The EP-FXT Users Guide

v1.30 (for fxtsoftv1.30)

EP-FXT Science Data Center

March 31, 2026

Contents

List of Figures

Chapter 1

Introduction

1.1 Scope

This document is meant as a guide and reference for users who want to use the Follow-up X-ray Telescope (FXT) data to extract scientific products.

Einstein Probe is an X-ray astronomy mission of the Chinese Academy of Sciences (CAS), and carries two payloads, the Wide-field X-ray Telescope (WXT, 0.5–4 keV) and the FXT. The FXT is used to conduct deep follow-up observations of interesting targets discovered by WXT and it also has the capability to observe an interesting target, which can be commanded from the ground segment via the command data uplink route. It consists of two X-ray mirror modules with 54 nested Wolter-I paraboloid-hyperboloid mirror shells for each. Figure 1.1 shows the two of the mirror modules of FXT. Furthermore, the PN-CCD is developed as a focal plane detector for FXT. It can detect single X-ray photon, which is imaged by the mirror to the focal plane, and provide energy, position and time of photons in the energy range from 0.3 up to 10 keV. The two PN-CCDs for the two modules are placed by rotating an angle of 90^o relative to each other in the focal plane.

Figure1.2 shows the the layout of the PN-CCD. The PN-CCD has an integrated image and a frame-store area. The integrated image has 384 $\times$ 384 pixels with pixel size of 75 μm $\times$ 75μm, while the frame-store area also has 384 $\times$ 384 pixels correspondingly, but with smaller pixel size 75 μm $\times$ 51μm. During the readout time, the integrated image is transferred to image frame, and then the PN-CCD channels are read out in parallel by three CAMEX chips.

This guide describes the principles of the processing and reduction of the FXT data. The FXT Data Analysis Software Package (FXTDAS) is to achieve the FXT data analysis processing and extract scientific products. Its purpose is to achieve scientific products, such as energy spectra, light curves, Ancillary Response Files (ARF), Redistribution Matrix Files (RMF) and background files. It provides several tasks with each task is to accomplish a step of data analysis. These tasks are written in ftools style and are fully compatible with the HEASoft software.

This guide assumes that the data have already been downloaded from the archive,

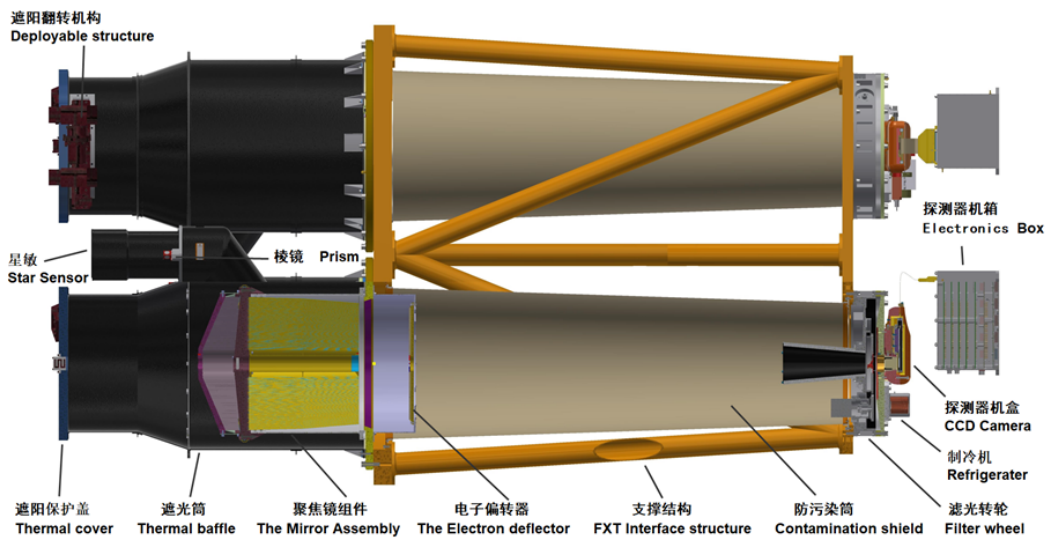

Figure 1.1: FXT mirror modules.

