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

#### Background/Introduction

The LOS model creation algorithm gathers the ancillary data and calibration parameters required to support geometric processing of the input image data set, validates the image time codes, extracts the corresponding ephemeris and attitude data from the ancillary data stream, performs the necessary coordinate transformations, and stores the results in a geometric model structure for subsequent use by other geometric algorithms. The OLI LOS model creation algorithm is derived from the ALI model creation algorithm used in ALIAS. Its implementation is very similar to the alinit application and the geometric sensor (axx) and spacecraft (exx) model libraries used in ALIAS, though much of the ephemeris and attitude preprocessing logic present in alinit has been moved to the (now separate) ancillary data preprocessing algorithm to better isolate the bulk of the geometric processing logic from the details of the incoming ancillary data stream. New attitude data processing logic has also been added to separate the high- and low-frequency attitude effects to allow the image resampling process to better correct for jitter at frequencies above the original 10 Hz algorithm design limit, without requiring an unreasonably dense resampling grid.

#### Dependencies

The LOS Model Creation algorithm assumes that the Ancillary Data Preprocessing algorithm has been executed to accomplish the following:

* Validated ephemeris data for the full imaging interval have been generated
* Validated attitude data for the full imaging interval have been generated
* The ancillary data have been scaled to engineering units

Whether or not “definitive” processing has been performed, the Ancillary Data Preprocessing algorithm will generate preprocessed smoothed and cleaned ephemeris and attitude data streams. The format will be the same for either validated spacecraft estimates or definitive processing.

####  Inputs

The LOS Model Creation 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) |
|  Acquisition Type (Earth, Lunar, Stellar) (optional, defaults to Earth) |
|  CPF File Name |
|  Ancillary Data Input File Name |
|  L0R/L1R Directory and File Names |
|  WRS Path/Row (stored in model and used for trending) |
|  Trending On/Off Switch (not implemented in prototype) |
|  L0Rp ID (for trending) |
|  Work Order ID (for trending) |
|  Optional Precision Model Input Parameters (see note 9) |
|  Input Precision Model Reference Time (optional) |
|  Input Precision Ephemeris Correction Order (optional) |
|  Input Precision X Correction Parameters (optional) |
|  Input Precision Y Correction Parameters (optional) |
|  Input Precision Z Correction Parameters (optional) |
|  Input Precision Attitude Correction Order (optional) |
|  Input Precision Roll Correction Parameters (optional) |
|  Input Precision Pitch Correction Parameters (optional) |
|  Input Precision Yaw Correction Parameters (optional) |
|  CPF Contents |
|  WGS84 Earth ellipsoid parameters |
|  Earth orientation parameters (UT1UTC, pole wander, leap seconds) (see note 1) |
|  Earth rotation velocity |
|  Speed of light |
|  ACS to OLI rotation matrix |
|  Spacecraft center of mass (CM) to OLI offset in ACS reference frame (meters) |
|  High-frequency attitude data cutoff frequency (Hz) |
|  Focal plane model parameters (Legendre coefficients) |
|  Detector delay table (now including whole pixel even/odd and deselect offsets) |
|  Nominal L0R fill (per band) |
|  Nominal OLI frame time nominal\_frame\_time (4.236 msec) |
|  Nominal OLI MS and pan integration times (msec) |
|  OLI MS and pan detector settling times (msec) |
|  Preprocessed Ancillary Data Contents |
|  Attitude Data |
|  Attitude data UTC epoch: Year, Day of Year, Seconds of Day |
|  Time from epoch (one per sample, nominally 50 Hz) |
|  ECI quaternion (vector: q1, q2, q3, scalar: q4) (one per sample) |
|  ECEF quaternion (one per sample) |
|  Body rate estimate (roll, pitch, yaw rate) (one per sample) |
|  Roll, pitch, yaw estimate (one per sample)  |
|  Ephemeris Data  |
|  Ephemeris data UTC epoch: Year, Day of Year, Seconds of Day |
|  Time from epoch (one per sample, nominally 1 Hz) |
|  ECI position estimate (X, Y, Z) (one set per sample) |
|  ECI velocity estimate (Vx, Vy, Vz) (one set per sample) |
|  ECEF position estimate (X, Y, Z) (one set per sample) |
|  ECEF velocity estimate (Vx, Vy, Vz) (one set per sample) |
|  L0R/L1R Data Contents |
|  Image Time Codes (one per line) |
|  Integration Time (one value for MS bands and one value for pan band) |
|  Detector Alignment Fill Table (see note 2) |

#### Outputs

|  |
| --- |
| OLI LOS Model (see Table 6‑3 below) |
|  WGS84 Earth ellipsoid parameters |
|  Earth Orientation Parameters (for current day) from CPF |
|  Earth rotation velocity |
|  Speed of light |
|  OLI to ACS reference alignment matrix/quaternion |
|  Spacecraft center of mass to OLI offset in ACS reference frame |
|  Focal plane model parameters (Legendre coefs) |
|  Detector delay table (now including whole pixel even/odd and deselect offsets) |
|  Nominal detector alignment fill table (from CPF) |
|  L0R detector alignment fill table (from L0R) |
|  ECI J2000 spacecraft ephemeris model (original and corrected) |
|  ECEF spacecraft ephemeris model (original and corrected) |
|  Spacecraft attitude model (time, roll, pitch, yaw) (orig and corr) (see note 4) |
|  High frequency attitude perturbations (roll, pitch, yaw) per image line (jitter table) |
|  Image time codes (see note 5) (in seconds) |
|  Integration Time (MS and pan) (in seconds) |
|  Sample Time (MS and pan) (in seconds) |
|  Settling Time (MS and pan) (in seconds) |
|  WRS Path/Row |
| Model Trending Data |
|  WRS Path/Row |
|  L0Rp ID |
|  Work Order ID  |
|  Image start UTC time (year, day of year, seconds of day) |
|  Computed image frame time (in seconds) |
|  Number of image lines |
|  Number of out-of-limit image time codes |

#### Options

Trending On/Off Switch

Optional precision model input parameters can be used to force model corrections.

#### Prototype Code

Input to the executable is an ODL file; output is a HDF4 formatted OLI model file.

