### OLI Line-of-Sight Model Correction Algorithm

#### Background/Introduction

The LOS model correction algorithm uses the results of GCP measurements in an image that was systematically and terrain corrected using the original LOS model to derive estimates for corrections to the model. Corrections to the spacecraft attitude and ephemeris are computed in a least squares procedure, which minimizes the differences between the measured locations of the control points and their true ground locations. This procedure includes the detection and removal of outlier GCPs, using a modified t-distribution test. These corrections are added to the LOS model structure to create a “precision” model for subsequent use by other geometric algorithms. Creating the precision model involves repeating some of the processing originally performed by the LOS model creation algorithm to incorporate the model corrections. Once the precision model corrections are computed, the algorithm performs simple threshold tests (e.g., on the pre-fit and post-fit Root-Mean Squared (RMS) GCP residuals and the percentage of GCPs declared outliers) to determine if the solution was successful. If the solution is not successful, the LOS model is not updated with the corrections.

The OLI LOS model correction algorithm is derived from the ALI precision correction algorithm used in ALIAS. Its implementation should be very similar to the aliprecision application. As with the LOS model creation algorithm, model correction makes extensive use of the ALIAS geometric sensor (axx) and spacecraft (exx) model libraries.

#### Dependencies

The LOS Model Correction algorithm assumes that GCPs exist for the ground site and that the Model Creation, LOS Projection and Gridding, and Image Resampling algorithms have been executed to create a systematic terrain-corrected image for GCP mensuration. Note that the band selection and resolution of this mensuration image will depend on the flow being executed/control source being used. For standard L1T product generation, the GLS control (SWIR1 band, 30m resolution) will be used, whereas for characterization and calibration flows, the DOQ control (panchromatic 15m) will be used. It further assumes that the GCP Correlation algorithm/utility has been executed to measure the GCP locations in the mensuration image. The mensuration image may be either SCA-separated or SCA-combined, though SCA-combined images will be the preferred mode of operation.

#### Inputs

The LOS Model Correction algorithm uses the inputs listed in the following table. Note that some of these “inputs” are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data; including data set IDs to provide unique identifiers for data trending).

|  |
| --- |
| **Algorithm Inputs** |
| ODL File (implementation) |
| Measured GCP file name |
| OLI LOS model file name |
| OLI grid file name |
| DEM file name |
| CPF file name |
| L1G image file name |
| Precision solution parameters: |
| Apriori weights for attitude correction parameters (in microradians and microradians/second) |
| Apriori weights for ephemeris correction parameters (in meters and meters/second) |
| Correction model parameterization options (att\_orb, eph\_yaw, both, weight - default is "both") |
| Bias correction or rate of change correction option (time flag) |
| Apriori weights for GCP measurements (in at-sensor microradians) |
| Iteration limit |
| Outlier threshold |
| Processing Options (implementation): |
| Residual Trending On/Off Switch (new) |
| Solution/Alignment Trending On/Off Switch (new) |
| L0Rp ID (for trending) |
| Work Order ID (for trending) |
| Measured GCP File Contents (see GCP Correlation ADD for additional details) |
| GCP image positions |
| GCP ground coordinates |
| OLI Grid File Contents (see LOS Projection ADD for additional details) |
| Arrays of Input/Output Mappings |
| Output Image Frame (e.g., corners, map projection) |
| OLI LOS Model Contents (see LOS Model Creation ADD (6.2.1) for additional details) |
| WRS Path/Row |
| Number of input image (L1R) lines |
| Smoothed image time codes |
| Integration Time (pan and multispectral bands) (new) |
| Smoothed ephemeris at 1 second intervals |
| Earth orientation parameters (UT1UTC, pole wander) |
| OLI to ACS reference alignment matrix/quaternion |
| Spacecraft CM to OLI offset in ACS reference frame (new) |
| Focal plane model parameters (number of SCAs, number detectors/band, Legendre coefficients) |
| Detector offset table (including detector deselect offsets) (new) |
| CPF File Contents |
| Pre-fit RMS threshold |
| Post-fit RMS threshold |
| Percent outlier threshold |
| Minimum number of valid GCPs threshold |
| L1G Image File Contents |
| L1G Metadata |
| DEM File Contents (see Maturity items 5 and 6) |
| DEM Metadata (new) |
| Elevation Data (new) |

#### Outputs

|  |
| --- |
| Precision LOS Model (only items that are updated from the input LOS model are listed below) |
| Updated corrected ephemeris at 1 second intervals |
| Updated corrected attitude data sequence |
| Precision correction reference date/time |
| Precision attitude and ephemeris corrections |
| LOS Model Correction Solution File (see Table 6‑6 below for additional details) |
| LOS model correction reference date/time |
| Final iteration precision correction values |
| Final iteration precision correction covariance |
| LOS Model Correction Residuals File (see Table 6‑7 below for additional details) |
| GCP residuals for each point for each iteration |
| Correction Solution and Alignment Trending Data (new) (see Table 6‑8 below for additional details) |
| Precision correction reference date/time |
| Precision attitude/ephemeris correction values (see note 1) |
| Reduced precision correction covariance (see note 2) |
| Solution quality metrics (see note 4) |
| Control type used (GLS or DOQ) |
| Off-nadir angle (in degrees) |
| L0Rp ID |
| Work Order ID |
| WRS Path/Row |
| Correction Residuals Trending Data (see note 3) (new) (see Table 6‑9 below for additional details) |
| WRS Path/Row |
| GCP ID |
| GCP Type (GLS or DOQ) |
| Date/Time of imaging |
| Spacecraft position/velocity at image time |
| GCP ground coordinates (lat,lon,height) |
| Apparent GCP position (lat, lon, height) in mensuration image |
| LOS Model Correction Success/Failure Status Return (new) |

#### Options

Solution/Alignment Trending On/Off Switch

Residual Trending On/Off Switch

#### Procedure

The LOS correction procedure uses the GCP measurements collected by the GCP Correlation algorithm to estimate updates to the spacecraft attitude and ephemeris data, which minimize the discrepancies between the actual (known) GCP locations and the apparent locations measured in the terrain-corrected L1G image. The solution method adopted for OLI is essentially the same as that used for Landsat 7 and for the ALI, wherein "truth" and "observed" LOS vectors are constructed in the orbital coordinate system, and a weighted least-squares solution is used to minimize the misalignments between the truth and observed vectors. The solution supports the estimation of offset and rate corrections for all three ephemeris position axes and for all three attitude angles (roll-pitch-yaw), though options are provided to reduce the number of parameters (e.g., solve for offsets only) to accommodate situations where the GCPs are few in number, poorly distributed, or inaccurate.

There are several differences in the implementation of the OLI LOS correction model as compared to the previous missions. The first is a change in the coordinate system in which the corrections are applied. For Landsat 7 and ALI data, the precision corrections were applied in the orbital coordinate system. For OLI, they are applied in the spacecraft body/attitude control system coordinate system (this was also noted in the Ancillary Data Preprocessing ADD (6.1.4)). For nadir-viewing scenes, there is little difference, but the case of off-nadir viewing leads to a few adjustments to the heritage algorithm in what follows.

The second significant difference is in the way that the corrections are reflected in the precision LOS model created as an output by this procedure. In the heritage implementation, the ephemeris corrections were used to update the model ephemeris data sequence, but the attitude corrections were stored as a separate correction model that was applied explicitly in the forward model. For OLI, corrected versions of both the ephemeris and attitude data sequences are computed using the LOS correction solution results. These corrected data are stored in the LOS model, along with the original ephemeris and attitude values. The parameters of the correction model are also included in the model, though they are there primarily for documentation purposes and are no longer used in the forward model computation.

The third difference is the inclusion of a portion of the sensor alignment calibration logic into the LOS correction algorithm. This logic uses the OLI to ACS alignment matrix stored in the OLI LOS model to convert the computed attitude offset corrections to OLI alignment angles. This yields updated estimates of the OLI to ACS alignment angles that are output to the characterization database for subsequent trending in the sensor alignment calibration procedure.

A fourth difference is the use of L1G terrain but not precision-corrected images to measure the control points (see the GCP Correlation ADD). This gives the apparent (measured) GCP location a non-zero height coordinate. The true GCP elevation (from the known GCP ground location) could be used, but it is more correct to interpolate the apparent point height from the DEM used to create the terrain-corrected L1G mensuration image. The use of terrain-corrected mensuration images also allows these images to be SCA-combined since the SCA overlap areas will be geometrically consistent.

The mathematical underpinnings of the LOS correction algorithm are presented first, followed by an overview of the procedure for implementing the algorithm.

*Mathematical Development*

The following subsections present the mathematical background of the LOS correction algorithm. In what follows, the equations presented are numbered so that they can be more easily referenced in the subsequent mathematical formulation and in the algorithm procedure sections.

1. Formulating the Observations

The geometric measurement in the OLI sensor system can be regarded as the look vector, ***l****sc*, in the spacecraft body-fixed system. This vector is transformed into the Orbit Reference Frame (OB) system (see Figure 6‑29), as described in the Ancillary Data Preprocessing ADD (6.1.4), through the spacecraft attitude parameters:

***l****ob* = **T**T(r, p, y) ***l****sc*. (1.1)

where r, p, and y are roll, pitch, and yaw angles, **T** is the transformation matrix, and can be expressed as

**T** (r, p, y) = **R**3(y)**R**2(p)**R**1(r)

**= **

**= ** (1.2)

where **R**1, **R**2, and **R**3 are the coordinate system transformation matrix for rotation around the x, y, and z-axis, respectively.

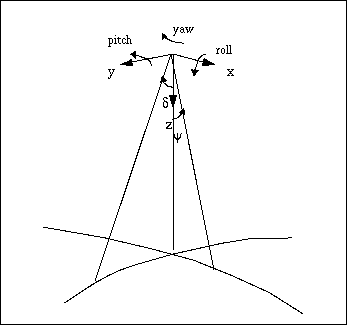


Figure 6‑29. Definition of Orbit Reference System

The vector ***l****ob* is further transformed into the ECF system

***l****ef* = **T**f(**r**ef, **v**ef) ***l****ob* (1.3)

where **T**f is the forward transformation for vectors from the OB system to the ECF system, as a function of the satellite position **r**ef and velocity **v**ef vectors in the ECF system. Note that **v**ef should be the "inertial" velocity expressed in the ECF system, as described in the Ancillary Data Preprocessing ADD (6.1.4). Vector ***l****ef*, together with the satellite position vector, **r**ef, is then used to intersect the ellipsoid Earth surface to pin down a point position, **R**ef, as the target point on the Earth. This is the common forward image pixel geolocation calculation (forward model). Note that when using a terrain-corrected L1G mensuration image, the **R**ef point represents the intersection of the LOS with the DEM used to create the L1G image, rather than the Earth ellipsoid surface. Thus, the target point will have a non-zero height coordinate.