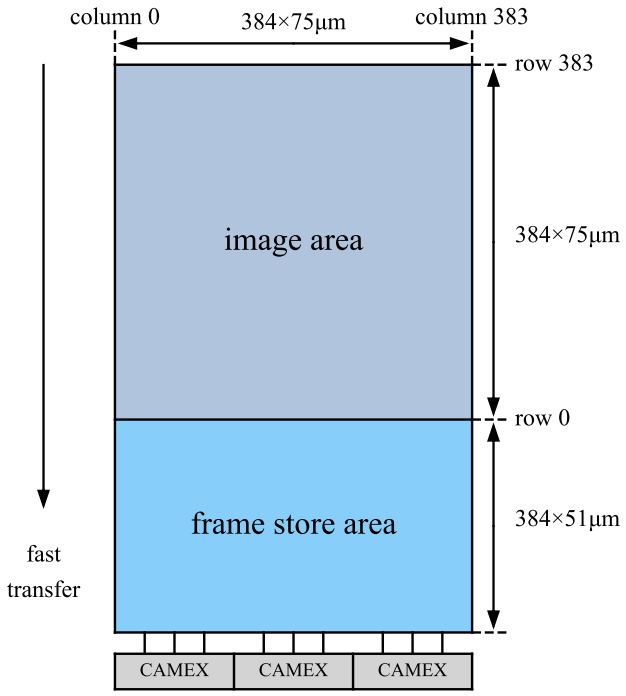

Figure 1.2: Layout of the PN-CCD of the FXT

and the FXT software and calibration data provided in CALDB are installed and initialized.

1.2 Organization of this Guide

The second chapter describes the data modes of the FXT and these modes are important issues to data reduction and analysis.

The third chapter is devoted to a brief description of the FXT data files and data formats.

The fourth chapter gives a description of the steps involved in the data reduction and the specific tools used for point sources and extended sources, respectively.

The fifth chapter describes the calibration files of the FXT, which are used in the data reduction software.

First appendix: installation of the FXTTDAS and its setup.

Second appendix: list of the individual helps for each of the FXT specific tasks.

The development task of FXT is undertaken by Institute of High Energy Physics (IHEP) , Chinese Academy of Sciences (CAS) , with participation of Technical Institute of Physics and Chemistry (TIP) , CAS. The European Space Agency (ESA) and the Max Planck Institute for Extraterrestrial Studies (MPE) have contributed to the successful development of FXT through international cooperation.

Chapter 2

FXT modes

Einstein Probe's Follow-up X-ray telescope (FXT) is designed to perform prompt follow-up observations of trigged sources from WXT and observe other interested targets during the all sky survey at the rest time. Table 2.1 summarise the FXT parameters.

Table 2.1: The FXT characteristics.

FXT parameters	Value
Energy range	0.3–10 keV
Mirror	Wolter-I
Detector	PNCCD
Effective area	600 cm^2 @1.25 keV (on-axis)
Field of view	$60' \times 60'$
Detector elements	384×384
pixel scale	$75\mu m \times 75\mu m$
HPD	$< 30''$ (on axis)
Sensitivity	2.4×10^{-13} erg cm^{-2} s^{-1} (1500 s exposure time)

The FXT is responsible for quick follow-up observations of the triggered sources by WXT and it can automatically select the science mode to fit the source count rate, while it will also observe other interested targets and the science mode of a given observation can be commanded from the ground segment via the command data uplink route. There are mainly three science modes depend on the source brightness (see table 2.2). The Full Frame mode (FF) and Partial Window mode (PW) are designed for weak source ($<$ 2 mCrab for FF; $<$ 40 mCrab for PW, thin filter and pile-up $<$2%), but a better position identified. The full field of view of FF mode is accumulated every 49.8848 ms and then read out by 0.1152 ms. Each CCD frame is rapidly transferred into the frame-store area, and then read out by clocking one row at a time into the serial register. For the PW mode, the exposure area is a 128×61 (x×y) region located in the center of CCD frame, and the exposure time is about 2.1319 ms and read time is 0.0681 ms. While, the Timing mode (TM) is

used for very bright sources (<100 mCrab for sepctral analysis) and for high time resolution. This mode performs one parallel clock shift row by row to frame-store area. When a frame is finished, a short reset time (about 342.08 μs) will be provided in order to reset the readout node and during this short integrated time, the photons will be collected from the source.