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

 -g -Wall –O2 -march=nocona -m32 –mfpmath=sse –msse2

The following text is a brief description of the main set of modules used within the prototype with each module listed, along with a very short description. It should be noted that not all library modules are referenced in the explanations below. The modules within the main create directory of the prototype are discussed, and any library modules that were determined to be important to the explanation of either results, input parameters, or output parameters.

model\_create – Main procedure that retrieves the input parameters and invokes the model generation and model output logic.

getpar – Retrieves the user-provided ODL parameters.

oli\_zero\_model – Library routine that initializes the model structure.

get\_path\_row\_l0ra - Designed to retrieve the WRS path and row numbers from the L0R data. In the baseline algorithm, these are ODL input parameters, but they should ultimately be extracted from the Level 0R data directly. This unit is a placeholder for the time being.

oli\_run\_model – Library routine that loads the CPF, L0R, and preprocessed ancillary data into the model structure.

oli\_get\_cpf – Library routine that reads the CPF.

oli\_get\_model\_sensor\_params – Library routine that loads the sensor section of the model structure, using data from the CPF and the L0R frame header.

oli\_get\_model\_image\_params – Library routine that loads the image section of the model structure, using data from the CPF, the L0R line headers, and the L0R detector offset fields. This unit also validates the image line time codes.

oli\_get\_model\_earth\_params – Library routine that loads the Earth model parameters from the CPF.

oli\_get\_ancillary\_pre – Library routine that loads the attitude and ephemeris model sections, using data from the preprocessed ancillary data file.

oli\_build\_jitter\_table – Library routine that splits the attitude data from the ancillary data into low- and high-frequency streams, interpolates the high frequency data to match the OLI panchromatic band line times, stores this per image line high-frequency attitude data in the jitter table structure, and replaces the original combined attitude data stream with the low-frequency stream.

remez – Library routine that uses the Remez exchange algorithm to synthesize the weights (taps) of a low pass finite impulse response digital filter based on input filter size and cutoff frequency parameters. GNU Public License code written by Jake Janovetz, formerly of UIUC, which is available online at his site: http://www.janovetz.com/jake/

and more specifically:

http://www.janovetz.com/jake/remez/remez-19980711.zip

l8\_correct\_attitude – Library routine that applies the user-input precision model attitude corrections (if any).

l8\_convert\_ephem – Library routine that applies the user-input precision model ephemeris corrections (if any).

oli\_put\_model – Library routine that writes the OLI model structure to the output HDF model file.

#### Procedure

The LOS model is stored as a structure and is created from information contained in the Level 0R or Level 1R image data, the CPF, and the Ancillary data. The model is subsequently used, along with the CPF, to create a resampling grid. Data present in the model structure includes satellite position, velocity, and attitude, LOS angles, timing references, precision-correction information (if any), and the software version. The LOS model is also used in several characterization and calibration routines for mapping input line/sample locations to geographic latitude/longitude.

The LOS model may be thought of in two parts, an instrument model that provides a line-of-sight vector for each OLI detector (and, hence, each image line/sample), and a spacecraft model that provides spacecraft ephemeris (position and velocity) and attitude as a function of time. These models are linked by the image time stamps that allow each Level 0R or Level 1R image sample to be associated with a time of observation.

##### Instrument Model

The model treats every band of every SCA independently. This is done by defining a set of 2nd order Legendre polynomials for each band of each SCA. Since the odd and even detectors are staggered for each band (Figure 6‑18) as well as there being multiple pixel selects, the set of Legendre polynomials represent a theoretical “nominal” set of detectors that are best-fit to the even detectors for the first pixel select. This approach treats the odd detectors and second and third pixel select detectors as though they were aligned with the even detectors for the first pixel select for purposes of sensor LOS generation. This approach also explicitly models the slight offsets caused by the actual odd detector offset, any offsets caused by detector deselect, and the subpixel deviations of each detector from its nominal location, for correction during image resampling. This leads to four detector types: nominal, actual, maximum, and exact. A nominal detector is calculated from the Legendre polynomials. An actual detector corrects the nominal detector location for the whole pixel odd/even and pixel select offsets. For the ALI, these offsets were band dependent. For the OLI, since individual detectors may be deselected, they are detector dependent. The maximum detector option uses the largest possible even/odd and pixel select offset for a given band. This is used to compute detector terrain parallax sensitivity coefficients when generating the line-of-sight grid. See the LOS Projection/Grid Generation Algorithm Description Document for additional details. An exact detector has the actual correction applied but also includes the specific individual (subpixel) detector offsets. The Legendre polynomials and a table of detector-offset values are stored in the CPF.

There is a slight angular difference between the line-of-sight vectors or angles associated with the odd/even and multiple pixel select detectors. If the nominal LOS, generated using the 2nd order Legendre model, is nominal, the look angles for the actual and exact detectors are as follows:

x\_actual = x\_nominal + round(detector\_shift\_x) \* IFOV

y\_actual = y\_nominal + round(detector\_shift\_y) \* IFOV

x\_exact = x\_nominal + detector\_shift\_x \* IFOV

y\_exact = y\_nominal + detector\_shift\_y \* IFOV

The maximum detector case uses the largest possible along-track offset and the nominal across-track offset, which is zero:

x\_maximum = x\_nominal + maximum\_shift\_x \* IFOV

y\_maximum = y\_nominal



Figure 6‑18. Detector Layout

The detector\_shift\_x and detector\_shift\_y values are the detector-specific offsets stored in the CPF detector delay tables. These offsets include both the whole-pixel even/odd and deselect offsets and the fractional-pixel detector placement effects, and must be rounded to extract the integer portion. Note that the integer portion of the detector\_shift\_y value is always zero since the even/odd and deselect effects are applicable only in the X direction.

The nominal LOS is used in most line-of-sight projection applications. The actual LOS is used in conjunction with the actual image time (see below) to model the errors introduced by trading time (sample delay) for space (detector offset) for purposes of correcting the nominal LOS model. The exact LOS is generally used only for data simulation and other analytical purposes rather than in the geometric correction model, as the subpixel portion of the detector delay is applied directly in the image resampler rather than being included in the LOS model.

#####  Sample Timing

The OLI provides a time stamp with each image frame collected. These time stamps make it possible to relate the image samples (pixels) to the corresponding spacecraft ephemeris and attitude data. Figure 6‑19 shows the OLI sample timing relationships. Several items in this figure are worthy of particular note. First, the time stamp associated with a data frame is recorded at the end of the detector integration time. Second, there is a small settling and sampling delay (MS SS and Pan SS in the figure) between the end of detector integration and time stamp generation. Third, the time stamps are delayed by one data frame so that date frame N contains the time stamp for data frame N-1. Fourth, the data frame associated with time stamp N contains the multispectral (MS) samples collected just prior to time stamp N as well as the panchromatic samples collected just prior to and just after time stamp N, rather than the two samples collected prior to time stamp N. This is important for relating the panchromatic sample timing to the multispectral sample timing.



Figure 6‑19. OLI Focal Plane Electronics Detector Sample Timing Diagram

One further complication to the problem of assigning times to image samples is the fact that the Level 0R/1R input imagery will include fill pixels inserted to achieve nominal even/odd detector alignment. This fill insertion allows the geometrically unprocessed 0R/1R imagery to be viewed as a spatially contiguous image without even/odd detector misalignments. The amount of detector alignment fill present will be indicated in the L0R/L1R image data (this is the purpose of the detector alignment fill table input noted above) so that the association of image samples with their corresponding time stamps can be adjusted accordingly. In the heritage ALIAS system, fill pixels were also inserted to achieve nominal band alignment. The assumption is that this will not be done for OLI data.

