LEVEL-2 Software PIPELINE FOR CADMIUM ZINC TELLURIDE IMAGER (CZTI) ABOARD ASTROSAT

Dipankar Bhattacharya
Santosh Vadawale

with software development team
at Space Application Centre, ISRO

Version 2.2
November 2015

Revision History

Version no.	Date	Status	Author	Remarks
1.0	Oct 2011	release	SAC team and D. Bhattacharya	Released as CZTI software reference document; derived from Astrosat Data Pipeline Software Requirement document v 2.5, 2009 (DB)
2.0	13 Apr 2015	release	D. Bhattacharya and Santosh Vadawale	Content trimmed, updated and new requirements added. Text revised, reformatted for wider readership and released with a different title.
2.1	28 Apr 2015	release	D. Bhattacharya	Added a missing step in cztbindata algorithm description
2.2	07 Nov 2015	release	D. Bhattacharya	Corrected a typographical error in cztgaas: sine changed to cosine.

I. Introduction

The Cadmium Zinc Telluride Imager (CZTI) aboard ASTROSAT employs an array of pixellated CZT detectors behind a Coded Aperture Mask (CAM) to detect and study hard X-ray sources. The instrument is configured as four independent quadrants, each with 16 detector modules. Each detector module has 256 pixels. There are collimator slats surrounding every detector module and a Coded Aperture Mask is placed above the collimator, about 48 cm above the detector plane. The combined geometric area of the all the CZT detectors in this payload is 976 cm^2. The primary energy range of operation of the CZTI would be about 15 to 150 keV. The CAM is made up of independent 256-element patterns for each detector module, derived from pseudo-noise Hadamard sets.

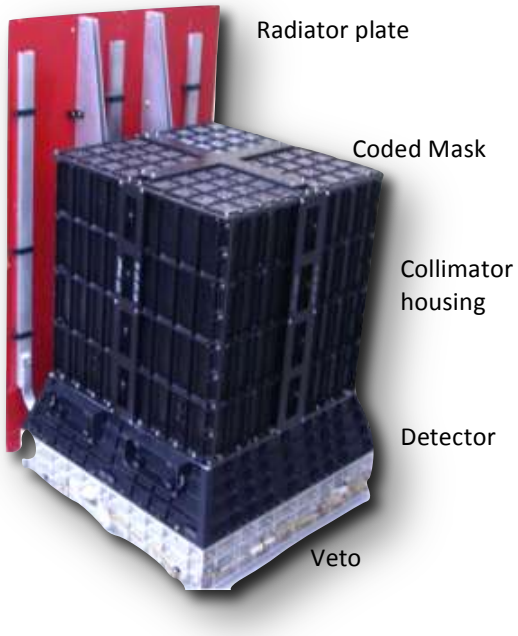

Figure 1: Configuration of the CZT Imager

II. CZTI Data Pipeline

The data received from the CZTI payload will pass through a series of steps in a processing pipeline. Three major data levels have been designated for long-term storage. These are:

Level 0
This is the raw data received from satellite telemetry, which is segregated by instrument, along with auxiliary data. This data will be archived internally and will not be available for public use.

Level 1
This is reorganized raw data, written in FITS format for Astronomical use. All auxiliary information necessary for further processing of this data will be collated at this level and packed along with the respective science data. This data will be released via the web for science use, first to the Principal Investigator (PI) of the corresponding observing proposal and, after a specified initial lock in period, to anyone interested in the data.

Level 2
This data will contain standard science products obtained from Level 1 data. Level 2 data would also be in FITS format and will be available for science use, with the same lock-in criteria and release mechanism as the Level 1 data.

The software to process data from Level 1 to Level 2 will contain user configurable elements. While a default configuration will be run at the Payload Operation Centre (POC) to create the automated Level 2 data products, the user would have the option to generate more customized products by using the same software with other configuration settings. The Level 2 pipeline software will be made available for download. For Input/Output, the implementation of the pipeline software makes use of the CFITSIO and the Parameter Interface Library developed and distributed by NASA High Energy Astrophysics Science Archive Research Centre (heasarc.nasa.gov).

The functionality of the CZTI Level 2 pipeline software is described in this document.

