

B2X Gimbal Misalignment Calibration

Edited [2019-11-22](#)

- [Summary](#)
- [Notation](#)
 - [In Documents and Equations](#)
 - [In Code](#)
- [Outline of Calibration Algorithm Derivation](#)
- [Definition of Coordinate Frames](#)
 - [Summary of Frames and Symbols](#)
 - [Canonical Loon Coordinate Frames](#)
 - [Attitude Measurement Frames](#)
 - [Gimbal Frames](#)
 - [Target Frames](#)
- [Gimbal Pointing Problem](#)
 - [Solution of Gimbal Pointing Problem](#)
 - [Solution to the Nominal Gimbal Pointing Problem](#)
 - [Small-Angle Solution to the Skewed Gimbal Pointing Problem](#)
 - [Matlab Simulation and Verification](#)
- [Calibration Problem](#)
 - [Solution of the Calibration Problem](#)
 - [Batch Least Squares Implementation](#)
 - [Recursive Least Squares Implementation](#)
 - [Observability and Selection of Parameters to Calibrate](#)
- [Table of Symbols and Names](#)
 - [Position of the pointing gimbal](#)
 - [Position of target](#)
 - [Payload Attitude](#)
 - [Nominal yaw of gimbal mount frame w.r.t attitude frame](#)
 - [Actual as-found azimuth and elevation of target](#)
 - [Misalignment Angles](#)

Summary

This document derives an algorithm for estimating the small-angle misalignments in the Loon B2X gimbals from flight data. The algorithm has been successfully implemented and used to significantly improve the pointing accuracy of the gimbals in flight – see go/gimbal-inflight-cal-perf.

Notation

The basis of the algorithm is a careful manipulation of coordinate reference frames, the transformations between them, and the representation of the pointing vector in those frames. Therefore we use (in this document and in the associated code) a notation designed to fully describe each vector, transformation, or angle in question and which frame it is represented in. Below is a summary of the notation; for more detail, see [Kinematics and Dynamics in Moving Coordinate Frames](#). For even more detail, see [Zipfel](#), which uses similar notation and is an excellent reference.

In Documents and Equations

1. We specify **position and attitude** by assigning coordinate frames to the various rigid bodies of interest, such that the origins of the frames define the relative positions of the bodies, and the orientation of the frames define the relative attitudes of the bodies.
2. **Vectors** are denoted by lower-case-bold-face letters.
3. **Matrices** are denoted by upper-case-bold-face letters.
4. Denoting position by r , $\mathbf{r}_{A/B}$ denotes the position of the origin of frame A with respect to frame B . Similarly, $\boldsymbol{\omega}_{A/B}$ denotes the angular velocity of frame A with respect to frame B . The scalar $r_{A/B}$ denotes the magnitude of $\mathbf{r}_{A/B}$. The notation $\hat{\mathbf{r}}_{A/B}$ denotes a unit vector in the direction of $\mathbf{r}_{A/B}$.
5. When necessary to coordinatize the vector quantities, we denote the vector $\mathbf{r}_{A/B}$ coordinatized in frame C as $[\mathbf{r}_{A/B}]_C$.

NOTE: Recall that a vector exists and is defined independent of a specific coordinate system, but to obtain a numerical representation, we need to choose a coordinate frame and calculate the vector's numerical components along the frame's axes. A given vector will have different components depending on the frame in which it is coordinatized.

6. $\mathbf{T}_{B/A}$ denotes a transformation from coordinate frame A to frame B : that is, given a vector coordinatized in frame A , left multiplication by $\mathbf{T}_{B/A}$ produces the same vector coordinatized in frame B :

$$[\mathbf{r}]_B = \mathbf{T}_{B/A} [\mathbf{r}]_A.$$

The inverse transformation is given by the transpose, i.e. $\mathbf{T}_{A/B} = \mathbf{T}_{B/A}^T$.

7. Notice that both vectors and transformations use a **with respect to** convention: $\mathbf{r}_{A/B}$ denotes the position of the origin of coordinate frame A with respect to frame B , and $\mathbf{T}_{A/B}$ denotes the attitude of coordinate frame A with respect to frame B .
8. We construct transformation matrices from [Elementary Transformations and Euler Angles](#). Measuring the transformation angle as positive from the original axes to the new axes, an elementary transformation by ϕ about the x -axis (also called the 1-axis) is given by:

$$\mathbf{T}_1(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix},$$

that is, $\mathbf{T}_1(\phi) \mathbf{r}$ gives vector \mathbf{r} coordinatized in terms of a frame rotated by ϕ about the x -axis of the original frame. Similarly, elementary transformations about the y - and z -axes (the 2- and 3-axes) are given by:

$$\mathbf{T}_2(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \quad \text{and} \quad \mathbf{T}_3(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

In general, a vector can be represented in any other frame by three successive elementary transformations -- applied through successive left multiplication. Such a transformation is called an **Euler-XYZ** transformation, where the numbers XYZ indicate the axes and order of rotation. The conventional aerospace order of rotation is 321, i.e. yaw, pitch, roll. Denoting the yaw, pitch, and roll angles by ψ , θ , and ϕ , respectively, the resulting *Euler-321* transformation is given by:

$$\mathbf{T}_{321}(\phi, \theta, \psi) = \mathbf{T}_1(\phi)\mathbf{T}_2(\theta)\mathbf{T}_3(\psi).$$

For small angles, it can be shown that:

$$\mathbf{T}_{321}(\delta\phi, \delta\theta, \delta\psi) = \begin{bmatrix} 1 & \delta\psi & -\delta\theta \\ -\delta\psi & 1 & \delta\phi \\ \delta\theta & -\delta\phi & 1 \end{bmatrix}.$$

In Code

In code, vector quantities must be coordinatized, i.e. represented by numerical values for each of their components. Therefore, to fully specify a vector, the notation must capture:

- the property represented (position, velocity, force, angular velocity, etc.),
- the object of whose property it is,
- the reference frame the property is measured with respect to, and
- the coordinate frame in which it is coordinatized.