Due to the staggered odd/even and multiple pixel select detectors, a nominal and an actual time can be found in a scene. The actual time reflects the time that the current detector was actually sampled, whereas the nominal time reflects the time at which the idealized detector represented by the OLI LOS model would have been sampled. There is also a “maximum” detector time option used in the computation of detector terrain parallax sensitivity coefficients during grid generation.

If the current position within the image is given as a line and sample location, the two different “types” of times for multispectral pixels are calculated by the following:

if detector\_type is set to MAXIMUM

l0r\_fill\_pixels = nominal\_fill\_pixels + round(maximum\_detector\_delay)

else

l0r\_fill\_pixels = Fill value for current detector from L0Rp

time\_index = round( MS\_line ) - l0r\_fill\_pixels + 1

if ( time\_index < 0 ) time\_index = 0

if (time\_index > (num\_time\_stamps - 1)) time\_index = num\_time\_stamps - 1

MS\_actual\_time = line\_time\_stamp[time\_index] - MS\_settle\_time

 - MS\_integration\_time/2

 + (MS\_line - l0r\_fill\_pixels – (time\_index-1)) \* MS\_sample\_time

MS\_nominal\_time = MS\_actual\_time

 + (l0r\_fill\_pixels – nominal\_fill\_pixels) \* MS\_sample\_time

where:

* MS\_line is the zero-referenced multispectral line number (N).
* nominal\_fill\_pixels is the amount of even/odd detector alignment fill to be inserted at the beginning of pixel columns that correspond to nominal detectors; that is, those detectors with a delay value of zero that are the basis for the Legendre polynomial LOS model. This value comes from the CPF.
* l0r\_fill\_pixels is the total amount of even/odd detector alignment fill inserted at the beginning of the pixel column associated with the current detector in the Level 0R image data. It includes both the nominal\_fill\_pixels and the detector-specific delay fill required to align even/odd detectors.
* num\_time\_stamps is the total number of time codes (data frames) in the image. It is tested to ensure that time\_index, the line\_time\_stamp index, does not go out of bounds.
* maximum\_detector\_delay is a constant offset that represents the largest amount of even/odd detector offset for any detector from the LOS model detector delay table. It is rounded to the nearest integer pixel because time offsets can only occur in whole line increments. The value of this parameter is not critical, as the line-of-sight offsets computed for “maximum” detectors are divided by the maximum delay to compute offset-per-unit-delay coefficients. This parameter is set in a #define statement.
* MS\_settle\_time is a small sample and hold time delay constant.

The MS\_settle\_time correction is expected to be a small (tens of microseconds) constant offset that will be captured in the CPF. The L0R/L1R data can be accessed by SCA making the association of sample number with detector index more straightforward. Note that the Level 0R data inverts the detector read-out order for the even-numbered SCAs so that the samples are numbered left-to-right for all SCAs (see note 6). This convention is also followed in the CPF detector offset tables. Also note that the non-imaging detector data (video reference pixels) are stored separately from the image data in the L0R and are not modeled in the CPF (see note 7). Therefore, there are 494 samples per SCA in the multispectral bands and 988 samples per SCA in the panchromatic band.

For the panchromatic band, the corresponding equations for a pan detector in the two pan lines (2N and 2N+1) associated with MS line N (Figure 6‑19) are computed as follows:

if detector\_type is set to MAXIMUM

l0r\_fill\_pixels = nominal\_fill\_pixels + round(maximum\_detector\_delay)

else

l0r\_fill\_pixels = Fill value for current detector from L0Rp

time\_index = floor( (round( pan\_line ) - l0r\_fill\_pixels)/2 ) + 1

if ( time\_index < 0 ) time\_index = 0

if (time\_index > (num\_time\_stamps - 1)) time\_index = num\_time\_stamps - 1

Pan\_actual\_time = line\_time\_stamp[time\_index] - Pan\_settle\_time

 - Pan\_integration\_time/2

 + (pan\_line - l0r\_fill\_pixels - 2\*(time\_index-1))\*Pan\_sample\_time

Pan\_nominal\_time = Pan\_actual\_time

 + (l0r\_fill\_pixels – nominal\_fill\_pixels) \* Pan\_sample\_time

where:

* pan\_line is the zero-referenced panchromatic line number (2N or 2N+1).
* nominal\_fill\_pixels is the amount of even/odd detector alignment fill to be inserted at the beginning of pixel columns that correspond to nominal detectors; that is, those detectors with a delay value of zero that are the basis for the Legendre polynomial LOS model. This value comes from the CPF.
* l0r\_fill\_pixels is the total amount of even/odd detector alignment fill to be inserted at the beginning of the pixel column associated with the current detector. It includes both the nominal\_fill\_pixels and the detector-specific delay fill required to align even/odd detectors. Note that these values will always be even for the panchromatic band.
* num\_time\_stamps is the total number of time codes (data frames) in the image. It is tested to ensure that time\_index, the line\_time\_stamp index, does not go out of bounds.
* maximum\_detector\_delay is a constant offset that represents the largest amount of even/odd detector offset for any detector from the LOS model detector delay table. It is rounded to the nearest integer pixel because time offsets can only occur in whole line increments. The value of this parameter is not critical as the line-of-sight offsets computed for “maximum” detectors are divided by the maximum delay to compute offset-per-unit-delay coefficients. This parameter is set in a #define statement.
* Pan\_settle\_time is a small sample and hold time delay constant.

For the panchromatic band, the l0r\_fill\_pixels, nominal\_fill\_pixels, and maximum\_detector\_delay parameters are in units of panchromatic pixels.

Note that when fill is used to align even and odd detectors, the spatial difference between the nominal and actual look vectors is approximately compensated by the time difference between tnominal and tactual.

Spacecraft Model

The spacecraft ephemeris and attitude models are constructed from the input preprocessed ancillary data by extracting the ancillary data that span the current image. Both ECI and ECEF versions of the ephemeris data are retained in the model structure to avoid the need to repeatedly invoke the ECI/ECEF coordinate system conversion. The ALIAS heritage roll-pitch-yaw representation of the attitude model is retained in the model structure, though a quaternion representation may be used in a future algorithm revision (see note 4).

###### Prepare LOS Model Sub-Algorithm

This function gathers the information from the preprocessed ancillary data and the Level 0R/1R image data needed to process model data and run the LOS model. Though it has the same overall purpose and function as the heritage axx\_run\_alimodel unit, differences in the details of how image timing and spacecraft telemetry information are provided for Landsat 8/9, as compared to EO-1, lead to extensive changes.

The main steps are as follows:

1. Load the image time codes and convert to seconds since spacecraft epoch.
2. Determine the image time window.
3. Validate/smooth the image time codes.
4. Extract the multispectral and panchromatic integration times from the Level 0R/1R image frame header data.
5. Extract the associated ephemeris and attitude data from the preprocessed ancillary data stream.
6. Preprocess the input attitude data into a low-frequency stream, used for basic geometric modeling, and a high-frequency stream, used as a fine correction in the image resampler. This preprocessing was added to improve the ability of the geometric correction system to compensate for jitter disturbance frequencies above 10 Hz.