III. CZTI Level-1 Data Content

The starting point of Level 2 data processing is the dataset produced at Level 1. This data consists of

1. **A Science Data File**: a FITS file containing sequentially stacked 2048-byte data frames generated by the CZTI payload. This data is mode segregated, i.e. a different FITS file is generated for each distinct data mode.
2. **A Time Calibration Table (**.tct): This file contains a list of time tags expressed in CZTI internal clock, Satellite On Board reference Time (OBT) and Universal Time (UT) derived from the SPS units on ASTROSAT. As all the CZTI science data is tagged with its internal clock, the TCT file is required to correlate them with other on-board events, as well as to obtain absolute timing.
3. **An Orbit File**: Gives time-tagged orbital position of the satellite in terms of geocentric x, y, z as a function of time.
4. **An Attitude File**: Gives time-tagged satellite attitude information, in terms of quaternions as well as the RA, Dec values of the pointing directions of the three reference axes of the satellite.
5. **An LBT Housekeeping file**: This gives a time-tagged recording of 65 different health parameters monitored by different sensors on the CZTI.
6. **A Make Filter (MKF) file**: This file collects together the time series of a number of selected parameters, including health parameters, Sun angle, Earth angle, charged particle count etc, based on which the quality of data obtained can be assessed.
7. **A Good Time Interval (GTI) file**: A list of time intervals during which the data acquired may be considered good for scientific purposes, arrived at on the basis of MKF file parameters and pre-set thresholds.
8. **A Bad Time Interval (BTI) file**: A complement of the GTI file, this file records the reason for data acquired in any time interval being considered not fit for scientific use.

