1 Introduction

The IIR Level 1 (L1) process converts IIR Level 0 (L0) packet data received from the Universal Space Network (USN) into IIR Level 1B products. The requirements of this process are based on three documents:

1. IIR/CALIPSO Level 1 Processing Requirements[1*] (DOC1)
2. CALIPSO Payload Data Measurements & Analysis Document[2] (DOC2)
3. CALIPSO Data Products Catalog[3] (DOC3)

The first document specifies the requirements of the process, the second document specifies the format of the L0 input data packet, and the third document specifies the parameters written to the output products.

This document describes the IIR Level 1 Process (IIRL1P) as implemented in the CALIPSO IIR Level 1 production code. Section Two describes the Overall Design, Section Three describes the Input Data, Section Four describes the IIRL1P, Section Five describes the Output Products, and Section Six provides the Structure Chart.

The IIR acquisition cycle is made of 30 images organized into 5 sequences of 6 images according to the **section 3.2 Sequences and cycles** from DOC1. The diagram is redisplayed in the **Figure 1**.

Cycle	40.92 s																													
Sequence	8.184 s						8.184 s						8.184 s						8.184 s						8.184 s					
View	Black Body			Earth			Deep Space			Earth			Deep Space			Earth			Deep Space			Earth			Deep Space			Earth		
Filter number	1	2	3	3	2	1	1	2	3	3	2	1	1	2	3	3	2	1	1	2	3	3	2	1	1	2	3	3	2	1
Image number + 1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

Figure 1 Image cycle description

A sequence is the period during which the IIR instrument looks at the Earth and at a calibration source (either Blackbody or Deep Space) for its 3 spectral bands: 1 = 8.65 μm, 2 = 12.05 μm, and 3 = 10.6 μm. The duration of a sequence is approximately 8.184 seconds.

The IIR L0 data are provided through data packets that are in Consultative Committee for Space Data Systems (CCSDS) format. Each packet is associated with one single IIR image. For more detail structure of the L0 data packet, see the **Appendix A, section 3.0** in DOC2.

There are three major processes when converting L0 data into L1B products. The three processes are check L0 data, calibration and count conversion, and registration. The requirements of these three processes are given in the DOC1 document.

The output products are written into Hierarchical Data Format (HDF) files. The output parameters and formats are described in the DOC3. The output products include a one-orbit calibration file and two half-orbit Level 1B data products, which contain geolocated, calibrated radiances.

* The number denotes the Reference Number.

The source codes of the IIR Level 1 process are written with the C++ computer language and interfaces with the common framework used by the other subsystems in CALIPSO.

2 Overall Design

The IIR Level 1 overall processing description is given in the **section 6 Processing description** from DOC1 and the flow chart diagram is redisplayed in **Figure 2**.

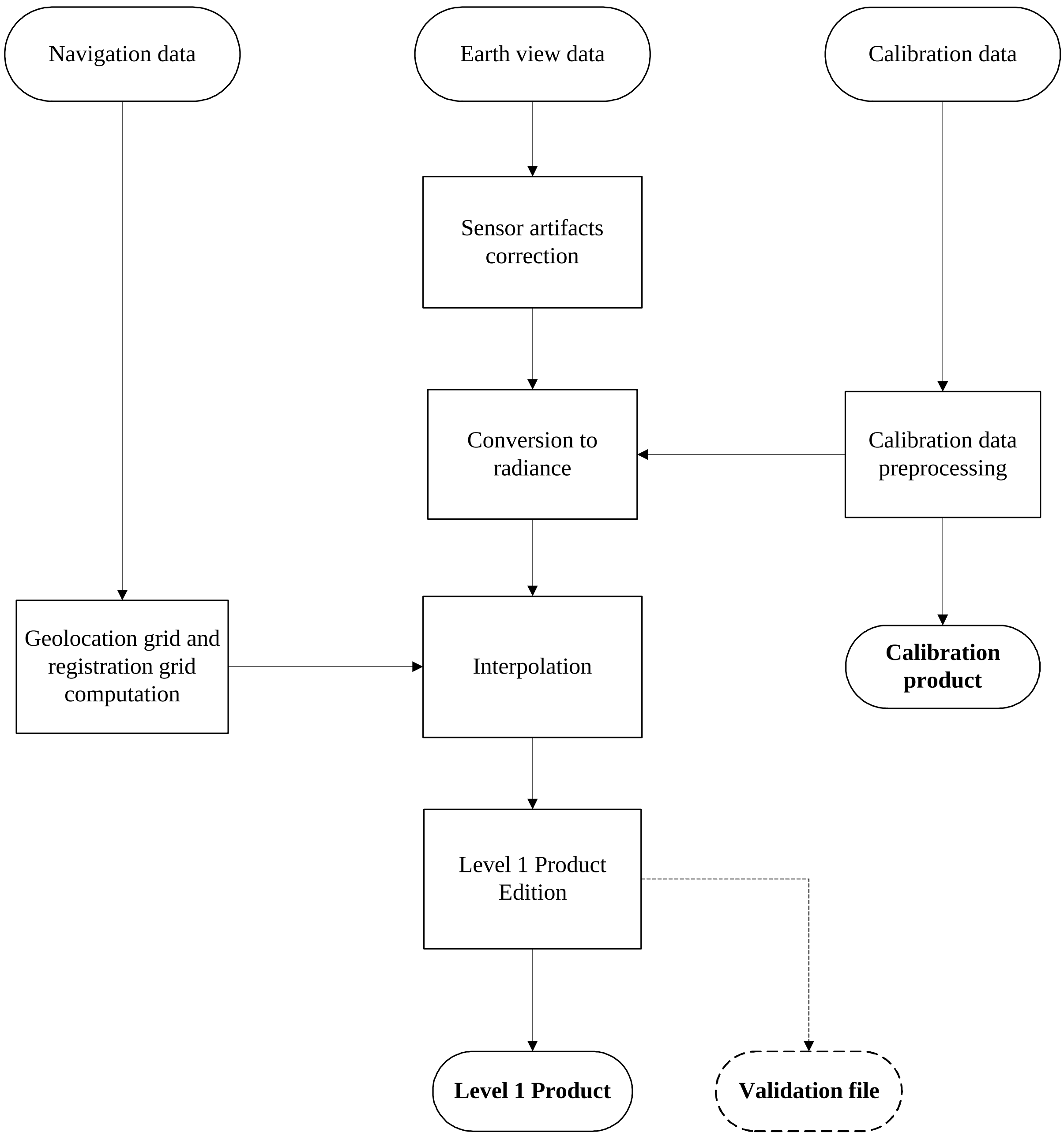

Figure 2 IIR Level 1 processing description

The input data include Navigation data (ephemeris and attitude data), Earth view data, and calibration data. The output products include Level 1B product, Calibration product, and an optional product, Validation file. A description of the IIR Level 1 processing requirements is given in DOC1. The document lists the required input data, processing, and output data products. The overall Design Diagram based on these requirements is given in **Figure 3**. The class diagram of ***IIRLevel1Process*** is given in **Figure 4**.