The input preprocessed ancillary data are stored in an HDF file. The attitude and ephemeris ancillary data streams each have an epoch time identifying the UTC date/time reference. Within these data streams, each attitude or ephemeris observation in the HDF file has a corresponding time offset relative to the epoch. This incoming ancillary data stream spans the entire imaging interval containing the image data represented in the Level 0R/1R input data. In creating the model, we identify and extract the ancillary data sequence required to process the current image data.

The input Level 0R/1R image data are also packaged in HDF files that include the image samples for each band and SCA, and the time codes assigned to each image line by the OLI instrument. These spacecraft time codes are provided by the OLI in CCSDS T-Field format, which includes days since epoch (16-bit integer), milliseconds of day (32-bit integer), and microseconds of millisecond (16-bit integer) fields:



Figure 6‑20. OLI Time Code Format

Level 0 processing will combine these raw time code fields to compute the time since the spacecraft epoch in the form: days since spacecraft epoch and seconds of day. Since they are derived from the spacecraft clock, the image time codes will be based on the same epoch used by the ancillary data (e.g., TAI seconds from J2000). Even though the initial time code conversion will occur in Level 0 processing, for completeness the processing described below begins with the raw time code fields shown in Figure 6‑20.

###### Process Image Time Codes

The image time codes are loaded from the input HDF Level 0R/1R data set. Even/odd detector alignment fill may be inserted into the Level 0R/1R imagery, as described above, so the image lines each contain samples collected at times that may be offset from the time specified by the corresponding time code. The relationship between these time codes, the OLI integration times, and the multispectral and panchromatic pixel center times has already been described above. The L0Rp data will contain one time code per multispectral image line, excluding fill, or a nominal 6701 time codes per scene. The image files themselves may be up to 10 lines longer to accommodate the even/odd detector alignment fill.

A defect in the OLI timing logic can lead to erroneous time codes being generated when the microseconds or milliseconds fields fail to roll over properly. In the first case, the microseconds field can reach 1000 and increment the millisecond field without rolling over to zero. In the second case, the milliseconds field can reach 86400000 and increment the day field without rolling over to zero. Though these errors should be detected and corrected during Level 0 processing, the following time code validation logic will detect and correct this effect, as well as other suspicious time codes.

1. Convert the time codes to seconds from spacecraft epoch:

Line\_time = TC\_Day\*86400 + TC\_MSec/1000 + TC\_Micro/1e6

Note that an IEEE 754 double precision (64-bit) number with a 52-bit fraction should provide sufficient precision to represent time differences from 01JAN2000 to 01JAN2050 with microsecond accuracy (1.6e15 microseconds < 2^51).

1. Validate the image time codes as follows:
	1. Loop through the time codes from 1 to N-1, where N is the number of image data frames/time codes, and test the difference between the current and previous time codes against the nominal frame time from the CPF using the #define tolerance DTIME\_TOL. The first of two consecutive time codes that are within the tolerance is the first valid time code.
	2. Initialize the OLI clock model by setting the least squares variables to zero: A00 = A01 = A11 = L0 = L1 = 0
		1. Since the normal equation matrix, A, is symmetric, A10 = A01 so it is not computed separately.
		2. Add the first valid time code observation by adding 1 to A00. This is all that is required since, by definition, the index difference and time difference (see below) are zero at the first valid point.
	3. For each subsequent time code:
		1. Compare the time difference from the previous time code to the nominal value using the DTIME\_TOL threshold.
		2. If a time code fails this check, see if the special conditions of the known OLI time code defect apply:
			1. If the time code difference deviates from the nominal value by more than 0.5 milliseconds:
				1. If the time code microseconds field = 1000, subtract 1000.
				2. If the time code milliseconds field = 86400000 and the microseconds field = 0, set the milliseconds field to zero.
				3. Recalculate the time code difference.
		3. Compare the time code difference to a larger outlier tolerance (OUTLIER\_TOL) chosen to bound the possible drift in the OLI clock relative to the spacecraft clock (currently set to 50 microsec).
		4. If the time code difference is within the outlier range, add the current time to a least squares linear OLI clock model:
			1. num = current index – first valid index
			2. time = current time – first valid time
			3. Accumulate:
				1. Valid point count: A00 += 1
				2. Index difference: A01 += num
				3. Squared index diff: A11 += num\*num
				4. Time difference: L0 += time
				5. Time diff\*index diff: L1 += num\*time
	4. Once all time codes have been analyzed, solve for the linear OLI clock model parameters:
		1. determinant = A00\*A11 – A01\*A01
		2. If abs(determinant) <= 0.0, return an error
		3. Offset = first valid time + (A11\*L0 – A01\*L1) / determinant
		4. Rate = (A00\*L1 – A01\*L0) / determinant
	5. Use the correction model to replace bad time codes:
		1. For each time code:
			1. Calculate the corresponding model time as:

Mtime = Offset + (current index – first valid index) \* Rate.

* + - 1. Calculate the actual time – model time difference.

Diff = abs( time code – Mtime )

* + - 1. Test the difference against DTIME\_TOL.
			2. If the difference exceeds DTIME\_TOL, replace the current time code with the model value, Mtime.
	1. If no valid time codes were found, return an error.
	2. Calculate the average observed frame time, delta\_time, by subtracting the first valid/corrected time code from the last valid/corrected time code and dividing by the number of time code minus one.
	3. Store delta\_time (MS frame time) and delta\_time/2 (pan frame time) in the model.
1. Set the image start time: image\_start = line\_time[0].
2. Subtract the image start time from the line time codes so that the times are seconds from the image start.
3. Store the image start UTC epoch (image\_year, image\_day, image\_seconds) and the image line offset times in the model structure.
4. Report/trend the results of the time code processing, including the following:
	1. WRS Path/Row (input parameters)
	2. Image UTC epoch (year, day, seconds of day)
	3. L0R ID (input parameter)
	4. Work order ID (input parameter)
	5. Computed frame time (delta\_time)
	6. Number of replaced time codes (bad\_image\_time\_count)
5. Check the pan and MS detector integration times in the L0Rp frame header and if they are valid (> 0), convert them to units of seconds and load them in the model. Otherwise, use the nominal values from the CPF converted to units of seconds.
6. Load the detector settling times from the CPF into the model after converting them to units of seconds.

Extract Ancillary Ephemeris and Attitude Data

The subset of ancillary ephemeris and attitude data needed to span the image data are extracted from the Level 0R data by the ancillary data preprocessing algorithm. The logic to do the required subsetting is reiterated below for reference, since that phase 3 algorithm has not yet been released. These data are read from the input preprocessed ancillary data stream and stored in the model structure during model creation.