IV. CZTI Level 2 Data Pipeline Workflow

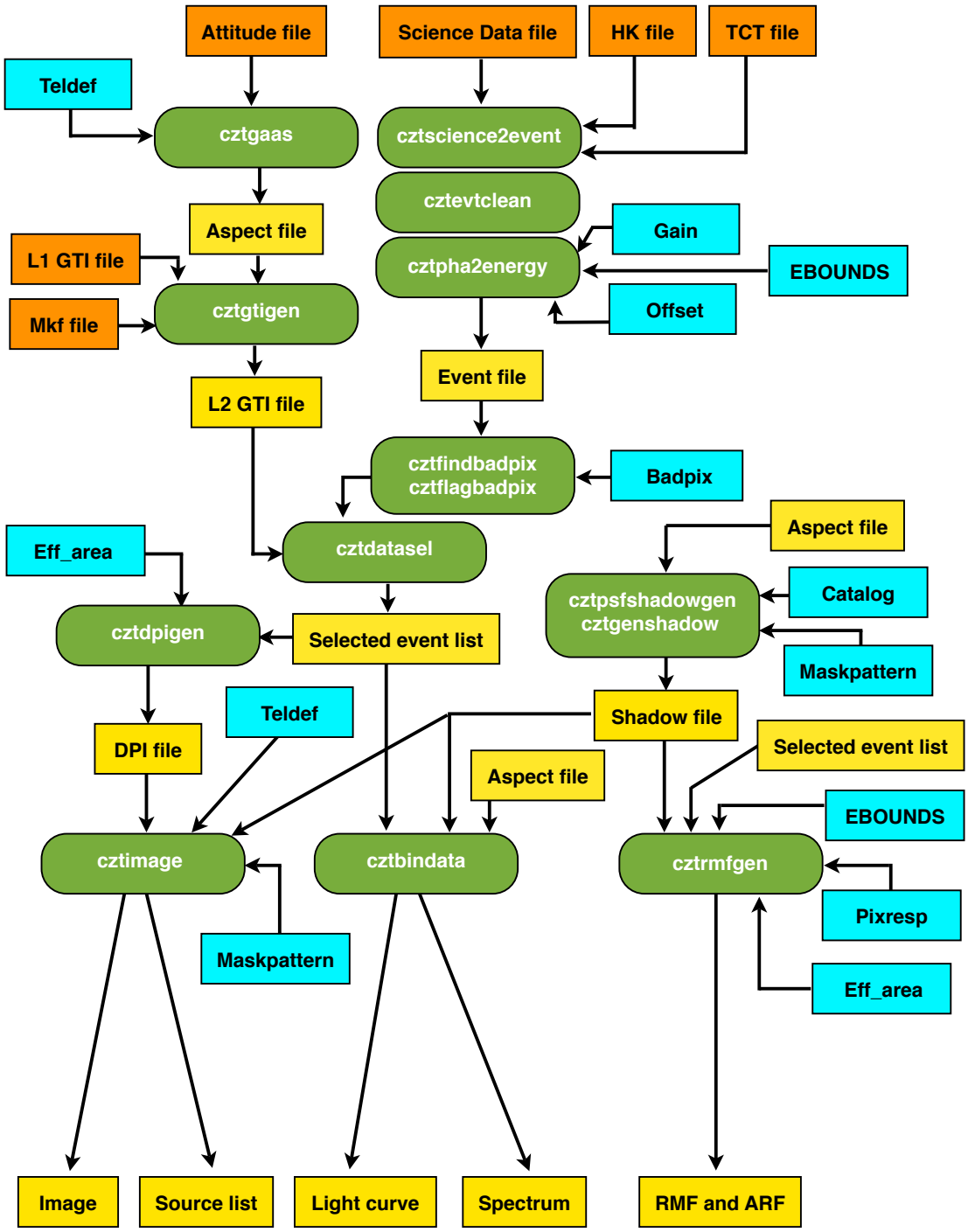

Figure 2: CZTI Level 2 Data Pipeline workflow. This starts with Level 1 data and generates Level 2 products. Shown in green are the different software modules. Orange boxes indicate Level 1 data files and yellow boxes the output files generated at Level 2. Cyan boxes are CALDB inputs.

Table 1: List of Software Modules in CZTI Level 2 Data Analysis Pipeline

Sr. No.	Functionalities in level-2 software	Module
1	a) Extraction and decoding of level-1 science data into events b) Mapping of quad_id, det_id and pix_id to (x,y) c) Mapping of 12 bit PHA to 10 bit PHA channels d) Time Calibration e) Recording of Temperature History	cztscience2event
2	PHA to PI and Energy conversion	cztpha2energy
3	a) User defined GTI generation based on MKF file and bad/flickering pixel data	cztgtigen
4	Filtering of events based on veto and alpha values	cztevtclean
5	Identification of dead, noisy and flickering pixels based on current algorithm	cztfindbadpix
6	Filtering out unusable pixels using bad pixel table	cztflagbadpix
7	Select data based on GTI	cztdatasel
8	Generate average aspect for CZTI	cztgaas
9	Generate Detector Plane Image	cztdpigen
10	Generate Light Curve/ Spectrum	cztbindata
11	Generic CZTI shadow generation module	cztgenshadow
12	Pick known sources in FOV and generate their shadows	cztpsfshadowgen
13	a) Create raw image b) Refine source list c) Create background subtracted image d) Convert camera coordinates to WCS coordinates	cztimage
14	Generate Response Matrix file	cztgenrmf

V. Description of CZTI Level 2 Software Modules

1 cztscience2event

This module reads the level-1 science data files and decodes the data packets into events. It creates an event file in fits format with events for each quadrant in different extensions.

1.1 Input

Level-1 science data file
Level-1 TCT file
Level-1 GTI file
Level-1 HK / SS mode data file

1.2 Background

The purpose of this is to organize the raw event data into a convenient tabular form with calibrated time tags for further processing. The event time tags recorded using CZTI clock are referenced and interpolated to UT using entries in the TCT file. The temperature history of each module during the period of observation is read from the HK/SS mode data file and appended as the last extension of the event file.

1.3 Method

1. Read the level-1 science data file
2. Read the data packets from the 'Data Array' column into a buffer.
3. Decode each of the 2048 byte data packet from the buffer as per the format given by onboard software.
4. Read the fields of the frame header. From the values of header fields, get the frame number, mode id, quad id, the number of events in the frame and read the event data accordingly into an event buffer.
5. Take the time for each event (fractional second) and the 'second count' value from its frame header and add both to get the event time in seconds. Using the CZTI time from level -1 TCT file, convert the time to UTC time using interpolation. Record this interpolated time as an additional column in the data. Create a time quality file as output that gives the nearest time found in TCT file for each frame time in the data.
6. Create the output event file with 4 extensions for four quadrants with extension name as Q1, Q2, Q3, Q4.
7. Create a binary table in each extension with columns 'time', 'CZtime','pha', det id', pixel id', 'veto', 'alpha' and add the events decoded from the buffer. CZtime is the CZTI time count (sec+frac.sec), with reference to a convenient offset CZTREFT mentioned in the FITS header.
8. Read the data from the event buffer and add the events to appropriate extensions (as per quad id) in the event file.
9. In each extension, add the keyword, Quad ID to the header giving the quadrant id of the data.
10. For each event in 4 quadrants, map the quad id, det id and pix id into (x, y) position on full 128 x 128 pixel array.
11. In each extension, add the column 'detx' and 'dety' and add the corresponding (x, y) values to each event.
12. Take 12 bit (4096) PHA value of each event and map it to 10 bit (1024) channels using the following equation.