Thus, we use the following notation:

- **Vectors:**

`property_body_reference__coordframe`

Where:

- `property` is the property denoted,
- `body` is the body of which this is the property,
- `reference` is the reference point or frame with respect to which the property is measured, and
- `coordframe` is the coordinate frame in which the vector is coordinatized.

For example: denoting position by r , the position of the target balloon, B , with respect to the pointing gimbal, G , coordinatized in the ECEF frame, E , would be:

$$[\mathbf{r}_{B/G}]_E = \mathbf{r}_{B_G_E}$$

If the name will be used in a wider scope, this might become:

$$[\mathbf{r}_{B/G}]_E = \text{position_balloon_gimbal_ecef}$$

- **Transformations**

$$T_{\text{frameB_frameA}}$$

For example, the orientation of the gimbal mount frame M with respect to the payload frame, P would be:

$$\mathbf{T}_{M/P} = T_{M_P} \text{ or } T_{\text{mount_payload}}$$

- **Euler 321 Vector and Euler Angles**

We denote an Euler 321 vector defining the transformation from frame A to frame B as:

$$[\phi_{B/A} \ \theta_{B/A} \ \psi_{B/A}]^T = \text{e321_B_A}$$

NOTE: We retain the x, y, z order for the angles in the vector, even though by definition, the rotations are applied in the order z, y, x in an Euler-321 transformation.

When we want to refer to the angles in code, we use the notation:

$$\begin{aligned} \phi_{B/A} &= \text{roll_B_A} \\ \theta_{B/A} &= \text{pitch_B_A} \\ \psi_{B/A} &= \text{yaw_B_A}. \end{aligned}$$

- **Latitude, Longitude, and Altitude**

We use GPS for position measurements, and for example, we denote the position of the G frame as follows:

$$\mathbf{l}l_{G/E} = \mathbf{l}l_{a_G_E}$$

NOTE: By definition, GPS latitude, longitude, and altitude give the position of the object with respect to the WGS 84 ECEF or E frame.

Outline of Calibration Algorithm Derivation

With notation defined, we move on to development of the algorithm. In outline, the development is as follows:

1. Define the frames involved.
2. Define the gimbal pointing problem, which is the problem of calculating the azimuth and elevation such that a 2-degree of freedom gimbal points towards its target.
3. Solve the gimbal pointing problem for the cases where the gimbal is perfect (no misalignments) and skewed (misaligned). We need a skewed gimbal problem solution in order to use the calibration, i.e. in order to calculate the azimuth and elevation needed to point a calibrated, but skewed gimbal (i.e. a gimbal with known misalignments) at its target.
4. Finally, we solve the calibration problem, which is the problem of estimating the misalignments from measurements when the gimbal is known to be pointing at its target.

Definition of Coordinate Frames

[Figure 1](#) shows the important frames involved in the B2X pointing calibration. These frames are defined and described below.

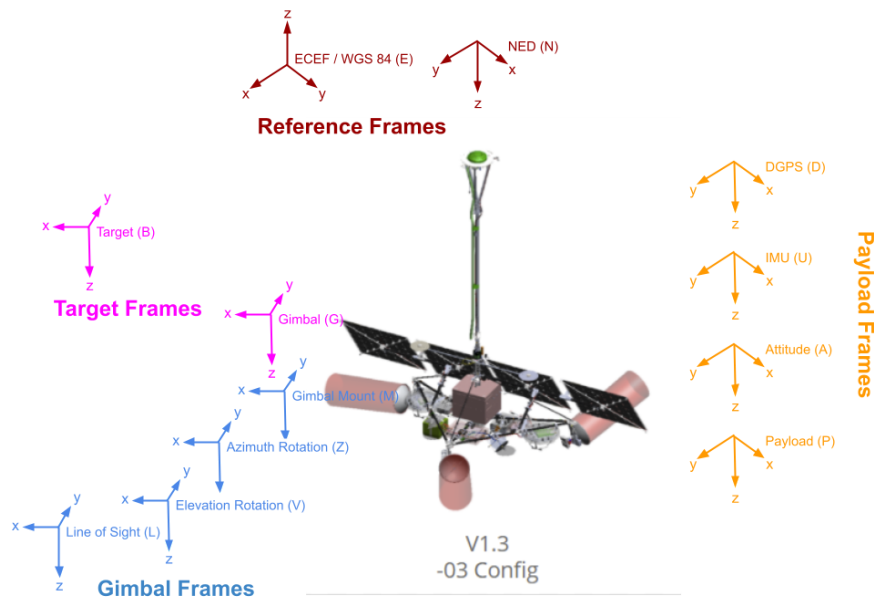


Figure 1: B2X pointing calibration frames.

Summary of Frames and Symbols

The table below summarizes the frames (in order of stack down from reference frames first through gimbal frames to gimbal line-of-sight frame). Detailed definitions of the frames are given below.