Mathematically, **R**ef is a function of ***l****sc*, r, p,y, **r**ef, and **v**ef.

**R**ef = *F*(***l****sc*, r, p,y, **r**ef, **v**ef )(1.4)

Because of errors in the satellite orbit ephemeris and attitude data, this calculated **R**ef is different from the true location of the image pixel. If we know the true location of a landmark pixel (**R**cp) from other sources (i.e., base map, survey), this point can be taken as a GCP to check the accuracy of the computed image pixel location. The precision correction process uses the GCP coordinates to estimate the correction to the satellite ephemeris and attitude data, so that with the corrected parameters in equation (1.4), the calculated image pixel location, **R**ef, will be close to its true location, **R**cp (depending on the GCP positional accuracy).

To calculate the precision correction, the difference between **R**ef and **R**cp is taken as the observable, and the observation equation becomes:

d**R = R**cp **-** *F*(***l****sc*, r, p,y, **r**ef, **v**ef )(1.5)

according to equation (1.4). However, the actual calculation of **R**ef is usually not an explicit function of the orbit and attitude parameters, especially for the intersecting procedure. Therefore, it is inconvenient to linearize equation (1.5) with standard estimation techniques. Instead, the calculation of look vector ***l****cp* corresponding to **R**cp, in the OB system, is much more explicit:

 (1.6)

where (**R**cp - **r**ef) is the LOS vector in the ECF system corresponding to **R**cp, and **T**i(**r**ef, **v**ef) is the inverse transformation for the look vector from the ECF system to the OB system. If all of the satellite attitude and ephemeris parameters are accurate, the ***l****cp* from equation (1.6) and ***l****ob* from equation (1.1) should be equal. Since the measurement ***l****sc* is accurate compared to the attitude and ephemeris information, any systematic difference between ***l****cp* and ***l****ob* can be attributed to the attitude and orbit errors. Thus, we can use the difference between ***l****cp* and ***l****ob* as the observable.

d**l**  = ***l****cp -* ***l****ob* = **T**i(**r**ef, **v**ef) - **T**(r, p, y) ***l****sc* (1.7)

The task of precision modeling is then to calculate the correction to those satellite ephemeris and attitude parameters (i.e., **r**ef, **v**ef and 's) so that the residuals of d**l** after correction are minimized for all selected GCPs. The orbit correction is modeled as a linear function of time for each component in the OB system. Referred to as the short arc method, this purely geometric method shifts and rotates the short arc of orbit defined by the original ephemeris points to fit the GCP measurements.

2. Linearizing the Observations

These observation equations can be linearized with the following steps. In equation (1.7), the calculation of ***l****ob* can also be carried out through the following:

***l****ob* = Ti(**r**ef, **v**ef)(**R**ef - **r**ef) / | **R**ef - **r**ef | (2.1)

if **R**ef is more conveniently accessible. Since equation (2.1) is simply the inverse of equation (1.4) and equation (1.3), the ***l****ob* calculated from equation (2.1) is the same as the one in equation (1.1), except for the possible inclusion of numerical errors. However, it should be mentioned that the true relationship between ***l****ob* and the parameters is always equation (1.1). Equation (2.1) should not be confused with this, because **R**ef in equation (2.1) is not an independent variable, but a function of equation (1.4). Therefore, in observation (1.7), information about the attitude parameters is contained in ***l****ob* and the information about orbit parameters comes from ***l****cp*.

Since the measurement of ***l****sc* is two-dimensional in nature, only two-dimensional information is contained in equation (1.7), although there are three components involved. If a look vector (either ***l****cp* or ***l****ob*) has the three components in the OB system.

**l** = {*xl, yl, zl*} (2.2)

The real information in these three components can be summarized in two variables like the original look angle measurements. We chose the following two variables:

atan (*yl / zl* ) (2.3)

atan (*xl / zl* ) (2.4)

So that the three components of equation (1.7) can be reduced to the two equations:

cp  ob (2.5)

cpob (2.6)

Note that in equation (2.3) and (2.4), the components of *xl, yl,* and *zl* can be that of LOS vector instead of unit look vector, so that cp and cp are explicit functions of orbit position. In that case, *zl* is approximately the height of the satellite.

If we define,

true value = approximate value + correction

and differentiate equations (2.3) and (2.4) with respect to the orbit position (for cp and cp), differentiate equation (1.1) with respect to the satellite attitude (for ob and ob) at their corresponding approximate values, then equations (2.5) and (2.6) can be linearized as the function of correction parameters.

cos2cp / *h*) dy  (coscp sincp / *h*) dz + dr (2.7)

*h*) dx  dp + tancp dy (2.8)

where dx, dy, and dz are the correction to satellite position vector **r**ob in the OB system, and d's are the corrections to the satellite attitude angle 's. Other quantities are functions evaluated at the approximate values of **r**ef, **v**ef, and 's.

The linearization above is done by directly differentiating equation (2.3) and (2.4), with transformation **T**i regarded unaffected by the error in **r**ef and **v**ef. This, however, ignores the curvature of the satellite orbit and the Earth, resulting in about 10 percent of error in the coefficients of dx, dy, and dz. A more accurate way to evaluate these coefficients is to examine the sensitivity terms dcp/dx, dcp/dy, and dcp/dz through the geometry of the look vector (see Figure 6‑30).

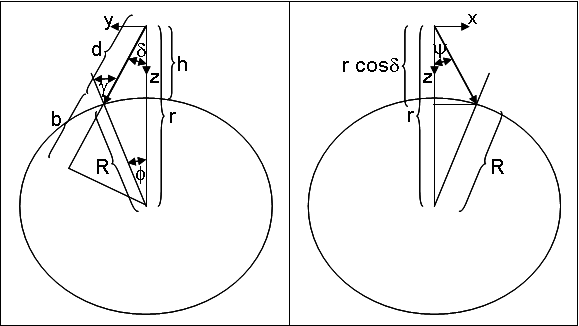


Figure 6‑30. Look Vector Geometry

*R* – the radius of the Earth

*r* – the radius of the satellite position

*h* – the altitude of the satellite

*d* – the magnitude of the look vector (from satellite to target)

 – the across-track angle of the look vector

 the Earth-centered angle between the satellite and the target

 – the zenith angle of the look vector at the target

x,y,z – the coordinates of the satellite position in the OB system

We have

*R* sin( + ) = *r* sin  (2.9)

Differentiating the equation (holding R and r constant) yields

*R* cos( + )(d + d) = *r* cos d (2.10)

Note that when  +  = , and d = dy / *r*, we have

 = d = (*b* / (*r d*)) dy (2.11)

Similarly for the along-track direction, we have

 = d = (-(*r* - *d* cos) / (r d cos)) dx (2.12)

For the effect of altitude error, differentiate equation (2.9) with respect to  and *r* (holding  constant) and noting dr = -dz, we have

 = d = (sin  / *d*) dz (2.13)

Note that the dx, dy, and dz in equations (2.11) through (2.13) are error terms, which are opposite in sign to the correction terms. With this in mind, we can replace the correction terms in equation (2.7) and (2.8) and rewrite the linearized observation equation as the following:

 = (*b* / (*r d*)) dy  (sin  / *d*) dz + dr (2.14)

 = ((*r - d* cos) / (*rd* cos)) dx - dp + tan  dy ( 2.15)

where:

*b* = *R* cos = sqrt(R2  (r2sin2)) (2.16)

*d* = *r* cos  - *b* (2.17)

This formulation does not account for the effects of applying the attitude correction in the ACS/body frame rather than the orbital frame. This is particularly significant in the case of off-nadir pointing. In the general case, applying the attitude correction in the ACS coordinate system leads to the following linearized observation equations:

 = (*b* / (*r d*)) dy  (sin  / *d*) dz + M11 dr + M12 dp + M13 dy (2.18)

 = ((*r - d* cos) / (*rd* cos)) dx + (M31 tan  - M21) dr

+ (M32 tan  - M22) dp + (M33 tan  - M23) dy (2.19)

where:

M11, M12, M13, M21, M22, M23, M31, M32, M33 are the elements of the ACS to Orbital rotation matrix **M**ACS2ORB at the time of the GCP observation. Thus, it is necessary to know the spacecraft roll-pitch-yaw corresponding to the GCP.

**M**ACS2ORB =



Note that for nominal nadir viewing M11 = M22 = M33 = 1 and M12 = M13 = M21 = M23 = M31 = M32 = 0 and equations (2.18) and (2.19) reduce to equations (2.14) and (2.15).

Both linearized observation equations (2.18) and (2.19) include all three attitude correction terms. This has the effect of linking the along- and across-track observations in the new OLI formulation, unlike the heritage implementation, which used separate along- and across-track solutions.

3. Weighted Least-Squares Solution

A weighted least-squares solution to the parameters is found using the following steps. The correction parameters in equations (2.18) and (2.19) can be expanded to include the correction to the change rates of the satellite attitude and position by defining

dx = dx0 + dxdotdt and dr = dr0 + drdotdt (3.1)

Since both the coordinates of the GCP and the measurement of the apparent GCP location in the image contribute random errors in computing  and , the covariance matrix for the observation equations (2.18) and (2.19) should be the sum of the covariance matrix of **R**cp in equation (1.6) and the covariance matrix of **R**ef in equation (2.1), mapped through equations (1.7), (2.3), and (2.4).

Note that in the observation equations (2.14) / (2.15) and (2.18a) / (2.19a),  is only related to parameters dy, dz, and dr, and  is only related to dx, dp, and dy. The parameters are uncoupled in the two observations. In the simplified case where observational error of and  are uncorrelated, the observation equations can be separated into two independent equations and solved individually. In the more general case of equations (2.18) and (2.19) the equations are coupled and must be solved together. This coupling is, in fact, present due to the yaw offsets introduced by yaw steering.

While it might be tempting to try to circumvent this complication by redefining the orbital coordinate system to be based on the Earth-rotation corrected ECEF velocity vector (thereby "yaw-steering" the orbital coordinate system), this would lead to a different set of complications in the application of the ephemeris corrections. In the baseline algorithm, we will adopt the general formulation of equations (2.18) and (2.19) and have adjusted the heritage separable least-squares solution formulation accordingly.

