### OLI Line-of-Sight Projection/Grid Generation Algorithm

#### Background/Introduction

The LOS projection and grid generation algorithm uses the OLI LOS model, created by the LOS model creation algorithm, to calculate the intersection of the projected lines-of-sight from selected OLI detector samples (pixels) with an Earth model (WGS84). The spacecraft position and pointing, OLI instrument alignment and offset information, and image timing data contained in the LOS model are used to construct the LOS for an individual OLI detector at a particular sample time. We then calculate the location where that line-of-sight intersects the Earth’s surface, as defined by the WGS84 Earth ellipsoid or a specified elevation above or below that ellipsoid. LOS intersections for an array of detector samples that span each OLI SCA/FPM and spectral band are computed at the WGS84 ellipsoid surface, as well as at a range of elevation levels selected to span the actual terrain elevations found in the image area. The resulting array of projected lines-of-sight forms a three-dimensional grid of input (Level 1R) image pixel line/sample to output space (Level 1G) mappings that can be used to interpolate input/output pixel mappings for intermediate points. The resulting ability to rapidly compute input/output mappings greatly facilitates image resampling.

The LOS projection and grid generation algorithm can also work in an “inertial direction” mode in which the output space is in angular units, with respect to a set of reference inertial directions. This mode is used to process lunar data wherein the inertial coordinates (declination and right ascension) of the moon, computed from a planetary ephemeris, are used as the reference to define the output image frame. In this case, the lines-of-sight are computed in inertial coordinates, but are not projected to the Earth’s surface.

Concerns about the temporal (line direction) grid density that would be required to adequately capture attitude deviations (jitter) at frequencies above 10 Hz motivated the addition of new grid functionality to support high-frequency image correction at image resampling time. Specifically, jitter sensitivity coefficients were added to each grid cell to allow the high-frequency attitude data in the OLI line-of-sight model jitter table to be converted to corresponding input image space line/sample offsets. These coefficients are used by the resampler to compute high-frequency line/sample corrections that refine the output-to-input space image coordinate mappings provided by the grid. This allows the grid to model only lower-frequency effects, making a sparser grid sampling in the time (line) direction possible.

Due to layout of the OLI focal plane, there are along-track offsets between spectral bands within each SCA, along-track offsets between even and odd SCAs, and a reversal of the band ordering in adjacent SCAs. This leads to an along-track offset in the imagery coverage area for a given band between odd and even SCAs, as well as an offset between bands within each SCA. To create more uniform image coverage within a geometrically corrected output product, the leading and trailing imagery associated with these offsets is trimmed (at image resampling time) based on image active area bounds stored in the grid.

#### Dependencies

The OLI LOS projection and grid generation algorithm assumes that the OLI LOS model creation algorithm has been executed to construct and store the OLI LOS model.

#### Inputs

The LOS projection and grid generation algorithm and its component sub-algorithms use 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).

|  |
| --- |
| **Algorithm Inputs** |
| ODL File (implementation) |
| CPF File Name |
| LOS Model File Name |
| DEM File Name |
| NOVAS Planetary Ephemeris File Name (for lunar processing) |
| Output Image Framing Parameters: |
| WRS Path for path-oriented scene framing (not necessarily the LOS model path) |
| WRS Row for path-oriented scene framing (not necessarily the LOS model row) |
| Map Projection (UTM, SOM, PS) |
| UTM Zone (use 0 to have code compute the zone) |
| Map Projection Parameters |
| Output Pixel Size(s) |
| Output Image Orientation |
| Frame Type (e.g., MINBOX) |
| Frame Bounds (e.g., corner coordinates, image size) |
| Grid Options: |
| Bands to Grid |
| CPF file contents |
| Thresholds and Limits (replaces System Table) |
| Grid Density (line/sample/height) |
| Default (WGS84) Spheroid parameter and Datum Codes |
| Scene framing band priority list |
| OLI LOS Model file contents (see LOS Model Creation ADD (6.2.1) for additional detail) |
| WGS84 Earth Ellipsoid parameters |
| Earth Angular Velocity (rotation rate) in radians/second |
| PAN and MS settling times |
| Speed of light (in meters/second) |
| Acquisition Type (Earth, Lunar, Stellar) |
| OLI to ACS reference alignment matrix |
| Spacecraft CM to OLI offset in ACS reference frame (new) |
| Focal plane model parameters (Legendre coefs) |
| Detector delay table |
| Smoothed ephemeris at 1 second intervals (original and corrected) |
| Low pass filtered attitude history (original and corrected) |
| High frequency attitude perturbations (roll, pitch, yaw) per image line (jitter table) |
| Image time codes |
| Integration Time (MS and Pan) |
| OLI MS and pan detector settling times (msec) |
| Nominal detector alignment fill table |
| L0R detector alignment Fill Table |
| DEM file contents |
| Min and Max Elevation |
| NOVAS Planetary Ephemeris file contents |
| JPL Ephemeris Table (DE405) for celestial bodies (i.e., the moon) (see note 1) |

#### Outputs

|  |
| --- |
| OLI Grid (see Table 6‑4 and Table 6‑5 below for detailed grid structure contents) |
| Grid Header (WRS path/row, acquisition type) |
| Output Image Framing Information (corner coordinates, map projection) |
| Image active area latitude/longitude bounds (for each band) |
| Grid Structure Information (number of bands/SCAs) |
| Grid Structures (one per SCA, per band) |
| Band number |
| Image dimensions (line/sample) |
| Pixel size |
| Grid cell size (image lines/samples per cell) |
| Grid dimensions (# rows/# columns/# Z-planes) |
| Z-plane zero reference and height increment |
| Arrays of input line/sample grid point coordinates |
| Arrays of output line and sample grid point mappings |
| Arrays of even/odd offset coefficients (2 per grid cell) |
| Arrays of forward (input/output) mapping polynomials (8 per grid cell per Z-plane) |
| Arrays of inverse (output/input) mapping polynomials (8 per grid cell per Z-plane) |
| Arrays of roll-pitch-yaw jitter line sensitivity coefficients (3 per grid cell per Z-plane) |
| Arrays of roll-pitch-yaw jitter sample sensitivity coefficients (3 per grid cell per Z-plane) |
| Rough mapping polynomials (one set per Z-plane) |

#### Options

A NOVAS planetary ephemeris file (JPL DE405) must be provided when the Acquisition Type (in the LOS model) is Lunar.

#### Prototype Code

Input to the executable is an ODL file; output is a HDF4 formatted resampling grid 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 oligrid 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.

**oligrid**

Main driver for generating the resampling grid. Calls modules to retrieve user parameters, establish the output image frame extent, and populate the grid structure with appropriate input to output, and output to input, mapping parameters.

**get\_parms**

This routine opens the input ODL parameter file, reads the grid parameters, closes the parameter file, and returns the parameters. It also will read the DEM, if the DEM is given as an input parameter, and determine the elevation extent within the DEM file. This elevation extent will then be used for establishing the z-plane parameters within the grid structure.

**oli\_get\_model**

Reads the OLI geometric model file and populates data within the OLI geometric model structure.

**read\_num\_ls\_l0ra**

This routine extracts the number of image lines from the Level 1R image and the number of samples per band per SCA from the sensor model portion of the LOS model. The routine then returns the number of lines and samples for the input band numbers. These values, along with the grid cell size, will be used to determine grid point locations. The number of lines and samples will be returned in their respective arrays, in band-referenced order. This is similar to the manner in which the grid is stored. Thus, the nlines and nsamps arrays must be of size nbands.

**det\_num\_grid\_ls**

This routine will determine the number of input points to be stored in the grid, according to the grid-sampling rate or grid cell size chosen.

**validate\_utm\_zone**

This routine validates the UTM zone that was entered as an ODL parameter. The scene center longitude will be used for this verification. The nominal UTM zone to use is computed from the scene center longitude, but the projection may be forced to an adjacent zone using input parameters. In particular, each WRS path/row may be preassigned to a UTM zone so the same zone is always used for scenes near UTM zone boundaries. This should not introduce a zone offset greater than 1. The validation is performed by computing the UTM zone in which the scene center falls and then determining whether the input UTM zone (if any) is within one zone of the nominal zone.

**oli\_malloc\_grid**

Allocates memory for the grid based on image size and output elevation extent.

**setup\_jpl\_solarsystem**

Initializes JPL routines needed to determine the position of the moon. This is only used for lunar acquisitions.

**calc\_active\_area**

This routine determines the bounds of that portion of the output image frame that contains actual OLI imagery, excluding "ragged" band/SCA edges. The resulting active area bounds for each spectral band are stored in the grid for subsequent use by the image resampling logic.

**north\_up**

This routine will determine the frame in output space for the north-up product. The actual frame is based on the output band's pixel size, but the frame is the same for every band. The method used to determine the scene corners depends on whether the corners were user input (PROJBOX) or calculated by projecting the Level 1R image corners (MAXBOX), but the framing logic is essentially the same in each case. Once given as input, or computed, the latitude/longitude scene corners are converted to the defined map projection, the extreme X and Y coordinates are found, and these extreme points are rounded to a whole multiple of the pixel size.

**calc\_stellar\_size**

Determines the output image extent for a stellar acquisition. Extent is based on SCA corners.

**calc\_lunar\_size**

Determines the output image extent for a lunar acquisition. Extent is based on either all of the SCA corners for all bands or only the SCA that contains the moon.

**point\_in\_polygon**

Simple point in a polygon check. Used with lunar process for determining if the moon lies within a SCA.

**oli\_moonpos\_ls**

Given a Level 1R line and sample location, this module calculates the relative line-of-sight between the moon and satellite sensor.

**oli\_moonpos**

Given a Julian day, this routine calculates the moon’s position. It calls the JPL NOVAS libraries to determine the moon’s position. Coordinates are given in terms of ECI true-of-date.

**maxbox**

This routine determines the frame in output space for the maxbox north-up product. Image framing is based on the maximum image extent derived from SCA corners.

**path\_oriented**

This routine provides a path-oriented projection that is framed to a nominal WRS scene. The user specifies only the projection, pixel size, and the path and row of the scene.

**det\_grid\_ls**

Given the number of grid lines and samples that will be sampled in the input imagery, this routine calculates where each grid cell point will fall in the input Level 1R image. These grid cell points will fall at integer locations in the input imagery.

**exx\_mapedg**

This routine calculates the minimum and maximum projection coordinates for the given upper-left and lower-right latitude, longitude coordinates.

**pad\_corners**

This routine pads the input corners by a defined factor of the pixel size. The x/y min and max values are input for the corner locations. These values are padded by PADVAL \* the pixel size.

**calc\_center\_and\_rotation\_angle**

This routine returns the scene center and rotation angle for a nominal WRS scene. The WRS path and row of the input scene and the projection parameters are needed as input. Note: The WRS\_Lat and WRS\_Long are the Center\_Lat\_Long that need to be returned from this routine. The Heading angle is the WRS rotation angle, i.e., the image orientation relative to geodetic north.

**calc\_path\_oriented\_frame**

Given the center point and rotation angle, this function calculates the image corner coordinates in an SOM or UTM product. It also calculates the first-order polynomial coefficients, which map output line/sample coordinates to their corresponding output projection coordinates. This routine will determine the frame in output space for the path-oriented product. The frame is calculated for each band, but the frame must be the same for every band.

**angle\_to\_map**

This routine converts the WRS rotation angle (from geodetic north) to a frame orientation angle in map coordinates. The orientation angle will be retained in the grid structure.

**path\_maxmin\_box**

This routine provides a path-oriented product whose frame is large enough to contain all bands (maxbox).

**calc\_path\_oriented\_maxbox\_frame**

This routine calculates the path-oriented frame for the maxbox approach.

**make\_grid**

This routine establishes the input-to-output mappings. It invokes make\_grid\_point for each point to compute the mapping, and then invokes make\_grid\_sensitivity for each point to compute the jitter sensitivity coefficients.

**make\_grid\_point**

Calculates the input-to-output space mapping for a single grid point. Calls oli\_forward\_model to perform input space location to output space location mappings.

**make\_grid\_sensitivity**

Calculates the roll-pitch-yaw to input space line/sample jitter sensitivity coefficients for one grid point. Calls oli\_forward\_model\_pert while varying the spacecraft attitude, the input space line number, and input space sample number to determine the corresponding output space sensitivity. It then finds the input space offsets that provide the same effect in output space as a given attitude perturbation, yielding the input space correction needed to compensate for a unit jitter disturbance for each spacecraft axis.