The ephemeris data extraction/subsetting procedure is as follows:

1. Compute the time offset from the ephemeris epoch time to the desired ephemeris start time for this image.

ephem\_start = image\_seconds – ancillary\_overlap – ephem\_seconds

Noting that image\_seconds and ephem\_seconds are the seconds of day fields from the image and ephemeris epoch times, respectively, and ancillary\_overlap is the desired extra ancillary data before and after the image window (set in a #define statement).

1. Loop through the ephemeris sample times to find the last entry that does not exceed ephem\_start. This is the ephemeris start index (eph\_start\_index).
2. Compute the time offset from the ephemeris epoch time to the desired ephemeris stop time for this image.

ephem\_stop = image\_seconds + line\_time[N-1] + ancillary\_overlap – ephem\_seconds

1. Loop through the ephemeris sample times to find the first entry that exceeds ephem\_stop. This is the ephemeris stop index (eph\_stop\_index).
2. Compute a new ephemeris UTC epoch for this image:

imgeph\_year = ephem\_year

imgeph\_day = ephem\_day

imgeph\_seconds = ephem\_seconds + ephem\_samp\_time[eph\_start\_index]

1. Load the ECI and ECEF ephemeris samples from eph\_start\_index to eph\_stop\_index (inclusive) into the preprocessed ancillary data output, adjusting the sample times so that they are offset from the UTC epoch computed in step 5.

The attitude data extraction/subsetting procedure is as follows:

1. Compute the time offset from the attitude epoch time to the desired attitude start time for this image.

att\_start = image\_seconds – ancillary\_overlap – att\_seconds

Noting that image\_seconds and att\_seconds are the seconds of day fields from the image and attitude epoch times, respectively.

1. Loop through the attitude sample times to find the last entry that does not exceed att\_start. This is the attitude start index (att\_start\_index).
2. Compute the time offset from the attitude epoch time to the desired attitude stop time for this image.

att\_stop = image\_seconds + line\_time[N-1] + ancillary\_overlap – att\_seconds

1. Loop through the attitude sample times to find the first entry that exceeds att\_stop. This is the attitude stop index (att\_stop\_index).
2. Compute a new attitude UTC epoch for this image:

imgatt\_year = att\_year

imgatt\_day = att\_day

imgatt\_seconds = att\_seconds + att\_samp\_time[att\_start\_index]

1. For Earth-view acquisitions, load the roll-pitch-yaw samples from att\_start\_index to att\_stop\_index (inclusive) into the preprocessed ancillary data output, adjusting the sample times so that they are offset from the UTC epoch computed in step 5.
2. For lunar/stellar acquisitions, convert the ECI quaternion samples from att\_start\_index to att\_stop\_index (inclusive) to ECI roll-pitch-yaw values, as described below, and store the computed roll-pitch-yaw values in the output, adjusting the sample times so that they are offset from the UTC epoch computed in step 5.

Converting ECI Quaternions to Roll-Pitch-Yaw

For lunar and stellar acquisitions, the ECI attitude representation is stored in the model structure. In the baseline model, this is done by converting the ECI quaternions to roll-pitch-yaw values relative to the ECI axes. This is one of the motivations for considering a transition to using a quaternion attitude representation in the model in the future.

The ECI quaternions are converted to roll-pitch-yaw values as follows:

1. Compute the rotation matrix corresponding to the ECI quaternion values:

**M**ACS2ECI =



1. Compute the corresponding ACS to ECI roll-pitch-yaw values:



Note that in implementing these calculations, it is important to use the ATAN2 rather than the ATAN arctangent implementation in order to retain the correct quadrants for the Euler angles. This is not a concern in Earth-view imagery where the angles are always small, but becomes an issue for these lunar/stellar ACS to ECI angles.

1. Store the ECI roll-pitch-yaw values in the model attitude data table.

At the completion of this sub-algorithm, the model structure contains the image frame time stamps, the multispectral and panchromatic sample, integration, and settling times, the ancillary ephemeris data, in both ECI and ECEF representations, covering the image, and the ancillary attitude data covering the image.

Jitter Correction Data Preprocessing

Jitter correction preprocessing operates on the roll-pitch-yaw attitude data stream extracted from the spacecraft ancillary data to separate the low-frequency spacecraft pointing effects from the higher-frequency jitter disturbances. The low-frequency pointing model is used for line-of-sight projection and other geolocation processing, while the high-frequency jitter effects are applied as per-line corrections during image resampling. To implement this frequency separation in the line-of-sight model, the original attitude sequence is passed through a low-pass filter with a cutoff frequency defined as a parameter in the CPF. This cutoff frequency will nominally be in the 1 Hz to 10 Hz range. The value ultimately selected for this cutoff frequency will depend on the actual disturbance profile observed in the spacecraft attitude data. The high-frequency data stream should be limited in magnitude to subpixel (ideally sub-half-pixel) effects, but the lower the cutoff frequency can be, the sparser (and smaller) the OLI resampling grid can be made in the line (time) dimension.

The low-pass filtered version of the attitude sequence is differenced with the original data to construct the complementary high-pass data sequence. The high-pass sequence is then interpolated at the image line times for the OLI panchromatic band to provide a table containing high-frequency roll-pitch-yaw corrections for each image line. This jitter table is stored in the OLI line-of-sight model. The original attitude sequence in the line-of-sight model is replaced with the low-pass filtered sequence to avoid double counting the high-frequency effects. Figure 6‑21 depicts this process.

****

Figure 6‑21. Jitter Correction Table Generation Data Flow

The jitter table construction processing sequence is as follows:

1. Extract a copy of the original attitude data sequence from the OLI line-of-sight model.
2. Retrieve the low-pass filter cutoff frequency from the CPF.
3. Design a low-pass filter with the desired cutoff frequency and apply it to the attitude data.
	1. Use the cutoff frequency and attitude data sampling time to compute the size of the desired filter, as follows:
		1. Compute the normalized cutoff frequency (the ratio of the cutoff frequency to the attitude data sampling frequency):

n\_cutoff = cutoff\_frequency / attitude\_sample\_frequency

 Note that this is the same as:

n\_cutoff = cutoff\_frequency \* attitude\_sample\_time

* + 1. Compute the number of samples per cycle at the cutoff frequency:

 Nsamp = 1 / n\_cutoff

* + 1. Multiply the number of samples per cycle by 3 and add 1 to yield the desired filter size:

FSize = 3\*Nsamp + 1

* + 1. If this results in an even filter size, add one:

