az_2^2} &= -\cos(el_1)\cdot\cos(el_2)\cdot\cos(az_1 - az_2) = \frac{\partial^2 \theta}{\partial az_1^2}
\end{aligned}$$

simplifying

$$
\begin{aligned}
\frac{d^2\theta}{dt^2} &= -x\left(\left(\frac{del_1}{dt}\right)^2+\left(\frac{del_2}{dt}\right)^2\right)\\
&- (\cos(el_1)\cdot\cos(el_2)\cdot\cos(az_1-az_2))\left(\left(\frac{daz_1}{dt}\right)^2+\left(\frac{daz_2}{dt}\right)^2-2\left(\frac{daz_1}{dt}\right)\left(\frac{daz_2}{dt}\right)\right)\\
&+ 2\left(\cos(el_1)\cdot\cos(el_2)+\sin(el_1)\cdot\sin(el_2)\cdot\cos(az_1-az_2)\right)\left(\frac{del_1}{dt}\right)\left(\frac{del_2}{dt}\right)\\
&+ 2\left(\sin(el_1)\cdot\cos(el_2)\cdot\sin(az_1-az_2)\right)\left(\left(\frac{del_1}{dt}\right)\left(\frac{daz_1}{dt}\right)-\left(\frac{del_1}{dt}\right)\left(\frac{daz_2}{dt}\right)\right)\\
&+ 2\left(\cos(el_1)\cdot\sin(el_2)\cdot\sin(az_1-az_2)\right)\left(\left(\frac{del_2}{dt}\right)\left(\frac{daz_1}{dt}\right)-\left(\frac{del_2}{dt}\right)\left(\frac{daz_2}{dt}\right)\right)\\
&+ (\cos(el_1)\cdot\sin(el_2)-\sin(el_1)\cdot\cos(el_2)\cdot\cos(az_1-az_2))\left(\frac{d^2el_1}{dt^2}\right)\\
&+ (\sin(el_1)\cdot\cos(el_2)-\cos(el_1)\cdot\sin(el_2)\cdot\cos(az_1-az_2))\left(\frac{d^2el_2}{dt^2}\right)\\
&- (\cos(el_1)\cdot\cos(el_2)\cdot\sin(az_1-az_2))\left(\frac{d^2az_1}{dt^2}-\frac{d^2az_2}{dt^2}\right)
\end{aligned}
$$