**oli\_init\_lunar\_projtran**

Initializes the position of the moon with respect the lunar acquisition. Needed for oli\_lunar\_projtran.

**oli\_forward\_model**

For a given a Level 1R line, sample, band, and SCA location, propagates the forward (geometric) model to determine a latitude and longitude for the specified point.

**oli\_forward\_model\_pert**

A variant of oli\_forward\_model that accepts an additional input roll-pitch-yaw attitude perturbation array. This perturbation is added to the spacecraft attitude interpolated from the OLI LOS model at the time corresponding to the input space line/sample point being projected. This capability is used by make\_grid\_sensitivity in determining the jitter sensitivity coefficients.

**oli\_findtime**

This function finds the time into the scene, given the Level 1R line, sample, and band. The input sample number is 0-relative and relative to the SCA.

**oli\_findlos**

This function finds the line-of-sight vector in sensor coordinates, using the Legendre polynomial LOS model stored in the LOS model.

**oli\_findatt**

This function computes the attitude, or roll, pitch, yaw, for a given time.

**oli\_findjit**

This function is invoked by oli\_forward\_model when the input detector type parameter is set to EXACT. This is currently only used by the OLI L0Rp data simulator. This unit uses the input time to extract the high-frequency attitude correction from the jitter table in the OLI LOS model, so that it can be added to the low-frequency spacecraft attitude result in oli\_forward\_model. This unit is not invoked by grid generation processing, where the detector type is NOMINAL, but as part of the forward line-of-sight model, it is described here for completeness.

**l8\_movesat**

This function computes the satellite position and velocity at a delta time from the ephemeris reference time using Lagrange interpolation.

**l8\_attitude**

This function finds the line-of-sight vector from the spacecraft to a point on the ground by transforming the line-of-sight vector in sensor coordinates to perturbed spacecraft coordinates.

**geo\_center\_mass\_corr**

Adjusts the observation vector according to the spacecraft center of mass.

**geo\_corr\_vel\_aberr**

Adjusts the line-of-sight vector for velocity aberration.

**geo\_findtarpos**

This function finds the position where the line-of-sight vector intersects the Earth's surface. Used only for Earth-based acquisitions.

**geo\_corr\_light\_travel\_time**

Adjusts the target location according to the light travel time. Used only for Earth-based acquisitions.

**geo\_centh2det**

This function converts between geocentric and geodetic coordinates. Used only for Earth-based acquisitions.

**exx\_cart2sph**

Converts between Cartesian and spherical coordinates. For grid generation, applies only towards stellar and lunar acquisitions.

**exx\_projtran**

This function converts coordinates from one map projection to another. The transformation from geodetic coordinates to the output map projection depends on the type of projection selected. The mathematics for the forward and inverse transformations for the UTM, Polar Stereo Graphic, and the Space Oblique Mercator (SOM) map projections are handled by USGS’s General Cartographic Transformation Package (GCTP), which may be obtained at [http://edcftp.cr.usgs.gov/pub/software/gctpc/](ftp://edcftp.cr.usgs.gov/pub/software/gctpc/).

**oli\_lunar\_projtran**

Calculates the output line and sample location, given the right ascension and declination angles associated with the sensor line-of-sight vector of a lunar acquisition. Serves as the equivalent exx\_projtran for a lunar-based acquisition.

**exx\_proj\_err**

This function reports projection transformation package errors. The function receives a GCTP error code and prints the correct error message.

**gctp**

Map projections are handled by USGS’s GCTP, which may be obtained at [http://edcftp.cr.usgs.gov/pub/software/gctpc/](ftp://edcftp.cr.usgs.gov/pub/software/gctpc/).

**xxx\_eval**

Applies a polynomial at a given point.

**calc\_map\_coefs**

This routine calculates the bilinear mapping coefficients for each grid cell. Coefficients are calculated for mapping from input location to output location (forward mapping) and for mapping from output location to input location (inverse mapping). A separate mapping function is used for lines and samples. This equates to four mapping functions. A set of four mapping functions is calculated for each grid cell, for each SCA, for every band, and for every elevation plane that is stored in the grid.

**exx\_calc\_forward\_mappings**

This function, given grid points in both input and output space, uses the Calculate Map Coefficients algorithm described in the Procedure subsection (6.2.2.7) to generate the mapping polynomial coefficients needed to convert from a line/sample in input space (satellite) to one in output space (projection). It generates these coefficients for every cell in the grid.

**exx\_calc\_inverse\_mappings**

This function, given grid points in both input and output space, uses the Calculate Map Coefficients algorithm described in the Procedure subsection (6.2.2.7)to generate the mapping polynomial coefficients needed to convert from a line/sample in output space (projection) to one in input space (satellite). It generates these coefficients for every cell in the grid.

**calc\_rough\_map\_coefs**

This routine finds the rough mapping coefficients for the grid.

**oli\_grid\_cell\_poly**

This utility function calculates a "rough" mapping of output-to-input lines/samples. The coefficients returned from this function are used as a rough estimate of an inverse model.

**calc\_det\_offsets**

This function computes the detector offset values and stores linear mapping coefficients associated with detector offsets in the grid structure.

**oli\_all\_ols2ils**

This utility routine maps an output space line/sample back into its corresponding input space line/sample. This is done using the "rough" polynomial from the grid to determine an initial guess at an input space line and sample. From this initial guess, a grid cell row and column is calculated, and the inverse coefficients for that cell are retrieved from the grid. These coefficients are used to determine an exact input space line and sample (in extended space).

**oli\_findgridcell**

This utility function finds the correct grid cell that contains the output line/sample location. It finds the correct grid cell containing the output pixel by first determining the set of grid cells to be checked. It then calls a routine to perform a "point in polygon" test on each of these grid cells to determine if the pixel does indeed fall within that grid cell.

#### Procedure

The LOS Projection algorithm uses the geometric LOS model created by the LOS Model Creation algorithm to relate OLI image pixels to ground locations, or in the case of lunar/stellar images, to ECI directions. The LOS model contains several components, including Earth orientation parameters, an image model (validated image time codes), a sensor model, an ephemeris model, and an attitude model. The Level 1R image line/sample location is used to compute a time of observation (from the image model), a LOS vector (from the sensor model), the spacecraft position (from the ephemeris model) at the time of observation, and the spacecraft attitude (from the attitude model) at the time of observation. The LOS vector is projected to the Earth's surface, either the topographic surface at a specified elevation (e.g., derived from an input Digital Elevation Model), or the WGS84 ellipsoid surface, to compute the ground position associated with that Level 1R image location. This LOS projection procedure relating an input image location to an output ground location is referred to as the forward model. In image resampling, we typically need to find the Level 1R input space line/sample location corresponding to a particular Level 1G output space location so that the corresponding image intensity can be interpolated from the Level 1R data. This "inverse model" computation must be performed for every pixel in the output Level 1G product. To make this computation efficient, we create a table, or grid, of input/output mappings, parameterized by height, for use by the image-resampling algorithm. This algorithm description document describes both the forward model and grid generation procedures.

##### The Geometric Grid

The geometric grid provides a mapping from input Level 1R line/sample space to output Level 1G line/sample space. As such, it incorporates not only the sensor LOS to Earth intersection geometry captured by the forward model, but also the output image framing information, such as scene corners, map projection, pixel size, image orientation, and the bounds of the active image area for each band. The gridding procedure generates a mapping grid that defines a transformation from the instrument perspective (input space) to a user-specified output projection on the ground (output space). This output frame may be map-oriented (north-up) or path-oriented for Earth-view acquisitions. Celestial (lunar/stellar) acquisitions use an output frame based on inertial right ascension and declination coordinates. Once the frame is determined in output space, the input space is gridded. Then, the grid in input space is mapped to the output space, using the forward model. Transformation coefficients to transform a grid cell from input to output space are determined, as well as coefficients to transform a grid cell from output to input space.

The concept behind creating this resampling grid is to define only a sparse set of points for the relationship between an input line and sample location to output line and sample location (see Figure 6‑22). Four grid points define a grid cell. A grid cell is defined as a rectangle in input space, but will be distorted when mapped to the output space. The sampling of points between grid cell points is chosen such that any two points defining a grid cell and a line in input space will map to a line in output space. Therefore, every grid cell defines a bilinear mapping between the input and output space, and vice versa. The method of only mapping and storing a small set of input points is much more efficient than trying to map points individually by invoking the LOS model for each point. This is especially the case since a rigorous implementation of the inverse model would have to be iterative.

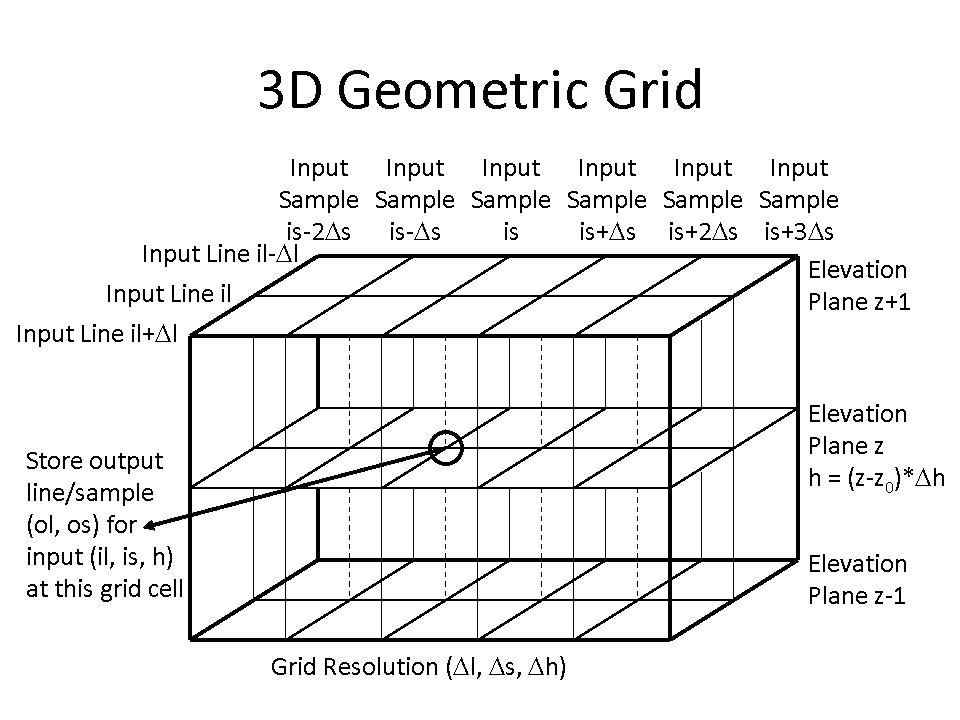


Figure 6‑22. 3D Grid Structure

The 3D grid structure stores the output space line/sample coordinates corresponding to an array of input space line/sample/height coordinates.

Input Space

Output Space

Forward mapping

Inverse mapping

Zn

Zn+1

Zn+2

Zn+3

Figure 6‑23. Forward and Inverse Mapping Using the Grid

The LOS projection grid contains projection information and three groups of mapping coefficients—one for mapping each grid cell from output space to input space (inverse), a second for mapping each grid cell from input space to output space (forward), and a third that gives an approximation or “rough” mapping of output space to input space. The first two mappings are described by a set of bilinear polynomials. The input space is represented by a line and sample location, while the output space is represented by a line and sample location, along with a Z component, where Z represents elevation. The output lines and samples can, in turn, be converted to X, Y projection space location by using the output image’s upper-left projection coordinate and pixel size information in the grid header. Figure 6‑24 shows how one input grid cell is mapped to a number of output grid cells, each grid cell representing a different elevation.

The number of grid cells is dependent on the line and sample size of each grid cell in the input image, elevation maximum, elevation minimum, and elevation increment. The input space is made up of evenly spaced samples and lines; values are associated with integer locations and can be indexed by an array of values: input\_line[row] and input\_sample[column]. Row refers to the index number, or row number, associated with the line spacing, while column refers to the index number, or column number, associated with the sample spacing. The output lines and samples typically do not fall on integer values (see Figure 6‑24). This creates a two-dimensional array of indices for output line and sample locations. Adding elevation indices produces a three-dimensional array for output line and sample locations. The output lines and samples are then indexed by output\_line[z][row][column] and output\_sample[z][row][column], where Z refers to an elevation value. The row and column are the indices associated with the gridding of the raw input space. Since there is a mapping polynomial for each grid cell, the mapping polynomial coefficients are indexed by the same method as that used for output lines and samples; i.e., there are z\*row\*column sets of mapping coefficients.