Symbol	Acronym	Description
<i>E</i>	ECEF or WGS-84	Earth-Centered-Earth-Fixed WGS-84 frame
<i>N</i>	NED	North-East-Down Local-Tangent-Plane frame
<i>P</i>	P	Payload body-fixed frame
<i>D</i>	DGPS	Primary DGPS antenna frame
<i>U</i>	IMU	IMU frame
<i>A</i>	A	Attitude frame
<i>M</i>	GM	Gimbal mount frame
<i>Z</i>	GZ	Gimbal azimuth rotation frame
<i>Z'</i>	GZP	Rotated gimbal azimuth rotation frame
<i>V</i>	GV	Gimbal elevation rotation frame
<i>V'</i>	GVP	Rotated gimbal elevation rotation frame
<i>L</i>	LOS	Gimbal antenna boresight or line-of-sight frame

Symbol	Acronym	Description
<i>G</i>	G	True gimbal-to-target frame
<i>B</i>	B	Target (other payload/beacon/ground station) frame

Canonical Loon Coordinate Frames

See [Loon Coordinate Frames](#) for further details on the following canonical frames:

- Earth-Centered-Earth-Fixed (ECEF or E) / WGS 84 Frame
- North-East-Down (NED or N) Frame
- Payload (P) Frame

Attitude Measurement Frames

DGPS (D) Frame

- This is the DGPS antenna frame.
- The dual-antenna DGPS receiver measures the yaw and pitch of the D frame with respect to the NED frame.
- The D frame is located at primary DGPS antenna.
- The D frame is nominally aligned with the P frame, but in practice, the D frame has yaw and pitch misalignments relative to the P frame.
- Dual antenna DGPS does not measure roll, therefore there is no roll misalignment.

IMU (U) Frame

- This is the IMU frame.
- The U frame is located at the IMU.
- The U frame is nominally aligned with the P frame, but in practice, the U frame has yaw, pitch, and roll misalignments relative to the P frame.

Attitude (A) Frame

- This is the effective measurement frame of the attitude determination system.
- The attitude determination system measures the yaw, pitch, and roll of the A frame relative to the NED reference frame.
- Depending on the implementation of the attitude sensor-fusion algorithm, the A frame is a merging of the D and U frames.
- The A frame is located at the IMU.
- Thus, the A frame is nominally aligned with the P frame, but in practice, the A frame has yaw, pitch, and roll misalignments relative to the P frame. Assuming that the misalignments are small, the yaw misalignment will be dominated by the D frame yaw misalignment, while the pitch and roll misalignments will be dominated by the U frame pitch and roll misalignments.

Gimbal Frames

Gimbal Mount (M) Frame

- This is the mounting frame of the gimbal.
- The M frame is located at the mounting point of the gimbal.
- Nominally, as-designed, the M frame is aligned with the P frame (with a possible exception of a known nominal yaw relative to the P frame).
- In practice, the M frame has yaw, pitch, and roll misalignments relative to the above nominal alignment.

Gimbal Azimuth Rotation (Z) Frame

- This frame defines the gimbal azimuth rotation axis. The gimbal should rotate about the z -axis of the M frame, but doesn't due to manufacturing misalignments.
- Located at the center of azimuth rotation.
- The gimbal rotates about z -axis of the Z frame by the gimbal azimuth control angle, α_C .

Rotated Gimbal Azimuth Rotation (Z') Frame

- The Z' frame is the frame obtained by rotating the Z frame about its z -axis by the gimbal azimuth control angle, α_C .

Gimbal Elevation Rotation (V) Frame

- This frame defines the gimbal elevation rotation axis. The gimbal should rotate about the y -axis of the Z' frame, but doesn't due to manufacturing misalignments.
- Located at the center of elevation rotation.
- The gimbal rotates about y -axis of the V frame by the gimbal elevation control angle, ϵ_C .

Rotated Gimbal Elevation Rotation (V') Frame

- The V' frame is the frame obtained by rotating the V frame about its y -axis by the gimbal elevation control angle, ϵ_C .

Line-of-Sight (L) Frame

- This is the RF line-of-sight frame.
- The x -axis points in the direction of the RF antenna boresight.
- Ideally the L frame is aligned with the V' frame, but in practice, the L frame has yaw and pitch misalignments relative to the V' frame.

Target Frames

Gimbal (G) Frame

- This is the gimbal target frame. It defines the true line-of-sight between the gimbal and the target.
- The G frame is located at the center of the pointing antenna.
- The G frame is obtained by rotating the Z frame through a *true* gimbal azimuth angle, α and a *true* gimbal elevation angle, ϵ , such that the x -axis of the G frame points to the target, B.
- The gimbal pointing problem is to align the x_L axis with the x_G axis.

Target (Balloon / Beacon / Ground Station) (B) Frame

- This is the target frame, i.e. the target payload or ground station.
- It is located at the center of the target.
- Since we consider the target to be a point source located at the origin of the B frame, we don't care about the orientation of this frame. To avoid leaving its orientation undefined, we choose the B frame to be aligned with the G frame.

Gimbal Pointing Problem

The gimbal pointing problem can be defined as follows:

- Given:
 - measurements of the positions of the pointing antenna and the target antenna in the ECEF reference frame, and
 - measurements of the orientation of the A frame relative to the NED reference frame,
- Calculate:
 - the gimbal azimuth control angle, α_C and the gimbal elevation control angle, ϵ_C ,
- Such that:
 - the pointing antenna line-of-sight is aligned with the true line-of-sight to the target (i.e. such that x_L -axis of the L frame is aligned with the x_G -axis of the G frame).