If ( FSize modulo 2 == 0 ) FSize = FSize + 1

* 1. Use the Remez exchange algorithm to design the filter and generate the filter weights. The standard Parks-McClellan Finite Impulse Response (FIR) digital filter design method uses the Remez exchange algorithm (ref. Theory and Application of Digital Signal Processing, Rabiner and Gold, Prentice-Hall, 1975). A C implementation of this algorithm called remez.c, authored by Jake Janovetz at the University of Illinois, is available as shareware. This implementation specifies the desired (low pass, in this case) filter response using the following parameters:
		1. Filter size (number of taps) – FSize computed in item a. above.
		2. Number of frequency bands to use – 2, one pass band (low frequency) and one stop band (high frequency).
		3. Band frequency bounds – 0 to the normalized cutoff frequency (n\_cutoff) for the pass band and 1.5\*n\_cutoff to 0.5 (normalized Nyquist frequency) for the stop band.
		4. Desired band gains – 1 for pass band (low) and 0 for stop band (high).
		5. Band weights (how tightly to constrain the actual filter response to the design filter response in each band) – 1 for pass band and 10 for stop band.
		6. Filter type – BANDPASS (the low-pass filter is a special case of the more general BANDPASS filter type supported by the remez algorithm.
	2. Make sure the synthesized filter is normalized (weights sum to 1) by adding the filter tap values and dividing each tap by the total.

sum = Σ h[i] where h[i] are the FSize filter taps.

h’[i] = h[i] / sum for i = 1 to FSize.

* 1. Convolve the filter with the roll-pitch-yaw attitude data one axis at a time:

 half\_size = FSize / 2

for index = 0 to num\_rpy – 1

 low\_roll[index] = low\_pitch[index] = low\_yaw[index] = 0

 for ii = -half\_size to half\_size

 if ( index + ii < 0 ) j = -index – ii

 else if ( index + ii < num\_rpy ) j = index + ii

 else j = 2\*num\_rpy – index - ii – 1

 low\_roll[index] += roll[j]\*h[ii+half\_size]

 low\_pitch[index] += pitch[j]\*h[ii+half\_size]

 low\_yaw[index] += yaw[j]\*h[ii+half\_size]

1. Subtract the low-pass filtered sequences from the original sequences to extract the high-frequency portion of the data, and transfer any residual bias (non-zero mean value) from the imaging portion of the high-frequency sequence to the low-frequency sequence:

roll\_bias = pitch\_bias = yaw\_bias = 0

att\_pts = 0

for index = 0 to nrpy–1

 high\_roll[index] = roll[index] – low\_roll[index]

 high\_pitch[index] = pitch[index] – low\_pitch[index]

 high\_yaw[index] = yaw[index] – low\_yaw[index]

 if ( image\_start\_time < attitude\_time[index] < image\_stop\_time )

 roll\_bias += high\_roll[index]

 pitch\_bias += high\_pitch[index]

 yaw\_bias += high\_yaw[index]

 att\_pts += 1

roll\_bias = roll\_bias / att\_pts

pitch\_bias = pitch\_bias / att\_pts

yaw\_bias = yaw\_bias / att\_pts

for index = 0 to nrpy-1

 high\_roll[index] -= roll\_bias

 low\_roll[index] += roll\_bias

 high\_pitch[index] -= pitch\_bias

 low\_pitch[index] += pitch\_bias

 high\_yaw[index] -= yaw\_bias

 low\_yaw[index] += yaw\_bias

1. Interpolate the high-frequency sequence values at the panchromatic band line sampling times to create the model jitter table:

For each panchromatic image line = 0 to number of pan lines:

 Compute the line sampling time as the following:

 index = line / 2

 pan\_time = line\_time\_stamp[index] - pan\_settle\_time

 - pan\_integration\_time/2

 + (line - 2\*time\_index)\*pan\_sample\_time

 Convert to time from attitude epoch:

 pan\_time += image\_epoch – attitude \_epoch

Interpolate high frequency roll-pitch-yaw values at this time using four-point Lagrange interpolation:

Compute starting index for interpolation:

 index = floor(pan\_time / attitude\_sample\_time) – 1

 Compute the fractional sample offset to the pan line time:

 w = pan\_time / attitude\_sample\_time – index – 1

 Compute the Lagrange weights:

 w1 = -w \* (w – 1) \* (w – 2) / 6

 w2 = (w + 1) \* (w – 1) \* (w – 2) / 2

 w3 = -w \* (w + 1) \* (w – 2) / 2

 w4 = (w + 1) \* w \* (w – 1) / 6

Interpolate:

roll = high\_roll[index]\*w1 + high\_roll[index+1]\*w2

 + high\_roll[index+2]\*w3 + high\_roll[index+3]\*w4

pitch = high\_pitch[index]\*w1 + high\_pitch[index+1]\*w2

 + high\_pitch[index+2]\*w3 + high\_pitch[index+3]\*w4

yaw = high\_yaw[index]\*w1 + high\_yaw[index+1]\*w2

 + high\_yaw[index+2]\*w3 + high\_yaw[index+3]\*w4

1. Replace the original model attitude data sequence with the low-pass filtered attitude data sequence.

###### Process LOS Model Sub-Algorithm

This function loads the LOS Legendre polynomial coefficients and other model components from the CPF, and performs additional processing on the attitude and ephemeris information in the LOS model structure. It invokes the following sub-algorithms.

###### Read CPF Model Parameters Sub-Algorithm

This function loads model components from the CPF. In the heritage ALIAS implementation, some of these model components either did not exist (e.g., instrument offset from the spacecraft center of mass) or were used for image resampling but not LOS model computations (e.g., detector offset table), and so were not included in the model. These are included in the OLI model to make it self-contained for purposes of line-of-sight computations.

Key CPF parameters loaded into the geometric model include the following:

* Earth orientation parameters – the UT1UTC and pole wander (x,y) parameters for the current day are stored in the model to avoid the necessity of repeatedly looking them up in the CPF. WGS84 ellipsoid parameters (semi-major and semi-minor axes and eccentricity) are also extracted from the CPF, as are physical constants such as the Earth rotation velocity and the speed of light.
* OLI offset from spacecraft center of mass – a 3-vector that captures the small offset, in spacecraft body coordinates, between the OLI instrument, where images are captured, and the spacecraft center of mass, the position of which is reported in the ancillary ephemeris data, making it possible to translate the ephemeris data to the OLI. Technically, this would be the vector from the spacecraft center of mass to the center of the OLI entrance pupil. Note that this formulation assumes that the spacecraft on-board GPS data processing includes the GPS to spacecraft Center of Mass (CM) offset and that the spacecraft is, in fact, reporting CM positions, not GPS antenna positions. If the ephemeris represents the GPS antenna location, then we would need to know the spacecraft CM to GPS antenna offset as well.
* OLI to ACS alignment matrix – a 3-by-3 matrix that captures the relative orientation of the OLI coordinate system to the ACS coordinate system, making it possible to rotate the OLI instrument-space line-of-sight vectors into the ACS reference system. In the heritage ALIAS system, this was actually represented in the CPF by an ACS-to-instrument-rotation matrix, which was inverted for each LOS model invocation. Whichever convention is used in the CPF, the LOS model should store the OLI-to-ACS rotation matrix.
* OLI sensor parameters, including the nominal detector sampling rate, integration times (pan and MS), settling times (pan and MS), and Instantaneous Fields Of View (IFOVs), as well as the number of bands, SCAs per band, detectors per SCA, and nominal detector fill values.
* OLI line-of-sight Legendre polynomials – a set of 6 coefficients (3 along-track and 3 across-track) for each band on each SCA. Each set of 3 forms a 2nd order Legendre polynomial that is used to evaluate a nominal LOS angle (along- or across-track) for the detectors in that band on that SCA. This is the heritage ALIAS implementation (see the Read LOS Vectors Sub-Algorithm description below).
* OLI detector delay table – a table consisting of two values (along- and across-track) per detector reflecting the offset of each actual detector from its nominal location (as modeled by the 2nd order Legendre polynomials – see below). In the heritage ALIAS implementation, these were small subpixel offsets that were applied in the image resampling procedure. With the OLI, this table will also contain the even/odd detector offsets, as well as any offsets due to detector deselect (i.e., the operational use of a detector from one of the redundant rows). The even/odd offset had been modeled separately as a single value for each band, but the possibility of per-detector deselect offsets led to their inclusion in the per-detector offset table. This table is therefore needed in those LOS projection algorithms that use either actual (whole pixel offsets) or exact (full subpixel offsets) detector locations.