Figure 6‑24. Mapping Integer Locations to “Non-integer” Locations

If a grid is being generated for a non-terrain-corrected image (i.e., no correction for relief is being applied), then the index for z is set such that zelev=0 = zero elevation. Note that zelev=0 does not necessarily have to be the first index in the array since there could be values for negative elevations. If the grid is being generated for a terrain-corrected image, then the indexes zn and zn+1 are used such that the elevation belonging to the output location falls between the elevations associated with the indexes n and n+1. When performing an inverse mapping for a terrain-corrected image, two sets of input lines and samples are calculated from the polynomials for n and n+1. The actual input line and sample is interpolated between these lines and samples.

Example:

Output line/sample has r = row, c = col and z=n, n+1. If the inverse mapping coefficients are *a* and *b* for line and sample respectively, then:

input\_linen = *bilinear*(*a*n,output\_line,output\_sample)

input\_samplen = *bilinear*(*b*n,output\_line,output\_sample)

input\_linen+1 = *bilinear*(*a*n+1,output\_line,output\_sample)

input\_samplen+1 = *bilinear*(*b*n+1,output\_line,output\_sample)

*bilinear* is the bilinear mapping function (described below) for each grid cell.

If *e* is the elevation for the output line and sample location, then the weights used to interpolate between the two input line/sample locations are as follows:

*e*n, *e*n+1 and *e* are the elevations associated with zn , zn+1 , and the output line and sample, respectively.

The final line/sample location is found from the following:

input\_line = *wn* \* input\_linen + *wn+*1 \* input\_linen+1

input\_sample = *wn* \* input\_samplen + *wn+*1 \* input\_samplen+1

The grid must contain a zero elevation plane. If the input minimum elevation is greater than zero, it is set to zero. If the input maximum elevation is less than zero, it is set to zero.

Given the elevation maximum, minimum, and increment, determine the number of z planes and the index of the zero elevation plane. Adjust the minimum and maximum elevations to be consistent with the elevation increment.

The number of z planes is determined from the following:



The plane for an elevation of zero is then found at:



The new minimum and maximum elevation due to the values calculated above are as follows:





##### LOS Projection/Grid Generation Procedure Overview

The LOS Projection/Grid Generation procedure is executed in the following five stages:

1. Data Input - First, the required inputs are loaded. This includes reading the processing parameters from the input ODL parameter file, loading the LOS model from its HDF file, reading static gridding parameters from the CPF, and loading the elevation data from the DEM.
2. Scene Framing - The parameters of the output image space are computed based on the scene-framing scheme specified in the input ODL file. This includes calculating bounds for the active image area that excludes the leading and trailing SCA imagery, and using one of several available methods for determining the Level 1G scene corners. The scene-framing parameters are stored in the grid structure for eventual inclusion in the geometric metadata for the Level 1G product.
3. Grid Definition - The grid parameters are established to ensure adequate density in the space (sample), time (line), and elevation (z-plane) dimensions. The required data structures are allocated and initialized.
4. Grid Construction - The forward model is invoked for each grid intersection to construct the array of input space to output space mappings. A separate grid structure is created for each SCA and each band. The grid mapping polynomial coefficients are computed from the input space to output space mapping results for each grid cell. Once the basic grid mappings are defined, the forward model is invoked with small attitude perturbations about each axis in order to evaluate the sensitivity of the input space to output space mapping to small attitude deviations. The resulting sensitivity coefficients are stored with each grid cell for subsequent use in computing high-frequency jitter corrections during image resampling. Figure 6‑25 shows a high-level data flow for the creation and use of these new coefficients.
5. Finalize and Output Grid - Derived grid parameters, such as the global rough mapping coefficients, are added to the grid structure, and the entire structure is written to a disk file. This also includes evaluating the small, but significant, parallax effects caused by the time delay between when adjacent even and odd detectors sample the same along-track location. These effects are modeled in the grid as along- and across-track sensitivity coefficients that are scaled by the output point elevation and the even/odd detector offset, which can vary by pixel for OLI (due to detector deselect), rather than by band.



Figure 6‑25. Jitter Correction Data Flow

Figure 6‑26 shows a block diagram for the LOS Projection algorithm.

###### Stage 1 - Data Input

The data input stage involves loading the information required to perform grid processing. This includes reading the framing parameters for the output scene from the ODL file, reading grid structural parameters from the CPF, loading the LOS model structure in preparation for invoking the forward model, and reading the DEM to determine the elevation range for the image.



Figure 6‑26. Line-of-Sight Projection Block Diagram

###### Stage 2 - Scene Framing

Framing the output image space involves determining the geographic extent of the output image to be generated by the resampler. This geographic extent of the output image space is referred to as the output space “frame,” and is specified in output image projection coordinates. Four different methods are used to determine the output frame for Earth-viewing acquisitions. Scene framing for lunar and stellar scenes uses either a maximum bounding rectangle (maxbox) or a minimum bounding rectangle (minbox) approach using inertial LOS declination and right ascension coordinates, and is discussed separately. These methods use the calculated coverage bounds of each band/SCA in different ways, with some excluding the leading and trailing SCA imagery based on a calculated active image area, and some including the leading/trailing imagery so as to preserve all available input pixels (e.g., for calibration purposes). Thus, the calculation of the active image area for each band is the first step in scene framing.

Calculating the Active Image Area

The along-track offsets between spectral bands and even/odd SCAs create an uneven coverage pattern when projected into output image space. In order to provide a more regular output image coverage boundary, we define a rectangular active image area that excludes the excess trailing imagery from even SCAs and the excess leading imagery from odd SCAs. This active area is used for the minbox framing methods, which seek to limit the output product area to provide consistent, contiguous coverage, but are ignored for maxbox framing methods, where all available imagery is desired.

The active image area is computed by constructing 8 critical SCA corner points, labeled C1 through C8 in Figure 6‑27. Points C1 and C2 define the top edge of the active area, C3 and C4 the right edge, C5 and C6 the bottom edge, and C7 and C8 the left edge. Note that points C1 and C8 are the same (the upper-left corner of SCA01), as are points C4 and C5 (the lower-right corner of SCA14). The forward model projects these eight line/sample locations to object space, computing the latitude/longitude coordinates of the WGS84 ellipsoid intersection for each point.

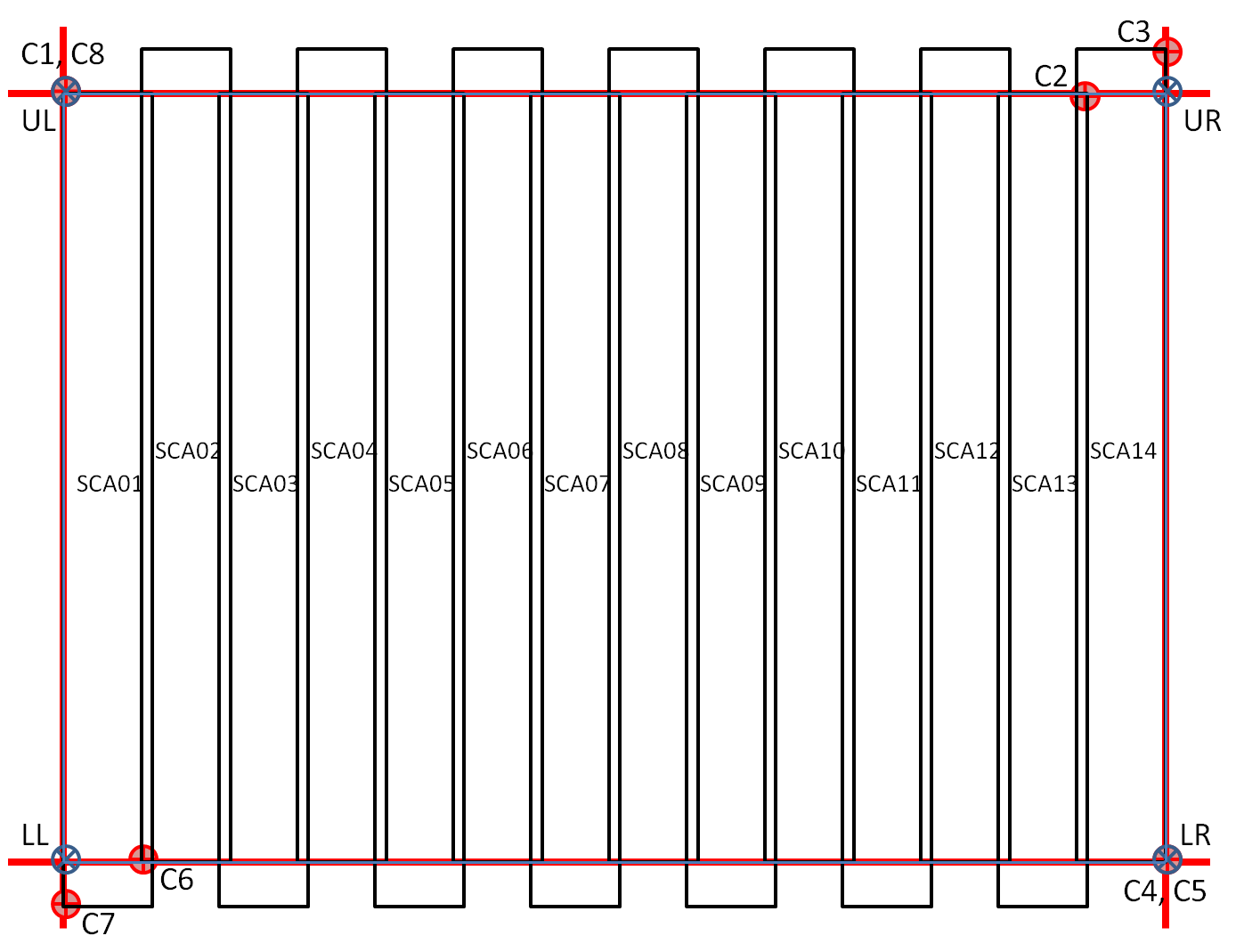


Figure 6‑27. Active Image Area Construction

The SCA and corner point assignments are made automatically by examining the SCA across-track and along-track Legendre coefficients to determine: 1) whether SCA01 is on the left (+Y) or right (-Y) side of the scene; 2) whether even or odd SCAs lead; and 3) whether the sample number increases in the –Y or +Y direction. If the across-track Legendre constant term (coef\_y0) for SCA01 is positive, then it is the left-most SCA and SCA14 is the right-most. If the along-track Legendre constant term (coef\_x0) for SCA01 is greater than that for SCA02, then the odd SCAs lead. If the across-track Legendre linear term (coef\_y1) for SCA01 is negative, then the sample number increases in the –Y direction.

Having determined the orientation of the SCAs, we assign the top edge of the active area to the left-most leading SCA UL corner and the right-most leading SCA UR corner, the right edge to the right-most SCA UR and LR corners, the bottom edge to the right-most trailing SCA LR corner and left-most trailing SCA Lower Left (LL) corner, and the left edge to the left-most SCA LL and UL corners. As shown in the figure, for the OLI: C1 = SCA01 (left-most odd SCA) UL, C2 = SCA13 (right-most odd SCA) UR, C3 = SCA14 (right-most SCA) UR, C4 = SCA14 (right-most SCA) LR, C5 = SCA14 (right-most even SCA) LR, C6 = SCA02 (left-most even SCA) LL, C7 = SCA01 (left-most SCA) LL, and C8 = SCA01 (left-most SCA) UL.

The geodetic latitudes computed by the forward model are converted to geocentric longitudes using the following:

 = arctan( (1-e2) tan() )

where:  = geocentric latitude

 = geodetic latitude

e2 = WGS84 ellipsoid eccentricity squared

This creates a set of 8 geocentric latitude/longitude (i, i) pairs, one for each “critical” corner, noting that geocentric longitude is equal to geodetic longitude.

Use the geocentric latitude/longitude to construct a geocentric unit vector for each corner:



Note that these vectors are inherently normalized.

Construct vectors normal to the top, right, bottom, and left edge great circles by taking cross products of the corner vectors:

Construct corner vectors from the edge vectors:

The top and bottom edges are next checked against all of the SCA corners to ensure that any curvature in the SCA field angle pattern is accounted for. This is done to suppress residual SCA edge “raggedness.”

Adjust the top edge:

Construct a vector in the plane of the top edge great circle:



Initialize the minimum “out of plane” distance: amin = 1

For each SCA:

For the two upper corners: UL (0,0) and UR (ns-1,0):

Use the forward model to project the corner.