Solution of Gimbal Pointing Problem

Calculate True Line-of-Sight

Given measurements of the positions of the gimbal (G) and target (B) in latitude, longitude, and altitude, i.e. $\text{lla}_{G/E}$ and $\text{lla}_{B/E}$, calculate the positions in the ECEF frame, i.e. $[\mathbf{r}_{G/E}]_E$ and $[\mathbf{r}_{B/E}]_E$. The true line-of-sight vector in the ECEF frame is then:

$$[\mathbf{r}_{B/G}]_E = [\mathbf{r}_{B/E}]_E - [\mathbf{r}_{G/E}]_E$$

Pointing Equation

Given the line-of-sight vector in the ECEF frame, and using the [definition of a coordinate transformation matrix](#), we can write the following gimbal pointing equation. This equation gives the true line-of-sight vector coordinatized in the gimbal line-of-sight (L) frame:

$$[\mathbf{r}_{B/G}]_L = \mathbf{T}_{LV'} \mathbf{T}_2(\epsilon_C) \mathbf{T}_{V/Z'} \mathbf{T}_3(\alpha_C) \mathbf{T}_{Z/M} \mathbf{T}_{M/A} \mathbf{T}_{A/N} \mathbf{T}_{N/E} [\mathbf{r}_{B/G}]_E \quad (1)$$

In the above equation:

- $[\mathbf{r}_{BG}]_E$ is known from the GPS position measurements of the gimbal and target.
- $\mathbf{T}_{N/E}$ is known from the GPS position of the gimbal.
- $\mathbf{T}_{A/N}$ is known from the attitude measurement.
- $\mathbf{T}_{M/A}$ has two components, a nominal component, which defines the as-designed orientation of the gimbal mount frame relative to the attitude frame, and a misalignment component, which captures the small-angle misalignments of the actual gimbal mount frame, i.e.:

$$\mathbf{T}_{M/A} = \mathbf{T}_{M/M_{nom}} \mathbf{T}_{M_{nom}/A} \quad (2)$$

where:

- $\mathbf{T}_{M_{nom}/A}$ is known from CAD, and
- $\mathbf{T}_{M/M_{nom}}$ is **unknown unless calibrated**.
- $\mathbf{T}_{Z/M}$ is **unknown unless calibrated**.
- $\mathbf{T}_{V/Z'}$ is **unknown unless calibrated**.
- $\mathbf{T}_{L/V'}$ is **unknown unless calibrated**.
- $[\mathbf{r}_{BG}]_L = [1 \ 0 \ 0]^T$ when the gimbal is pointed correctly.
- α_C and ϵ_C are the gimbal azimuth and elevation control angles for which we wish to solve.

To solve the pointing problem we need calculate α_C and ϵ_C , such that such that $[\mathbf{r}_{BG}]_L = [1 \ 0 \ 0]^T$, i.e. such that the x_L -axis of the L frame is aligned with the true line-of-sight.

Solution to the Nominal Gimbal Pointing Problem

In the case where there are no misalignments, equation 1 simplifies to

$$[\mathbf{r}_{BG}]_L = \mathbf{T}_2(\epsilon_C) \mathbf{T}_3(\alpha_C) \mathbf{T}_{M_{nom}/A} \mathbf{T}_{A/N} \mathbf{T}_{N/E} [\mathbf{r}_{BG}]_E \quad (3)$$

which can be rewritten

$$[\mathbf{r}_{BG}]_Z = \mathbf{T}_3(\alpha_N)^T \mathbf{T}_2(\epsilon_N)^T [1 \ 0 \ 0]^T \quad (4)$$

and for which there is a closed-form solution, i.e.

$$\alpha_N = \arctan\left(\frac{[y_{BG}]_Z}{[x_{BG}]_Z}\right) \quad (5)$$

$$\epsilon_N = \arctan\left(\frac{-[z_{BG}]_Z}{([x_{BG}]_Z^2 + [y_{BG}]_Z^2)}\right) \quad (6)$$

Small-Angle Solution to the Skewed Gimbal Pointing Problem

In the case where the misalignments are not zero, but are known (for example from a calibration), we can rearrange equation 1 to give:

$$\mathbf{T}_{Z/M} \mathbf{T}_{M/A} \mathbf{T}_{A/N} \mathbf{T}_{N/E} [\mathbf{r}_{BG}]_E = \mathbf{T}_3(\alpha_C)^T \mathbf{T}_{V/Z'}^T \mathbf{T}_2(\epsilon_C)^T \mathbf{T}_{L/V'}^T [\mathbf{r}_{BG}]_L \quad (7)$$

Multiplying out the left hand side of equation 7, and substituting for $[\mathbf{r}_{BG}]_L$, we have the following equation to be solved for α_C and ϵ_C :

$$[\mathbf{r}_{BG}]_Z = \mathbf{T}_3(\alpha_C)^T \mathbf{T}_{V/Z'}^T \mathbf{T}_2(\epsilon_C)^T \mathbf{T}_{L/V'}^T [1 \ 0 \ 0]^T \quad (8)$$

Let's call this the *skewed gimbal pointing problem*.

Small Angle Simplification

Noting that $\mathbf{T}_{V/Z'}^T$ and $\mathbf{T}_{L/V'}^T$ are small angle transformations (resulting from misalignments in manufacture), the skewed gimbal pointing problem can be solved as follows.