Table 2.2: FXT science mode

Mode	Image capability (x×y)	Read time per frame	Flux level (thin filter, pileup<2%)	Fraction of OoT
FF	384 × 384, 2D	50 ms	< 2 mCrab	0.23%
PW	128×61, 2D	2.2 ms	< 40 mCrab	<1%
TM	128×384, 1D	23.68 μs (Read time per row)	< 100 mCrab (spectral analysis) < 1 Crab (timing analysis)	-

2.1 The Full Frame mode

The full frame mode retains full imaging but the time resolution is limited. A full field of view is accumulated every 49.8848 ms and frame-transfer is about 0.1152 ms. Each CCD frame is rapidly transferred into a framestore area and then read out by three CAMEX (CAMEX_L, CAMEX_M and CAMEX_R).

In the FF mode, the energies and the positions of the signals exceeding the threshold are measured in the whole image area at a frame rate of 20 frames/s. During the exposure time, the frame area is read out before transfer of the next image to the frame store area.

The pixels are processed on board where the bias is subtracted and the lower level discriminator is applied. However, the events are not reconstructed.

The photons are not only registered during the integration time but also during the readout time of PN-CCD. These events during the readout time are aclled out-of-time events (OoT events). The PN-CCD PHA value of each event is always converted to PI (Pulse Invariant) value based on its position, due to the gain and Charge Transfer Inefficiency (CTI). But the OoT events is assigned a wrong position, leading to a wrong PI correction and will broaden spectral features. The OoT fraction is the ratio of the readout time to integration time and for this mode, the fraction is about 0.23% (see table 2.2).

The initial data processing for this mode has to do for the following:

- Flag bad pixels
- Calculate and Flag hot and flickering pixels
- Assign grade and PI values

Additional, FXT also records events of the frame store area during the integration time of the image area. These frame store area events mainly be caused by space particles and are labeled with the keyword “fsaevt” in their filenames. They can be used to estimate the instrumental background of the image area.

2.2 The Partial Window mode

The partial window mode only readouts the pixels in the central part 128×61 controlled by the middle CAMEX (CAMEX_M). The time resolution is 2.2 ms. Figure 2.1 shows the readout region of PW mode.

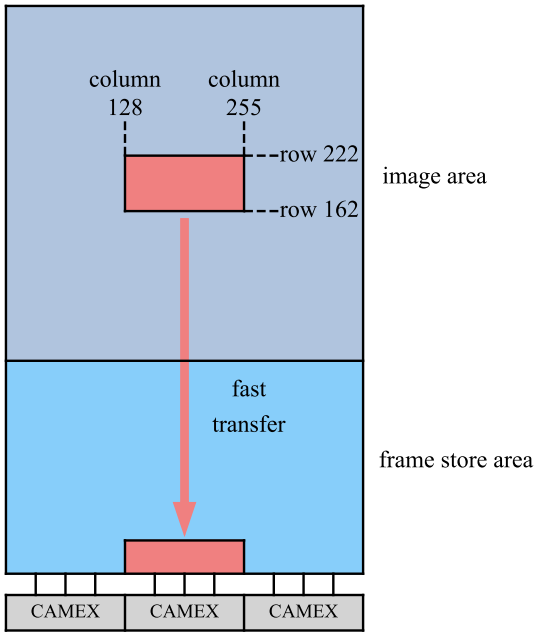

Figure 2.1: The readout region of partial window mode

For this mode, the sub-image can be transferred to frame-store area by 227 serial clock shifts (about 0.0681 ms) and during this time, the OoT events are still registered in the CCD. However, the photons only located in the region of 162–222 rows can be read out and other photons will be discarded. Therefore, the fraction of OoT correlates with the source position. The fraction for a source in the sub-image can be estimated by $f = 0.832\% - (N - 161) \times 0.01364\%$, where N means the Y position of the source and has the range from 162 to 222.