Convert the geodetic latitude to geocentric latitude, as above.

Construct a geocentric unit vector, Xi, as above.

Project the unit vector onto the Xg and XT vectors and compute the ratio:



If ai < amin

amin = ai

Xmin = Xi

Next corner

Next SCA

If amin < 0, then the innermost corner lies inside the current active area, and we need to adjust the top edge:





Update the top corner vectors using the adjusted edge vectors:

Adjust the bottom edge:

Construct a vector in the plane of the bottom edge great circle:



Initialize the minimum “out of plane” distance: amin = 1

For each SCA:

For the two lower corners: LL (0,nl-1) and LR (ns-1,nl-1):

Use the forward model to project the corner.

Convert the geodetic latitude to geocentric latitude, as above.

Construct a geocentric unit vector, Xi, as above.

Project the unit vector onto the Xg and XB vectors and compute the ratio:



If ai < amin

amin = ai

Xmin = Xi

Next corner

Next SCA

If amin < 0, then the innermost corner lies inside the current active area, and we need to adjust the bottom edge:





Update the bottom corner vectors using the adjusted edge vectors:

Convert the four corner vectors to the corresponding geodetic latitude/longitude:

 = atan2( X.y, X.x )

 = atan2( X.z, )

 = atan( tan(  ) / (1-e2) )

The four latitude/longitude corners are the bounds of the active image area.

Once the active image area bounds are calculated, the output product frame is determined using one of the following methods:

*Method 1: PROJBOX*

The user defines the upper-left and lower-right corner coordinates of the area of interest in target map projection coordinates. These coordinates are then projected to the output projection coordinate system using the Projection Transformation Package (see the Projection Transformation sub-algorithm below). This usually results in a non-rectangular area, so a minimum-bounding rectangle is found (in terms of minimum and maximum X and Y projection coordinates) in the resulting output space. This minimum-bounding rectangle defines the output space frame. The output image pixel size is then applied to the projection space to determine the number of lines and samples in the output space. This creates an output image that is map projection north-up.

*Method 2: MINBOX*

The image active areas for each band, calculated previously, are converted to the specified output map projection coordinate system and used in a minimum bounding rectangle computation to create an output image frame that includes the active area for each band. The computed (latitude/longitude) active area corners are maintained in the grid for subsequent use by the image resampler, so that the output product image will not include leading/trailing SCA imagery.

*Method 3: MAXBOX*

The four corners of each SCA in each band are projected to the Earth. The maximum and minimum latitude and longitude found across all SCAs and all bands are used to establish the output scene frame in the manner described above for the PROJBOX method. This creates an output frame that contains all input pixels from all bands. The previously calculated image active areas are ignored in this process, and the band active area corners are all set equal to the output product corners. Leading and trailing SCA imagery is thereby not excluded from MAXBOX framed products.

*Method 4: PATH*

The user specifies a path-oriented Landsat product in either the SOM or UTM projection. In this case, the framing coordinates are not user-specified. The standard path-oriented frame is a preset number of lines and samples based on the Landsat WRS scene size and the maximum rotation needed to create a path-oriented product. Additional options exist to apply either MINBOX or MAXBOX logic in determining the path-oriented product frame.

*Method 5: LUNAR*

Lunar image framing applies either the same framing methodology as MAXBOX, defining the maximum and minimum corners in right ascension and declination angles with respect to the ECI coordinate system determined by the corners of all the SCAs for all bands, or with a similar framing methodology as MINBOX, determining the corners based solely on the SCA that contains the moon. The right ascension and declination angles are adjusted according to the change in orbit of the moon during image acquisition.

*Method 6: STELLAR*

Stellar image framing applies the same framing methodology as MAXBOX, only the output space frame defining the maximum and minimum corners are in right ascension, and declination angles with respect to the ECI coordinate system.

The scene framing logic uses the following sub-algorithms/routines:

**a)Validate UTM Zone**

The nominal UTM zone to use is computed from the scene center longitude, but the projection may be forced to an adjacent zone using input parameters. In particular, each WRS path/row may be preassigned to a UTM zone so that the same zone is always used for scenes near UTM zone boundaries. This should not introduce a zone offset greater than 1. The validation is performed by computing the UTM zone in which the scene center falls and then determining whether the input UTM zone (if any) is within one zone of the nominal zone.

Shift the scene center longitude to put it in the range 0-360 degrees:

SC\_long = mod( SC\_long + 540, 360 )

where: SC\_long is the scene center longitude in degrees

Compute the nominal UTM zone (note that UTM zones are six degrees wide):

SC\_zone = (int)floor( SC\_long/6 ) + 1

See if the input zone is within one zone of the nominal zone:

if ( abs( input\_zone - SC\_zone ) < 2 or (60 - abs( input\_zone - SC\_zone )) < 2 )

then input\_zone is valid.

**b) North Up Framing**

Determine the scene corners. Scene corners depend on whether the corners were user input (PROJBOX) or calculated by projecting the Level 1R image corners (MAXBOX), but the framing logic is essentially the same in each case. Once given as input or computed, the latitude/longitude scene corners are converted to the defined map projection, the extreme X and Y coordinates are found, and these extreme points are rounded to a whole multiple of the pixel size. The following sub-algorithms describe each of the north-up framing methods.

**b).1. Map Edge/PROJBOX Framing**

Calculates the minimum and maximum projection coordinates for given upper-left and lower-right latitude, longitude coordinates.

* Calculate the min/max coordinates along the east edge of the output area by computing latitude/longitude to map x/y projections for a series of points from (minimum latitude, maximum longitude) to (maximum latitude, maximum longitude).
* Calculate the min/max coordinates along the west edge of the output area by computing latitude/longitude to map x/y projections for a series of points from (minimum latitude, minimum longitude) to (maximum latitude, minimum longitude).
* Calculate the min/max coordinates along the south edge of the output area by computing latitude/longitude to map x/y projections for a series of points from (minimum latitude, minimum longitude) to (minimum latitude, maximum longitude).
* Calculate the min/max coordinates along the north edge of the output area by computing latitude/longitude to map x/y projections for a series of points from (maximum latitude, minimum longitude) to (maximum latitude, maximum longitude).

Note that since lines of constant latitude and/or longitude may be curved in map projection space, the extreme map x/y points may not correspond to the four PROJBOX corners.

**b).2. Minbox/Maxbox Framing** Determine the frame in output space for the minbox or maxbox north-up product. The actual frame is determined based on the optimal band's pixel size, but the frame is the same for every band.

**b).2.1 Minbox Framing** Calculate the MINBOX frame bounds, using the active area corner points for each band.

1. Call projtran (see below) to get the output map projected x/y for each active area corner point for each image band.
2. Find the minimum and maximum output proj x/y from the full set of active area corner points.
3. Pad the min and max output projection x/y to make them a multiple of pixsize.
4. Fill in the corners for the grid in the order of UL, LL, UR, LR, and Y/X coords.

UL = min x, max y

UR = max x, max y

LL = min x, min y

LR = max x, min y

1. Find the number of lines and samples for the grid, for each specified band number.

lines = (max y - min y)/pixsize + 1

samples = (max x - min x)/pixsize + 1

**b).2.2 Maxbox Framing** Calculate the MAXBOX product frame bounds, using the projected corners of each band/SCA.

1. Find the four image corners in input space for each SCA and band.

UL - (1, first\_pixel)

UR - (1, last\_pixel)

LL - (NLines, first\_pixel)

LR - (NLines, last\_pixel)

1. Call the forward model (see below) to get the output lat/long, for each corner point.
2. Call projtran (see below) to get the output map projected x/y, for each corner point.
3. Find the minimum and maximum output proj x/y from the full set of corner points.
4. Pad the min and max output projection x/y to make them a multiple of pixsize.
5. Fill in the corners for the grid in the order of UL, LL, UR, LR and Y/X coords.

UL = min x, max y

UR = max x, max y

LL = min x, min y

LR = max x, min y

1. Find the number of lines and samples for the grid, for each specified band number.

lines = (max y - min y)/pixsize + 1

samples = (max x - min x)/pixsize + 1

1. Call projtran to convert the map projection Y/X coordinates of the output product corners to latitude/longitude.
2. Replace the active area corner coordinates for each band with the converted output product corner coordinates.

**b).2.3. Pad Corners** Pad the input corners by a defined factor of the pixel size. The x/y min and max values are input for the corner locations. These values are padded by PADVAL \* the pixel size. The newly padded x/y min and max values are returned, replacing the original values.

ixmin = int (Xmin/(PADVAL\*pixsize))

Xmin = ixmin\*PADVAL\*pixsize

ixmax = int (Xmax/(PADVAL\*pixsize))+1

Xmax = ixmax\*PADVAL\*pixsize

iymin = int (Ymin/(PADVAL\*pixsize))

Ymin = iymin\*PADVAL\*pixsize

iymax = int (Ymax/(PADVAL\*pixsize))+1

Ymax = iymax\*PADVAL\*pixsize

**c) Path-oriented Framing**

Provide a path-oriented projection that is framed to a nominal WRS scene. The projection, pixel size, and the path and row of the scene must be defined.

**c).1. Calculate Center and Rotation Angle**

Calculate the scene center and rotation angle for a nominal WRS scene. The WRS path and row of the input scene and the projection parameters are needed as input. The nominal WRS scene center lat/long and rotation angle for the given projection are returned. The algorithm has the following steps:

Convert input angles to radians:

Inclination\_Angle\_R = Pi / 180 \* Inclination\_Angle

Long\_Path1\_Row60\_R = Pi / 180 \* Long\_Path1\_Row60

Compute the Earth's angular rotation rate:

earth\_spin\_rate = 2 \* Pi / (24 \* 3600)

Note: We use the solar rotation rate rather than the sidereal rate in order to account for the orbital precession, which is designed to make the orbit sun synchronous. Thus, the apparent Earth angular velocity is the inertial (sidereal) angular velocity plus the precession rate, which, by design, is equal to the solar angular rate.

Compute the spacecraft's angular rotation rate:

SC\_Ang\_Rate = 2 \* Pi \* WRS\_Cycle\_Orbits / (WRS\_Cycle\_Days\*24\*3600)

Compute the central travel angle from the descending node:

Central\_Angle = (Row - Descending\_Node\_Row)/Scenes\_Per\_Orbit\*2\*Pi

Compute the WRS geocentric latitude:

WRS\_GCLat = asin( -sin(Central\_Angle) \* sin(Inclination\_Angle\_R) )

Compute the longitude of Row 60 for this Path:

Long\_Origin = Long\_Path1\_Row60\_R - (Path-1) \* 2\*Pi/WRS\_Cycle\_Orbits

Compute the WRS longitude:

Delta\_Long = atan2( tan(WRS\_GCLat)/tan(Inclination\_Angle\_R),

cos(Central\_Angle)/cos(WRS\_GCLat) )

WRS\_Long = Long\_Origin - Delta\_Long - Central\_Angle \*

Earth\_Spin\_Rate / SC\_Ang\_Rate

Make sure the longitude is in the range +/- Pi:

While ( WRS\_Long > Pi )

WRS\_Long = WRS\_Long - 2\*Pi

While ( WRS\_Long < -Pi )

WRS\_Long = WRS\_Long + 2\*Pi

Compute the scene heading:

Heading\_Angle = atan2( cos(Inclination\_Angle\_R)/cos(WRS\_GCLat),

-cos(Delta\_Long)\*sin(Inclination\_Angle\_R) )

Convert the WRS geocentric latitude to geodetic latitude:

WRS\_Lat = atan( tan(WRS\_GCLat) \* (Semi\_Major\_Axis/Semi\_Minor\_Axis) \*

(Semi\_Major\_Axis/Semi\_Minor\_Axis) )

Convert angles to degrees:

WRS\_Lat = WRS\_Lat \* 180 / Pi

WRS\_Long = WRS\_Long \* 180 / Pi

Heading\_Angle = Heading\_Angle \* 180 / Pi

Round WRS lat/long off to the nearest whole arc minute:

WRS\_Lat = round( WRS\_Lat\*60 ) / 60

WRS\_Long = round( WRS\_Long\*60 ) / 60

**c).2. Calculate Path-Oriented Frame**

Calculate the center point and rotation angle, and the image corner coordinates in an SOM or UTM projection. Also calculate the first-order polynomial coefficients, which map output line/sample coordinates to their corresponding output projection coordinates. Determine the frame in output space for the path-oriented product. Calculate the frame for each band. The frame must be the same for all bands.