Proceeding with the integrated formulation, we define the parameter vector as the following:

**X**’ = {dr0, dp0, dy0, dx0, dy0, dz0, drdot, dpdot, dydot, dxdot, dydot, dzdot} (3.2)

(deleted) (3.3)

where ' means transpose of a vector or matrix. Then, the two observation equations can be written as follows:

 = *h*1 **X** + a (3.4)

 = *h*2 **X** + b (3.5)

where:

*h*1 = { M11, M12, M13, 0.0, b/(d r), -sin /d,

M11 dt, M12 dt, M13 dt, 0.0, b dt/(d r), -sin  dt/d} (3.6)

*h*2 = {(M31 tan - M21, (M32 tan - M22, (M33 tan - M23,

(r - d cos/(r d cos), 0.0, 0.0,

(M31 tan - M21dt, (M32 tan - M22 dt, (M33 tan - M23 dt,

(r - d cos) dt/(r d cos), 0.0, 0.0} (3.7)

with M11, M12, M13, M21, M22, M23, M31, M32, M33 the elements of the ACS to Orbital rotation matrix **M**ACS2ORB at the time of the GCP observation.

a and b are the random error of  and  respectively. With all GCPs included, the along- and across-track observation equation can be written as follows:

***A*** = *H*1**X** + A (3.8)

***B*** = *H*2**X** +B (3.9)

and the integrated parameters can be solved by WLS estimation as follows:

**X** = (*H*1’*WaH*1 + *H*2’*WbH*2)-1 (*H*1’*Wa****A*** + *H*2’*Wb****B***) (3.10)

where ***A*** and ***B*** are the observation vectors, composed of and  for all the GCPs, respectively. *H*1 and *H*2 are corresponding coefficient matrix, with *h1* and *h2* as rows corresponding to each  and , *Wa* and *Wb* are the diagonal weight matrix for ***A*** and ***B****,* respectively, composed of inverse of the variance of each individual a and b.

4. Parameter Correlation and Covariance Estimation

One problem in this solution is the nearly linear correlation between parameter dx and dp in the observation equation (3.7). The along-track orbit error and the pitch angle error have the very similar effect on  The two parameters cannot be well separated in the solution without additional information – including both parameters in the observation equations results in a near-singular normal equation and therefore an unstable solution of the parameters. Similarly, high correlation exists between the cross-track position and the roll attitude errors in equation (3.6), and an ill-conditioned normal equation would result.

To correct the image, we do not have to distinguish between orbit position correction and attitude correction parameters. Letting either the orbit or the attitude correction parameters absorb the existing errors will correct the image in a similar manner. Therefore, we can choose to estimate either dx and dy or dp and dr. This can be done by setting those coefficients in *h1* and *h2,* corresponding to the unwanted parameters to zero.

One of the challenging tasks is to distinguish satellite attitude error from the orbit positional error. The purpose of precision correction estimation is not only to correct the image but also to extract information about the sensor alignment, which is reflected in the attitude correction parameters. In order to separate the ephemeris error from the attitude error as much as possible, we should first use the most precise ephemeris data available and correct systematic errors with available models. Second, we should use available a priori information, in addition to the observation, to cure the ill condition of the normal equation in statistical estimation.

Let the observation equation be:

***Y*** = *H****X*** + ;

[] = 0, Cov[] = s2*C* (4.1)

where ***Y*** is the measurement vector, ***X*** the parameter vector, *H* the coefficient matrix,  the residual error vector, and s2 is a covariance scaling factor

and the a priori information of the parameters are as follows:

***X\_*** = ***X*** + x;

[x] = 0, Cov[x] = q2*C*x (4.2)

where ***X***\_ is the apriori parameter vector, x is the residual vector, and q2 is a covariance scaling factor;

then the normal equation for the Best Linear Unbiased Estimate (BLUE) ***X***^ of the unknown parameter vector ***X*** is as follows:

*((l/s2)H’WH* + *(l/q2)Wx*)***X^*** = *(l/s2)H’W****Y*** + *(l/q2)Wx****X*\_** (4.3)

where *W* and *Wx* are weight matrices.

*W = C-1*;

*Wx = Cx-1* (4.4)

The covariance matrix of ***X^*** is as follows:

Cov[***X^***] = ((*l/s2*)*H’WH* + *(l/q2)Wx)-1* (4.5)

Usually, the Cov[] and Cov[x] cannot be exactly known. In the case of GCP, for example, the position error involves many factors, such as base map error, and human marking error, etc. If there are unknown scale factors *s2* and *q2*, we can still obtain the WLS estimate from the normal equation.

*(H’WH + Wx)****X^*** *= H’W****Y*** *+ Wx****X\_*** (4.6)

In such case, the inverse of the normal matrix cannot be taken directly as the Cov[**X^**]. Factor *s2* and *q2* should be estimated with appropriate variance component estimation from the residual of the solution of equation (4.6). The weighted residual square summation can be calculated as follows:

V*’W****V*** *=* ***Y’****W****Y*** *- 2* ***X^****’M +* ***X^****’ N****X^*** (4.7)

V**x**’*Wx****Vx*** = ***X\_****’Wx****X\_*** *- 2* ***X^****’Wx****X\_*** *+* ***X^****’Wx****X^*** (4.8)

where:

V = ***Y*** *- H****X^*** the measurement residual vector (4.9)

***V****x =* ***X\_*** *-* ***X^*** the apriori parameter residual vector (4.10)

*N = H’WH* (4.11)

*M = H’W****Y*** (4.12)

When the factors *s2* and *q2* are appropriately estimated, the weight matrix ***W*** and ***Wx*** should be correspondingly corrected by factors 1/*s2* and 1/*q2*, respectively. Equation (4.6) should be resolved with the new weight matrices. In the new solution, information from the observation and the a priori information are appropriately combined and the

(*H' WH + Wx*)-1 is the Cov[***X^***].

5. Weight Factor Estimation

One of the estimates of *s2* and *q2* is the Helmert type estimate. For the problem here, the equation for the estimate can be derived following Helmert's variance component analysis,

*E s2 + D q2 =* ***V’****W****V*** (5.1)

*D s2 + G q2 =* ***Vx’****Wx****Vx*** (5.2)

where:

*E = n - 2 tr{Q N} + tr{Q N Q N}* (5.3)

*G = m - 2 tr{Q Wx} + tr{Q Wx Q Wx}* (5.4)

*D = tr{Q N Q Wx}* (5.5)

*Q = (H’ W H + Wx)-1* (5.6)

n = number of observations

m = number of parameters

*tr{A}* indicates the trace of matrix *A*

Equation (5.1) and (5.2) do not guarantee positive solution of *s2* and *q2*. In some cases, especially for small *s2* and *q2*, noise can drive the solution negative. Another type of estimate, the iterative Maximum Likelihood Estimate (MLH), guarantees a positive solution, though the estimate *s2* and *q2* may not be statistically unbiased. The MLH solution is obtained by iteratively solving equation (4.6) and

s2 = ***V’****W****V*** */ n* (5.7)

q2 = ***Vx’****Wx****Vx*** */ m* (5.8)

*W = W / s2* (5.9)

*Wx = Wx / q2* (5.10)

until *s2* and *q2* converge.

The solution above provides the estimate of the corrections to the ephemeris and attitude data, as well as to their covariance matrix. The covariance information can be used as a measure of precision for assessing the alignment errors of the sensor system. It can also be propagated to any pixel in the scene to evaluate the pixel location error after the precision correction.

6. Covariance Propagation

Given the sample time and across-track look angle of a pixel, the coefficients h1 and h2 can be calculated for  and  according to equation (3.6) and (3.7). The variance of  and  are then calculated as:

2 = *h1* Cov[**X^**]*h1*’ (6.1)

2 = *h2*Cov[***X^***]*h2’* (6.2)

These are the variance of the pixel location in sample and line directions due to the uncertainty of the estimated precision-correction parameters. They are in angles, but can be easily converted into IFOV according to the sensor system specifications.

7. Outlier Detection

Outlier detection for the precision-correction solutions seeks to identify GCPs that are likely to be in error due to miscorrelation. This is done by analyzing the GCP residuals, taking into account the relative importance of the GCP as reflected in the precision solution normal equation matrix.

Definitions:

**A** = matrix of coefficients (partial derivatives) relating parameters to observations

 = parameter vector

X = observation vector

V = residual vector

**C** = observation covariance matrix

n = the number of observations

p = the number of parameters

**A** is n x p,  is p x 1, X and V are n x 1, and **C** is n x n

Observation Equation:

**A** = X - V (7.1)

X = Xtrue + E where E = error vector ~ G(0,**C**) (7.2)

**A**true = Xtrue where true is the “true” parameter vector (7.3)

**A** = Xtrue + E - V (7.4)

so V = E if  = true

Minimum Variance Parameter Estimate:

’ = [**A**T**C**-1**A**]-1**A**T**C**-1X (7.5)

Estimated Residual Error:

V’ = X - **A**[**A**T**C**-1**A**]-1**A**T**C**-1X (7.6)

Define Projection matrix **P**:

**P** = **A**[**A**T**C**-1**A**]-1**A**T**C**-1 (7.7)

This matrix projects the observation vector into the parameter subspace (the column space of **A**). This projection is only orthogonal if **C** has the special structure described below.

Substituting:

V’ = X - **P**X = [**I** - **P**]X (7.8)

[**I** - **P**] projects X into the parameter null space.

Looking at the Error Estimate V’:

V’ = [**I** - **P**]X = [**I** - **P**][Xtrue + E] = [**I** - **P**]Xtrue + [**I** - **P**]E (7.9)

but [**I** - **P**]Xtrue = 0 since Xtrue lies entirely within the parameter subspace.

so V’ = [**I** - **P**]E = E - **P**E (7.10)

Here are some comments about V’ and E:

For a given precision solution, the elements of E are not random variables; they are realizations of random variables.

V’ is an estimate of the actual (realized) error E, which includes an estimation error equal to **P**E.

We cannot exactly recover E from [**I** - **P**]-1V’ because [**I** - **P**] is singular (it is an n x n matrix of rank n-p).

We can attempt to predict how accurate our estimate (V’) of E is likely to be by looking at the estimation error R = **P**E.

Since we want the predicted accuracy to apply in general, we treat R as a random vector, which is a function of another random vector E.

Expected Value: E[ R ] = E[ **P** E ] = **P** E[ E ] = **P** 0 = 0 (7.11)

Variance: E[ R RT ] = E[ **P** E ET **P**T ] = **P** E[ E ET ] **P**T = **P** **C** **P**T (7.12)

Special Structure of Observation Covariance Matrix for Precision Correction:

**C** = 2**I** (7.13)

since the observation errors are realizations of independent and identically distributed zero mean Gaussian random variables with variance 2.