In this mode, there are also OoT events from the outside of the sub-image region and the fraction can be obtained if the source position is known. If the

source's position is in rows 0–161, the probability of a photon being recorded is $61 \times 0.3/2200 = 0.832\%$. If the source's position is in rows 223–318, the photon will not be recorded. if the source's position is in rows 319–379, the probability of a photon being recorded is $(N - 318) \times 0.01364$. if the source's position is in rows 380-383, the probability of a photon being recorded is $61 \times 0.3/2200 = 0.832\%$.

The pixels are processed on board where the bias is subtracted and the lower level discriminator is applied. However, the events are not reconstructed.

The initial data processing for this mode has to do for the following:

- Flag bad pixels
- Calculate and Flag hot and flickering pixels
- Assign grade and PHA values

2.3 The Timing mode

The timing mode is designed for very bright sources and for high time resolution. This mode performs consecutive row readouts and the shift clock, which corresponds to the readout of one row, is about 23.68 μs. However, after reading out 384 rows (a frame), there will be a pause time (reset time) about (6×23.68+200) μs. Figure 2.2 shows the readout region in the PN-CCD.

In this mode, photons from source are uniformly distributed along the readout direction, but there is an count enhancement at the position of the source during the reset time.

This mode does not record the arrival time of each photon, but records the frame time and the order of photons at the CAMEX_M. Therefore, we need to calculate the arrival time of each photon through algorithms.

The pixels are processed on board where the bias is subtracted and the lower level discriminator is applied. However, the events are not reconstructed.

The initial data processing for this mode has to do for the following:

- Assign the proper arrival time to each event
- Flag bad pixels
- Calculate and Flag hot and flickering pixels
- Assign grade and PHA values

In the timing mode, there is an inherent periodic signal with a period of approximately 9.4352 ms ($\sim$ 105.986 Hz). Figure 2.3 illustrates the profile of this periodic signal. This periodic signal is caused by readout system and photon arrival time calculation. If there are signals with frequencies close to this signal, users need to handle them with caution.

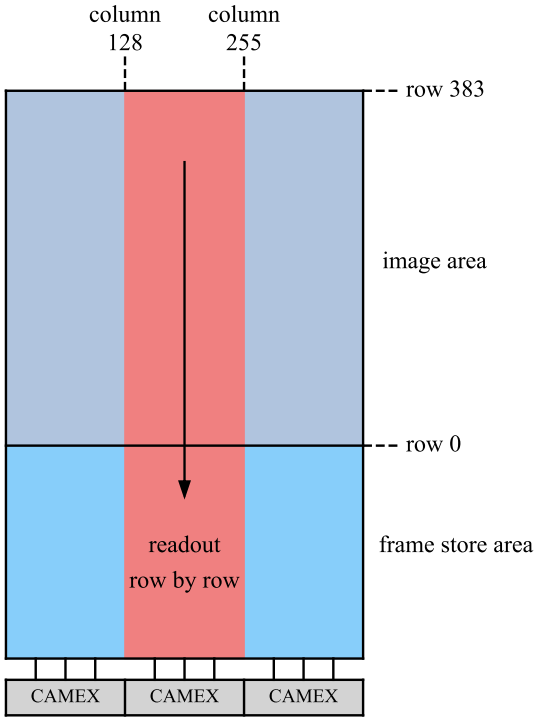

Figure 2.2: The readout region of timing mode

When a charge packet is transferred from one pixel to another, a portion of it is lost. The proportion of charge loss depends on various factors, one of which is the intensity of the incident photon flux——especially under high flux conditions. Therefore, in timing mode, as the source flux gradually increases, the detector's energy-channel relationship will undergo some degree of change. Currently, we recommend performing spectral analysis when the flux is below 100 mCrab. Conducting spectral analysis at fluxes above 100 mCrab may introduce systematic errors.