**c).2.1. Angle to Map**

Convert the WRS rotation angle (from geodetic north) to a frame orientation angle in map coordinates. The following is an algorithm to compute this:

Convert the WRS scene center latitude/longitude to map projection x/y (X1, Y1) using the projtran routine.

Add 1 microradian (0.2 seconds) to the WRS scene center latitude and convert this point to map projection x/y (X2, Y2).

Compute the azimuth of this line in grid space as the arctangent of (X2-X1)/(Y2-Y1). This is the grid azimuth of geodetic north at the WRS scene center.

Add this angle to the WRS rotation angle to give the grid heading. A standard framed scene puts the satellite direction of flight at the bottom of the scene, so the scene orientation angle is the grid heading + or - 180 degrees. If the grid heading is <0, then subtract 180 degrees. If the grid heading is >0, then add 180 degrees. This is the scene orientation angle to use with the WRS scene center.

**c).2.2. Path-oriented Minbox/Maxbox Frame**

Calculate the path-oriented frame that is large enough to contain all bands.

**c).2.2.1. Calculate the Path-oriented Minbox Frame**

Calculate the path-oriented frame for the minbox approach.

1. Compute the map projection coordinates of the four image active area corners for each band, as described in step 1 of Minbox Framing.
2. Offset and rotate the scene corners to the path-oriented frame using the WRS scene center map projection coordinates (X1, Y1) and orientation angle:
   1. X' = (X - X1) cos(angle) - (Y - Y1) sin(angle) + X1
   2. Y' = (X - X1) sin(angle) + (Y - Y1) cos(angle) + Y1
3. Compute the minbox frame as described in steps 2-4 of Minbox Framing.
4. Convert the rotated minbox corners back to the unrotated map projection coordinate system:
   1. X = (X' - X1) cos(angle) + (Y' - Y1) sin(angle) + X1
   2. Y = -(X' - X1) sin(angle) + (Y' - Y1) cos(angle) + Y1

**c).2.2.2. Calculate Path-oriented Maxbox Frame**

Calculate the path-oriented frame for the maxbox approach.

1. Compute the map projection coordinates of the four image corners for the optimal band, as described in steps 1-3 of Maxbox Framing.
2. Offset and rotate the scene corners to the path-oriented frame, using the WRS scene center map projection coordinates (X1, Y1) and orientation angle:
   1. X' = (X - X1) cos(angle) - (Y - Y1) sin(angle) + X1
   2. Y' = (X - X1) sin(angle) + (Y - Y1) cos(angle) + Y1
3. Compute the maxbox frame, as described in steps 4-6 of Maxbox Framing.
4. Convert the rotated maxbox corners back to the unrotated map projection coordinate system:
   1. X = (X' - X1) cos(angle) + (Y' - Y1) sin(angle) + X1
   2. Y = -(X' - X1) sin(angle) + (Y' - Y1) cos(angle) + Y1
5. Call projtran to convert the map projection Y/X coordinates of the output product corners to latitude/longitude.
6. Replace the active area corner coordinates for each band with the converted output product corner coordinates.

**d) Celestial Acquisitions**

Celestial acquisitions use the same framing logic as Earth acquisitions (namely maxbox), but the output space coordinate systems are sufficiently different to merit separate discussion. For both lunar and stellar acquisitions, the output space is defined in terms of directions in inertial space, defined by the ECI J2000 right ascension and declination of the OLI look vectors. In the case of stellar acquisitions, the output space "projection" uses the ECI J2000 right ascension and declination directly. For lunar acquisitions, the output coordinate system is modified to use the LOS right ascension and declination offset from the lunar right ascension and declination at the time of observation. This creates a slowly rotating coordinate system that tracks the moon and is the reason for having a planetary ephemeris file as an input to this algorithm. These differences emerge in the forward model computations for celestial acquisitions, where the LOS intersection logic used for Earth acquisitions is replaced by operations on the inertial lines-of-sight (after conversion to inertial right ascension and declination angles), with the resulting map projection x/y coordinates used in the Earth-view algorithms replaced by right ascension and declination (or delta-right ascension and delta-declination). Either the maxbox or minbox framing logic applied to the x/y map projection coordinates in Earth-view acquisitions is then applied to these angular celestial coordinates.

**e) Lunar Acquisitions**

Lunar acquisitions use either a MAXBOX or MINBOX framing type. For MAXBOX, the framing logic is the same as that used for Earth viewing acquisitions; determine bounding viewing angles based on all SCAs of all bands. For MINBOX, the minimum box, or viewing angles, is based on the SCA within which the moon resides. The bounding viewing angles for this SCA for all bands define the frame for the output image. The moon is found to reside within an SCA by using the moon’s coordinates, defined by the center of acquisition time of the scene, and checking to see if these coordinates fall within a SCA. A simple point in a polygon routine is used for the check:

If coordinate to check is defined as Xm and Ym

e1) Define rectangle using SCA corners, Xi and Yi.

e2) Find maximum Y coordinate of rectangle, Ymax.

e3) Define a new coordinate as the following:

Xn = Xm

Yn = Delta + Ymax

Where Delta is large enough to put point (Xn, Yn) outside of polygon

e4) Define a line from (Xm,Ym) to (Xn,Yn).

e5) Determine the number of times the line defined in e4) intersects sides of the rectangle from e1). If the number of intersections is an odd number, then the point is within the rectangle.

**Stage 3 - Grid Definition**

The grid definition stage determines the required size of the grid, allocates the grid structure, and computes the input space (Level 1R) line/sample locations for each grid cell.

**a) Determine the Number of Grid Input/Output Lines/Samples**

Determine the number of input points to be stored in the grid according to the grid-sampling rate or grid cell size chosen.

Loop through each band stored in the grid.

Loop through each SCA stored in the grid.

Calculate the number of lines and samples stored in the grid, according to the size of each grid cell and the size of the input image to be processed. Store the number of grid lines and samples calculated in the grid.

Calculate the number of times grid cell size divides into Level 1R imagery



where:

number of image lines = number of lines in Level 1R (LOS model)

number of detectors per SCA = number of samples per SCA (LOS model)

grid cell size line direction = number of lines in one grid cell

grid cell size sample direction = number of samples in one grid cell

If the grid cell size in the line direction does not divide evenly into the number of lines in the Level 1R, then increment the number of grid lines by one.

If the grid cell size in the sample direction does not divide evenly into the number of samples in the Level 1R, then increment the number of grid samples by one.

**b) Determine Grid Lines/Samples**

Determine where each grid cell point will fall in the input Level 1R image. These grid cell points will fall at integer locations in the input imagery.

Loop through each band that is stored in the grid.

Loop through each SCA stored in the grid.

Initialize the first grid cell line location to zero relative.

input line location grid cell0 = 0

Loop until the grid cell line location is greater than or equal to the number of Level 1R lines, incrementing each new grid cell line location by the appropriate grid cell size in the line direction for the current band and SCA.

input line location grid celln = input line location grid celln-1

+ grid cell size line direction

Set the last grid cell line location to the last line in Level 1R image.

input line location grid celllast = number of lines in Level 1R imagery

Initialize the first grid cell sample location to zero relative.

input sample location grid cell0 = 0

Loop until the grid cell sample location is greater than or equal to the number of Level 1R samples, incrementing each new grid cell sample location by the appropriate grid cell size in the sample direction for the current band and SCA.

input sample location grid celln = input sample location grid celln-1

+ grid cell size sample direction

Set the last grid cell sample location to the last sample in Level 1R image.

input sample location grid celllast = number of samples in Level 1R imagery

**Stage 4 - Grid Construction**

Once the grid structures are created (one per SCA per band), the forward model is evaluated at every grid intersection; that is, for every Level 1R line/sample location at every elevation plane. The forward model computes the WGS84 latitude/longitude coordinates associated with each input line/sample/height point. These latitude/longitude positions are then converted to output space line/sample by projecting them to map x/y, computing the offsets (and rotation if path-oriented) from the upper-left scene corner, and scaling the offsets from meters to pixels using the pixel size.

**a) Make Grid** Given the number of grid lines and samples that will be sampled in the input imagery, loop on each band of each SCA, loop on the number of z-planes, loop on the number of input grid lines and samples calculating the corresponding output line and sample location. For each input line, sample location, and elevation, the instrument forward model function is called. The steps below outline this forward model function. Additional detail on the sub-algorithms, which comprise the forward model, is provided in the subsection titled "Forward Model" later in this document.

The forward model uses the LOS model structure and the CPF to map an input line and sample location to an output geographic location. These are the steps performed whenever calculating an output geodetic latitude and longitude from an input line and sample by invoking the instrument “forward model.” The GCTP function can then be used to transform the geographic latitude and longitude to a map projection X and Y coordinate. If the output image has a “North up” orientation, then the upper-left projection coordinate of the output imagery and the output pixel size can be used to transform any projection coordinate to an output line and sample location. If the map projection space is in a rotated projection space, such as having a satellite path orientation, then a transformation handling rotation is established between the projection space and output pixel location. This transformation is then used in converting projection coordinates to output pixel line and sample locations.

The process listed below is performed on all bands, all elevation planes, and all SCAs present in the grid. The detector type used in the process is nominal (see the LOS Model Creation ADD (6.2.1) for a discussion of detector types). The list explains the actions taken if a detector type other than nominal is chosen, so that it can be referenced later.

Loop on the number of input grid lines.

Loop on the number of input grid samples.

Read the input space (Level 1R) line/sample coordinate for this grid point.

Loop on the number of elevation planes.

Compute the height of the current elevation plane:



where:

z is the index of the current z-plane and

zelev=0 is the index of the zero elevation z-plane.

Invoke the forward model to compute the corresponding ground position latitude/longitude for this point. The general steps of the forward model are described here and are presented in more detail below.

*Find Time*

Find the nominal time of the input sample relative to the start of the imagery. The LOS Model Creation ADD (6.2.1) describes this procedure, and it is presented below in the Find Time sub-algorithm description.

*Find LOS*

Find the LOS vector for the input line/sample location using the Legendre polynomial coefficients, as described below in the Find LOS sub-algorithm.

*Find Attitude*

Calculate the spacecraft attitude corresponding to the LOS, i.e., for the line/sample location, at the time computed above, using the Find Attitude sub-algorithm described below. Note that for Earth acquisitions, the roll-pitch-yaw attitude sequence in the LOS model is relative to the orbital coordinate system, whereas for celestial (lunar/stellar) acquisitions, the LOS model roll-pitch-yaw sequence is with respect to the ECI J2000 coordinate system. The operations applied by the Find Attitude sub-algorithm are the same in either case.

*Find Ephemeris*

Calculate the satellite position for the line/sample using Lagrange interpolation. Reference the move\_sat sub-algorithm described in the LOS Model Creation ADD (6.2.1) and repeated below. Note that for Earth acquisitions, the move\_sat sub-algorithm is provided with the corrected ECEF ephemeris data from the LOS model, whereas for celestial (lunar/stellar) acquisitions, it will be passed the corrected ECI ephemeris.

*Rotate LOS to ECEF (Earth-view) or ECI (Celestial)*

Use the OLI alignment matrix in the LOS model to convert the LOS vector from sensor to ACS/body coordinates. Then, apply the interpolated roll, pitch, and yaw to the LOS to convert ACS/body to orbital (Earth-view) or ECI (celestial). If Earth-view, use the ephemeris to construct the orbital to ECEF rotation matrix and use it to transform LOS to ECEF. The Attitude sub-algorithm below describes the procedure for Earth-view scenes. For celestial acquisitions, the procedure is complete once the LOS has been rotated to ECI using the roll-pitch-yaw perturbation matrix.

*Spacecraft Center of Mass to OLI Offset Correction*

Adjust the spacecraft position for the offset between the spacecraft center of mass and the OLI instrument. This offset, in spacecraft body coordinates, is stored in the LOS model structure. First, convert the offset from spacecraft body frame to ECEF using the attitude perturbation matrix (body to orbital) and the orbital to ECEF matrix:





Add the offset to the ECEF spacecraft position vector. This correction is not used for celestial (lunar/stellar) acquisitions.

*Correct LOS for Velocity Aberration*

The relativistic velocity aberration correction adjusts the computed LOS (ECEF for Earth-view and ECI for celestial) for the apparent deflection caused by the relative velocity of the platform (spacecraft) and target. The preparatory computations are somewhat different for Earth-view and celestial acquisitions due to the differences in target velocity.