Substituting (7.13) into equation (7.7) for **P** yields:

**P** = **A**[(1/2)**A**T**IA**]-1**A**T**I**(1/2) = **A**2[**A**T**A**]-1**A**T(1/2) = **A**[**A**T**A**]-1**A**T (7.14)

And the equation for the variance of R:

E[ R RT ] = 2 **P** **I** **P**T = 2 **P** (7.15)

noting that **P**T = **P** and **P** **P** = **P**

so R ~ G( 0, 2 **P** )

For a particular component of R ri:

E[ ri ] = 0 (7.16)

E[ ri2 ] = 2 pii (7.17)

Where pii is the ith diagonal component of **P**

Looking at the equation for **P**, we see that:

pii = AiT [**A**T**A**]-1 Ai (7.18)

Where AiT is the ith row of **A**

Considering a particular component of the Residual Error Vector V’:

vi = ei - ri (7.19)

Where ei is the corresponding component of the observation error vector

so vi is an unbiased estimate of ei with variance 2 pii

If we knew what ei was, we could test it against a probability threshold derived from its standard deviation, , to determine if it is likely to be an outlier. Instead of ei, we have vi, which includes the additional error term ri. Including the additional estimation error in the threshold computation leads to the following:

v2 = 2 + 2 pii (7.20)

Where 2 is the term due to the actual error variance and 2 pii is the term due to the estimation error variance.

This may seem like cheating since ei and ri are not independent for a given realization.

E[ vi2 ] = E[ (ei - ri)2 ] = E[ ei2 - 2eiri + ri2 ] and ri = j pij ej

E[ vi2 ] = 2 (1 - pii) (7.21)

It is tempting to use vi / (1 - pii)1/2 for ei in the outlier test (or, equivalently, to test vi against a threshold based on 2 (1 - pii)), but this becomes dangerous as pii approaches 1. The factor pii can be interpreted as a measure of the uniqueness of, or as the information content of, the ith observation. As pii approaches 1, the ith observation lies almost entirely within the parameter subspace, which implies that it is providing information to the solution that the other observations do not. Note that such “influential” observations can be identified from the structure of the coefficient matrix, **A**, without reference to the observation residuals. Attempting to use 1/(1 - pii)1/2 to rescale the residual vi to better approximate ei will, in a sense, punish this observation for being important. Instead, we view pii as a measure of how poor an estimate of the actual error, ei, the residual, vi, is and ignore the fact that vi will tend to be an underestimate of ei. We therefore use v2 (= 2 (1 + pii) as shown above) to construct the outlier detection threshold.

One remaining problem is that we do not know exactly what 2 is and must estimate it from the observation residuals. This is done by scaling the a priori observation variance using the variance of unit weight that was computed in the precision solution. The fact that we are using an estimated variance to establish our outlier detection threshold modifies the algorithm in two ways: 1) we compensate for the fact that removing a point as an outlier will alter the computation of the variance of unit weight by removing one residual and reducing the number of degrees of freedom; and 2) we base the detection threshold computation on student’s t-distribution rather than the Gaussian distribution.

The variance of unit weight is computed as follows:

var0 = VT**C**-1V/(n - p) = VTV/02(n - p) = vj2/02(n - p) (7.22)

Where: n = number of observations,

p = number of parameters, and

02 is the a priori variance.

The estimated variance is as follows:

var = var0 02 = vj2/(n - p) (7.23)

Removing the kth observation makes this:

vark = (vj2 - vk2)/(n - 1 - p) = (n - p)/(n - p - 1) \* (vj2 - vk2)/(n - p)

vark = (n - p)/(n - p - 1) \* var - vk2/(n - 1 - p) (7.24)

To normalize the kth residual, we divide it by the estimated standard deviation ’ = (var)1/2:

wk = vk / ’ (7.25)

We can rescale this normalized residual to reflect the removal of this observation from the variance estimate without having to actually compute a new variance:

wk’ = vk / k’ = wk ’/k’ = wk (var/vark)1/2

var/vark = 1 / [(n - p)/(n - p - 1) - vk2/var (n - p - 1)] = (n - p - 1)/(n - p - vk2/var)

var/vark = (n - p - 1)/(n - p - wk2)

noting that vk2/var = wk2

wk’ = wk [(n - p - 1)/(n - p - wk2)]1/2 (7.26)

Finally, we include the (1 + pkk) factor discussed above and our normalized and reweighted residual becomes:

wk’ = wk [(n - p - 1)/(1 + pkk)(n - p - wk2)]1/2 (7.27)

where: wk = vk / ’

This normalized and reweighted residual is compared against a probability threshold computed using Student’s t-distribution with (n - p) degrees of freedom.

*LOS Correction Procedure Overview*

The precision-correction procedure developed mathematically above is implemented as an iterative solution to account for the non-linearity of the observation equations presented in (2.5) and (2.6) above. Each step in the iteration solves the linearized correction problem using equation (3.10) above, using the current correction estimates, to compute incremental corrections for the current iteration. These corrections are used to update the current estimates, and the iteration continues until the incremental corrections are smaller than some threshold (or the iteration limit is exceeded).

An additional layer of iteration is introduced by the need to perform outlier filtering on the input GCP data. This, the procedure includes two levels of iteration: 1) use the current active set of GCPs to perform the iterative weighted least-squares solution (the linearization iteration); 2) filter the resulting GCP residuals for outliers, remove those exceeding the specified tolerance, and iterate the weighted least-squares procedure with the new (reduced) active set until no new outliers are found.

The LOS correction procedure can be viewed as a five-phase process in which the third and fourth phases are nested:

1. Phase 1 - Load the necessary data and initialize the solution procedure.
2. Phase 2 - Load and initialize the GCPs. For each GCP, use the geometric grid to compute the input space (L1R) location and time of observation. Interpolate the spacecraft position, velocity, and attitude at the time of observation.
3. Phase 3 - Use the current active set of GCPs to form and solve the linearized weighted least-squares equation. Use the computed corrections to update the current estimates. Iterate the linearized solution procedure until the incremental corrections are below the convergence threshold. Compute and write residuals for each iteration (the initial pre-correction and final iteration residuals are both used in geodetic accuracy assessment). Note that the residuals file is reinitialized at the beginning of each phase 4 loop so that the output residual file will reflect only the final pass through the outlier detection loop (phase 4).
4. Phase 4 - Run the outlier detection and removal iteration loop using the results of the iterative weighted least-squares solution procedure (phase 3) by testing the resulting residuals for outliers. Remove any newly detected outliers from the active GCP list and recompute the phase 3 solution with the reduced GCP set. Continue to iterate until no new outliers are detected.
5. Phase 5 - Write the precision solution file to document the result, update the LOS model, and, if requested, convert the attitude corrections to OLI alignment angles and write the resulting alignment calibration information to the characterization database.

Figure 6‑31 shows a block diagram of the LOS correction procedure in which the individual process steps are identified by phase using the codes P1 through P5.

P3

P3

P3

Retrieve

P

arameters

ODL

L1G

Geometric

Grid

DEM

Calculate ILS from

GCP OLS

Get L1G Metadata

Read GCP

Correlation Results

Get GCP Heights

Get Satellite Position

& Velocity

Calculate Correction

Update LOS Model

Write Precision

Solution

Precision

Solution

Precision

LOS

Model

Get LOS Model

Get Geometric Grid

Get ILS Position &

Time

Initialize Precision

Process One GCP

Initialize Residuals

Solve for Corrections

Update Observation

Angles

Write Final Residuals

Calculate ECF to ORB

Calculate LOS

Relate Observable

to Correction

LOS

Model

Residuals

GCPs

Check Convergence

Write Iteration

Residuals

P1

P1

P2

P2

P2

P2

P2

P2

P2

P3/4

P4

P3

P3

P3

P3

P3

P3

P3

P4

P5

P5

Remove GCP Outliers

Figure 6‑31. LOS Correction Algorithm Block Diagram

The following text contains the individual process steps in the prototype LOS correction procedure.

#### Prototype Code

Inputs to the executable include an ODL parameter file, an ASCII GCP measurement file created by the GCP correlation algorithm, the L1G image used to measure the GCPs, the OLI LOS model used to create the L1G image, the LOS projection grid file used to create the L1G image, the CPF used to create the L1G image, and the DEM file (if any) used to terrain correct the L1G image. Note that only the L1G image metadata is used, not the imagery itself. The outputs are an updated (precision corrected) OLI LOS model file, an ASCII report file containing a standard header that identifies the data set analyzed, and the results of the precision-correction solution, and an ASCII file containing the computed GCP residual errors at each iteration of the solution. The GCP residuals for the first and last iteration are subsequently used by the geodetic accuracy characterization algorithm. The prototype implementation also includes an option to generate trending data, which, rather than being stored in a database, is written to the standard output in a comma-delimited format.

The prototype code accesses two environment variables to populate fields used in the standard report header. These are IAS\_REL, which contains the IAS software version number, and IAS\_SITE, which contains a text string identifying the processing center.

The prototype code was compiled with the following options when creating the test data files:

-g -Wall -march=nocona -m32

**Get Precision Parameters (get\_prec\_par)**

This function gets the precision-correction algorithm parameters.

**Read Precision Parameters (read\_prec\_parm)**

This function reads all of the parameters from ODL and CPF files that precision requires.

**Get Position (get\_position)**

This function finds the satellite position, velocity, attitude, and reference time for each GCP.

**Add Position (add\_position)**

This function adds the position to the ground control point structure and assigns the reference time to the time structure, using the following steps:

1. From the 1G line and sample, find the latitude and longitude, and use the DEM to find the height at that line/sample (was new for Landsat 8); then, transform to Earth-fixed.
2. From the L1G line and sample, use the geometric grid and the ols2ils routine (reference the LOS Projection ADD) to compute the corresponding input space line and sample. Note that this computation includes the DEM height interpolated in step 1 above, which was new for Landsat 8.
3. From the input space line and sample, calculate the reference year, day, and seconds, satellite position and velocity, and spacecraft attitude (roll-pitch-yaw). Note that the inclusion of roll-pitch-yaw here was new for Landsat 8.
4. Calculate the transformation matrix from Earth-fixed to orbit-oriented.
5. Calculate the line-of-sight.

**Calculate Position (oli\_calc\_position)**

This function finds the satellite position, velocity, attitude, and time, using the forward model. This unit invokes oli\_findtime to get the time, oli\_findatt to get the attitude, and l8\_movesat to compute the position and velocity. The OLI LOS Projection ADD describes these sub-algorithms.