Make the substitutions:

$$\begin{aligned}\alpha_C &= \alpha_N + \delta\alpha \\ \epsilon_C &= \epsilon_N + \delta\epsilon \\ \mathbf{T}_{V|Z'} &= \begin{bmatrix} 1 & \delta\psi_{V|Z'} & -\delta\theta_{V|Z'} \\ -\delta\psi_{V|Z'} & 1 & \delta\phi_{V|Z'} \\ \delta\theta_{V|Z'} & -\delta\phi_{V|Z'} & 1 \end{bmatrix} \\ \mathbf{T}_{L|V'} &= \begin{bmatrix} 1 & \delta\psi_{L|V'} & -\delta\theta_{L|V'} \\ -\delta\psi_{L|V'} & 1 & \delta\phi_{L|V'} \\ \delta\theta_{L|V'} & -\delta\phi_{L|V'} & 1 \end{bmatrix}\end{aligned}$$

where:

- $\delta\alpha$ and $\delta\epsilon$ are small angles,
- $\delta\psi_{V|Z'}$, $\delta\theta_{V|Z'}$, $\delta\phi_{V|Z'}$ are the (small) yaw, pitch, and roll of the V frame relative to the Z' frame,
- $\delta\psi_{L|V'}$, $\delta\theta_{L|V'}$, $\delta\phi_{L|V'}$ are the (small) yaw, pitch, and roll of the L frame relative to the V' frame,

and we have used the [small angle approximation to the Euler-321 yaw, pitch, and roll transformation](#).

Substituting into into equation 8, multiplying out, and dropping higher-order terms of small angles, we obtain ([see this Colab notebook](#)):

$$[\mathbf{r}_{BG}]_Z = \begin{bmatrix} \begin{pmatrix} -\delta\alpha \sin(\alpha_N) \cos(\epsilon_N) - \delta\epsilon \sin(\epsilon_N) \cos(\alpha_N) \\ -\delta\phi_{V|Z'} \sin(\alpha_N) \sin(\epsilon_N) - \delta\psi_{L|V'} \sin(\alpha_N) - \delta\psi_{V|Z'} \sin(\alpha_N) \cos(\epsilon_N) \\ -\delta\theta_{L|V'} \sin(\epsilon_N) \cos(\alpha_N) - \delta\theta_{V|Z'} \sin(\epsilon_N) \cos(\alpha_N) + \cos(\alpha_N) \cos(\epsilon_N) \end{pmatrix} \\ \begin{pmatrix} \delta\alpha \cos(\alpha_N) \cos(\epsilon_N) - \delta\epsilon \sin(\alpha_N) \sin(\epsilon_N) \\ +\delta\phi_{V|Z'} \sin(\epsilon_N) \cos(\alpha_N) + \delta\psi_{L|V'} \cos(\alpha_N) + \delta\psi_{V|Z'} \cos(\alpha_N) \cos(\epsilon_N) \\ -\delta\theta_{L|V'} \sin(\alpha_N) \sin(\epsilon_N) - \delta\theta_{V|Z'} \sin(\alpha_N) \sin(\epsilon_N) + \sin(\alpha_N) \cos(\epsilon_N) \end{pmatrix} \\ \begin{pmatrix} -\delta\epsilon \cos(\epsilon_N) \\ -\delta\theta_{L|V'} \cos(\epsilon_N) - \delta\theta_{V|Z'} \cos(\epsilon_N) - \sin(\epsilon_N) \end{pmatrix} \end{bmatrix} \quad (9)$$

Equation 9 is an overdetermined system of 3 equations in 2 unknowns, $\delta\alpha$ and $\delta\epsilon$. We can thus solve for $\delta\alpha$ and $\delta\epsilon$ from which we then obtain the desired control angles:

$$\begin{aligned}\alpha_C &= \alpha_N + \delta\alpha \\ \epsilon_C &= \epsilon_N + \delta\epsilon\end{aligned}$$

Handling Zone Flips

Note, this same solution can be used in the case of a zone-flipped gimbal by inserting the zone-flipped equivalents of the nominal azimuth and elevation angles:

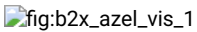
$$\alpha_{N_{zf}} = \pi + \alpha_N \quad (10)$$

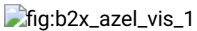
$$\epsilon_{N_{zf}} = \pi - \epsilon_N \quad (11)$$

into equation 9 and solving for the zone-flipped correction angles, $\delta\alpha_{zf}$ and $\delta\epsilon_{zf}$.

Matlab Simulation and Verification

The above solution was verified using a [Matlab simulation](#). The pointing error histogram below shows the improvement in pointing error when using the small-angle-skewed-gimbal solution instead of the nominal azimuth and elevation angles, for the case where the misalignment angles are drawn from Gaussian normal distributions with mean 0 and standard deviation 2 degrees.

fig:b2x_azel_vis_1

fig:b2x_azel_vis_1

Calibration Problem

The calibration problem is the inverse of the gimbal pointing problem. Specifically, given a set of measurements where we know that the gimbal is tracking, i.e. where we know that the gimbal pointing equation (equation 1):

$$[\mathbf{r}_{BG}]_L = \mathbf{T}_{L|V'} \mathbf{T}_2(\epsilon_C) \mathbf{T}_{V|Z'} \mathbf{T}_3(\alpha_C) \mathbf{T}_{Z|M} \mathbf{T}_{M|A} \mathbf{T}_{A|N} \mathbf{T}_{N|E} [\mathbf{r}_{BG}]_E \quad (12)$$

is satisfied, we attempt to estimate the unknown misalignments. That is:

1. Given:

- measurements of the positions of the pointing antenna and the target antenna in the ECEF reference frame,
 - measurements of the orientation of the A frame relative to the NED reference frame,
 - knowledge of the gimbal azimuth control angle, α_C and the gimbal elevation control angle, ϵ_C , required to track the target.
2. Attempt to calibrate the small misalignment angles which define the misalignment transformations: \mathbf{T}_{ZIM} , $\mathbf{T}_{M/A}$, $\mathbf{T}_{VIZ'}$, and $\mathbf{T}_{LIV'}$.