    New PHA = [{1024 * (PHA+1)} / 4096] - 1

13. Replace the PHA values with this new PHA value.

14. Copy required keywords from science data file to the output event file headers.
15. Create an output GTI file having 4 extensions for four quadrants. Copy required keywords and data from level-1 GTI file to this GTI file.
16. Read temperature of each module recorded at 100-s intervals in the SS mode data file. Record these as a time series in the last extension of the event file called "TEMP".

1.4 Output

Level-2 event file
Level-2 GTI file
Time Quality Tag file

2 cztpha2energy

The module converts the PHA value in event file to PI and energy value for each event.

2.1 Input

Level-2 event file
Gain file from CALDB
Offset file from CALDB
EBOUNDS file from CALDB

2.2 Background

The energy of each X-ray photon striking the CZT detector is measured as a voltage, which through the Analog-to-Digital Convertors in each module is recorded as a Pulse Height Amplitude (PHA) channel value. This software module uses pixel-wise calibration data to estimate the incident energy of the photon from the recorded PHA, and express it on a Pulse-Invariant and Pixel-Invariant scale, called a PI channel value.

2.3 Method

1. Copy the input event file to output event file
2. Read the EBOUNDS file from CALDB
3. Create three columns, PI, Energy and ECONV_FLAG in the output file
4. Read the PHA, detID and pixID for each event.
5. Read the temperature of the detector at event time from the TEMP extension.
6. Read Gain and Offset at a temperature nearest to the actual detector temperature from the corresponding CALDB files.
7. Read the corresponding Q_FLAG from the CALDB files. Assign this value to ECONV_FLAG
8. Use E = Gain*PHA + Offset to calculate Event Energy
9. From EBOUNDS, find which PI bin the computed E belongs to
10. Write the PI bin number, Event Energy and ECONV_FLAG to the event file

2.4 Output

Event file with PI, Energy and ECONV_FLAG columns added

3 cztgtigen

This module generated a Good Time Interval (GTI) file based on the Level-1 GTI, instrument health parameters recorded in the MKF file and the time-dependent bad pixel list.

3.1 Input

Level-1 GTI file

Level-1 MKF file
Bad pixel list

3.2 Background

The GTI file generated during Level-1 processing is created on the basis of orbital, attitude and spacecraft control-related parameters. This needs to be augmented with additional checks on the detector performance, which is done here. The housekeeping parameters are recorded in the MKF file and pixel behaviour history in the Bad pixel list.

3.3 Method

1. Create a GTI file with four extensions each for a quadrant.
2. Get the required threshold values for all HK parameters from the user.
3. Read the HK parameters from the Level-1 MKF file for each of the quadrant.. Take the time intervals when the parameters are within threshold limits and write these intervals in GTI file.
4. If required by user, take the level-1 GTI file and further exclude the time interval from the output GTI file which is not in Level-1 GTI file.
5. If required by user, further refine the time intervals based on bad pixel information
6. Every interval that is not included in the GTI, should be recorded in an output BTI (Bad Time Interval) file along with the reason for the interval being considered bad.

3.4 Output

Level-2 GTI file
Level-2 BTI file

4 cztevtclean

This module filters the events based on veto and alpha tag value.

4.1 Input

Level-2 event file with veto and alpha columns

4.2 Background

Events with simultaneous veto count or alpha particle detection are special, and may be used for calibration. The alpha particles are generated by an on-board radioactive source, which also generates simultaneous photon events. These events need to be separated from the rest of the events which constitute science data. This module provides the functionality to select/reject alpha and veto-tagged events in various combinations according to user-supplied choices.

4.3 Method

1. Create the output event file with same format as input event file. Copy the header and create the event table same as input event file.
2. Read the value of two boolean parameters viz. 'alphaclean' and 'vetoclean'
3. Read the 'alpha' value (0 or 1) for accepting events
4. Read the veto range that needs to be accepted or rejected.
5. Read the value of flag 'vetoaccept'
6. If (alphaclean = true and vetoclean = true) then
    a. If (vetoaccept = true) then select events with the same alpha flag given as input and veto value within the input veto range
    b. Else select events with the same alpha flag given as input and veto value outside the given range
7. Else If (alphaclean = true and vetoclean = false) then select all the events with the same alpha flag given as input.

8. Else If (alphaclean = false and vetoclean = true) then
    a. If (vetoaccept = true) then select events with veto value within the input veto range
    b. Else select events with veto value outside the given range
9. Else copy all the events to output file.

4.4 Output

Cleaned Event file

5 cztfindbadpix

This module identifies dead and noisy pixels in the data

5.1 Input

Event File

5.2 Background

Some fraction of pixels in the CZT detector shows time-variable misbehaviour. It is observed that certain pixels may become Noisy, i.e. they tend to show very high counts, and some pixels may become Dead, showing very low counts. Over time, some of these pixels may also recover from such anomalous states. This module identifies and lists the occurrence of such hot and dead pixels during an observation.