**Get Latitude/Longitude (getlatlong)**

This function finds the latitude/longitude given the L1G line/sample.

1. Find the first-order rotation coefficients if there is a rotation.
2. Find the output projection coordinate of pixel.
3. Call the projtran routine (see the LOS Projection ADD for details) to convert projection X/Y coordinates to the corresponding latitude/longitude.
4. Access the DEM to interpolate the height at the L1G line/sample coordinates. Note that this is a departure from the ALIAS heritage approach and is a consequence of using terrain-corrected mensuration images.
5. Convert the latitude, longitude, and height into Cartesian ECEF coordinates (x,y,z), as described below.

**Geodetic to Cartesian (geo\_geod2cart)**

This function converts geodetic coordinates (lat, lon, height) into Cartesian coordinates (x, y, z), as described in the LOS Projection ADD and reiterated below. Input latitude and longitude are in radians; height, semi-major axis, and output Cartesian position vector are in meters; flattening is a dimensionless number.

b = a (1 - f)

*e*2 = 1 - b2 / a2

N = a / (1 - *e*2 sin2(*φ*))1/2

X = (N + *h*) cos(*φ*) cos(*λ*)

Y = (N + *h*) cos(*φ*) sin(*λ*)

Z = (N (1-*e*2) + *h*) sin(*φ*)

where:

X, Y, Z - ECEF coordinates

*φ*, *λ*, *h* - Geodetic coordinates (lat **, long **, height *h*)

N - Ellipsoid radius of curvature in the prime vertical

f - Ellipsoid flattening (f = 1 - b/a)

*e*2 - Ellipsoid eccentricity squared

a, b - Ellipsoid semi-major and semi-minor axes

**Calculate Line-of-Sight (calc\_line\_of\_sight)**

This function calculates the line-of-sight angles from the satellite position to the ground point from their position coordinates.

For x, y, and z, assign: ecf\_look = pixpos - satpos.

Perform matrix multiplication to transform Earth-fixed look vector to orbit-oriented look vector (see the Earth-Fixed to Orbit-Oriented sub-algorithm below for the construction of the Tecf2oo transformation matrix):

[Tecf2oo]3x3 [ecf\_look]3x1 = [oo\_look]3x1

Compute the along- and across-track angles:

psi = arctan(oo\_look[0] / oo\_look[2])

delta = arctan(oo\_look[1] / oo\_look[2])

**Calculate Correction (calc\_correction)**

This function solves for the attitude and/or ephemeris correction using the Ground Control Points.

1. Initialize the correction parameter structure.
2. Allocate memory for the residuals structure.
3. Begin the outlier detection and rejection iteration loop.
4. Prepare the residual file to be written to.
5. Reset the GCP information to its original state.
6. Initialize the weight factor for observation and a priori parameters.
7. Iterate the precision correction solution process.
   1. Initialize the normal equations.
   2. For each GCP, compute the observables ( and ), relate them to the correction parameters, and then form the normal equation to accumulate.
   3. Accumulate the normal equations by adding up information from each GCP.
   4. compute diff\_time = gcps[gcp\_num].time - ref\_time[2]
   5. Write the residual information for this iteration. This will be done for each iteration. The structure get\_residuals must be filled before writing to this file. We store the RMS residuals for the first and last iteration as solution quality metrics.
   6. Solve the Normal equations. Solve for the corrections from the normal equation using the Weighted Least-Square sub-algorithm.
   7. If the parameter flag is 4 (weight factor estimation option):

1. Estimate the variance factor with the Minimum Norm Quadratic Unbiased Estimate (MINQUE).

2. If the MINQUE solution is obtained, compute the residual square sum.

3. If the MINQUE solution failed, try the Maximum Likelihood Estimate (MLHE) solution.

4. If MLHE fails, the solution cannot be obtained.

5. Calculate the posteriori standard error.

Else If the parameter flag is not 4 (no a priori weight factor estimation is used):

1. Compute the residual square sum.

2. Calculate the posteriori standard error.

* 1. Update the total correction estimate.
  2. Update the observable and orbit state for each GCP.
  3. If the sum of the absolute values of the elements of the across-track and along-track solution vectors are greater than 1 and the number of iterations is less than max iterations, iterate again; otherwise, end iteration.

1. Calculate the residual in alpha and beta for each GCP.
2. Check the residuals for new outliers; if any are found, continue the outlier iteration loop from step 3 above.
3. Extract the final solution and update the correction parameters.
4. Free memory.

**Write Residuals (write\_residuals)**

This function writes out the residual for along- and across-track components for each GCP to the residual file.

For each GCP:

Rescale the residuals to meters.

Compute the projection space value of the residuals.

Copy the information to the residual structure.

Write out the residual information.

**Get Ground (get\_ground)**

This function calculates the projection (x/y) residual values in meters from the Earth orbit delta and psi residual values.

1. Calculate the Earth Centered Fixed (ECF) to Orbit Oriented (OO) transformation system and transpose the matrix to get the OO to ECF matrix.
2. Given the satellite position and correction terms, calculate a new look vector.
3. Transform the vector from OO to ECF.
4. Convert the ECF latitude and longitude to projection in meters.
5. Convert the true latitude and longitude to projection in meters.
6. Subtract the projection and assign to residual.

**Earth-fixed to Orbit-oriented (xxx\_earth2orbit)**

This function generates the transformation matrix from the Earth-fixed Cartesian system to the OO Cartesian system, as described in the Ancillary Data Preprocessing ADD (6.1.4) and reiterated below. Note that the ECEF velocity vector is really the ECI velocity vector rotated into the ECEF coordinate system (i.e., it is still an inertial velocity) and does not include the relative Earth rotation velocity. This is done so that the ECEF velocity vector remains parallel to the attitude control reference X axis, which is defined in ECI coordinates.

The relationship between the orbital and Earth-Centered coordinate systems is based on the spacecraft's instantaneous ECEF position and velocity vectors. The rotation matrix to convert from orbital to ECEF can be constructed by forming the orbital coordinate system axes in ECEF coordinates:



where:

*p* = spacecraft position vector in ECEF

*v* = spacecraft velocity vector in ECEF

*n* = nadir vector direction

*h* = negative of angular momentum vector direction

*cv* = circular velocity vector direction

[ORB2ECEF] = rotation matrix from orbital to ECEF

The transformation from orbital to ECEF coordinates is the inverse of the ECEF to orbital transformation matrix. Since the ECEF to orbital matrix is orthogonal, the inverse is also equal to the transpose of the matrix.



**Detect Outliers (det\_outliers)**

This function detects GCP outliers using the residuals and normal equations. Given a tolerance value, outliers are removed within the data set until all values deemed as “non-outliers” or “valid” fall inside the confidence interval of a T-distribution. The tolerance, or associated confidence interval, is specified per run and usually lies between 0.9-0.99. The default value is 0.95. The number of degrees of freedom of the data set is equal to the number of valid data points minus one. The steps involved in this outlier procedure are as follows:

1. Calculate the standard deviation of all valid points in the data set.
2. Loop on “valid” data points until no outliers are found.
   1. Find the two-tailed T-distribution (*T*) value for the current degree of freedom and confidence level specified α.
   2. Calculate the largest deviation allowable for the specified degree of freedom and α. This is not scaled by  since the residuals themselves are normalized by  in step c below.

Δ = *T*

* 1. For each data point, compute the along- and across-track weight factors using equation (7.18) above and the normalized and weighted along- and across-track residuals using equation (7.27) above.
  2. Find the data point with the largest normalized and weighted residual.
  3. If the maximum residual value found in step d is less than Δ, then exit.
  4. If the value found in step d is greater than Δ, then flag the data point as an outlier and calculate the standard deviation of the new set of “valid” data points.

**Get Correction (get\_correction)**

This function extracts the estimated correction parameters and their covariance matrix from the Weighted Least-Square solution, and updates the correction parameter structure.

1. Record the reference time for the correction.
2. Extract the satellite position corrections.
3. Extract the satellite velocity corrections.
4. Extract the satellite attitude angle corrections.
5. Extract the satellite attitude angle rate corrections.
6. Record the covariance matrix for the correction parameters.

**Reset Observations (resetobserv)**

This function resets the satellite state vector and the look angles corresponding to each GCP to their original values. This resets the inputs for the next iteration of the outlier loop.

**Initialize Precision (initial\_precision)**

This function initializes the normal matrix for the attitude and ephemeris correction estimate by least-square solutions.

1. Initialize the observational and a priori part of the normal equation, obs\_mx, obs\_rgt, apr\_corr, apr\_wgt\_par, to zero or almost zero.
2. if param\_flag = both or input weights are provided, estimate all corrections.
3. Form the a priori normal matrix for the parameters.
4. Form the a priori right-side term for the parameters.
5. Subtract the current net correction (Yb) terms from the right-hand side to restrain the magnitude of the net correction.
6. if param\_flag = eph\_yaw, estimate the orbit corrections:
7. set zero a priori mean for roll
8. set zero a priori mean for roll dot
9. set huge a priori weight for roll
10. set huge a priori weight for roll dot
11. set zero a priori mean for pitch
12. set zero a priori mean for pitch dot
13. set huge a priori weight for pitch
14. set huge a priori weight for pitch dot
15. if param\_flag = att\_orb, estimate attitude corrections:
16. set zero a priori mean for dy
17. set zero a priori mean for dy dot
18. set huge a priori weight for dy
19. set huge a priori weight for dy dot
20. set zero a priori mean for dx
21. set zero a priori mean for dx dot
22. set huge a priori weight for dx
23. set huge a priori weight for dx dot
24. if time\_flag = FALSE
25. Block out the rate terms by setting a huge weight for zero apriori mean.
26. Initialize the number of observations.
27. Initialize the weighted residual square summation.

**Process One GCP (process\_one\_gcp)**

This function updates the normal equation of the least-square problem for correction solution by adding one GCP.

Calculate the transformation matrix from ECF to Orbit system

Calculate the line-of-sight angles for GCP

Note: The look vectors here should be in the Orbit reference system.

If the line-of-sight angle for the pixel Pi is from the forward model (in the spacecraft-fixed system), then it should be transformed into the Orbit reference system (through matrix A(roll, pitch, yaw)) first before the observable alpha and beta can be formed.

Compute the observable alpha and beta

If not an outlier:

Relate the observable to correction parameters

Update the weighted square summation of observation

Accumulate the normal equation contribution for alpha

Accumulate the normal equation contribution for beta

**Partial (partial)**

This function composes the partial coefficients matrix of the observation equation, given the angle delta for one GCP.