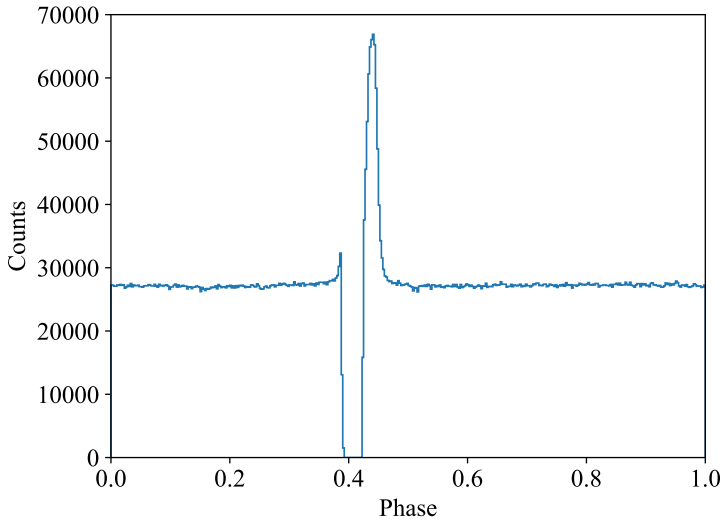

Figure 2.3: The periodic signals caused by the readout system

Chapter 3

Data Files

The FXT science data can be collected using three readout modes, as discussed in the chapter 2. There are the full frame (FF) mode, partial window (PW) mode and timing (TM) mode. This chapter includes the descriptions of the FXT science data files, the file naming convention and main columns included in an observation.

3.1 Level 1 Data Product

For scientific user, data analysis starts from Level 1 data product. The root directory of every product is named by the Observation ID and is orginized as fllowed:

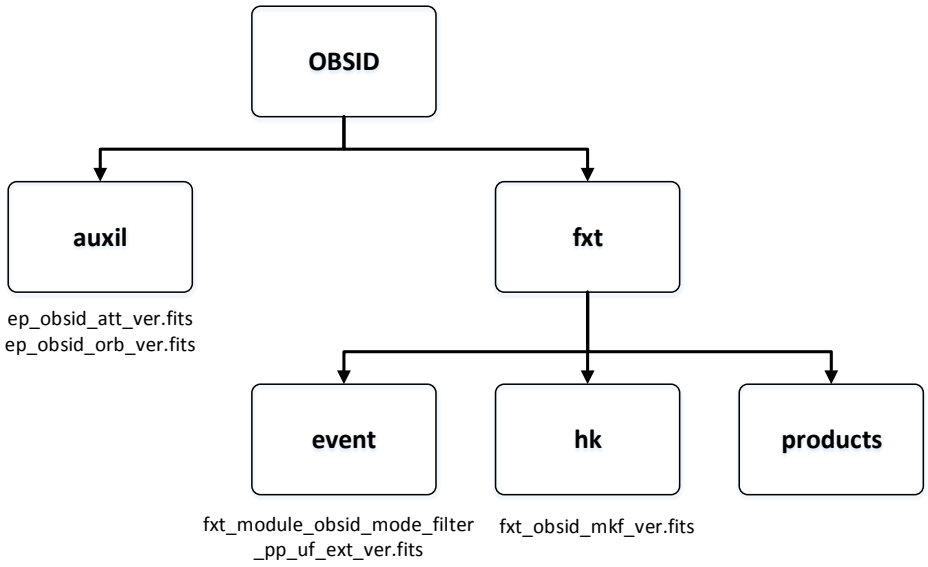

Figure 3.1: FXT observation ID and the data files.(Figure taken from Zhao et al. 2025)

The 'auxil' directory stores attitude and orbit data. While the 'fxt' directory includes the 'event' and 'hk' directory. 'event' contains the event files obtained by the three science modes and 'hk' stores the make filter file (MKF) of fxt data.