From **section 7 Pseudo-code** in DOC1, the entire data set is processed through a multiple pass procedure. This process requires a large allocation of computer resources. The design of IIRL1P follows a different approach. TheCheck L0 Data process (**section 7.2 Check of level 0 data**), processes the entire data file record by record without storing any of the input data. For the remaining processes, IIRL1P uses a moving windows method to process the entire data set. The number of sequences in the window depends upon the number of missing data, **CS_Interp_Threshold** parameter, and **BB_Interp_Threshold** parameter. The minimum and maximum sequences are 35 and 45, respectively, for the current settings (**CS_Interp_Threshold** = 7 and **BB_Interp_Threshold** = 3). All the sequences in the window send to the Calibration and Count Conversion process, but only the sequences 11 through 15 send to the Registration process. Once one window process is complete, the window moves five sequences down the data stream.

The Initialize process retrieves runtime parameters from the Process Control File (PCF) and input parameters from the input parameter file. IIRL1P creates two L1B output files; night portion and day portion. The time ranges of these two portions are obtained from the PCF. The settings of these time ranges are retrieved from the Payload Health and Status data (PHSD). The PHSD contains the day/night partition times of LIDAR data. Once runtime and input parameters are defined, the Initialize process allocates memory for all the variables, performs parameter initialization, creates all the objects, and opens all input and output files.

The Check L0 Data process (**section 7.2 Check of level 0 data**) reads the L0 data, checks the data consistency, sorts the 3 types of incoming products (Earth, Blackbody, and Deep Space), and updates the sequence number. It will perform the Blind Pixel Detection and/or Dead Pixel Detection if the control parameters are set to “On” value.

The **change_equalization** parameter is used to indicate the Equalization mode of operation that occurred (**change_equalization** = 1) in the data stream. As given in **section 7.2 Check of level 0 data**, if equalization number has changed (Equalization mode operation occurred), no processing is performed and the calibration image is added to the calibration file.

The **has_input** parameter indicates the input data conditions and is giving as follows:

- 0: No more input data.
- 1: Has input data without/with missing data that are less than need sequences.
- 2: Has input data with missing data that are more than need sequences but less than maximum sequences.
- 3: Has input data with missing data that are more than maximum sequences but less than gain interpolation threshold.

4: Has input data with missing data that are more than gain interpolation threshold with some sequences of data needed to calibrate.
5: Has input data with missing data that are more than gain interpolation threshold without any sequence of data needed to calibrate; no data is needed to process.

The IIRL1P will restart when the **has_input** parameter is equal to 4 or 5 because no interpolation could be performed to make a regular period between images.

If the IIRL1P detects missing any normal science packets, it will fill these missing packets with fill data to make the data stream continuous. The missing Earth images will not be converted into radiances so that no missing Earth radiances will be written to L1B output products.

The calibration images (Deep Space and Blackbody) are written to the calibration file (**section 6.5.2 Calibration product**), and the satellite information is written to the L1B output products (**section 6.5.1 Level 1 product**) for each sequence has been read.

If enough sequences of data have been read into the window, these data send to Calibration and Count Conversion process (**section 6.3 Radiometric calibration**) to calibrate the data and convert the Earth raw data into radiances. The mean and standard deviation of the calibration images (Deep Space and Blackbody) are written to the calibration file (**section 6.5.2 Calibration product**).

The calibrated radiances of the three channels send to Registration process (**section 6.4 Geometrical processing**) to register these three channels radiances to a reference grid that based on the track of LIDAR. The results of this process are written to the L1B output files (**section 6.5.1 Level 1 product**).

If the data stream is still within the processing time range, the window moves five sequences down the data stream and read the next sequences of data into the window to continue the process. Otherwise the process stops, releases all the memory that has been allocated, and closes all the input and output files.