Solution of the Calibration Problem

Consider equation 12, and assume we have measurements at time instants when the gimbal is known to be accurately tracking its target. In this case:

- $[\mathbf{r}_{BIG}]_E$, $\mathbf{T}_{N/E}$, $\mathbf{T}_{A/N}$, are known from measurements.
- α_C and ϵ_C are known from the gimbal control algorithm.
- $[\mathbf{r}_{BIG}]_L = [1\ 0\ 0]^T$ because the gimbal is tracking.
- The misalignment transformations (\mathbf{T}_{ZIM} , $\mathbf{T}_{M/A}$, $\mathbf{T}_{VIZ'}$, and $\mathbf{T}_{LIV'}$) contain 12 unknown angles.
- However, any small errors in $\mathbf{T}_{M/A}$ can be subsumed into \mathbf{T}_{ZIM} . Thus,

$$\mathbf{T}_{M/A} = \mathbf{T}_{M/A}(0, 0, \psi_{Mnom/A}) = \mathbf{T}_{Mnom/A}$$

is known from the payload design (recall from equation 2 that $\mathbf{T}_{M/A} = \mathbf{T}_{M/Mnom} \mathbf{T}_{Mnom/A}$).

Therefore we are reduced to 9 unknown angles – specifically those defining the misalignment transformations \mathbf{T}_{ZIM} , $\mathbf{T}_{VIZ'}$, and $\mathbf{T}_{LIV'}$, and using the notation

$$\mathbf{T}_{B/A} = \mathbf{T}_{B/A}(\phi, \theta, \psi),$$

we rewrite equation 12 as

$$[\mathbf{r}_{BIG}]_L = \mathbf{T}_{LIV'}(\check{\phi}, \check{\theta}, \check{\psi})\mathbf{T}_2(\epsilon_C)\mathbf{T}_{VIZ'}(\check{\phi}, \check{\theta}, \check{\psi})\mathbf{T}_3(\alpha_C)\mathbf{T}_{ZIM}(\check{\phi}, \check{\theta}, \check{\psi})\mathbf{T}_{M/A}\mathbf{T}_{A/N}\mathbf{T}_{N/E}[\mathbf{r}_{BIG}]_E \quad (13)$$

where the $\check{\sim}$ symbol indicates angles which are small and unknown. These are the angles we wish to calibrate.

Now, we can rearrange and simplify 13 to obtain:

$$\mathbf{T}_{LIV'}(\check{\phi}, \check{\theta}, \check{\psi})\mathbf{T}_2(\epsilon_C)\mathbf{T}_{VIZ'}(\check{\phi}, \check{\theta}, \check{\psi})\mathbf{T}_3(\alpha_C)\mathbf{T}_{ZIM}(\check{\phi}, \check{\theta}, \check{\psi})[\mathbf{r}_{BIG}]_M = [\mathbf{r}_{BIG}]_L \quad (14)$$

where the unknowns are $\check{\phi}_{ZIM}$, $\check{\theta}_{ZIM}$, $\check{\psi}_{ZIM}$, $\check{\phi}_{VIZ'}$, $\check{\theta}_{VIZ'}$, $\check{\psi}_{VIZ'}$, $\check{\phi}_{LIV'}$, $\check{\theta}_{LIV'}$ and $\check{\psi}_{LIV'}$.

At this point we can replace \mathbf{T}_{ZIM} , $\mathbf{T}_{VIZ'}$, and $\mathbf{T}_{LIV'}$ by [their small angle transformation matrix equivalents](#), e.g.:

$$\mathbf{T}_{ZIM} = \begin{bmatrix} 1 & \delta\psi_{ZIM} & -\delta\theta_{ZIM} \\ -\delta\psi_{ZIM} & 1 & \delta\phi_{ZIM} \\ \delta\theta_{ZIM} & -\delta\phi_{ZIM} & 1 \end{bmatrix}$$

etc., and multiply out to obtain an equation of the form:

$$\mathbf{H}_k \mathbf{x} = \mathbf{y}_k \quad (15)$$

where \mathbf{x} is the vector of unknown misalignment angles, i.e. the parameters we wish to calibrate:

$$\mathbf{x} = \begin{bmatrix} \check{\phi}_{ZIM} \\ \check{\theta}_{ZIM} \\ \check{\psi}_{ZIM} \\ \check{\phi}_{VIZ'} \\ \check{\theta}_{VIZ'} \\ \check{\psi}_{VIZ'} \\ \check{\phi}_{LIV'} \\ \check{\theta}_{LIV'} \\ \check{\psi}_{LIV'} \end{bmatrix} \quad (16)$$

[Here is a Colab](#) which uses SymPy to calculate the matrices \mathbf{H}_k and \mathbf{y}_k in equation 15.