5.3 Method

1. Create an output bad pixel file with two extensions
    a. First extension having an image of 128 x 128 pixels.
    b. Second extension with a binary table which contains the list of bad pixels (dead/noisy) with columns 'detx', 'dety' and quality (1/2)
2. Read the threshold value "t" from the user.
3. Detect dead and noisy pixels with the following procedure
    a. Bin the counts of the pixel into a raw DPI.
    b. Make an exposure map e(i,j) by creating an array of transmission fraction to each pixel from the direction of the source.
    c. Find the mean source count S and background count B by making a least square fit of the DPI count matrix count(i,j) to the relation Cmodel(i,j) = S*e(i,j)+B. Find deviation D(i,j) = count(i,j)-Cmodel(i,j) and store in an array.
    d. Separate the pixels into two sets – one for which e(i,j) > 0.5 and the other consisting of the rest. Carry out the following steps separately for these two sets.
    e. Find rms of the pixel set by averaging $|D(i,j)|^2$ over the members of the set and taking a square root. Store this in variable rms_pix.
    Repeat f thru h for each pixel in the set
    f. If the criteria "|D(i,j)| < t * rms_pix" , holds true then the pixel is good. Update the bad pixel file accordingly.
    g. If "D(i,j) < - t * rms_pix" then the pixel is dead, update the output file accordingly
    h. If "D(i,j) > + t * rms_pix" then the pixel is noisy, update the output file accordingly
4. Finally the bad pixel file will give the bad pixels in the image (dead/noisy). The identification may be carried out in multiple time blocks and time-tagged history of pixel performance may be recorded in the output file.

5.4 Output

Bad pixel file

6 cztflagbadpix

This module attaches quality flag to each event based on the quality of the pixel at which the event occurred.

6.1 Input

Event File
Bad pixel file created by cztfindbadpix
Bad pixel legacy table from CALDB

6.2 Background

The list of bad pixels identified in the present run, and of those already marked as bad based on earlier performance history, are combined and each event in the event file is attached with a tag corresponding to the respective pixel quality.

6.3 Method

2. Copy the input event file as output event file.
3. Create a column 'Quality' in output event file.
4. Read the detx and dety value for each event from the event file.
5. Check whether this pixel falls in dead pixel. If it is found in dead pixel map, then add quality value as 1 to the Quality column
6. Check whether the pixel falls in noisy pixel. If it is found in noisy pixel map, then add quality value as 2 to the Quality column
7. If it is not found in any of dead and noisy pixel map, then add quality value as 0 to the Quality column.

6.4 Output

Event file with pixel quality column added

7 cztdatasel

Selects a subset of data based on GTI and pixel quality information

7.1 Input

Event file
GTI file
Bad pixel file

7.2 Background

Some of the science data may be recorded in non-optimal conditions with the pointing or instrument operating parameters out of acceptable range. The time intervals during which these parameters are within acceptable limits are listed in a GTI (Good Time Interval) file or a GTI extension within the event file. This module helps retain only those events which fall within the Good Time Interval and reject the rest. Data within time range affected by noisy pixels can also be rejected by this module.

7.3 Method

1. Create an output event file with same format as input event file leaving the event data.
2. For each of the quadrant, read the corresponding good time intervals from GTI file and select only those events which fall under GTI.
3. If filtering by noisy pixel interval is desired then read pixel performance history from bad pixel file and reject events that fall within the time interval where data may be affected by noisy pixels
4. Write the selected events to output file
5. Add the GTI extensions from GTI file to output event file.

7.4 Output

Event file with selected events and GTI extensions

8 cztgaas

This module computes the average pointing direction of the CZTI during the observation.

8.1 Input

Level-1 Attitude file
Teldef file from CALDB
Event file

8.2 Background

In order to find the position and orientation of the CZTI field of view one needs to utilize the satellite aspect information given in the attitude file and account for the alignment of the CZTI payload with respect to the satellite reference axes. The latter is given in the Teldef file stored in CALDB.