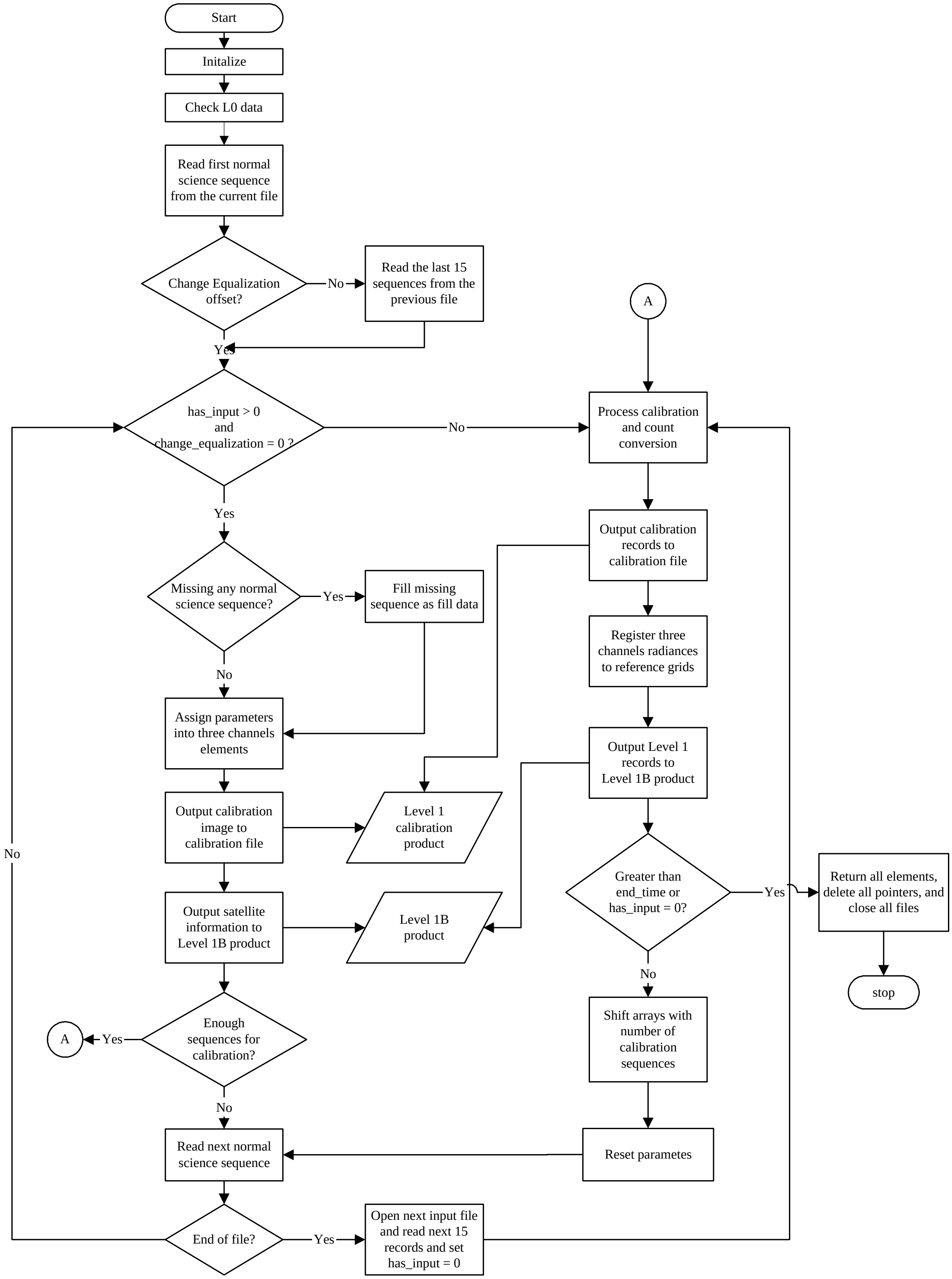

Figure 3 Design diagram of IIR Level 1 Processing

IIRLevel1Process
-*cs_output_manager : OutputManager = NULL -*reg_output_manager : OutputManager = NULL -*manager : L0InputManager = NULL -*reader : IIRreader = NULL -*checkl0data_ptr : IIRCheckL0Data = NULL -*calibration_ptr : Calibration = NULL -*registration_ptr : Registration = NULL -*satellite_ptr : Satellite = NULL -cal_channel_8_5 : vector<Element*> -cal_channel_11 : vector<Element*> -cal_channel_12 : vector<Element*> -swaths : vector<Element*> -*L0_data_ptr : Element = NULL -*dadta_ptr : Element = NULL -total_packet_count : int = 0 -test_mode_count : int = 0 -change_equalization_count : int = 0 -num_earth_image : int = 0 -num_bad_pixel : int = 0 -num_good_pixel : int = 0 -num_missing_pixel : int = 0 -num_bad_cs_image : int = 0 -num_bad_bb_image : int = 0
+process() : int -openOutputFile() : int -openSecondOutputFile() : int -openCurrentFile() : int -closeOutputFile() : void -doCalibration() : void -doRegistration() : void -*getDataPtr() : Element -getStartTime() : int -setMetadata() : void -setCalMetadata() : void -getCheckL0Result() : void -getSumEarthRecord() : void -setDQPFlag() : void -checkMissingCSImage() : void -checkMissingBBImage() : void

Figure 4 ***IIRLevel1Process*** **class diagram**

3 Input Data

The IIR L0 data packet is in CCSDS format. Each packet includes a Payload Data Header and Payload Source Data. The detail packet layout is defined in **Table A. 3-2 in Appendix A section 3.0** from the DOC2. The Payload Data Header follows the CCSDS primary header definition and occupies the first three words (each word is 16 bits). The next four words contain the secondary header and used to time tag the Payload Data. The time tag in the secondary header uses the CCSDS Day Segmented (CDS) Time Code. The time is from January 1, 1958.

The Payload Source Data has four components: Payload Source Packet ID, Unique Packet Sequence Identifier (UPSI), Payload Source Data, and Payload Packet Cyclic Redundancy Check (CRC32). For IIR L0 data packet, the Application ID (APID) is 13 and Packet ID is 1. Since the Packet ID for IIR L0 data is always equal to 1, the IIR mode of operation (normal, test, or equalization) is identified from the Science Ancillary Data (SAD) words.

Instead of using the Packet Sequence Count field in the primary header, the UPSI is used as sequential counter. This counter is used to detect lost packets within the Payload Data stream. The Payload Source Data are packed after the UPSI and follow with the CRC32. The CRC32 is used to valid each of the Payload Data packets on the ground.

In the IIRL1P codes, the ***IIRreader*** object performs the input of L0 data packets from IIR Level 0 data files. The ***addPacket*** function uses the Input System from the CALIPSO Generic Framework to read these L0 data packets. The ***L0InputManager*** reads one packet at a time based on the input configuration file specification. The input configuration file specifies the structure of one IIR packet. After one packet has been read, the ***L0InputManager*** separates the variables based on the description of the input configuration file and then these variables are parsed into ***Parameters*** that are attached to an ***Element***, **L0_data_ptr**, which represents an image. For more detail description of the Input System of the Generic Framework, one can look at the Data Management System Framework Architecture and Design User s Guide[4].

The ***addPacket*** function also checks the consistency of the L0 data packets (time, APID, Packet ID, and packet length), to sort the 3 types of incoming products (Earth, Deep Space, and Blackbody), to check the instrument modes (normal, test, and equalization), and to update the sequence numbers. It puts all six images with same sequence number into another ***Element***, **data_ptr**, and then passes this ***Element*** into ***process*** function to complete the input process. The ***process*** function extracts the data from **data_ptr**, sorts the data into three channels, and assigns them to three different ***Element***s (**cal_channel_8_5, cal_channel_11, and cal_channel_12**). The **Figure 5** shows the ***addPacket*** function flowchart of the ***IIRreader*** object and **Figure 6** shows the class diagram of ***IIRreader***.