*Earth-view Case*

The LOS intersection sub-algorithm described below (see Find Target Position) is invoked with an elevation of zero to find the approximate ground target position. The ground point velocity is then computed as:

**V**g =  × **X**g

where:

**V**g = ground point velocity

**X**g = ground point ECEF position

**** = Earth rotation vector = [ 0 0 e ]T

e = Earth rotation rate in radians/second (from CPF)

The relative velocity is then:

**V** = **V**s - **V**g

where Vs is the spacecraft ECEF velocity from the ephemeris data.

*Correcting the Earth-View LOS*

The LOS vector is adjusted based on the ratio of the relative velocity vector to the speed of light (from the CPF):

 where: **l** = uncorrected LOS and **l'** = corrected LOS

Note: In this case, the LOS velocity aberration correction is negative since we are correcting the apparent LOS to the true (aberration corrected) LOS. The correction is positive if we are computing the apparent LOS from the true (geometrical) LOS (see lunar case below).

*Celestial (Lunar/Stellar) Case*

Both lunar and stellar acquisitions use the spacecraft inertial velocity from the ephemeris data as the relative velocity. This is justified by the use of a lunar ephemeris (using the Naval Observatory's NOVAS-C package) that returns apparent places. The apparent location of the moon is already corrected for light travel time (see below) and velocity/planetary aberration due to the motion of the moon around the Earth. Thus, the residual aberration is due only to the motion of the spacecraft relative to the Earth. Thus, for both lunar and stellar acquisitions:

**V** = **V**s

where Vs is the spacecraft ECI velocity from the ephemeris data.

*Correcting the Celestial LOS*

For stellar acquisitions, the LOS is corrected for aberration in the same manner as for Earth-view scenes. For lunar acquisitions, rather than correct the LOS vector, we adjust the apparent location of the moon. The lunar vector is thus adjusted based on the ratio of the relative velocity vector to the speed of light (from the CPF) as the following:

 where: **l** = uncorrected LOS and **l'** = corrected LOS

The correction is positive in this case since we are computing an apparent location rather than correcting one.

*LOS Intersection*

For Earth-view acquisitions, intersect the LOS in ECEF with the Earth model as described in the Find Target Position sub-algorithm below. This yields the geodetic latitude, longitude, and height of the ground point.

For celestial acquisitions, convert the ECI LOS to Right Ascension (RA) and declination () angles:



where the ECI los vector is [ x y z ]T.

*Correct Ground Position for Light Travel Time*

Since the light departing the ground point takes a finite time to arrive at the OLI sensor, there is a slight discrepancy in the corresponding time at the ground point and at the spacecraft. Since the LOS intersection logic assumed that these times were the same, a small correction can be made to correct for this light travel time delay.

Given the ECEF positions of the ground point and the spacecraft, compute the light travel time correction as follows:

Compute the distance from the ground point to the spacecraft:



where:

**X**s is the spacecraft ECEF position and

**X**g is the ground point ECEF position.

Compute the light travel time using the speed of light (from CPF):



Compute the Earth rotation during light travel:

θ = ltt \* e where e is the Earth angular velocity from the CPF.

Apply the light travel time Earth rotation:



where:

**X**g**'** is the corrected ECEF position

Xg is the uncorrected ECEF position

Convert the corrected ECEF position to geodetic latitude, longitude, and height.

Note that the light travel time correction for lunar observations due to the difference between the Earth-moon distance and the spacecraft-moon distance is neglected. This is justified by the fact that that the lunar angular rate is less than 3 microradians per second and the maximum LTT difference is about 25 milliseconds, making the magnitude of this effect less than 0.1 microradians.

*Convert Position to Output Space Line/Sample*

The angular geodetic (latitude/longitude) or celestial (RA/declination) coordinates must be converted to the corresponding output space line/sample coordinate to complete the input space to output space mapping.

For Earth-view acquisitions, this is accomplished as follows:

Calculate the map projection X/Y for the geodetic latitude and longitude.

Convert the map X/Y coordinate to the output line/sample location:

If the output map projection is of a path-oriented projection, then the X/Y coordinate is transformed to output space with a bilinear transformation.



where:

ai = polynomial coefficients that map X/Y to an output line location

bi = polynomial coefficients that map X/Y to an output sample location

X,Y = map projection coordinates

The polynomial transformation is set up to handle the rotation involved in rotating a “Map North” projection to Satellite of “Path” projection (i.e., one that has the output line coordinate system more closely aligned with the along flight path of the satellite).

If the output map projection is not path-oriented, but “North up,” the relationship between X/Y and output line/sample does not involve any rotation, and the following equation is used:



where:

upper left Y = the upper-left Y projection coordinate of the output image

upper left X = the upper-left X projection coordinate of the output image

pixel size Y = the output pixel size in Y coordinates

pixel size X = the output pixel size in X coordinates

Note that these line and sample pixel coordinates are (0,0) relative (i.e., the center of the upper-left pixel is at line,sample 0,0).

For lunar acquisitions, the right ascension and declination angles derived from the inertial LOS are offset from the nominal lunar inertial position to establish an output frame that "tracks" the apparent location of the moon. This is done as follows:

a) Compute the apparent ECI J2000 position of the moon.

1. Use the input JPL lunar ephemeris data in the NOVAS-C package to compute the ECITOD apparent location of the moon at the time corresponding to the current LOS (lxx\_moonpos). This apparent location is provided as an ECITOD vector (i.e., including both direction and distance).
2. Apply the nutation and precession corrections (see the Ancillary Data Preprocessing ADD (6.1.4) for additional information) to convert the ECITOD vector to ECI J2000.
3. Subtract the current spacecraft ECI J2000 position vector from the lunar ECI J2000 vector to compute the spacecraft-lunar vector.
4. Compute the apparent (parallax corrected) right ascension, declination, and spacecraft-lunar distance from the spacecraft-lunar vector (by invoking exx\_cart2sph.

b) Compute the differences between the LOS right ascension and declination and the apparent lunar right ascension and declination.

c) Normalize the nominal angular pixel size by the ratio of the current spacecraft-moon distance (computed above) and the nominal spacecraft-moon distance. The nominal distance is computed at the acquisition center time.

psizecurrent = psizenominal \* distancenominal / distancecurrent

d) Divide the angular distances computed in b) above by the normalized pixel size computed in c) above. This yields the moon-relative line/sample coordinate. This is the coordinate space in which lunar images are framed, so the offset between these coordinates and the lunar scene upper-left corner coordinates yields the output space line/sample for the current grid point.

For stellar acquisitions, the right ascension and declination angles derived from the inertial LOS are used directly. The offsets relative to the scene upper-left corner (in right ascension/declination space) are computed and divided by the angular pixel size to compute output space line/sample coordinates.

One additional note regarding the celestial acquisition scene framing is in order. Since right ascension, like longitude, increases eastward, and declination, like latitude, increases northward, and given that celestial images are looking "up" rather than "down," the right ascension-x, declination-y coordinate system is left-handed. This can lead to the moon being apparently inverted left-to-right in the output image. This is not important for the applications (e.g., band registration characterization) in which the lunar images are to be used. If "anatomically correct" lunar images are required, some changes to the framing logic may be necessary.

The line and sample location calculated is stored in the grid structure. This line/sample location is then the output location for the corresponding input line/sample and the current elevation (current grid line/sample input locations).

**b) Calculate Jitter Sensitivity Coefficients** The forward model is invoked multiple times at each grid intersection to compute the effect that small attitude perturbations about each spacecraft axis have on the input space to output space line/sample mapping. This is done at each grid point as follows:

Save the current grid point input line/sample as in\_line/in\_samp and the current grid point output line/sample as line0/samp0.

For each spacecraft axis (roll-pitch-yaw) :

1. Perturb the attitude about the selected axis by 1 microradian.
2. Use the forward model to compute the output line/sample corresponding to the current input line/sample using the perturbed attitude, and store the result in line[0]/samp[0].
3. Perturb the input line number by 1 line (delta\_line = 1) and recompute the corresponding output line/sample, storing the result in line[1]/samp[1].
4. Restore the input line number to in\_line and perturb the input sample number by 1 sample (delta\_samp = 1), and recompute the corresponding output line/sample, storing the result in line[2]/samp[2].
5. Calculate the output space to input space line/sample sensitivities as follows:
   1. delta\_oline\_per\_iline = (line[1]–line[0]) / delta\_line
   2. delta\_oline\_per\_isamp = (line[2]–line[0]) / delta\_samp
   3. delta\_osamp\_per\_iline = (samp[1]–samp[0]) / delta\_line
   4. delta\_osamp\_per\_isamp = (samp[2]–samp[0]) / delta\_samp
6. Invert the resulting 2-by-2 sensitivity matrix to find the input line/samp per output line/samp sensitivities:
   1. determinant = delta\_oline\_per\_iline \* delta\_osamp\_per\_isamp – delta\_oline\_per\_isamp \* delta\_osamp\_per\_iline
   2. delta\_iline\_per\_oline = delta\_osamp\_per\_isamp / determinant
   3. delta\_iline\_per\_osamp = -delta\_oline\_per\_isamp / determinant
   4. delta\_isamp\_per\_oline = -delta\_osamp\_per\_iline / determinant
   5. delta\_isamp\_per\_osamp = delta\_oline\_per\_iline / determinant
7. Apply the input line/samp per output line/samp sensitivities to the output line/samp offset due to the attitude perturbation, to find the equivalent input space offset :
   1. d\_iline = delta\_iline\_per\_oline \* (line[0] – line0) + delta\_iline\_per\_osamp \* (samp[0] – samp0)
   2. d\_isamp = delta\_isamp\_per\_oline \* (line[0] – line0) + delta\_isamp\_per\_osamp \* (samp[0] – samp0)
8. Divide by the attitude perturbation to compute the input line/sample to attitude jitter sensitivities for this axis at this grid point:
   1. line\_sens[axis] = -d\_iline / perturbation
   2. samp\_sens[axis] = -d\_isamp / perturbation

Where :

line\_sens[] is the array of roll-pitch-yaw line sensitivities for the grid.

samp\_sens[] is the array of roll-pitch-yaw sample sensitivities for the grid.

perturbation is the 1 microradian attitude perturbation introduced in step 1.

Note that the sign of the sensitivities is inverted in this calculation. This is done because the sensitivities will be used to compute the equivalent input space corrections needed to compensate for an attitude disturbance. Therefore, since d\_iline is the input space line offset that is equivalent to one microradian of jitter for the current axis, an offset of –d\_iline will compensate for this jitter.

A 2-by-3 array containing the line and sample sensitivity coefficients for the roll, pitch, and yaw axes is stored for each grid point.

**c) Calculate Map Coefficients** Bilinear mapping coefficients for each grid cell are calculated for mapping from the input location to output location (forward mapping) and for mapping from the output location to input location (inverse mapping). A separate mapping function is used for lines and samples. This equates to four mapping functions. A set of four mapping functions is calculated for each grid cell, for each SCA, for every band, and for every elevation plane that is stored in the grid.

The following methodology is used for calculating one set of four bilinear mapping equations:

A 9x4 matrix is used to fit nine points within a grid cell. The matrix equation takes the form of:



In this equation, matrix A is 9x4, vector b is 9x1, and the coefficient matrix is 4x1. The coefficient matrix, [*coeff*], can be solved to obtain the mapping coefficients as follows:



In solving for an equation to map an input line and sample location to an output sample location, belonging to one grid cell, the matrices can be defined as follows:

An,0 = 1 where n=0,8

A0,1 = upper-left input sample location for the current grid cell

A1,1 = upper-right input sample location for the current grid cell

A2,1 = lower-left input sample location for the current grid cell

A3,1 = lower-right input sample location for the current grid cell

A4,1 = (A0,1+A1,1+A2,1+A3,1)/4

A5,1 = (A0,1+A1,1)/2

A6,1 = (A1,1+A3,1)/2

A7,1 = (A2,1+A3,1)/2

A8,1 = (A2,1+A0,1)/2

A0,2 = upper-left input line location for the current grid cell

A1,2 = upper-right input line location for the current grid cell

A2,2 = lower-left input line location for the current grid cell

A3,2 = lower-right input line location for the current grid cell

A4,2 = (A0,2+A1,2+A2,2+A3,2)/4

A5,2 = (A0,2+A1,2)/2

A6,2 = (A1,2+A3,2)/2

A7,2 = (A2,2+A3,2)/2

A8,2 = (A2,2+A0,2)/2

An,3 = An,1\*An,2 where n=0…8

b0 = upper-left output sample location for the current grid cell

b1 = upper-right output sample location for the current grid cell

b2 = lower-left output sample location for the current grid cell

b3 = lower-right output sample location for the current grid cell

b4 = (b0+b1+b2+b3)/4

b5 = (b0+b1)/2

b6 = (b1+b3)/2

b7 = (b2+b3)/2

b8 = (b2+b0)/2

The line and sample locations listed above are defined at the grid cell corners coordinates. The points interpolated in between the grid cell line segments provide stability for what could be, most notably, a mapping that involves a 45o rotation, an ill-defined solution if only four points were used in the calculation. The set of coefficients define a bilinear mapping equation of the form:

sampleo = coeff0 + coeff1 \* samplei + coeff2 \* linei + coeff3 \* samplei \* linei

where:

sampleo = output sample location

samplei = input sample location

linei = input line location

The forward mapping equations, mapping input line and sample locations to output line locations, can be solved by swapping output line locations for output sample locations in the matrix [b]. The reverse mapping equations, mapping output locations to input line and sample, can be found by using output line and sample locations in the [A] matrix and the corresponding input sample and then line locations in the [b] matrix.