Batch Least Squares Implementation

Equation 15 is a vector equation which holds for each measurement sample time, k . One way to use this to solve the calibration problem is to concatenate multiple measurements into a large over-determined set of equations, and use Batch Ordinary Least Squares to solve for the misalignment angles. Thus, the concatenated measurement equation is:

$$\begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \vdots \\ \mathbf{H}_n \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_n \end{bmatrix} \quad (17)$$

which has the well-known least-squares solution:

$$\hat{\mathbf{x}} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{y}$$

A batch least-squares version of the calibration algorithm has been implemented in Python:

 and has been verified to work (i.e. calibrated gimbals have significantly improved pointing accuracy) – see [go/gimbal-inflight-cal-perf](https://github.com/Space-Force/gimbal-inflight-cal-perf).

Recursive Least Squares Implementation

Another alternative is to implement an online continuous calibration using a recursive least squares estimator. This is equivalent to formulating the problem as a Kalman filter for a system with stationary dynamics:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k && \text{state equation} \\ \mathbf{y}_k &= \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k && \text{measurement equation} \end{aligned}$$

where:

- \mathbf{x}_k is the state to be estimated (in least-squares terms, the parameter vector),
- \mathbf{H}_k is the measurement matrix (in least-squares terms, the regressor matrix),
- \mathbf{y}_k is the measurement vector, and
- \mathbf{v}_k is the measurement noise.

For an ordinary (unweighted) least-squares solution we assume that measurement noise is Gaussian with unit variance, in which case the Kalman filter and recursive least squares solutions are identical, and given by:¹

$$\begin{aligned} \hat{\mathbf{x}}_{k+1} &= \hat{\mathbf{x}}_k + \mathbf{K}_p (\mathbf{y}_k - \mathbf{H}_k \hat{\mathbf{x}}_k) \\ \mathbf{K}_p &= \hat{\mathbf{P}}_k \mathbf{H}_k^T \mathbf{R}_e^{-1} \\ \mathbf{R}_e &= \mathbf{I} + \mathbf{H}_k \hat{\mathbf{P}}_k \mathbf{H}_k^T \\ \hat{\mathbf{P}}_{k+1} &= \hat{\mathbf{P}}_k - \mathbf{K}_p \mathbf{H}_k \hat{\mathbf{P}}_k \end{aligned}$$

For exponentially-weighted least squares, we can introduce an exponential decay factor, λ :

$$\begin{aligned} \hat{\mathbf{x}}_{k+1} &= \hat{\mathbf{x}}_k + \mathbf{K}_p (\mathbf{y}_k - \mathbf{H}_k \hat{\mathbf{x}}_k) \\ \mathbf{K}_p &= \lambda^{-1} \hat{\mathbf{P}}_k \mathbf{H}_k^T \mathbf{R}_e^{-1} \\ \mathbf{R}_e &= \mathbf{I} + \lambda^{-1} \mathbf{H}_k \hat{\mathbf{P}}_k \mathbf{H}_k^T \\ \hat{\mathbf{P}}_{k+1} &= \lambda^{-1} (\hat{\mathbf{P}}_k - \mathbf{K}_p \mathbf{H}_k \hat{\mathbf{P}}_k) \end{aligned}$$

Observability and Selection of Parameters to Calibrate

Equation 14 contains 9 unknown parameters. However, not all of these parameters are independently observable, because some have similar effects on the pointing vector. Specifically:

- The two small azimuth angles $\check{\psi}_{ZIM}$ and $\check{\psi}_{VIZ'}$ have similar effects and can be calibrated together as $\check{\psi}_{VIZ'}$.
- The two small azimuth angles $\check{\theta}_{VIZ'}$ and $\check{\theta}_{LIV'}$ have similar effects and can be calibrated together as $\check{\theta}_{LIV'}$.
- $\check{\phi}_{LIV'}$ has no effect on the direction of the pointing vector and can be ignored.

Setting angles which we choose not to estimate to zero, the pointing equation (equation 14) reduces to:

$$\mathbf{T}_{LIV'}(0, \check{\theta}, \check{\psi}) \mathbf{T}_2(\epsilon_C) \mathbf{T}_{VIZ'}(\check{\phi}, 0, \check{\psi}) \mathbf{T}_3(\alpha_C) \mathbf{T}_{ZIM}(\check{\phi}, \check{\theta}, 0) [\mathbf{r}_{BG}]_M = [\mathbf{r}_{BG}]_L \quad (18)$$

and the parameter vector we choose to calibrate becomes:

$$\mathbf{x} = \begin{bmatrix} \check{\phi}_{ZIM} \\ \check{\theta}_{ZIM} \\ \check{\phi}_{VIZ'} \\ \check{\psi}_{VIZ'} \\ \check{\theta}_{LIV'} \\ \check{\psi}_{LIV'} \end{bmatrix} \tag{19}$$

Table of Symbols and Names

The Loon Mission Control implementation of the algorithm described above touches many aspects of the Loon system – Avionics flight code, Avionic flight parameters, telemetry reports, and the MC estimator itself. This section summarizes the correspondence between the symbols used above and the names, parameters, and telemetry reports used in the various code bases.