8.3 Method

1. Read the CZTI alignment matrix elements from teldef file. The matrix elements define the transformation of a vector (DETX, DETY, DETZ) defined in detector coordinates to satellite body coordinates (SATX, SATY, SATZ) as:
   SATX = ALIGNM11*DETX + ALIGNM12*DETY + ALIGNM13*DETZ
   SATY = ALIGNM21*DETX + ALIGNM22*DETY + ALIGNM23*DETZ
   SATZ = ALIGNM31*DETX + ALIGNM32*DETY + ALIGNM33*DETZ
   i.e.: {SAT} = **[ALIGNM]***{DET}
2. From the Attitude file, for every recorded sample, read (RA, Dec) values of the Yaw, Roll, Pitch axes. From these, derive their Direction Cosines in the Inertial Coordinate System.
   Ly = cos(RA_y)cos(DEC_y) ; My = sin(RA_y)cos(DEC_y) ; Ny = sin(DEC_y)
   Lr = cos(RA_r)cos(DEC_r) ; Mr = sin(RA_r)cos(DEC_r) ; Nr = sin(DEC_r)
   Lp = cos(RA_p)cos(DEC_p) ; Mp = sin(RA_p)cos(DEC_p) ; Np = sin(DEC_p)
3. Construct a transformation matrix from satellite body coordinates to the Inertial Coordinate System (ICS):
   [M] = Ly Lr Lp
   My Mr Mp
   Ny Nr Np
4. For CZTI-z: {zDET} = (0, 0, 1) and CZTI-x: {xDET} = (1, 0, 0) find the components in the ICS:
   {N} = (Nx, Ny, Nz) = **[M]* [ALIGNM]***{zDET} and
   {Nt} = (Nxt, Nyt, Nzt) = **[M]* [ALIGNM]***{xDET}
   Record (Nx, Ny, Nz) and (Nxt, Nyt, Nzt) along with the timestamp of the sample read from the Attitude file, in the output CZTI aspect array file. Repeat for all the samples in the Attitude file.
5. Average the components individually over the duration of the observation to obtain vectors (<Nx>, <Ny>, <Nz>) and (<NXt>, <Nyt>, <Nzt>). Normalize the vectors:
   (Ax=<NX>/Rn, Ay=<NY>/Rn, Az=<NZ>/Rn), where Rn=$(<NX>^2 + <NY>^2 + <NZ>^2)^{1/2}$ and
   (Tx=<NXt>/Rt, Ty=<NYt>/Rt, Tz=<NZt>/Rt) where Rt=$(<NXt>^2 + <NYt>^2 + <NZt>^2)^{1/2}$.
6. Find the average pointing direction of the CZTI:
   DEC = asin(Az); RA = atan2(Ay,Az) mod 360 deg
   and The average TWIST angle from:
   TWIST=atan2[{Ty cos(RA)-Tx sin(RA)},{Tz cos(DEC)–sin(DEC)(Tx cos(RA)+Ty sin(RA))}]
7. Record the above normalized vectors, RA, Dec and TWIST values in the FITS header of the Event file

8.4 Output

CZTI average aspect file
Event File with recorded average aspect

9 cztdpigen

This module bins the count data of the pixel based on given time and energy ranges and generates a detector plane image.

9.1 Input

Event File with PI, energy, quality columns
GTI file
QE map from CALDB

9.2 Background

The Detector Plane Image (DPI) gives the shadow of the coded mask recorded on the CZT detector. The pattern of total counts on each pixel gives a Detector Plane Histogram (DPH), which is then corrected for difference in quantum efficiency of different pixels to yield the DPI.

9.3 Method

1. Read the event data from event file into a buffer.
2. Read the time binning type. (Binning type can be uniform/gti. In uniform binning it makes equal time intervals based on time in size. In gti binning, intervals are taken from gti file specified)
3. Read the time bin size from user. (time bin size is used for uniform binning. If it is 0, full time range is taken)
4. Get the energy ranges from the user.
5. Get acceptable quality range from the user.
6. Generate a DPH for each of the combination of time and energy range by counting the number of events of desired quality at each detector pixel. Write the DPH in an output file.
7. Divide the count at each pixel by the relative QE of that pixel to yield the DPI
8. Write the generated DPIs in extensions of a file.
9. Create an EBOUNDS extension and write the energy ranges used into it.
10. Create gti extensions and write the time intervals in this extension.

9.4 Output

Detector Plane Image file
Detector Plane Histogram file

10 cztbindata

Generates Light Curve and Spectrum for the dominant source in the Field of View by directly binning counts in time and energy.

10.1 Input

Event file
EBOUNDS file from CALDB

10.2 Background

This module works by evaluating weighted sum of event counts. The weights are computed for each pixel based on the relative exposure the pixel receives to the source, given its direction in camera coordinates, and the coded mask pattern. This module is

designed to work up to photon energy of 100 keV, where partial transmission through closed elements of the coded mask plate may be ignored.