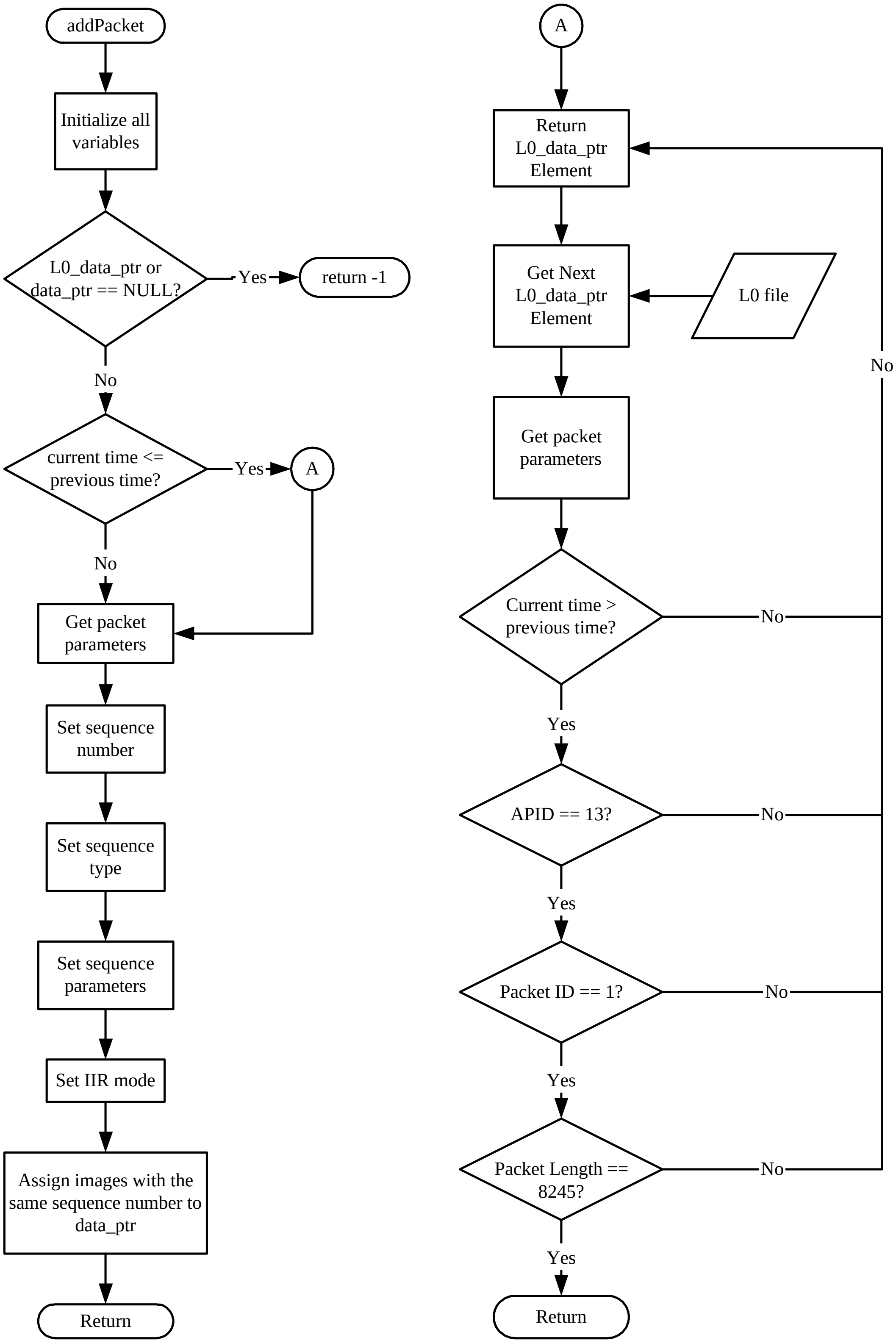

Figure 5 ***addPacket*** **function flow chart**

IIRreader
-total_packet_read : int = 0 -total_bad_time_packet : int = 0 -total_bad_apid_packet : int = 0 -total_bad_packetid_packet : int = 0 -total_bad_packet_length_packet : int = 0 -num_earth_image : int = 0 -num_blackbody_image : int = 0 -num_spacelook_image : int = 0 -*image8_5 : Element = NULL -*image11 : Element = NULL -*image12 : Element = NULL -cal_image8_5 : Element = NULL -cal_image11 : Element = NULL -cal_image12 : Element = NULL
+*addPacket () : Element +*readNextPacket() : Element +process() : void +getCoefficient() : int +getNumberEarthImage() : int +getNumberBlackbodyImage() : int +getNumberSpacelookImage() : int +getTotalBadTimePacket() : int +getTotalBadAPIDPacket() : int +getTotalBadPacketIDPacket() : int +getTotalBadPacketLengthPacket() : int -assignParameter() : int -getImageID() : unsigned short -getCycleNumber() : unsigned short -getEqualizationNumber() : unsigned int -checkPacket() : void

Figure 6 *IIRreader* class diagram

4 Processes

There are three major processes within IIRL1P and they are Check Level 0 Data process (**Section 7.2**), Calibration and Count Conversion process (**Section 6.3**), and Registration process (**Section 6.4**).

4.1 Check Level 0 Data Process

Input: IIR Level 0 data
Output: checked Level 0 data, Quality Control Product information

As given in the **section 7.2 Check of level 0 data** from DOC1, The **Principle** of Check Level 0 Data Process (***IIRCheckL0Data***) is to read the L0 data of the IIR, check the data consistency, sort the 3 types of incoming products (Earth, Blackbody, and Deep Space), and update the sequence number. It also includes two runtime parameter controlled processes, Blind Pixel Detection and Dead Pixel Detection. These two processes will be performed if the runtime parameters set to “ON” (**blind_pixel_detect** = 0 and **dead_pixel_detect** = 0).

The process reads record by record from the entire data set and performs the following:

1. check instrument mode
2. sort and count the 3 types of images
3. check missing packet, sequence, and cycle
4. perform Dead Pixel Detection
5. perform Blind Pixel Detection

The results of this check will be written into metadata of the output products. The **Figure 7** shows the flowchart of the ***IIRCheckL0Data*** object and **Figure 8** shows the class diagram.

4.1.1 Blind Pixel Detection

Input: Calibration data, Blackbody Temperature
Output: Noise Equivalent Temperature, Blind Pixel Grid

The blind pixels are characterized according to two criteria, which are given in the **section 7.4 Calibration data pre-processing** from DOC1:

1. A value of the signal of the pixel is much higher or lower than the average value of all the pixels when a blackbody is seen.
2. The noise equivalent temperature (i.e. the signal of the pixel divided by the gain when a blackbody is seen) is higher than a specified value.

If the pixel is detected as blind, the value of this pixel in the **blind_pixel_grid** is set to 1, otherwise is set to 0. If a blind pixel is detected, the parameter, **Blind_pixel_alert**, will be set to 1 to alert user. Also if the Blackbody temperature and/or Deep Space image are out of range, the corresponding parameters, **Alert_BB_temp** and **Alert_CS**, will be set to 1 to alert user.

4.1.2 Dead Pixel Detection

Input: Deep Space Calibration data
Output: Dead Pixel Grid

The dead pixels are characterized by Deep Space Calibration data and are equal to either 0 or 4095 as provided in the **section 7.4 Calibration data pre-processing** from DOC1. If the pixel is detected as dead, the value of this pixel in the **dead_pixel_grid** is set to 1, otherwise is set to 0. If a dead pixel is detected, the parameter, **Dead_pixel_alert**, will be set to 1 to alert user.