3.2 FXT file naming convention

The file name format for the FXT science files uses the following convention:
fxt_[module]_[obs_id]_[mode]_[filter]_[pp]_[uf]_[ext]_[lev].fits.
The filename contains several keywords:

- [fxt] is a prefix to indicate the mission name.
- [module] indicates the instrument (a/b) of FXT.
- [obs_id] contains an 11 digits number to identify the observation ID.
- [mode] indicates the scientific mode to obtain the data (ff/pw/tm).
- [filter] gives the thickness of the filter of FXT with values 01/02/03, indicating thin/medium/hole
- [pp] identifies if the event data were taken with the satellite in pointing mode (po) or during a slew (sl).
- [uf] indicates the file is an unfiltered file.
- [ext] indicates the file type and is typically set to 'evt'. However, it can sometimes be set to 'fsaevt', which means events originating from the frame-store area.
- [lev] gives the file version.

The file version definition rule is as follows: (1) represented by three characters; (2) the first character indicates data integrity (2, uncompleted; 3, completed; 6, merged); the second character indicates the data version (the number of times the data is produced under a certain software version); the third character represents the software version.

3.3 Main columns in FXT FITS events files

This section describes the important columns found in the event files.

The RAWX and RAWY columns give the discrete CCD pixel location of each event processed by the on-board electronics. The data processing using the ground calibrations, produces the DETX and DETY focal plane coordinates and X and Y sky coordinates. For the FF and PW modes, the RAWX and RAWY correspond

to spatial information and RAWY is along the readout direction. In the TM mode, the RAWY is a counter incremented by one when a row is read out.

The TIME column contains the time assigned to each event and it is given in seconds after the reference time.

CCDFrame contains the frame number.

The Channel range is from 0 to 4095. And the PI (pulse invariant) column is derived by gain-correcting the Channel values.

The GRADE describes the grade of the event and these values are calculated by "fxtgrade" tool. Figure 3.2 shows the definition of the grades of FXT.

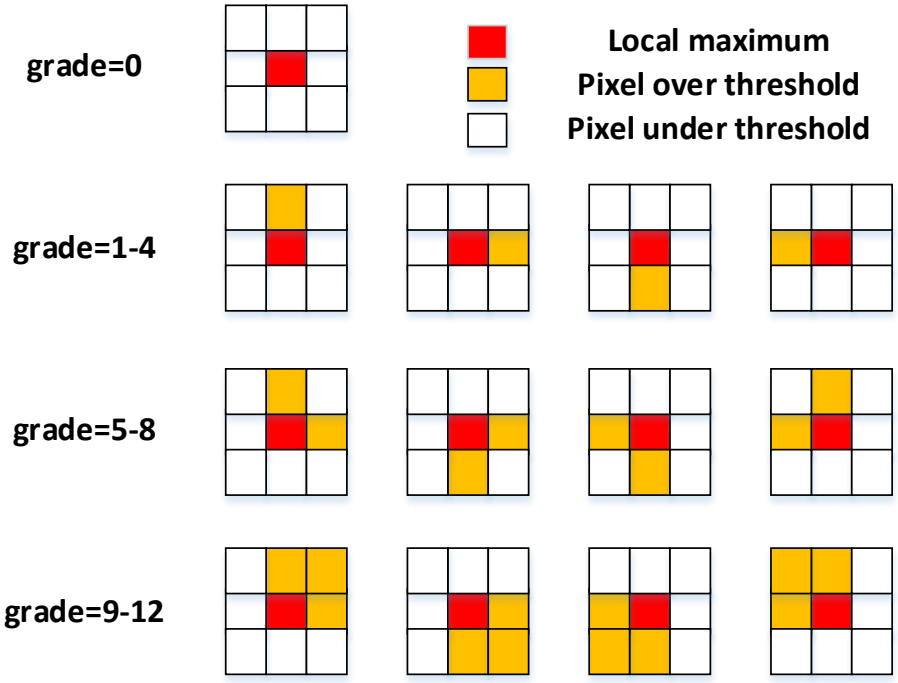

Figure 3.2: The definition of the FXT grades. (Figure taken from Zhao et al. 2025)

The STATUS column contains a bit mask flag describing the quality of the event. The column is populated during the data calibration in the "fxtgrade" step. The possible flags are:

- GOODSTATUS 0x00: good event
- GROUNDSTATUS 0x01: bad pixel from on-ground CALDB bad pixel file
- BOARDSTATUS 0x02: disabled pixel from on-board software
- USERSTATUS 0x04: bad pixel in the file provided by the user