**Partial Attitude (partial\_att)**

This function composes the partial coefficients matrix of the observation equation for param\_flag = "att\_orb," estimating attitude plus height corrections.

Calculate the constants needed for the partial derivative (H) matrix calculation, such as sin(delta), cos(delta), and satellite radius.

The side perpendicular to the look vector = satellite\_radius \* sindelta

Compose the H matrix by finding:

alpha w.r.t roll, microradian

alpha w.r.t pitch, microradian

alpha w.r.t yaw, microradian

alpha w.r.t. dz, meter scaled to microradian

beta w.r.t roll, microradian

beta w.r.t. pitch, microradian

beta w.r.t. yaw, microradian

**Partial Ephemeris (partial\_eph)**

This function composes the partial coefficients matrix of the observation equation for param\_flag = "eph\_yaw," estimating ephemeris plus yaw corrections.

Calculate the constants needed for H calculation by assigning sin(delta), cos(delta), and satellite radius.

Compose the H matrix by finding:

alpha w.r.t. dy, meter scaled to microradian

alpha w.r.t. dz, meter scaled to microradian

alpha w.r.t. yaw, microradian

beta w.r.t. dx, meter scaled to microradian

beta w.r.t. yaw, microradian

**Partial All (partial\_all)**

This function composes the partial coefficients matrix of the observation equation for param\_flag = "both" or "weight," estimating both attitude and ephemeris corrections. Note that this is the normal case.

Calculate the constants needed for H calculation sin(delta), cos(delta), and satellite radius (see equations (3.6) and (3.7) above).

Compose the H matrix:

alpha w.r.t. roll, microradian

alpha w.r.t. pitch, microradian

alpha w.r.t. yaw, microradian

alpha w.r.t. dy, meter scaled to microradian

alpha w.r.t. dz, meter scaled to microradian

beta w.r.t. dx, meter scaled to microradian

beta w.r.t. roll, microradian

beta w.r.t. pitch, microradian

beta w.r.t. yaw, microradian

**Accumulate Normal Equation (accum\_normal\_equation)**

This function accumulates the normal equation of the least-square problem by adding one observation.

Update the n x n normal matrix by accumulating:

H\_transpose \* wo \* H

Update the n x 1 right-hand-side array of the normal equation by adding:

H\_transpose \* wo \* obs

where:

H is the matrix of partial derivatives

wo is the observation weight

obs is the observation value

**Weighted Least Square (weighted\_least\_square)**

This function solves the weighted least-square problem with nxn normal matrix.

Form the normal equation for the Weighted Least-Square (WLS) problem, including any weight factors:

A[i][j] = weight\_factor\_for\_observation \* normal\_matrix\_for\_observation[i][j]

Augment the diagonal terms using the apriori observations:

A[i][i] += weight\_factor\_for\_apriori \* normal\_matrix\_for\_apriori[i]

Form the constant vector, including both observations and apriori contributions:

L[i] = weight\_factor\_for\_observation \* observation\_rhs[i]

+ weight\_factor\_for\_apriori \* apr\_corr[i]

Solve the equation:

solution = sol\_Ya = A-1 L

Note that the inverted normal equation matrix (A-1) is returned along with the solution so that it can be used to construct the solution aposteriori covariance matrix.

**MINQUE (minque)**

This function estimates the variance factor with MINQUE.

let:

wght\_rss\_obs = weighted residual square for observation

cov\_mx = Inverse of the WLS problem normal matrix

obs\_mx = the observation part of the normal matrix

apr\_wgt\_par = the a priori weights loaded into a diagonal weight matrix

wgt\_fact\_obs = the estimated variance factor for the observation

wgt\_fact\_apr = the estimated variance factor for the a priori variance

compute the weighted residual square for the observation (rss\_obs)

compute the weighted residual square for the a priori parameters (rss\_apr)

Allocate memory for arrays

compute the trace coefficients for the weight estimate equation

cc2 = cov\_mx \* apr\_wgt\_par

cc1 = cov\_mx \* obs\_mx

s1 = ngcp - 2tr[cc1] + tr[cc1 \* cc1] ref. equation (5.3)

s2 = n\_aprior - 2tr[cc2] + tr[cc2 \* cc2] ref. equation (5.4)

s12 = tr[cc1 \* cc2] ref. equation (5.5)

solve for the weight factors:

ss1 = s1 \* s2 - s12 \* s12

wgt\_fact\_obs = (rss\_obs \* s2 - rss\_apr \* s12) / ss1

wgt\_fact\_apr = (rss\_apr \* s1 - rss\_obs \* s12) / ss1

If wgt\_fact\_obs and wgt\_fact\_apr are less than 0.0--return, minque failed.

Run WLS where the scale factor for the weight of observation and a priori are 1/wgt\_fact\_obs and 1/wgt\_fact\_apr, respectively.

If WLS fails, return minque with failed status.

If WLS returns a non-error value, assign wght\_rss\_obs.

**Residual Square Sum (resquare)**

This function computes the residual square sum by adding the dot product of:

sol\_YaT \* obs\_mx \* sol\_Ya - 2 \* obs\_rgtT \* sol\_Ya

where:

sol\_Ya is the weighted least-squares solution vector

obs\_mx is the normal equation matrix

obs\_rgt is the right hand side vector of the normal equations

to the observation square sum (post\_sig).

**MLHE (mlhe)**

This function estimates the variance factor with MLHE.

Initialize the weight factor to zero.

Iterate the estimation of the weight factors.

Compute the weighted residual square for the observation and for the apriori parameters.

Compute the weight factor estimate.

Compute the weight factor difference for this iteration.

Solve the new WLS solution with the new weight factors.

Compute the final variance factor estimate.

**New Observation Angle (new\_observ\_angle)**

This function updates the satellite state vector and the look angles corresponding to each GCP, according to the correction parameters, for the purpose of iteration.

Extract the orbit and attitude correction parameters from the solution vectors.

Orbit corrections:

dorbit[0] = sol\_Ya[3]

dorbit[1] = sol\_Ya[4]

dorbit[2] = sol\_Ya[5]

orbit\_rate[0] = sol\_Ya[9]

orbit\_rate[1] = sol\_Ya[10]

orbit\_rate[2] = sol\_Ya[11]

Attitude corrections:

datt[0] = sol\_Ya[0]

datt[1] = sol\_Ya[1]

datt[2] = sol\_Ya[2]

att\_rate[0] = sol\_Ya[6]

att\_rate[1] = sol\_Ya[7]

att\_rate[2] = sol\_Ya[8]

For each GCP

Calculate the orbit perturbation and update the orbit state vector

Calculate the attitude perturbation and update the look angles

**Update Ephemeris (update\_eph)**

This function calculates the orbit position change and updates the ephemeris data in the Earth-Fixed system.

Construct the ECF to orbital transformation Tef2oo from the input position and velocity vectors (using xxx\_earth2orbit).

Take the transpose of (the orthogonal matrix) Tef2oo to find the inverse Too2ef.

Transform the input orbital position and velocity corrections to ECF using Too2ef.

Update the input ECF position and velocity by adding the transformed position and velocity corrections.

**Calculate New Look Angles (newlook)**

This function calculates the new look angles by adding the attitude angle perturbation. The heritage ALIAS implementation was modified as described below to account for applying the attitude corrections in the ACS rather than the orbital coordinate system.

Convert the units of the attitude corrections (to radians).

Construct the look vector from the two look angles.

look\_vector[0] = tan(psi)

look\_vector[1] = tan(delta)

look\_vector[2] = 1.0

Convert the orbital look vector to the ACS coordinate system:

Use the roll-pitch-yaw values for this GCP to construct the orbital to ACS rotation matrix **M**ORB2ACS = [**M**ACS2ORB]T.

Where: **M**ACS2ORB =



Convert look\_vector to ACS\_look\_vector by multiplying it by [**M**ACS2ORB]T:

ACS\_look\_vector = [**M**ACS2ORB]T look\_vector

Use the attitude corrections to construct the ACS correction rotation matrix **M**Precision:

1. Compute the precision correction at the time (t\_att = att\_seconds + att\_time) corresponding to the attitude sample:
2. roll\_corr = roll\_bias + roll\_rate \* (t\_att – t\_ref – image\_seconds)
3. pitch\_corr = pitch\_bias + pitch\_rate \* (t\_att – t\_ref – image\_seconds)
4. yaw\_corr = yaw\_bias + yaw\_rate \* (t\_att – t\_ref – image\_seconds)

Note that only the seconds of day fields are needed for the attitude and image epochs, as they are constrained to be based on the same year and day.

1. Compute the rotation matrix corresponding to roll\_corr, pitch\_corr, and yaw\_corr (**M**Precision), using the same equations used for **M**ACS2ORB above.

Apply the attitude corrections to the look vector by multiplying by **M**Precision:

ACS\_pert\_look\_vector = **M**Precision ACS\_look\_vector

Rotate the line-of-sight back to the orbital coordinate system, using the transpose of the **M**ORB2ACS matrix, which is the same as **M**ACS2ORB:

pert\_look\_vector = **M**ACS2ORB ACS\_pert\_look\_vector

Note that this can be achieved with a single rotation of:

**M**corr = **M**ACS2ORB **M**Precision[**M**ACS2ORB]T

pert\_look\_vector = **M**corr look\_vector

Calculate the new look angles:

psi = arctan(pert\_look\_vector[0]/pert\_look\_vector[2])

delta = arctan(pert\_look\_vector[1]/pert\_look\_vector[2])

**Calculate Observation Residual (observation\_residual)**

This function corrects the final values of alpha and beta for all GCPs for the effects of the final solution iteration. These values are updated by process\_one\_gcp for all but the final iteration.

For each GCP

Calculate the full partial coefficients matrix for alpha and beta.

For all 6 elements

Calculate the residual for alpha by subtracting the calculated observation.

gcps.va = gcps.va - H1 \*Ya

Calculate the residual for beta by subtracting the calculated observation.

gcps.vb = gcps.vb - H2 \*Ya

Where Ya is the vector containing the incremental parameter corrections for the last iteration.

**Finish Processing (finish\_processing)**

This function updates the OLI model file and writes to the solution and residual files. It has new functions added to check the solution quality statistics (pre-fit RMS, post-fit RMS, outlier percent, number of valid GCPs) to determine if the solution was successful.