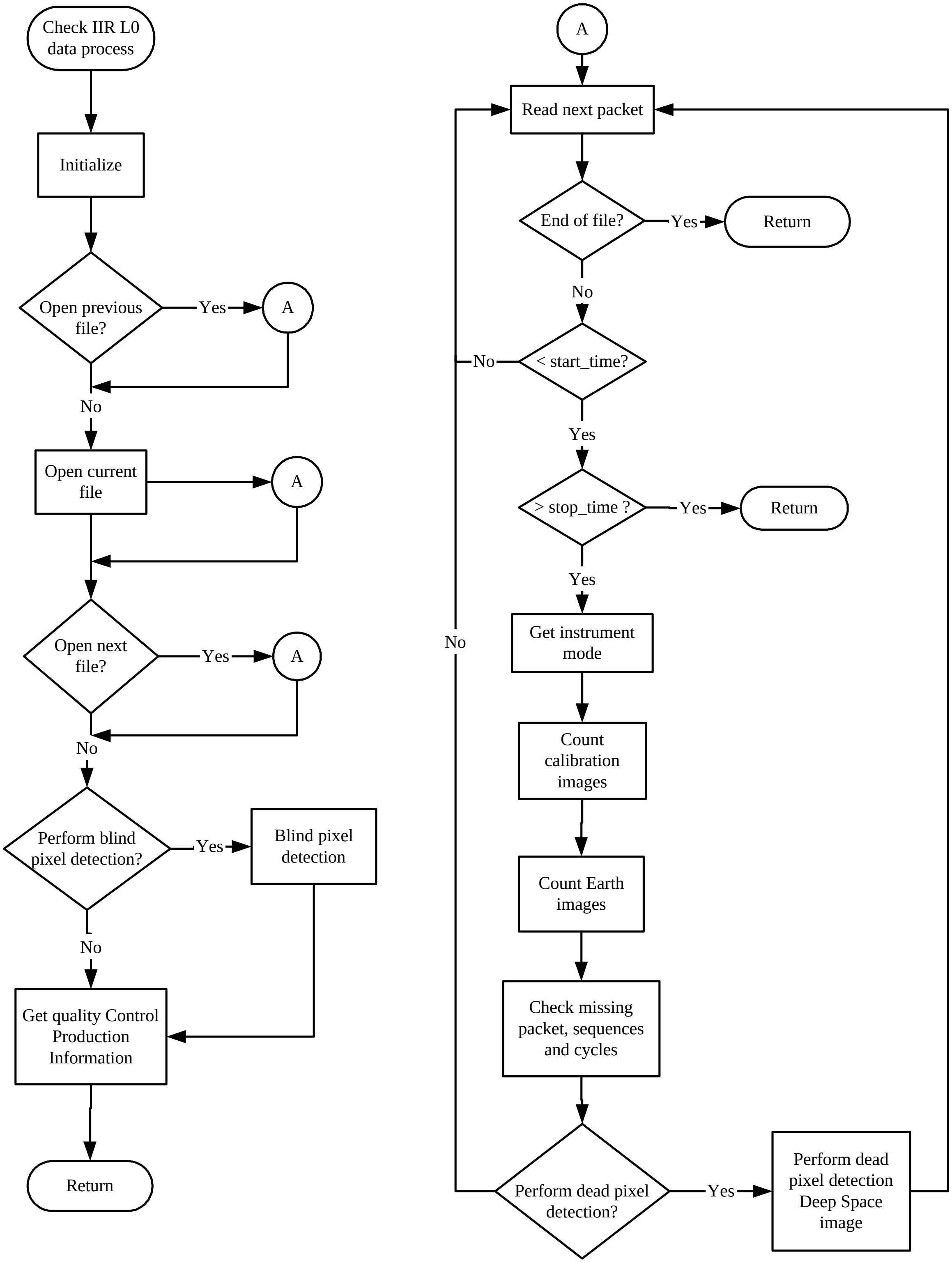

Figure 7 Check Level 0 Data Process flowchart

IIRCheckL0Data
-current_dead_pixel : unsigned char = 0 -current_blind_pixel : unsigned char = 0 -dead_pixel_alert : int = 0 -blind_pixel_alert : int = 0 -bb_temperature_alert : int = 0 -cc_alert : int = 0 -test_mode_count : int = 0 -change_equalization_count : int = 0 -num_black_image : int = 0 -num_space_image : int = 0 -num_earth_image : int = 0 -num_packet_read : int = 0 -num_sequence_count : int = 0 -num_cycle_count : int = 0 -num_bad_time : int = 0 -num_bad_apid : int = 0 -num_bad_packet_id : int = 0 -num_bad_packet_length : int = 0
+process() : int +getCoefficient() : int +*getCurrentDeadPixel() : unsigned char +*getCurrentBlindPixel() : unsigned char +getDeadPixelAlert() : int +getBlindPixelAlert() : int +getBBTemperatureAlert() : int +getCSAlert() : int +getNumEarthImage() : int +getNumBlackbodyImage() : int +getNumColdSpaceImage() : int +getNumPacketRead() : int +getNumCycleRead() : int +getNumSequenceRead() : int +getNumBadTimePacket() : int +getNumBadAPIDPacket() : int +getNumBadPacketIDPacket() : int +getNumBadPacketLengthPacket() : int +getNumTestMode() : int +getNumEqualizationMode() : int -*getDataPtr() : Element -checkDeadPixel() : void -checkBlindPixel() : int -CheckL0Data(): int

Figure 8 ***IIRCheckL0Data*** **class diagram**

4.2 Calibration and Count Conversion Process

Input: Calibration data, Earth Image Digital Count data
Output: Mean and standard deviation of Calibration data, Radiances

The **Basic Principle** of this process is given in **Section 6.3.1 Basic Principle** from DOC1 and is redisplayed as the following:

For each spectral band (index k = 1 for 8.65 µm, 2 for 12.05 µm and 3 for 10.6 µm)

$$R_E^k = \underbrace{(X_E^k - \hat{X}_{DSE}^k)}_{\text{offset}} . \frac{R_{BB}^k(T_{BB}^k)}{\underbrace{(X_{BB}^k - \hat{X}_{DSBB}^k)}_{\text{gain}}} . l^k [X_E^k - \hat{X}_{DSE}^k]$$

with

X_E^k	scene measurement
$\hat{X}_{DSE}^k$	average on N_{DSE}^k Deep Space measurements around X_E^k
$\hat{X}_{DSBB}^k$	average on N_{DSBB}^k Deep Space measurements around X_{BB}^k
X_{BB}^k	Blackbody measurement
$R_{BB}^k(T_{BB}^k)$	computed Blackbody radiance at measured T_{BB}^k
$l[X_E^k - \hat{X}_{DSE}^k]$	Non-linearity function depending on the input radiance (a table, LINEAR(k,s), in the processing, with $[X_E^k - \hat{X}_{DSE}^k]$ rounded to the closest value)

The Gain is averaged on N_{BB}^k Blackbody measurements around X_E^k

This process includes two processes, Calibration process and Count Conversion process. The Calibration process is to calibrate the calibration data (Deep Space and Blackbody) and the Count Conversion process is to convert the Earth Image Digital Count data into radiances. The **Figure 9** shows the flowchart of the ***Calibration*** object and **Figure 10** shows the class diagram.

The **num_mean** parameter indicates number of sequences of calibration data from the input window needed to calculate the mean and standard deviation. The **num_calibrate** parameter indicates number of sequences of calibration data from the input window need to be calibrated. The results are stored back into input ***Element***s (**cal_channel_8_5, cal_channel_11, and cal_channel_12**) and passed back to ***IIRLevel1Process*** to continue the processing and write the results to calibration output file.

4.2.1 Calibration

Input: Calibration data
Output: Averaged Deep Space calibrated data, Averaged Gain value

From **section 6.3.2 Pre-processing of calibration data**: As for every instrument data, the calibration data are not perfect. Using them directly would increase the noise on the Earth view images. The calibration data are smoothed in order to reduce the noise, without eliminating the low frequency variation of the calibration. The missing Deep Space images have to be interpolated in order to have a regular period between images.

The Gain value is computed for every Blackbody measurement. The Gain values will be averaged and then this averaged Gain is applied for all the Earth images of the cycle. In case of a missing Blackbody image, the corresponding Gain value for the cycle has to be interpolated using the other existing Gain values.