**c).1. Calculate Forward Mappings**

Using the Calculate Map Coefficients algorithm described above, generate the mapping polynomial coefficients needed to convert from a line/sample in input space (satellite) to one in output space (projection). Coefficients for every cell in the grid are generated.

**c).2. Calculate Inverse Mappings**

Using the Calculate Map Coefficients algorithm described above, generate the mapping polynomial coefficients needed to convert from a line/sample in output space (projection) to one in input space (satellite). Coefficients for every cell in the grid are generated.

**Stage 5 - Finalize the Grid**

The final stage of grid processing generates the global (rough) mapping coefficients, used to initially identify the appropriate grid cell, and computes the parallax sensitivity coefficients, used to correct for even/odd detector offset effects, for each grid cell.

**a) Calculate Rough Mapping Coefficients**

Calculate the rough mapping coefficients for the grid. The rough polynomial is a set of polynomials used to map output line and sample locations to input line and sample locations. The rough polynomial is generated using a large number of points distributed over the entire scene, and by calculating a polynomial equation that maps an output location to an input location. The rough polynomial is only meant to get a “close” approximation to the input line and sample location for a corresponding output line and sample location. Once this approximation is made, the value can be refined to get a more accurate solution. A rough mapping polynomial is found for every SCA, for every band, and for every elevation plane that is stored in the grid file.

Bilinear mapping was found to be sufficient for the rough mapping. Therefore, the mapping function looks like the ones used for each individual grid cell. However, the setup of the matrices to solve for the mapping coefficients is different:



Where the matrix [A] is defined by the output line and sample locations, matrix [b] is defined by either the input lines or input samples, and N is equal to the total number of points stored in the grid for one elevation plane, of one band, for a single SCA. Therefore, the rough polynomial is found by using all the point locations stored in the grid for a given band and elevation plane for a single SCA. There is one mapping for output line and sample location to input sample location, and one mapping for output line and sample location to input line location.

**Grid Cell Polynomial**

Calculate a "rough" mapping of output to input lines/samples. These coefficients are used as a first-order approximation to an inverse line-of-sight model. This polynomial is used to initially locate the grid cell to be used in the resampling process, providing a starting point for the more accurate inverse model based on individual grid cell parameters.

**b) Calculate Detector Offsets** Computes the detector offset values and stores linear mapping coefficients associated with detector offsets in the grid structure. Using the zero elevation plane, for each band and each SCA, loop on the input lines and samples, calculating the odd/even detector offsets. The detector offsets are set up to account for the geometric differences between the odd/even, secondary, and tertiary detectors and the “nominal” set of detectors. (See the LOS Model Creation ADD (6.2.1)). These differences are considered to be consistent between actual and nominal detectors when they occur under the same acquisition conditions, i.e., they are slowly varying. These actual to nominal detector differences are due to the imperfect trade-off between space (detector offset) and time (detector delay) that is made when we temporally shift (through the use of Level 1R image fill) the even/odd and deselected detectors to compensate for their spatial offsets on the focal plane. The degree to which this time/space trade is imperfect varies with height, and so the corrections derived here and stored in the grid structure are functions of detector offset and height.

The subpixel detector-specific offsets are stored in the CPF. These "exact" detector-specific offsets are accounted for in the resampling process. Note that the potential for deselected detectors has made it necessary to also store per-detector full-pixel offsets in the CPF (and LOS model). As a result, this detector offset sensitivity logic has been changed to compute the offset sensitivity per pixel of detector offset rather than a fixed value derived from the static even/odd detector offset. The Image Resampling ADD discusses the routine ols2ils listed below, used for mapping an output line and sample to an input line and sample using the geometric grid.

Loop on the number of bands stored in the grid

Loop on the number of SCAs stored in the grid

Loop on the lines and samples stored in the grid

Get the maximum detector offset value for this band from the CPF.

Calculate the output line/sample location for the current input line and sample and the zero elevation plane, calculated using the forward model (see below) with the detector location set to MAXIMUM. This detector type is the same as ACTUAL but uses the maximum detector offset rather than the detector-specific value.

Map the calculated output line/sample back to input space using the geometric grid and ols2ils.

Delta line/sample per pixel of offset are calculated as follows:

line0 = (nominal line - mapped line) / max offset

sample0 = (nominal sample - mapped sample) / max offset

where:

nominal line = current grid cell line location

mapped line = input line location from ols2ils mapped "maximum" output line

nominal sample = current grid cell sample location

mapped sample = input sample location from ols2ils mapped “maximum” output sample

max offset = detector offset used in the MAXIMUM forward model calculations

These delta lines and samples represent the input space correction necessary to compensate for the difference between nominal and actual detectors per pixel of detector offset, for the zero elevation plane.

Repeat these calculations for the maximum elevation plane to compute lineH and sampleH, where H is the elevation corresponding to the maximum z-plane.

Compute the line and sample even/odd offset sensitivity coefficients:

c0 = line0

c1 = (lineH - line0) / H

d0 = sample0

d1 = (sampleH - sample0) / H

Note that c0 and d0 are in units of pixels per pixel and c1 and d1 are in units of pixels per meter per pixel.

These ci and di coefficients are stored in the projection grid to be used during the resampling process.

**Output Line/Sample to Input Line/Sample**

Map output space line/sample locations back into its corresponding input space line/sample locations. This is done using the "rough" polynomial from the grid to determine an initial guess at an input space line and sample. From this initial guess, a grid cell row and column is calculated and the inverse coefficients for that cell are retrieved from the grid. These coefficients are used to determine an exact input space line and sample (in extended space).

**Find Grid Cell**

This utility function finds the correct cell that contains the output line/sample. It finds the correct grid cell containing the output pixel by first determining the set of grid cells to be checked. It then calls a routine to perform a "point in polygon" test on each of these grid cells to determine if the pixel does indeed fall within that grid cell.

**Forward Model**

Having described the grid generation procedure, we now turn to the forward model, referred to extensively above, in more detail.

For a given line, sample and band, propagate the forward model to determine a latitude and longitude for the specified point. This involves finding the time of the observation, constructing the instrument line-of-sight, calculating the spacecraft attitude and ephemeris for the observation time, and intersecting the projected line-of-sight with the Earth’s surface. The entire forward model procedure, referred to as LOS projection, is described step-by-step below.

**a) Project LOS**

Find the position where the line-of-sight vector intersects the Earth's surface. It invokes the following sub-algorithms:

**a).1. Find Time** Find the time into the scene, given the line, sample, and band. The input sample number is 0-relative and relative to the SCA. The accounting for the odd/even, secondary, and tertiary detector offsets is based on the value of the dettype variable, which may be NOMINAL, ACTUAL, MAXIMUM or EXACT. Note that the EXACT selection is treated the same as ACTUAL. This is because fractional-pixel detector offsets can occur, but the compensating time shifts implemented by inserting fill pixels can only be introduced in whole-line increments. Therefore, the subpixel difference between the ACTUAL and EXACT detector types affects only the LOS angle, not the time. The MAXIMUM detector type represents a theoretical offset that is used to calculate the odd/even offset, or parallax, coefficients within the grid. This maximum is stored as #define in the prototype code, called MAX\_DET\_DELAY.

Due to the staggered odd/even and multiple pixel select detectors, a nominal and an actual time can be found in a scene. 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 as follows:

if detector type is set to MAXIMUM

detector\_shift\_x = maximum\_detector\_shift

l0r\_fill\_pixels = round(detector\_shift\_x) + nominal\_fill

else

detector\_shift\_x = shift stored in the geometric model

l0r\_fill\_pixels = Fill from L0Rp (also stored in the geometric model)

time\_index = MS\_line - l0r\_fill\_pixels

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) \* MS\_sample\_time

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

where:

* MS\_line is the zero-referenced multispectral line number (N).
* 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. This table is stored in the LOS model.
* 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.
* detector\_shift\_x (unless type is MAXIMUM) is the amount of even/odd detector offset for the current 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. This detector shift is stored within the geometric model.
* MS\_settle\_time is a small sample and hold time delay constant.
* nominal\_fill is the nominal fill associated with current band and SCA.
* maximum\_detector\_shift is the theoretical offset used in calculating the geometric effects associated with the odd/even offset of the detectors.

The MS\_settle\_time correction is expected to be a small (tens of microseconds) constant offset that should be captured in the CPF. The detector\_shift\_x offset parameter from the LOS model detector delay table is rounded to include the effects of even/odd detector stagger and detector deselect, but not the detector-specific subpixel offsets.

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 are computed as follows:

if detector type is set to MAXIMUM

detector\_shift\_x = maximum\_detector\_shift

l0r\_fill\_pixels = round(detector\_shift\_x) + nominal\_fill

else

detector\_shift\_x = shift stored in geometric model

l0r\_fill\_pixels = Fill from L0Rp (also stored in geometric model)

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

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)\*Pan\_sample\_time

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

where:

* pan\_line is the zero-referenced panchromatic line number (2N or 2N+1).
* 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. These values are stored in the LOS model. 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.
* detector\_shift\_x (unless type is MAXIMUM) is the amount of even/odd detector offset for the current 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. This detector shift is stored within the geometric model.
* Pan\_settle\_time is a small sample and hold time delay constant.
* nominal\_fill is the nominal fill associated with the current band and SCA.
* maximum\_detector\_shift is the theoretical offset used in calculating the geometric effects associated with the odd/even offset of the detectors.

For the panchromatic band, the l0r\_fill\_pixels and detector\_shift\_x parameters are in units of panchromatic pixels.

**a).2. Find LOS** Find the line-of-sight vector in sensor coordinates, using the Legendre polynomial LOS model stored in the LOS model, as follows:

Find the normalized detector for the Legendre polynomial:



where:

current detector = sample location (in the range 0 to number of detectors-1)

number of detectors = number of detectors (samples) for current band and SCA (from LOS model)

Find across-track (y) and along-track (x) angles:



where:

coef\_x = Legendre coefficients for the along-track direction

coef\_y = Legendre coefficients for the across-track direction

(Note: coef\_x and coef\_y are read from the CPF and stored in the LOS model)

If LOS requested is ACTUAL, add the whole pixel detector shift (detector, band, and SCA dependent for OLI) from the LOS model. This detector shift is only in the along-track direction. Note that the LOS model contains the combined whole pixel and subpixel detector offset, so it must be rounded to the integer part for the ACTUAL detector type and left unrounded for the EXACT detector type.

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

If LOS requested is EXACT, then add individual detector offsets (detector number, band, and SCA dependent). This detector shift is in both the along- and across-track directions. These values are stored within the LOS model.

x = x + (detector\_shift\_x) \* IFOV

y = y + (detector\_shift\_y) \* IFOV

Note that the detector\_shift\_y parameter, from the LOS model detector delay table, is always subpixel. See the LOS Model Creation ADD (6.2.1) for a further explanation of NOMINAL/ACTUAL/EXACT line of sight.

If the LOS request in MAXIMUM, then add the maximum, or theoretical, detector offset.

x = x + (maximum\_detector\_shift\_x) \* IFOV

Calculate LOS vector.



Normalize LOS.



**a).3. Find Attitude**

Find the precise roll, pitch, and yaw at the specified time. This routine uses the "corrected" version of the attitude data stored in the OLI LOS model. This attitude data sequence includes the effects of ground control point precision correction (if any).