Compute the percentage of GCPs that were declared outliers:

percent\_outlier = num\_outlier / num\_GCP \* 100

Compute the number of valid GCPs:

num\_valid = num\_GCP – num\_outlier

Check the pre-fit RMS, post-fit RMS, percent\_outlier, and num\_valid metrics against the thresholds (maximum pre-fit RMS, maximum post-fit RMS, maximum outlier percentage, minimum number of valid GCPs) from the CPF.

If the pre- and post-fit RMS values are both below the thresholds, and either the percent\_outlier metric is below threshold or the num\_valid metric is above the threshold:

Update the model to make a precision model.

Fill the gcp\_solution structure with the appropriate values.

Write to the solution file.

Return success status.

Else return failure status.

**Update LOS Model (oli\_update\_model)**

This function updates the LOS model file with the precision-correction values. The LOS model will be read from the LOS model file, the new precision correction values will be placed in the LOS model structure, the LOS model will be processed with the new precision-correction values, and the new precision LOS model structure will be output to the precision LOS model file.

Unlike the heritage ALIAS approach, not only are the precision-correction parameters stored in the LOS model, they are also applied to both the ephemeris and attitude data sequences. This is captured in the LOS model by storing both original and corrected attitude and ephemeris data sequence. This update procedure operates as follows:

**Correct Attitude Sub-Algorithm (l8\_correct\_attitude)**

This function applies the ACS/body space attitude corrections computed by the LOS/precision correction procedure to the attitude data sequence. It outputs a parallel table of roll-pitch-yaw values with the precision corrections applied. This "corrected" table is created by the LOS Model Creation algorithm, but initially it is identical to the original attitude data sequence.

The sequence of transformations required to convert a line-of-sight in the OLI instrument coordinate system, generated using the Legendre polynomials, is as follows:

**x**ECEF = **M**ORB2ECEF **M**ACS2ORB **M**Precision **M**OLI2ACS **x**OLI

where: **x**OLI is the Legendre-derived instrument LOS vector

**M**OLI2ACS is the OLI to ACS alignment matrix from the CPF

**M**Precision is the correction to the attitude data, computed by the LOS/precision correction procedure

**M**ACS2ORB is the spacecraft attitude (roll-pitch-yaw)

**M**ORB2ECEF is the orbital-to-ECEF transformation computed using the ECEF ephemeris

**x**ECEF is the LOS vector in ECEF coordinates

Note that in the heritage ALIAS implementation, the sequence was:

**x**ECEF = **M**ORB2ECEF **M**Precision **M**ACS2ORB **M**OLI2ACS **x**OLI

For nadir-viewing imagery, the **M**ACS2ORB matrix is nearly identical, so there is little difference. Since OLI will occasionally be viewing off-nadir and it is more natural to model attitude errors in the ACS/body coordinate system, the order has been reversed for Landsat 8/9. The impact is minimal in the model and LOS projection, but becomes more important for the LOS/precision correction algorithm.

This new sub-algorithm pre-computes the **M**ACS2ORB **M**Precision combination and stores the corresponding corrected roll-pitch-yaw attitude sequence in the model structure. This approach has the following advantages:

1. It streamlines the application of the model for LOS projection by removing the step of explicitly applying the precision correction.
2. It allows for the use of a more complex correction model in the future since the application of the model is limited to this unit. Note that the Earth-view attitude correction model consists of the following model parameters:

Precision reference time: t\_ref in seconds from the image epoch (nominally near the center of the image time window)

Roll bias and rate corrections: roll\_bias, roll\_rate

Pitch bias and rate corrections: pitch\_bias, pitch\_rate

Yaw bias and rate corrections: yaw\_bias, yaw\_rate

This model is dealt with in more detail in the line-of-sight correction algorithm description.

1. Retaining both the original and corrected attitude sequences in the model makes the model self-contained and will make it unnecessary for the LOS/precision correction algorithm to access the preprocessed ancillary data.

The disadvantage is that it doubles the size of the attitude data in the model structure.

The construction of the corrected attitude sequence proceeds as follows:

For each point in the attitude sequence j = 0 to K-1:

1. Compute the rotation matrix corresponding to the jth roll-pitch-yaw values:

**M**ACS2ORB =



1. Compute the precision correction at the time (t\_att = att\_seconds + att\_time(j)) corresponding to the attitude sample:
2. roll\_corr = roll\_bias + roll\_rate \* (t\_att – t\_ref – image\_seconds)
3. pitch\_corr = pitch\_bias + pitch\_rate \* (t\_att – t\_ref – image\_seconds)
4. yaw\_corr = yaw\_bias + yaw\_rate \* (t\_att – t\_ref – image\_seconds)

Note that only the seconds of day fields are needed for the attitude and image epochs, as they are constrained to be based on the same year and day.

1. Compute the rotation matrix corresponding to roll\_corr, pitch\_corr, and yaw\_corr (**M**Precision), using the same equations presented in step 1 above.
2. Compute the composite rotation matrix: **M** = **M**ACS2ORB **M**Precision
3. Compute the composite roll-pitch-yaw values:



1. Store the composite roll’-pitch’-yaw’ values in the jth row of the corrected attitude data table.

**Correct Ephemeris Sub-Algorithm (l8\_convert\_ephem)**

The heritage ALIAS function converted the ephemeris information (position and velocity) from the Earth-Centered Inertial (ECI J2000) system to the ECEF system and applied the ephemeris corrections computed in the LOS/precision correction procedure to both ephemeris sets. Since both ECI and ECEF representations of the ephemeris are now provided by the ancillary data preprocessing algorithm (6.1.4), the first portion of the heritage algorithm is no longer necessary.

The precision-correction parameters are stored in the LOS model in the spacecraft orbital coordinate system as three position (x\_bias, y\_bias, z\_bias) corrections and three velocity (x\_rate, y\_rate, z\_rate) corrections that, like the attitude corrections, are relative to t\_ref. These values must be converted to the ECEF and ECI coordinate systems. Once the precision correction is determined in the ECEF/ECI coordinate system, the ECEF/ECI ephemeris values can be updated with the precision parameters.

Loop on LOS model ephemeris points j = 0 to N-1

Compute the precision correction:

Calculate delta time for precision correction:

dtime = ephem\_seconds + ephem\_time(j) – t\_ref – image\_seconds

Calculate the change in X, Y, Z due to precision correction. Corrections are in terms of spacecraft orbital coordinates.

dx orb = model precision x\_bias + model precision x\_rate \* dtime

dy orb = model precision y\_bias + model precision y\_rate \* dtime

dz orb = model precision z\_bias + model precision z\_rate \* dtime

where:

model precision x\_bias = precision (orbital coord sys) update to X position

model precision y\_bias = precision (orbital coord sys) update to Y position

model precision z\_bias = precision (orbital coord sys) update to Z position

model precision x\_rate = precision (orbital coord sys) update to X velocity

model precision y\_rate = precision (orbital coord sys) update to Y velocity

model precision z\_rate = precision (orbital coord sys) update to Z velocity

Construct precision position and velocity “delta” vectors.





Calculate the orbit to ECF transformation [ORB2ECEF] using ECEF ephemeris (See the ancillary data preprocessing ADD (6.1.4) for this procedure).

Transform precision “delta” vectors to ECEF.



Adjust ECEF ephemeris by the appropriate “delta” precision vector and store the new ephemeris in the model. These ephemeris points will be used when transforming an input line/sample to an output projection line/sample.



where:

All parameters are 3x1 vectors

ephemeris ecef values are the interpolated one-second ephemeris values in ECEF coordinates

Calculate the orbit to ECI transformation [ORB2ECI] using ECI ephemeris.

Transform precision “delta” vectors to ECI.



Adjust ECI ephemeris by the appropriate “delta” precision vector and store the new ephemeris in the model. These ephemeris points will be used with lunar/stellar observations.



where:

All parameters are 3x1 vectors

ephemeris eci values are the interpolated one-second ECI ephemeris

**Convert the Net Attitude Corrections to Alignment Angles (calc\_alignment)**

This new sub-algorithm combines the newly computed attitude correction with the OLI sensor alignment matrix from the LOS model to construct corrected alignment angles.

Compute the precision correction at the reference time t\_ref:

roll\_corr = roll\_bias

pitch\_corr = pitch\_bias

yaw\_corr = yaw\_bias

Compute the rotation matrix corresponding to roll\_corr, pitch\_corr, and yaw\_corr (**M**Precision), using the standard rotation matrix equations:

**M**Precision =



Extract the ACS to OLI alignment matrix, **M**ACS2OLI, from the OLI LOS model, and take the transpose to compute **M**OLI2ACS.

Compute the composite alignment matrix: **M** = **M**Precision **M**OLI2ACS

Compute the composite roll-pitch-yaw alignment angles:



Extract the orbital ephemeris biases from the precision solution: (x\_bias, y\_bias, z\_bias).

Extract the attitude bias correction and ephemeris bias correction covariance terms from the precision solution covariance matrix:

The solution provides:  
  with covariance Cov

We want to form:



With (see note #2): 

The covariance matrix captures the correlations between the attitude and ephemeris correction parameters (e.g., roll-Y and pitch-X).

The following fields are output to the alignment characterization database:

Reference time: image epoch year, image epoch day, image epoch second + t\_ref

Alignment vector: X above

Alignment covariance: CovX above

RMS GCP fit

Number of GCPs used

Outlier threshold used

Scene off-nadir roll angle

Control type flag (DOQ or GLS)

L0Rp ID

Work Order ID

WRS Path/Row

**LOS Model Correction Output Summary**

The primary output of the LOS model correction algorithm is the updated "precision" LOS model. This model has the same structure as the input LOS model, which is described in the LOS Model Creation ADD (6.2.1). Although the model structure is the same, the corrected ECI position and velocity and the corrected ECEF position and velocity sections of the Ephemeris Model, the corrected roll-pitch-yaw section of the Attitude Model, and the Precision Correction Model all contain updated values as a result of the LOS model correction algorithm.