From **section 6.3.5 Statistics over calibration (DS or BB) images**: The mean and standard deviation of the Deep Space, Blackbody, and Gain images will be calculated and written to calibration output products. These results will compare with predefined values.

4.2.2 Count Conversion

Input: Earth Image Digital Count data, Averaged Deep Space calibrated data, Averaged Gain value
Output: Radiances

From **section 6.3.4 Conversion to radiance**: After the Offset and Gain computations along with the input instrument non-linearity function, the Earth Image Digital Count data are converted into radiances. Once radiances are computed, the dead and/or blind pixels of the Earth images are obtained by averaging the closest valid radiances according to the **section 7.7 Dead and Blind Pixels interpolation**.

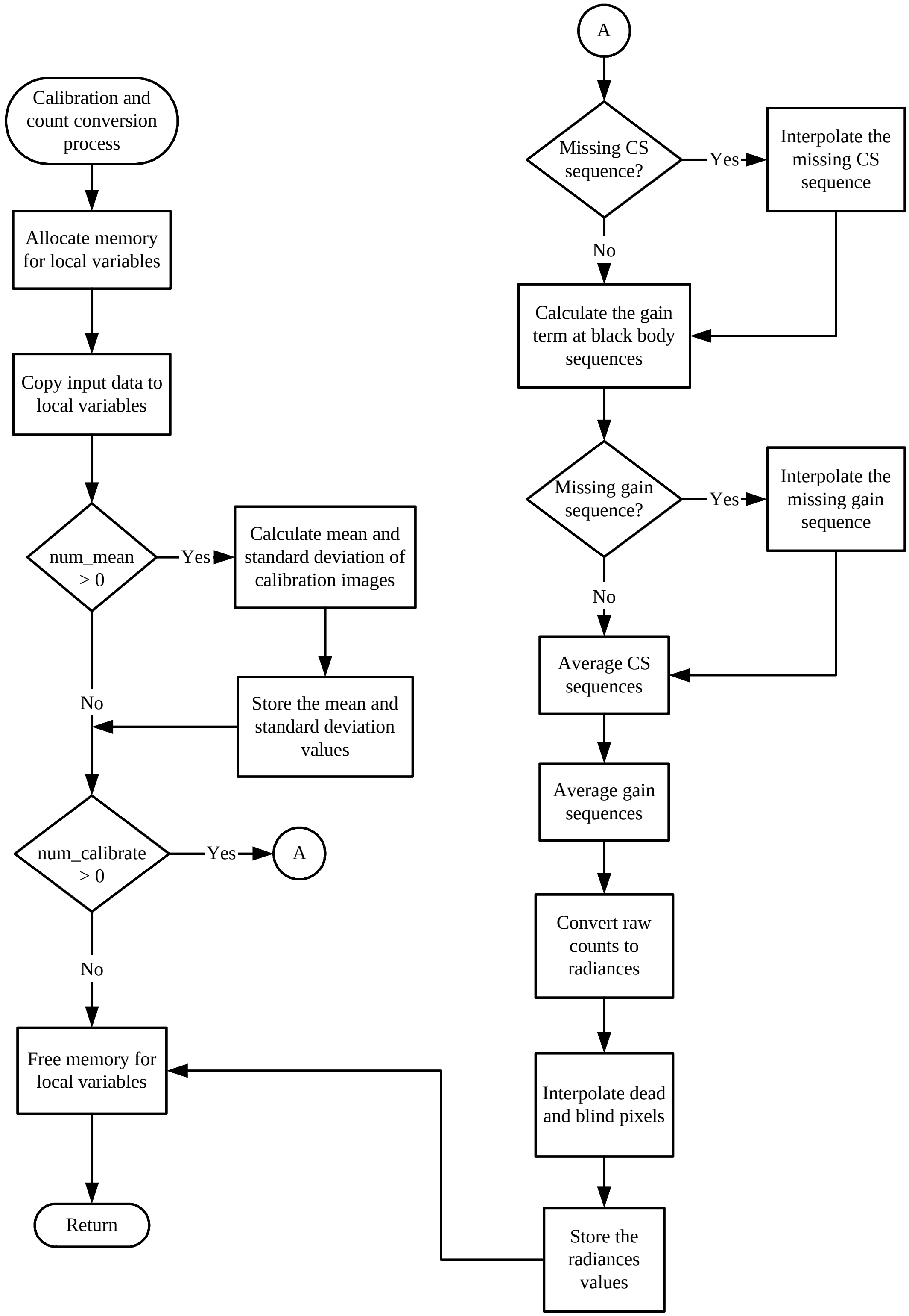

Figure 9 Calibration and Count Conversion Process flowchart

Calibration
-*L_raw_geo : float = NULL -*gain : double = NULL -*gain_status : short = NULL -nonlinearity_table : float -mean_CS : float = -9999 -mean_BB : float = -9999 -mean_gain : float = -9999 -stdv_CS : float = -9999 -stdv_BB : float = -9999 -stdv_gain : float = -9999 -*digital_radiance : float = NULL -*CS_interm : float = -9999 -*BB_interm : float = -9999 -*cal_quality_flag : unsigned int = NULL -*CS_interm_status : short = NULL -*BB_interm_status : short = NULL -dead_pixel_grid : unsigned char = 0 -blind_pixel_grid : unsigned char = 0
+process() : int +getCoefficient() : int -setArrayDimension() : int -calculateMeanStdv() : void -findCSSequence() : void -calculateGain() : void -doRadConversion() : void -doCSInterpolation() : void -doGainInterpolation() : void -getCSAverage() : int -getGainAverage() : int -freeArrayDimension() : void -setQualityFlag() : void

Figure 10 *Calibration* class diagram

4.3 Registration

Input: Calibrated Radiances for three bands
Output: Regis rated Radiances, Latitude, Longitude, Viewing Angles

From **section 6.4 Geometrical processing**: This process has two goals:

1. to register the data from the 3 spectral bands to a reference grid
2. to provide the geometrical coordinates (longitude, latitude) of each pixel and viewing angles

The raw data of the three spectral bands are miss-registered since they are not acquired simultaneously due to the filter wheel concept of the instrument. To produce accurate Level 2 data, it is necessary to register the three spectral bands to one common reference grid and this must be done in the Level 1 product. The IIR Mission Working Group has decided to define the reference grid as follows:

- The lines in the product are orthogonal to the LIDAR track
- The center column is the LIDAR track (not taking into account the fluctuations of this track related to the pointing errors of the satellite)
- For each line, the projection is defined so that the center pixel of the line is registered with a LIDAR shot (the time of acquisition of the LIDAR shot is used as a parameter of the projection definition)
- The lines are separated by about 1 km, but the sampling is exactly determined by the translation of the sub-satellite point during a time Δt equivalent to 3 LIDAR shots, i.e. about 148.8 ms.
- It is not necessary to resample the product with each LIDAR shot (about 300m), since the IIR geometrical resolution is about 1 km. Only one LIDAR shot out of three will be chosen. If the LIDAR shots are averaged by groups of three in the LIDAR processing; the idea is to use the center LIDAR shot as the parameter for the computation of the projection for a line of the IIR product.