10.3 Method

1. Read time range, energy range, binsize (time bin width in case of light curve and energy bin width in case of spectrum).
2. Get source RA and Dec from event file header or the user.
3. Find camera coordinates theta_x, theta_y of the source (using the average aspect).
4. Compute shift of mask pattern as:
   Delta_x = -h*tan (theta_x), Delta_y = -h*tan (theta_y);
   where h= height of mask pattern from the shadow.
5. Shift the mask pattern by Delta_x and Delta_y on the detector plane. Discard the portion of the pattern that falls outside the detector due to this shift. Assign zero weight to detector pixels falling in the area not covered by the shifted mask pattern. For detector pixels within the shifted pattern determine the fraction of open area f_i within each detector pixel (as in the cztgenshadow module). Assign a weight $w_i' = 2f_i - 1$ to each of these pixels.
6. Assign a weight of zero to all dead, noisy and bad pixels.
7. Evaluate renormalization offset $D=\sum w_i' \, a_i / \sum a_i$, where a_i is the geometric area of individual pixels, and the sum runs over all active pixels. Redefine their weights as $w_i = w_i' - D$.
8. Compute weighted area $A = \sum w_i f_i a_i$, Also compute a reliability index $R = 1 - (\sum w_i)/(\sum w_i f_i)$. Report the values of A and R in screen output/log file as well as in the header of the output file.
9. If required output is light curve, then follow steps for subsequent time bins until bin end time exceeds the observation end time or user supplied end time
   a. Generate a DPI array C_i by binning all the events satisfying the energy range criteria, and falling within one bin time width after the bin start time
   b. Find the flux value $F = \sum w_i C_i / A$ (in counts per unit area)
   c. Compute the standard error $E = \sum w_i^2 \, C_i / A$
   d. Find mid UT for the bin as (bin start time + bin width/2).
   e. Output mid UT, F and E in light curve data file.
10. If required output is spectrum, then follow steps for subsequent energy bins until the end of specified energy range is reached.
    a. Generate a DPI array C_i by binning all the events within the desired time range and falling within one energy bin width after the bin start energy
    b. Find the flux value $F = \sum w_i C_i / A$ (in counts per unit area)
    c. Compute the standard error $E = \sum w_i^2 \, C_i / A$
    d. Find mid energy of the bin as (bin start energy + bin width/2).
    e. Output mid energy, F and E in spectrum data file.
11. Create output file and write the required data into it.

10.4 Output

Light Curve / Spectrum file

11 cztgenshadow

This module computes the shadow of the coded mask cast by a source on the detector, given its location in camera coordinates and its flux.

11.1 Input

Source location in camera coordinates (theta_x, theta_y)
Source flux
Mask pattern
Photon energy range
EFFECTIVE_AREA file from CALDB

11.2 Background

Coded Mask cameras work by casting the shadow of the mask pattern on the detector by sources in the sky. The observed shadow pattern (count distribution on detector pixels) is a linear combination of shadows cast by multiple sources in the field of view. The image reconstruction works by fitting shadow sums to the observed count distribution. A key element of this process is computing shadow templates accurately, ready for fitting. In order to generate the shadow pattern this module uses a geometric ray tracing algorithm and computes the transmission fraction through the mask to each detector pixel.

11.3 Method

1. Using theta_x and theta_y project the four corners of a detector pixel *i* onto the mask plane by shifting positions by h*tan(theta_x) in the x direction and h*tan(theta_y) in the y direction, where h is the distance between the mask plate and the detector. Do this for each module of the detector separately.
2. Find the source angle with the camera normal: theta = arctan[sqrt($\tan^2$ theta_x + $\tan^2$ theta_y)]
3. Find the optical depth of the mask plate in the given energy range: tau = mu*t/cos(theta) where mu is the effective absorption coefficient of Tantalum in the energy range and t is the thickness of the mask plate.
4. In the mask area bounded by the four projected corners, find the fraction of the area *r* that is blocked by tantalum. $f_i = (1 - r) + r*\exp(-\text{tau})$ is then the transmission fraction to this pixel. The f_i array is the normalized shadow of a source in this direction.
5. Convert the supplied source flux into counts per unit area *C*.
6. $S_i = C * f_i * a_i(\text{theta,energy})$ is the final shadow pattern, where a_i(theta, energy) is the effective area of the respective pixel in the given photon energy range and angle of incidence theta. Write out the S_i array

11.4 Output

The computed shadow pattern in a FITS file

12 cztpsfshadowgen

The module identifies the known sources in the field of view of the camera using the catalog file. It generates the shadows/DPIs for the known sources and stores in a file.