Table 6‑6 shows the contents of the output LOS Model Correction Solution File. This report file documents the results of the LOS model correction procedure. It contains standard header fields common to all geometric report files.

|  |  |
| --- | --- |
| **Field** | **Description** |
| 1. Date and time | 1. Date (day of week, month, day of month, year) and time of file creation. |
| 1. Spacecraft and instrument source | 1. Landsat 8/9 and OLI |
| 1. Processing Center | 1. EROS Data Center SVT |
| 1. Work order ID | 1. Work order ID associated with processing (blank if not applicable) |
| 1. WRS path/row | 1. WRS path and row |
| 1. Software version | 1. Software version used to create report |
| 1. Off-nadir angle | 1. Off-nadir roll angle of processed image file |
| 1. Acquisition Type | 1. Earth viewing or Lunar |
| 1. L0Rp ID | 1. Input L0Rp image ID |
| 1. L1G image file | 1. Name of L1G used to measure GCPs |
| 1. Precision solution reference time | 1. Time reference for model correction parameters as year, day of year and seconds of day. |
| 1. Roll-pitch-yaw attitude corrections | 1. Attitude bias corrections in microradians |
| 1. Roll-pitch-yaw rate corrections | 1. Attitude rate corrections in microradians/second |
| 1. Roll-pitch-yaw standard deviations | 1. Attitude bias parameter sigmas in microradians |
| 1. Roll-pitch-yaw rate std. devs. | 1. Attitude rate parameter sigmas in microrads/sec |
| 1. Ephemeris position corrections | 1. Ephemeris X-Y-Z bias corrections in meters |
| 1. Ephemeris velocity corrections | 1. Ephemeris Vx-Vy-Vz corrections in meters/second |
| 1. Position standard deviations | 1. Ephemeris X-Y-Z sigmas in meters |
| 1. Velocity standard deviations | 1. Ephemeris Vx-Vy-Vz sigmas in meters/second |
| 1. Across-track covariance matrix | 1. 6-by-6 covariance matrix for roll, Y, Z, roll rate, Vy, Vz correction parameters. |
| 1. Along-track covariance matrix | 1. 6-by-6 covariance matrix for X, pitch, yaw, Vx, pitch rate, yaw rate correction parameters. |
| 1. Spacecraft roll-pitch-yaw at solution reference time | 1. Spacecraft attitude at solution reference time in microradians |

Table 6‑6. LOS Model Correction Solution Output File Contents

Table 6‑7 shows the contents of the LOS Model Correction Residuals file. This file documents the GCP residuals for the final set of GCPs (after the outlier rejection loop has found no additional outliers), including the residuals for each iteration of the weighted least-squares solution procedure. Thus, it contains both the initial (pre-correction) and final (post-correction) residuals. This file is used as an input by the Geodetic Accuracy Assessment algorithm. The output residual file also contains the standard report header mentioned above.

|  |  |
| --- | --- |
| **Field** | **Description** |
| 1. Date and time | 1. Date (day of week, month, day of month, year) and time of file creation. |
| 1. Spacecraft and instrument source | 1. Landsat 8/9 and OLI |
| 1. Processing Center | 1. EROS Data Center SVT |
| 1. Work order ID | 1. Work order ID associated with processing (blank if not applicable) |
| 1. WRS path/row | 1. WRS path and row |
| 1. Software version | 1. Software version used to create report |
| 1. Off-nadir angle | 1. Off-nadir roll angle of processed image file |
| 1. Acquisition Type | 1. Earth viewing or Lunar |
| 1. L0Rp ID | 1. Input L0Rp image ID |
| 1. L1G image file | 1. Name of L1G used to measure GCPs |
| 1. Number of GCPs used | 1. Number of valid (non-outlier) GCPs |
| 1. Heading for individual GCPs | 1. One line of ASCII text containing column headings for the individual GCP fields. |
| For each iteration: |  |
| Iteration number | Starts with 0 for initial (uncorrected) residuals and ends with "Final" for results of last iteration. |
| For each GCP: |  |
| Point ID | GCP ID (see GCP Correlation ADD for details) |
| Predicted L1G Line | Predicted L1G line location |
| Predicted L1G Sample | Predicted L1G sample location |
| GCP Time of Observation | Seconds from image epoch time |
| Latitude | GCP latitude in degrees |
| Longitude | GCP longitude in degrees |
| Height | GCP height in meters |
| Across-track Angle (delta) | Across-track LOS angle in degrees |
| Across-track Residual | Residual on delta converted to meters |
| Along-track Residual | Residual on psi converted to meters |
| Y Residual | Residual in Y/line direction in meters |
| X Residual | Residual in X/sample direction in meters |
| Outlier Flag | 0 for outlier, 1 for valid GCP |
| GCP Source | DOQ or GLS |

Table 6‑7. LOS Model Correction Residuals Output File Contents

Table 6‑8 lists the fields stored in the characterization database for future sensor alignment calibration operations.

|  |  |
| --- | --- |
| **Field** | **Description** |
| 1. Work order ID | 1. Work order ID associated with processing |
| 1. WRS path/row | 1. WRS path and row |
| 1. L0Rp ID | 1. Input L0Rp image ID |
| 1. Control Type | 1. DOQ or GLS |
| 1. Off-nadir angle | 1. Off-nadir roll angle of scene (in degrees) |
| 1. Number of GCPs used | 1. Number of valid (non-outlier) GCPs |
| 1. Outlier threshold used | 1. Confidence level used for outlier rejection threshold |
| 1. RMS GCP Fit | 1. RMS of final iteration across- and along-track residuals in meters. This field would subsequently be used to identify entries that may be suspect due to poor fits to the ground control. |
| 1. Precision solution reference time | 1. Time reference for model correction parameters as year, day of year and seconds of day. |
| 1. Roll-pitch-yaw alignment angles | 1. Composite alignment angles in microradians |
| 1. Ephemeris position corrections | 1. Ephemeris X-Y-Z bias corrections in meters |
| 1. Alignment covariance matrix | 1. 6-by-6 covariance matrix for roll, pitch, yaw, X, Y, Z correction parameters. |

Table 6‑8. Model/Alignment Characterization Database Output Fields

Table 6‑9 lists the fields stored in the characterization database to support future GCP quality assessment and improvement activities (see note #3). Only the residuals for non-outlier GCPs from the initial (zeroth) iteration are written to the characterization database.

|  |  |
| --- | --- |
| **Field** | **Description** |
| For each GCP: |  |
| 1. Work order ID | 1. Work order ID associated with processing |
| 1. WRS path/row | 1. WRS path and row |
| 1. L0Rp ID | 1. Input L0Rp image ID |
| Point ID | GCP ID (see the GCP Correlation ADD for details) |
| GCP Time of Observation | Year, day of year, and seconds of day |
| Ephemeris Position | Spacecraft ECEF position at GCP time (meters) |
| Ephemeris Velocity | Spacecraft ECEF velocity at GCP time (meters/sec) |
| Spacecraft Roll-Pitch-Yaw | Spacecraft roll-pitch-yaw at GCP time (radians) |
| True Latitude | GCP latitude in radians |
| True Longitude | GCP longitude in radians |
| True Height | GCP height in meters |
| Apparent Latitude | Latitude measured in L1G image in radians |
| Apparent Longitude | Longitude measured in L1G image in radians |
| Apparent Height | Height interpolated from DEM in meters |
| GCP Source | DOQ or GLS |

Table 6‑9. GCP Residual Characterization Database Output Fields

#### Maturity

Although much of the ALI model correction algorithm was reusable, changes were required in the following areas:

* The logic that computes the OLI sensor alignment corrections implied by the precision attitude and ephemeris corrections has been added to this algorithm (it runs as a pre-process in the ALIAS alignment calibration algorithm) to ensure that the computed corrections are applied to the proper sensor alignment matrix. Storing only the corrections leaves open the question of what alignment they are relative to. This was not a problem in the heritage systems (L7 IAS, ALIAS) because the alignment calibration process was run as one continuous flow, using the same set of data throughout. This approach limited the number of scenes that could be processed and restricted the order of processing to be in data acquisition order. This restriction will be lifted for OLIAS so a much larger volume of data can be reduced and trended for subsequent offline analysis. This requires the trended data to be converted to apparent alignment angles so acquisitions processed using different alignment calibrations can be compared.
* The covariance data that are trended for subsequent use in alignment calibration are a subset of the full precision solution covariance.
* Trending of a slightly enhanced version of the initial (zeroth) iteration GCP residuals has been added to support offline research into large-scale area triangulation. The path/row, date/time, GCP ID, true position, and apparent (mensuration image) position are recorded for all non-outlier GCPs.
* Since the precision correction process will likely be run prior to any cloud screening and will therefore frequently fail due to the inability to correlate GCPs in cloud-covered imagery, thresholds and bounds will need to be developed to detect cases in which the solution has failed. In this case, scene processing would fail over to a terrain-corrected systematic data flow. The prototype computes and reports three quality metrics, including prefit GCP RMS, postfit GCP RMS, and percent GCP outliers, but does not apply any threshold logic. The operational version should apply thresholds on the pre-fit and post-fit GCP RMSE values, and make sure that either a sufficient number of valid GCPs were used or that the percentage of GCPs declared outliers was not too high.
* Using a systematic terrain-corrected image for GCP mensuration instead of the heritage systematic image required some modifications to the GCP processing logic. Specifically, the DEM elevation associated with the measured GCP image position is used to construct the “apparent” LOS instead of using zero for a LOS projected to the ellipsoid surface as the heritage algorithm does. Note that, while the actual GCP elevation could be used, this would introduce error that would grow with the misregistration between the systematic image and the DEM, making the simpler approach less robust. This change was motivated by the large GCP search areas that would be required in systematically corrected images for off-nadir scenes in high elevation areas.
* Using the DEM as the source of “apparent” GCP height allows the algorithm to support either terrain-corrected or systematic image inputs. If an input DEM is not provided, the “apparent” GCP height will be set to zero as it is now. If an input DEM is provided, it will be used as the source of the “apparent” GCP height. Note that the capability to use SCA-combined mensuration images only applies to terrain-corrected images.
* The heritage ALIAS implementation generates a fatal error if the square root of a negative number is encountered while computing partial derivatives. This can happen in the case of an invalid GCP measurement. This will be enhanced for OLI to adopt the logic used in Landsat 7, wherein this condition is detected and used to declare the offending point an outlier rather than generating a terminal error condition.

#### Notes

Some additional background assumptions and notes include the following:

* The heritage aliprecision process uses the DDR for the L1G mensuration image to retrieve the image framing and projection parameter information necessary to convert output space line/sample coordinates to latitude/longitude, but the same information is available in the grid file, so either could be used.
* The extent to which the model creation logic must be rerun was scaled back as compared to the heritage implementation. The precision ephemeris corrections are embedded in the model ephemeris so it must be regenerated, but full model reprocessing is not truly necessary. This was done in the past for convenience (because it is fast and was easy to simply invoke the model creation logic as a subroutine – Update LOS Model).
* Possible additional solution quality metrics include initial vs. final GCP distribution metrics, but are not implemented in the baseline version.
* The heritage ALIAS (and Landsat) LOS model correction algorithms required the L1R file as input data to provide the L1R input space image dimensions. This information will now be available in the LOS model in the image sub-model, so the L1R input is no longer required.