The **Figure 11** shows the requirement of the registration process. The dash lines represent the reference grid lines. The **Figure 12** shows the flowchart of the ***Registration*** object and **Figure 13** shows the class diagram.

Two sequences of images are passed to the ***process*** function of the ***Registration*** object because there is an overlap region between two consecutive image sequences; see **Figure 11**. According to **section 8.6. choose_sequence**, the best sequence of these two sequences is chosen to perform the bicubic interpolation to calculate the radiances at the reference grid.

The **t_cur** parameter represents the current time to build the reference gridline. The **stop_time** parameter represents the stop time to build the reference gridline for the current sequences (55 km). The **numGrids** parameter represents number of grid points in each gridline. The **numGrids** parameter is set to 69 according to **Annex 3.5. Constants for geometry**.

If the results of a gridline are all bad data, the results are thrown away. Otherwise the results are written to a new ***Element***, **swath**, and passed back to ***IIRLevel1Process*** to write the results to L1B output file.

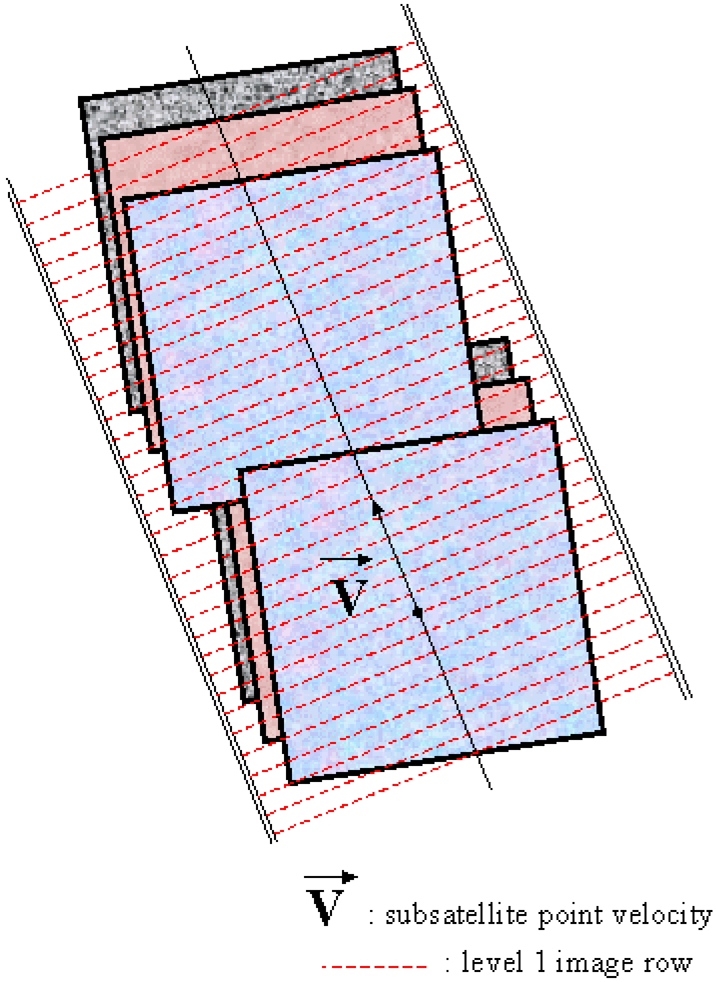

Figure 11 Registration process

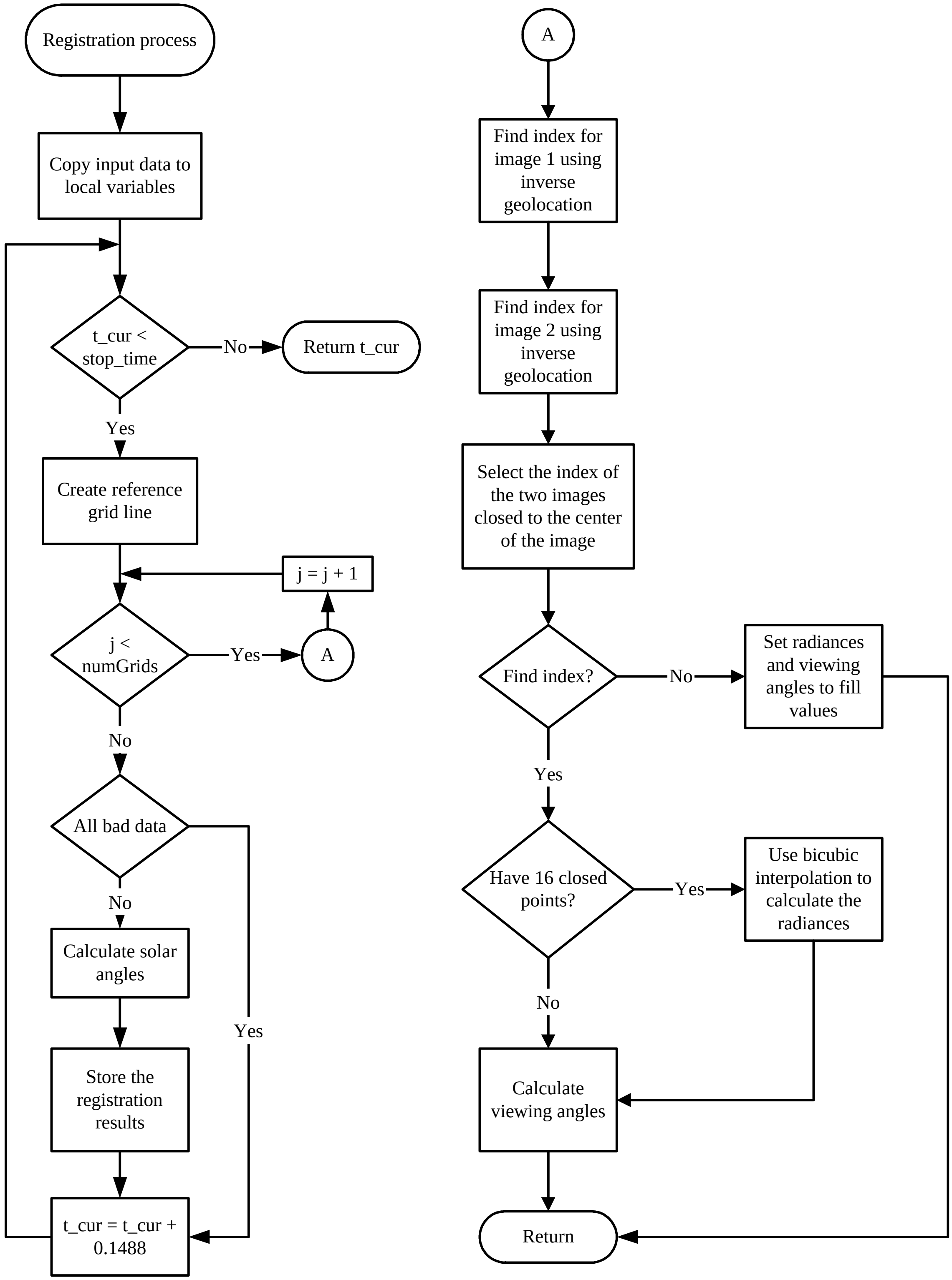

Figure 12 Registration process flowchart

Registration
-numGrids : int = 0 -dmax : int = 0 -h_proj : double = 0 -slope : double = 0 -band_time : double = 0 -location : double = 0 -sc_lookvector : double = -9999 -ref_longitude : double = 0 -ref_latitude : double = 0 -rad_quality_flag : unsigned int = 0 -ref_radiance : double = -9999 -view_zenith : double = -9999 -view_azimuth : double = -9999 -sequence_number : int = -9999 -row_number : double = -9999 -column_number : double = -9999 -coeff : double -mat_payload_iir : double -mat_sat_payload : double -mat_iir_image : double