###### Read LOS Vectors Sub-Algorithm

This function retrieves the line-of-sight vectors from the CPF. The line-of-sight vectors are stored as sets of 2nd order Legendre polynomial coefficients. There is a unique set of 6 coefficients for each band of each SCA, 3 for the along-track polynomial and 3 for the across-track polynomial. These values are read from the CPF and stored in the LOS model. The polynomials are used to compute along- and across-track viewing angles for each nominal detector.

###### Initialize the Precision Model Sub-Algorithm

This function initializes the precision LOS correction model parameters. If the optional precision model input parameters are provided, those values are used. In the normal case, those parameters are absent and the correction model is initialized as follows:

Set the precision correction reference time to the beginning of the scene:

 t\_ref = 0.0

Set the ephemeris correction model order to zero: eph\_order = 0

Set both ephemeris X correction parameters to zero:

 x\_corr[0] = 0.0, x\_corr[1] = 0.0

Set both ephemeris Y correction parameters to zero:

 y\_corr[0] = 0.0, y\_corr[1] = 0.0

Set both ephemeris Z correction parameters to zero:

 z\_corr[0] = 0.0, z\_corr[1] = 0.0

Set the attitude correction model order to zero: att\_order = 0

Set all three attitude roll correction parameters to zero:

 roll\_corr[0] = 0.0, roll\_corr[1] = 0.0, roll\_corr[2] = 0.0

Set all three attitude pitch correction parameters to zero:

 pitch\_corr[0] = 0.0, pitch\_corr[1] = 0.0, pitch\_corr[2] = 0.0

Set all three attitude yaw correction parameters to zero:

 yaw\_corr[0] = 0.0, yaw\_corr[1] = 0.0, yaw\_corr[2] = 0.0

Note that these parameters are used to compute the corrected ephemeris and attitude data sequences, which are also stored in the model. The parameters themselves are included in the model primarily to document the magnitude of the corrections applied and to facilitate more advanced uses of the model creation logic. For example, it is sometimes useful to be able to force a particular model bias (e.g., a roll angle) into a model that is to be used for data simulation (see note 9). Therefore, though not strictly necessary for operational data processing, these parameters aid in anomaly resolution, data simulation, and algorithm development. In normal operations, these initial correction parameters are all zero and the "corrected" attitude and ephemeris data sequences are identical to the "original" attitude and ephemeris data prior to the execution of the LOS model correction algorithm. Subsequent algorithms (e.g., LOS projection) operate on the corrected data.

###### Correct Attitude Sub-Algorithm

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. In the model creation context, the precision corrections are zero, so the two sets of attitude data are identical. Although applying the precision corrections to construct the corrected attitude sequence could be said to be overkill for model creation (since the corrections are nominally zero at this point), this capability is required for LOS model correction and is used here to support the use of the model creation algorithm for data simulation and anomaly resolution, as it makes it possible to force initial biases into the model. This sub-algorithm will also be used by the LOS/precision correction algorithm to create the precision model. Note that the formulation is somewhat different for Earth-view scenes (Acquisition Type = Earth) than it is for lunar and stellar observations.

Earth Scenes

For Earth-view scenes, 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 as follows:

**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 is 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:

* It streamlines the application of the model for LOS projection by removing the step of explicitly applying the precision correction.
* 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 (at the center of the image time window)

Attitude model order: att\_order = 2

Roll bias and rate corrections: roll\_corr[] = roll\_bias, roll\_rate

Pitch bias and rate corrections: pitch\_corr[] = pitch\_bias, pitch\_rate

Yaw bias and rate corrections: yaw\_corr[] = yaw\_bias, yaw\_rate

The line-of-sight correction algorithm description describes this model in more detail.

* 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:
	1. 
	2. 
	3. 

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\_correction (r), pitch\_correction (p), and yaw\_correction (y) (**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.

Lunar and Stellar Scenes

For celestial (lunar or stellar) observations, 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**ECI = **M**ACS2ECI **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**ACS2ECI is the spacecraft attitude in the ECI frame derived from the ECI quaternions in the preprocessed ancillary data

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

The advantage of modeling the precision attitude corrections in ACS rather than orbital coordinates becomes apparent here, since the orbital frame is not used in the lunar case.

This sub-algorithm pre-computes the **M**ACS2ECI **M**Precision combination and stores the corresponding corrected attitude sequence (as roll-pitch-yaw values relative to ECI) in the model structure. Another difference between the Earth-view and lunar/stellar models is in the formulation of the precision model. The lunar attitude correction model adds an acceleration term to the Earth-view correction model parameters:

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

Attitude correction model order: att\_order = 3

Roll bias, rate, and acceleration corrections: roll\_corr[] = roll\_bias, roll\_rate, roll\_acceleration

Pitch bias, rate, and acceleration corrections: pitch\_corr[] = pitch\_bias, pitch\_rate, pitch\_acceleration

Yaw bias, rate, and acceleration corrections: yaw\_corr[] = yaw\_bias, yaw\_rate, yaw\_acceleration

Due to the different orders of the Earth-view and lunar correction models, this model is stored as an array in the model structure, along with a field defining the model order. The line-of-sight correction algorithm description describes the precision model in more detail.

The processing steps to construct the corrected attitude sequence are the same for lunar/stellar acquisitions, although the interpretation of the roll-pitch-yaw values is slightly different, and proceeds as follows:

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

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

**M**ACS2ECI =



1. Compute the precision correction at the time (t\_att = att\_seconds + att\_time[j]) corresponding to the attitude sample:
	1. 
	2. 
	3. 