Find the current time relative to attitude data start time stored in the LOS model.

dtime = time + image epoch time – attitude epoch time

Note:

time = nominal time of input sample relative to the start of the image epoch time = image start time from LOS model, only need seconds of day field since all epochs are adjusted to the same day.

attitude epoch time = attitude data start time from LOS model, only need seconds of day field since all epochs are adjusted to the same day.

Find index into attitude data (stored in model) corresponding to dtime:



where:

attitude sampling rate = sample period from LOS model

This attitude index determination could also be implemented as a search through the attitude data time stamps, which are stored in the LOS model. The selected index would be the index of the last time that does not exceed dtime.

Attitude is found by linearly interpolating between the attitude values located at index and index+1, using the corrected attitude sequence from the LOS model:





**a).3.i. Find Jitter** Find the high-frequency roll, pitch, and yaw corrections at the specified input image line/sample coordinate. This routine uses the jitter table stored in the OLI LOS model. This table is time aligned with the OLI panchromatic band line sample times, so the jitter table look-up proceeds directly from the input line/sample coordinates:

Find the current detector number from the input sample location:

detector = round(sample)

Verify that the detector is in the valid range for this band (return error if not).

Look up the number of L0R fill pixels for this detector (from the fill table).

Calculate the jitter table index:

If (band = pan)

Index = round(line) – l0r\_fill\_pixels

Else

Index = 2\*(round(line) – l0r\_fill\_pixels)

Verify that jitter table index is within the valid range for the table (return zeros if not).

Extract the roll-pitch-yaw jitter values for the current index from the jitter table and return these values.

Note that the jitter values are a direct look-up without interpolation. This does not compromise accuracy because this function is only used for cases of EXACT detector projection (e.g., the OLI data simulator) for which the input line/sample coordinates are integers. The jitter values extracted by Find Jitter are added to the low-frequency roll-pitch-yaw values interpolated by Find Attitude by the calling procedure Get LOS when the EXACT option is in force.

**a).4. Move Satellite Sub-Algorithm** Compute 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 "corrected" version of 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.

Calculate the time of the current line/sample relative to the start time of the ephemeris start time.

reference time = time + image epoch time – ephemeris epoch time

where:

time = nominal time of input sample relative to the start of the imagery

image epoch time = image start time from LOS model, only need seconds of day since all epochs are on same day.

ephemeris epoch time = ephemeris start time from LOS model, only need seconds of day since all epochs are on same day.

Find index into ephemeris data stored in model.



where:

ephemeris time steps = time between ephemeris samples

number of Lagrange points = number of points to use in Lagrange interpolation

Use Lagrange interpolation to calculate satellite position and velocity in ECEF (or ECI, depending on which sequence is provided) coordinates at time of current line/sample.

X = Lagrange(model satellite ECEF/ECI x[index])

Y = Lagrange(model satellite ECEF/ECI y[index])

Z = Lagrange(model satellite ECEF/ECI z[index])

XV = Lagrange(model satellite ECEF/ECI vx[index])

YV = Lagrange(model satellite ECEF/ECI vx[index])

ZV = Lagrange(model satellite ECEF/ECI vx[index])

where:

X = satellite x coordinate

Y = satellite y coordinate

Z = satellite z coordinate

XV = satellite x velocity

YV = satellite y velocity

ZV = satellite z velocity

**a).5. Convert Sensor LOS to Geocentric**

Find the line-of-sight vector from the spacecraft to a point on the ground by transforming the line-of-sight vector in sensor coordinates to perturbed spacecraft coordinates.

Use the OLI alignment matrix in the LOS model to convert the LOS vector from sensor to body. Then, apply roll, pitch, and yaw to the LOS to convert body to orbital. Finally, use the ephemeris to construct the orbital to ECEF rotation matrix and use it to transform LOS to ECEF.

First, using the 3x3 ACS to instrument alignment transformation matrix stored in the LOS model, calculate the instrument to ACS transformation matrix.



Transform LOS from Instrument to ACS/body coordinates.



Calculate the attitude perturbation matrix, using interpolated attitude values. Note that these values include the effects of precision LOS correction (if any), as these will be built into the "corrected" attitude stream in the LOS model. For the Earth-view acquisitions, the roll-pitch-yaw values will be with respect to the orbital coordinate system, but for celestial acquisitions, they will be with respect to ECI.

Calculate the perturbation matrix, [perturbation], due to roll, pitch, and yaw:



Calculate the new LOS in orbital coordinates (Earth-view) or ECI (celestial) due to attitude perturbation:



For Earth-view acquisitions, calculate the transformation from Orbital Coordinates to ECEF. The position and velocity vectors used in calculating the transformation are those calculated above. These vectors are in ECEF, allowing the LOS to be transformed from the instrument coordinate system to the ECEF coordinate system.

Transform perturbed LOS from Orbital to ECEF.



For celestial acquisitions, the ECI los ([perturbation los]) is returned.

**a).6. Find Target Position**

Finds the position where the line-of-sight vector intersects the Earth's surface.

Intersect the LOS in ECEF with the Earth model calculating the target ECEF vector. The ECEF vector is then used to compute the geodetic latitude and the longitude of the intersection point.

CPF

Projection

Grid

Grid Input Space

Figure 6‑28. Intersecting LOS with Earth model

Where:

rs = satellite position vector

re = geocentric Earth vector

los = line-of-sight vector

Intersect LOS with ellipsoid

1. Rescale vectors with ellipsoid parameters.







where:

*a* = semi-major axis of Earth ellipsoid

*b* = semi-minor axis of Earth ellipsoid

rs' = rescaled satellite position vector

re' = rescaled geocentric Earth vector

los' = rescaled LOS vector

1. From the Law of Cosines



where:

d = los’ vector length

δ = angle between rs’ and los’



By definition | re’ | = 1

Rearranging the equation determined from the Law of Cosines in terms of the constant d.



Solving for d using the quadratic equation.



1. Compute the new target vector.



1. Rescale the target vector.



1. Compute the Geodetic coordinates (see Geocentric to Geodetic below).



If target height (H), or elevation corresponding to current z plane, is not zero:

Initialize:

Target vector: rt=re

Target height: h0=0

Iterate until Δh =(hi-H) is less than TOL

1. Calculate the delta height.

Δh=hi-H

1. Compute the length of LOS.



where:

d = length of the LOS vector

rt = target vector

rs = spacecraft position vector

1. Compute the LOS /height sensitivity.



Where **n** is a vector normal to the ellipsoid surface.



and:

*q* = LOS height sensitivity coefficient

*los* = LOS unit vector

*i* = current estimate of ground point latitude

*i* = current estimate of ground point longitude

1. Adjust LOS.



1. Re-compute the target vector.



1. Calculate the new geodetic coordinates and corresponding height above ellipsoid.



Calculate the geodetic latitude and longitude from the final ECEF vector.

**a).7. Geocentric to Geodetic** The relationship between ECEF and geodetic coordinates can be expressed simply in its direct form:

*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

*e*2 - Ellipsoid eccentricity squared

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

The closed-form solution for the general inverse problem (which is the problem here) involves the solution of a quadratic equation, and is not typically used in practice. Instead, an iterative solution is used for latitude and height for points that do not lie on the ellipsoid surface, i.e., for h ≠ 0.

To convert ECEF Cartesian coordinates to spherical coordinates:

Define:



Initialize:



Iterate until abs(hi-hi+1) < TOL



**Projection Transformation**

Convert coordinates from one map projection to another. The transformation from geodetic coordinates to the output map projection depends on the type of projection selected. The mathematics for the forward and inverse transformations for the UTM, Lambert Conformal Conic, Transverse Mercator, Oblique Mercator, Polyconic, and the Space Oblique Mercator (SOM) map projections are handled by USGS’s General Cartographic Transformation Package (GCTP), at <http://edcftp.cr.usgs.gov/pub/software/gctpc/>.

**Grid Structure Summary**

Table 6‑4 and Table 6‑5 below show the detailed contents of the geometric grid structure.

|  |
| --- |
| **Geometric Grid Structure Contents** |
| Satellite Number (8) |
| WRS Path |
| WRS Row (may be fractional) |
| Acquisition Type (Earth, Lunar, Stellar) |
| Scene Framing Information: |
| Frame Type: PROJBOX, MAXBOX, PATH\_MAXBOX, LUNAR, or STELLAR |
| Projection Units (text): METERS, RADIANS, ARCSECONDS |
| Projection Code: GCTP integer code for UTM, SOM, etc... |
| Datum: WGS84 |
| Spheroid: GCTP integer code = 12 (WGS84/GRS80) |
| UTM Zone: UTM zone number (or 0 if not UTM) |
| Map Projection Parameters: 15-element double array containing parameters |
| Corners: 4 by 2 array of projection coordinates for UL, LL, UR, and LR corners |
| Path-oriented Framing Information: |
| Center Point: latitude and longitude of WRS scene center |
| Projection Center: Map x/y of WRS scene center |
| Rotation Angle: Rotation (from true north) of the path frame (degrees) |
| Orientation Angle: Rotation (from grid north) of the path frame (degrees) |
| Active Image Areas: latitude and longitude (in degrees) of the four corners of the active image area (excluding leading and trailing SCA imagery) for each band |
| Grid Structure Information: |
| Number of SCAs |
| Number of Bands |
| Band List: array of band IDs included in grid |
| Array of band grid structures, one for each SCA in each band (see Table 6‑5) |

Table 6‑4. Geometric Grid Structure Contents

|  |
| --- |
| **Grid Structure Contents for Each SCA in Each Band** |
| Band number |
| Grid cell size: number of image lines and samples in each grid cell |
| Grid cell scale: 1/lines per cell and 1/samples per cell |
| Pixel size: in projection units (usually meters) |
| Number of lines in the output image |
| Number of samples in the output image |
| Number of lines in the grid (NL) |
| Number of samples in the grid (NS) |
| Number of z-planes (NZ) |
| Index of zero-elevation z-plane |
| Z-plane spacing: elevation increment between z-planes |
| 1D array of input line numbers corresponding to each grid row |
| 1D array of input sample numbers corresponding to each grid column |
| 3D array of output lines for each grid point (row-major order) (NS\*NL\*NZ) |
| 3D array of output samples for each grid point (row-major order) (NS\*NL\*NZ) |
| Array of line c0, c1 even/odd offset coefficients (row-major order) (2\*NS\*NL) |
| Array of sample d0, d1 even/odd offset coefficients (row-major order) (2\* NS\*NL) |
| 3D array of forward mapping (ils2ols) coefficient sets (NS\*NL\*NZ) |
| 3D array of inverse mapping (ols2ils) coefficient sets (NS\*NL\*NZ) |
| 3D array of line jitter sensitivity coefficient vectors (note 2) (3\*NS\*NL\*NZ) |
| 3D array of sample jitter sensitivity coefficient vectors (note 2) (3\*NS\*NL\*NZ) |
| Degree of rough polynomial |
| Array of rough line polynomial coefficients ((degree+1)2 \* NZ values) |
| Array of rough sample polynomial coefficients ((degree+1)2 \* NZ values) |

Table 6‑5. Per Band Geometric Grid Structure Contents

**Geometric Grid Size**

Fully capturing the potential variability of the 50 Hz attitude data that will be available within the L8/9 ancillary data stream would require a grid spacing of 5 lines. This may be impractical. Fortunately, the OLI error budgets assumed that attitude variations at frequencies up to only 10 Hz would be corrected in the LOS model. Such variations can be captured by sampling at 20 Hz or higher. This corresponds to a grid spacing of 11-12 lines. The grid has been successfully tested down to a line sampling of 10, but this does make for a large grid structure. The inclusion of a high-frequency jitter table in the OLI model and jitter sensitivity coefficients in the grid structure allow the grid to be less dense in the time (line) dimension. The baseline assumption is that attitude frequencies above 3 Hz will be relegated to the jitter table, allowing the grid density to be reduced to 30 lines, thus saving grid space even with the addition of the new jitter sensitivity fields.

#### Notes

The following are additional background assumptions and notes:

1. The NOVAS planetary ephemeris file provides the lunar ephemerides used to define the reference output space for lunar image processing. This file is in the original JPL format and is provided to the NOVAS routines as an input.
2. The TIRS implementation of the LOS projection grid will include another new feature that has also been applied to OLI as of version 3.4 of this algorithm – a set of sensitivity coefficients that maps roll-pitch-yaw deviations to the corresponding line and sample differences, for each grid cell. The resampler will use these sensitivity coefficients to convert high-frequency (per image line) attitude variations to line and sample adjustments. Modeling the high-frequency deviations separately and correcting them in the resampler allows for a sparser and more manageable-sized grid.