Position of the pointing gimbal

Symbol	Name in Code	Description	Units
$lla_{G/E}$	lla_G_E	geodetic [latitude, longitude, altitude] of pointing gimbal	[deg, deg, m]

- Telemetry reports:

```
lat_G_E: report.orient.dgps.position_packed
lon_G_E: report.orient.dgps.position_packed
alt_G_E: report.orient.dgps.elevation
```

Position of target

Symbol	Name in Code	Description	Units
$lla_{B/E}$	lla_B_E	geodetic [latitude, longitude, altitude] of target	[deg, deg, m]

- Telemetry reports (note: these are in radians and must be converted to degrees):

```
lat_B_E: report.gimbal*.pointing.target_position.latitude
lon_B_E: report.gimbal*.pointing.target_position.longitude
alt_B_E: report.gimbal*.pointing.target_position.elevation
```

Payload Attitude

Symbol	Name in Code	Description	Units
n/a	e321_A_N	vector of Euler-321 angles in the order [roll, pitch, yaw]; defines the measured attitude of the payload	[rad, rad, rad]

- Telemetry reports:

```
roll_A_N: report.pointing.est_attitude.roll
pitch_A_N: report.pointing.est_attitude.pitch
yaw_A_N: report.pointing.est_attitude.yaw
```

Nominal yaw of gimbal mount frame w.r.t attitude frame

Symbol	Name in Code	Description	Units
$\psi_{M/M_{nom}}$	yaw_Mnom_A	nominal yaw of the gimbal mount w.r.t. the attitude frame	[rad]

- This is a mechanical angle determined by the payload design. See [redacted]

- Definition in code:

```
yaw_Mnom_A: defined in B2xCalParams proto as a per gimbal parameter.
```

Actual as-found azimuth and elevation of target

Symbol	Name in Code	Description	Units
α_C	az_B_Z	actual azimuth of target as reported by gimbal control algorithm	[rad]
ϵ_C	el_B_V	actual elevation of target as reported by gimbal control algorithm	[rad]

NOTE There is some subtlety here, because the azimuth and elevation must be extracted from the Avionics commanded gimbal angles, which use different reference frames for the angles, and which are dependent on the mechanical zone in which the gimbal is operating.

- In zone 0:

```
az_B_Z = -gimbal.pointing().gimbal_angles_cmd().azimuth() - M_PI / 2.0
el_B_V = gimbal.pointing().gimbal_angles_cmd().elevation() - M_PI / 2.0
```

- In zone 1:

NOTE The expressions below produce the zone-flipped equivalents α_{N_zf} and ϵ_{N_zf} defined in equation 11.

```
...
az_B_Z = -gimbal.pointing().gimbal_angles_cmd().azimuth() + 3.0 * M_PI / 2.0
el_B_V = gimbal.pointing().gimbal_angles_cmd().elevation() + 3.0 * M_PI / 2.0
...
```

- To determine the zone:

```
report.gimbal*.pointing.zone
```

Misalignment Angles

Calibration angles

Index	Symbol	Name in Code	Description	Units
0	$\check{\phi}_{Z/M}$	roll_Z_M	roll of frame Z relative to frame M	[rad]
1	$\check{\theta}_{Z/M}$	pitch_Z_M	pitch of frame Z relative to frame M	[rad]
2	$\check{\psi}_{Z/M}$	yaw_Z_M	yaw of frame Z relative to frame M	[rad]
3	$\check{\phi}_{V/Z'}$	roll_V_Zp	roll of frame V relative to frame Zp	[rad]
4	$\check{\theta}_{V/Z'}$	pitch_V_Zp	pitch of frame V relative to frame Zp	[rad]
5	$\check{\psi}_{V/Z'}$	yaw_V_Zp	yaw of frame V relative to frame Zp	[rad]
6	$\check{\phi}_{L/V'}$	roll_L_Vp	roll of frame L relative to frame Vp	[rad]

Index	Symbol	Name in Code	Description	Units
7	$\tilde{\theta}_{LV'}$	pitch_L_Vp	pitch of frame L relative to frame Vp	[rad]
8	$\tilde{\psi}_{LV'}$	yaw_L_Vp	yaw of frame L relative to frame Vp	[rad]

Correspondence between misalignment angles and Avionics gimbal parameters

Four parameter calibration

Currently we are using a 4 parameter calibration, because the Avionics code only supports the 4 parameters listed below. We calibrate and set the following 4 parameters (indices refer to the index in the above table).

```
iii_est = [0, 1, 5, 7]
```

Calibration Angle	Corresponding Avionics Parameter	Sign
roll_Z_M	gimbal_az_in_mount_angle.roll	same sign
pitch_Z_M	gimbal_az_in_mount_angle.pitch	same sign
yaw_V_Zp	hardstop_offset_angle.azimuth	same sign
-pitch_L_Vp	hardstop_offset_angle.elevation	note opposite sign

Six parameter calibration

We would like to switch to a 6 parameter calibration which calibrates the 6 parameters identified as [observable](#). In that case, we envisage calibrating the following 6 parameters (again indices refer to the index in the above table).

```
iii_est = [0, 1, 3, 5, 7, 8]
```

Calibration Angle	Corresponding Avionics Parameter	Sign
roll_Z_M	gimbal_az_in_mount_angle.roll	same sign
pitch_Z_M	gimbal_az_in_mount_angle.pitch	same sign
roll_V_Zp	N/A (no parameter exists or planned)	same sign
yaw_V_Zp	hardstop_offset_angle.azimuth	same sign
pitch_L_Vp	rf_line_of_sight_offset_angle.elevation	same sign
yaw_L_Vp	rf_line_of_sight_offset_angle.azimuth	same sign

-
1. See e.g. Section 2.6 of '[Linear Estimation](#)' by Kailath, Sayed, and Hassibi: ↩