12.1 Input

Event File with aspect information
Catalogue file

12.2 Background

Given the average aspect, the pointing direction of the imager is defined. The field of view is projected in the sky around the central pointing direction. From the catalogue, known sources that fall within this FOV are picked and the shadows of the coded mask cast by them on the detector are computed by this module.

12.3 Method

1. From the average aspect and twist angle read from the Event file header, compute the Inertial frame direction cosines (Tx, Ty, Tz), (Bx, By, Bz) and (Ax, Ay, Az) of the CZTI x, y and z axes respectively.
2. Construct a transformation matrix between sky coordinates and camera coordinates:

   [P] = Tx Ty Tz
   Bx By Bz
   Ax Ay Az

3. For each source in the catalog construct its inertial vector {SRC} from RA, Dec value.
4. Transform the inertial vector to camera coordinate system as {CAM} = **[P]***{SRC}.
5. From the components of the vector {CAM} evaluate theta_x, theta_y
6. If theta_x, theta_y are within the specified FOV limits then save the details of the source (theta_x, theta_y and flux) in an output file of selected sources.
7. For each source in the selected sources list call **cztgenshadow** to generate a shadow for the corresponding location and flux.

12.4 Output

List of selected sources with coordinates and flux (ascii file)
PSF shadows of known sources in a FITS file, one extension per source

13 cztimage

This module reconstructs the image of the target by correlating the mask pattern and the DPI. It uses DPI fitting to estimate the background and then creates a background subtracted image.

13.1 Input

DPI file with CZTI average aspect recorded in the header
Mask pattern file from CALDB
TELDEF file from CALDB

13.2 Background

This module works in two steps. First a cross-correlation is performed between the mask pattern and the observed shadow using a fourier technique. This yields a crude image. Significant peaks above noise in this image are then located, and accurately computed shadows for those locations are fit to the observed shadow via least squares technique. The source list is iteratively pruned to a minimum set that still yields a good fit. The fit coefficients resulting from this are the flux values of the detected sources.

13.3 Method

1. Oversample the DPI and mask array by 5 and make an array of 640 x 640 (128*5) pixels.
2. Take Fourier Transform of DPI and mask array and multiply the arrays element by element. Take the inverse transform of this array resulting from the product.
3. Retain just the central 181 x 181 elements of the resulting array. This gives the raw image in camera coordinates.
4. Compute the rms of this image and locate significant peaks in the image above preset threshold times the rms. Store the positions (theta, phi) and counts in saved source list.
5. Use DPI fitting to further refine the source list and compute the background:
    a. Model the DPH as a linear combination of the shadows of the saved sources, plus a six-element background component. This is done by performing a least-squared fit, with the source strengths and the six background coefficients as fit parameters.
    b. Examine the values of the fitted strengths of the sources. Those falling below a pre-set threshold times image rms are removed from the saved source list. Go back to step “a” and re-fit with the remaining sources. Repeat until no further sources are dropped from the list.
    c. If fluxes (or upper limits) are required to be determined for a set of known sources then retain their camera coordinates in the fitting process.
6. Subtract the fitted background from the DPI and repeat steps 1 thru 3 to get a background subtracted image.
7. Compute transformation between camera coordinates and WCS using the average aspect and TELDEF file.
8. Add RA, Dec as new columns in saved source list.
9. Add WCS keywords to headers of raw and background subtracted image.

10. Convert the images to jpg, also as level-2 products.

13.4 Output

CZTI raw and background subtracted image files in FITS and JPG
Source list

14 cztrmfgen

This module creates spectral response of the CZTI in the RMF format for use in spectral fitting by packages like XSPEC.

14.1 Input

PIXRESP file from CALDB
EFF_AREA file from CALDB
Bad pixel file
Shadow file for the source
Spectrum (.pha) file for the source

14.2 Background

This module performs a weighted combination of spectral responses of individual pixels to generate an overall response matrix file for the CZTI instrument. Pixels are assigned weights proportional to their exposure fraction to the source and their net effective area at a given energy band. Pixels flagged with poor spectral quality are left out of this computation, and should be ignored in subsequent spectral analysis.

14.3 Method

1. Read PIXRESP, EFF_AREA from the respective CALDB files and exposure fraction from the Shadow file.
2. Remove pixels with poor spectral quality, as flagged in the Bad pixel file.
3. Combine the spectral response of the remaining pixels, weighted by their exposure fraction and effective area at the relevant angle and energy, and compute overall response for the given spectrum.
4. Write the result in standard response matrix format in an output RMF file.

14.4 Output

Response Matrix File