+process() : double +getCoefficient() : int + getFOVPerimeterVector() : void + getPointingVector() : void - setChVariable() : void - createRefGrid() : int - getIndex() : void - findIndex() : void - doBicubicInterp() : void - getViewAngle() : void - setQualityFlag() : void - convertShort() : void - convertIIRToCeres() : void - convertCeresToIIR() : void - convertSCToIIR() : void - convertIIRToImage() : void - getLocation() : void - checkQualityFlag() : int - getImageVector() : void

Figure 13 *Registration* class diagram

5 Output Products

The output products of IIR Level 1 are written into HDF files. The output parameters and formats are detail described in the DOC3. The output products include a one-orbit calibration file and two half-orbit Level 1B data products, which contain geolocated, calibrated radiances. The Output System from the CALIPSO Generic Framework is used to create these output products. The ***OutputManager*** bases on the output configuration files to write the data records to output files. The output configuration files can also specify the data attributes and File Metadata that are written to output products. For more detail description of the Output System of the Generic Framework, one can look at the Data Management System Framework Architecture and Design User's Guide.

Since the time interval covered for the final product of a calibration file is one UTC day (24 hours), the ***ProcessIIRCalAssember*** process combines all the calibration output products that are within the UTC day into one file.

6 Structure Chart

The structure chart of IIR Level 1 process is shown in **Figure 14**. Each box except "main" is represented a class. The names below the line of each box are functions within the class. The boxes with **PGS Toolkit** are represented the external SDP PGS Toolkit functions called in the functions.

7 Reference:

1. IIR/CALIPSO Level 1 Processing Requirements, PIC-FDS-N1-SP-251-CNES, Edition 3, Revision 1, Draft, June 24, 2004.

2. DRD-14 CALIPSO Payload Data Measurements & Analysis Document, IN0110-DRD-14, Revision F, October 16, 2003.

3. CALIPSO Data Management System Data Product Catalog, PC-SCI-503, Release 1.1, August 2004.

4. CALIPSO Data Management System Framework Architecture and Design User's Guide, Preliminary, December 2003.

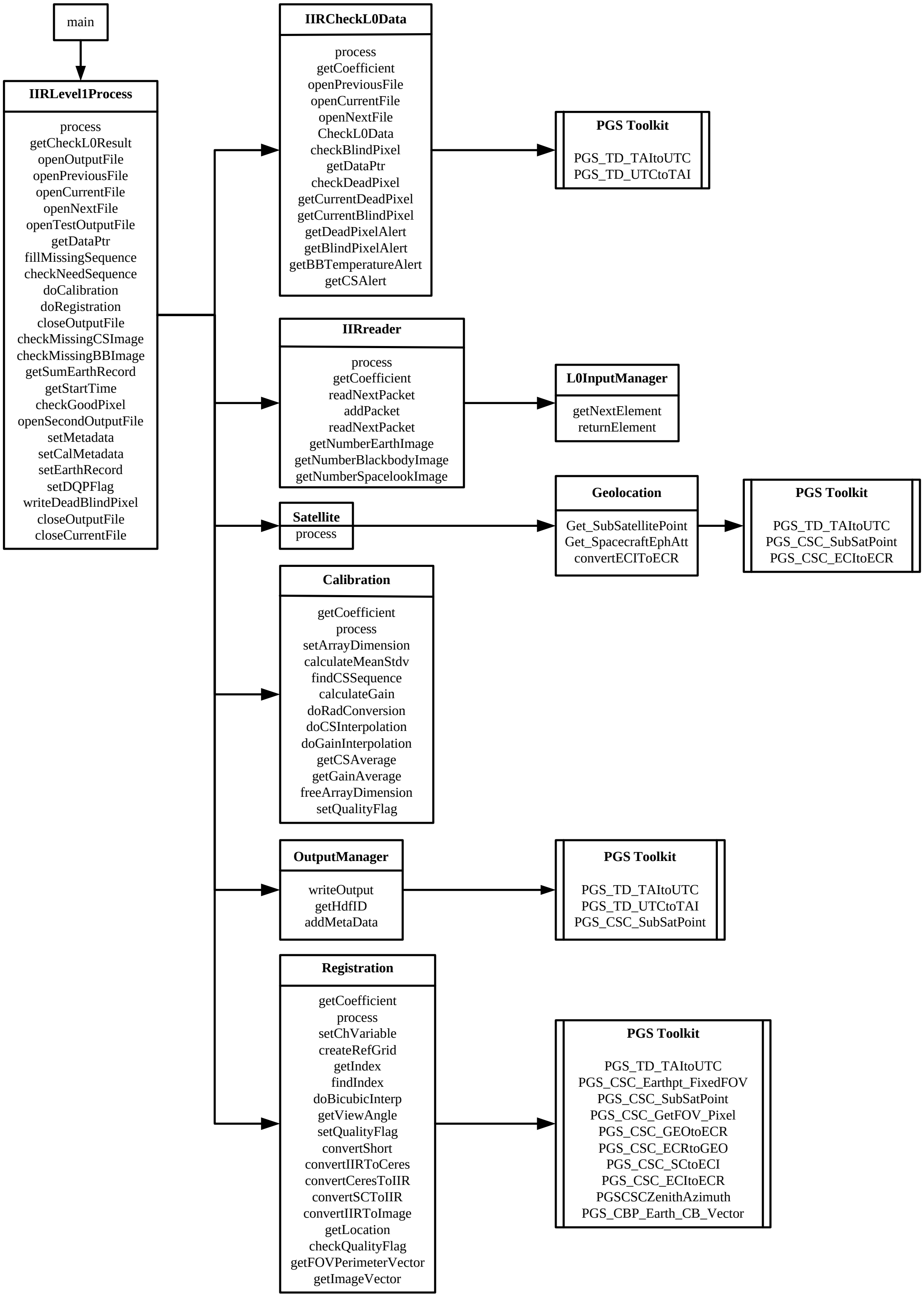

Figure 14 IIR Level 1 structure chart