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\_correction (r), pitch\_correction (p), and yaw\_correction (y):

**M**Precision =



1. Compute the composite rotation matrix: **M** = **M**ACS2ECI **M**Precision.
2. Compute the composite ACS to ECI roll-pitch-yaw values:



Note that in implementing these calculations, it is important to use the ATAN2 rather than the ATAN arctangent implementation in order to retain the correct quadrants for the Euler angles. This is not a concern in Earth-view imagery where the angles are always small, but becomes an issue for these lunar/stellar ACS to ECI angles.

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

###### Correct Ephemeris Sub-Algorithm

The heritage ALIAS function converts the ephemeris information (position and velocity) from the Earth-Centered Inertial (ECI J2000) system to the ECEF system and applies 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, the first portion of the heritage algorithm is no longer necessary (or could be reused in the ancillary data preprocessing algorithm). Though applying the precision corrections to construct the corrected ephemeris sequence could be said to be overkill for model creation (since the corrections are nominally zero at this point), this capability is required for LOS model correction and is used here to support the use of the model creation algorithm for data simulation and anomaly resolution, as it makes it possible to force initial biases into the model. This sub-algorithm will also be used by the LOS/precision correction algorithm to create the precision model.

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\_corr[0] + model precision x\_corr[1] \* dtime

dy orb = model precision y\_corr[0] + model precision y\_corr[1] \* dtime

dz orb = model precision z\_corr[0] + model precision z\_corr[1] \* dtime

where:

 model precision x\_corr[0] = precision (orbital) update to X position

 model precision y\_corr[0] = precision (orbital) update to Y position

 model precision z\_corr[0] = precision (orbital) update to Z position

 model precision x\_corr[1] = precision (orbital) update to X velocity

 model precision y\_corr[1] = precision (orbital) update to Y velocity

 model precision z\_corr[1] = precision (orbital) 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

###### Move Satellite Sub-Algorithm

This function computes the satellite position and velocity at a delta time from the ephemeris reference time using Lagrange interpolation. This is a utility sub-algorithm that accesses the model ephemeris data to provide the OLI position and velocity at any specified time. Since the model ephemeris arrays are inputs to this sub-algorithm, it will work with either the ECI or ECEF ephemeris data.

Table 6‑3 below summarizes the contents of the LOS model structure. The estimated size of this structure is approximately 1.5 megabytes.

|  |
| --- |
| **LOS Model Structure Contents** |
| Satellite Number (8) |
| Format Version Number (for documentation and backward compatibility) |
| WRS Path |
| WRS Row (may be fractional) |
| Acquisition Type (Earth, Lunar, Stellar) |
| Earth Parameters |
|  UT1UTC Correction (in seconds) |
|  Pole Wander X Correction (in arc seconds) |
|  Pole Wander Y Correction (in arc seconds) |
|  WGS84 Ellipsoid Semi-Major Axis (in meters) |
|  WGS84 Ellipsoid Semi-Minor Axis (in meters) |
|  WGS84 Ellipsoid Eccentricity (dimensionless) |
|  Earth Angular Velocity (radians/second) |
|  Speed of Light (meters/second) |
| Image Model |
|  Number of image lines |
|  Image UTC epoch: image\_year, image\_day, image\_seconds |
|  For each line: frame time offset (in seconds) from image epoch |
|  For each line: roll, pitch, yaw high frequency jitter correction (in radians) |
|  Nominal alignment fill table (from CPF) one value per band per SCA (in pixels) |
|  Detector alignment fill table (from L0R/L1R) one value per detector (in pixels) |
| Sensor Model |
|  OLI to ACS reference alignment matrix [3x3] |
|  Spacecraft center of mass to OLI offset in ACS reference frame [3x1] in meters |
|  Integration Times (MS and pan) in seconds |
|  Computed Sample Times (MS and pan) in seconds |
|  Detector Settling Times (MS and pan) in seconds |
|  Number of SCAs (14) |
|  Number of Bands (9) |
|  Along-Track IFOVs (MS and pan) in radians |
|  Across-Track IFOVs (MS and pan) in radians |
|  Number of Detectors per SCA Per Band (9x1 array) |
|  Focal plane model parameters (Legendre coefs) [NSCAxNBANDx2x3] (in radians) |
|  Detector delay table [NSCAxNBANDx2xNDET] (in pixels) |
| Ephemeris Model |
|  Scene ephemeris data UTC epoch: imgeph\_year, imgeph\_day, imgeph\_seconds |
|  Number of ephemeris samples |
|  Time from epoch (one per sample, nominally 1 Hz) (in seconds) |
|  Original ECI position estimate (X, Y, Z) (one set per sample) (in meters) |
|  Original ECI velocity estimate (Vx, Vy, Vz) (one set per sample) (in meters/sec) |
|  Original ECEF position estimate (X, Y, Z) (one set per sample) (in meters) |
|  Original ECEF velocity estimate (Vx, Vy, Vz) (one set per sample) (in meters/sec) |
|  Corrected ECI position estimate (X, Y, Z) (one set per sample) (in meters) |
|  Corrected ECI velocity estimate (Vx, Vy, Vz) (one set per sample) (in meters/sec) |
|  Corrected ECEF position estimate (X, Y, Z) (one set per sample) (in meters) |
|  Corrected ECEF velocity estimate (Vx, Vy, Vz) (one set per sample) (in meters/sec) |
| Attitude Model |
|  Scene attitude data UTC epoch: imgatt\_year, imgatt\_day, imgatt\_seconds |
|  Number of attitude samples |
|  Time from epoch (one per sample, nominally 50 Hz) (in seconds) |
|  Original Roll, pitch, yaw estimate (one per sample) (in radians) |
|  Corrected Roll, pitch, yaw estimate (one per sample) (in radians) |
| Precision Correction Model |
|  Precision reference time (t\_ref) seconds from image epoch |
|  Ephemeris correction order: eph\_order (0 none, 2 for Earth-view and lunar/stellar) |
|  X correction model: x\_bias, x\_rate (meters, meters/sec) |
|  Y correction model: y\_bias, y\_rate (meters, meters/sec) |
|  Z correction model: z\_bias, z\_rate (meters, meters/sec) |
|  Attitude correction order: att\_order (0 none, 2 for Earth, 3 for lunar/stellar) |
|  Roll correction model: roll\_bias, roll\_rate, roll\_acc (rad, rad/sec, rad/sec2) |
|  Pitch correction model: pitch\_bias, pitch\_rate, pitch\_acc (rad, rad/sec, rad/sec2) |
|  Yaw correction model: yaw\_bias, yaw\_rate, yaw\_acc (rad, rad/sec, rad/sec2) |

Table 6‑3. LOS Model Structure Contents

Note that in the precision correction model, only the first att\_order correction model array elements are valid. For example, for Earth-view scenes att\_order = 2 and roll\_corr[0] = roll\_bias, roll\_corr[1] = roll\_rate and roll\_corr[2] is not used.