- HOTSTATUS 0x08: pixel hot found in the current observation
- FLICKSTATUS 0x10: pixel flickering found in the current observation
- NEIGHBORSTATUS 0x20: neighbor close to a bad pixel 3*3 reconstructed
- NNEIGHBORSTATUS 0x40: neighbor close to a bad pixel 5*5 reconstructed

Chapter 4

Analysis Guide

4.1 Data reduction

The FXT database provides users with Level 1 data products as described in Chapter 3. It is always the first step to calibrate and filter the Level 1 events file to produce the Level 2 events file in the beginning of the analysis. Then scientific products like images, spectra, and light curves can be extracted from the calibrated data at any time. This step only needs to be done once especially when there is no update of the CALDB.

The data files typically have long file names, as described in the previous chapter. To simplify operations, we create symbolic links to shorten the file names in this example[1]. Below is an example command sequence (assuming the files are located in the `fxt/event` folder):

```
ln -s fxt_a*uf* fxta-ori.fits
ln -s fxt_b*uf* fxtb-ori.fits
cd ../hk
ln -s *hk* hk.fits
ln -s *mkf* mkf.fits
cd ../../auxil
ln -s *att* att.fits
ln -s *orb* orb.fits
cd ../fxt/event
```

And here shows the whole thread to produce a Level 2 event file using FXTA as an example:

- *fxtcoord* updates the SKY coordinate of each event using the latest calibration file and recalculates the GTI of events based on the event file and attitude file

[1]Creating symbolic links is optional and is adopted here for demonstration purposes. Users can also directly use the original file names if preferred.

```
fxtcoord evtfile=fxta-ori.fits attfile=../../auxil/att.fits outfile=fxta-coord.fits
```

- For TM mode, the event time should be corrected based on their `Y` values. Observations in FF or PW mode can skip this step. [ra] and [dec] are coordinates of the target in units of degree.

  ```
  fxttimecorr ra=[ra] dec=[dec] evtfile=fxta-coord.fits
  ```

- *fxtpical* to apply the energy calibration:

  ```
  fxtpical evtfile=fxta-coord.fits outfile=fxta-pi.fits
  ```

- *fxtparticleidentify* to identify fake events from cosmic rays:

  ```
  fxtparticleidentify evtfile=fxta-pi.fits outfile=fxta-rp.fits xlength=11 ylength=11
  ```

  It should be noted that for FF and PW mode, `xlength` and `ylength` equal 11 and for TM mode, this value is 35 (Cosmic rays cross more rows in TM mode).

- *fxtbadpix* and *fxthotpix* search bad and hot pixels, respectively.

  ```
  fxtbadpix evtfile=fxta-rp.fits outfile=fxta-badpix.fits
  fxthotpix evtfile=fxta-badpix.fits outfile=fxta-hotpix.fits
  ```

- *fxtgrade* calculates the event grades:

  ```
  fxtgrade evtfile=fxta-hotpix.fits outfile=fxta-grade.fits
  ```

- *fxtgtigen* produces the Good Time Interval (GTI) file:

  ```
  fxtgtigen mkffile=../hk/mkf.fits module=fxta outfile=fxta.gti expr=DEFAULT
  ```

  The `expr=DEFAULT` means that the selection criterion for the GTIs is `ELV>5 && DYE_ELV>30 && SAA==0 && STABILITY==3 && ANG_DIST<0.2`.

  Last, the events file can be filtered with selected grades, status, and GTIs with `xselect`:

  ```
  xselect
  > Enter session name >[xsl]
  xsl:SUZAKU > read event fxta-grade.fits
  > Enter the Event file dir >[./]
  > Reset the mission ? >[yes]
  xsl:EP-FXTA-FF > filter grade 0-12
  xsl:EP-FXTA-FF > select event "status==b0"
  ```

```
xsl:EP-FXTA-FF > filter time file fxta.gti
xsl:EP-FXTA-FF > extract events copyall=yes
xsl:EP-FXTA-FF > save events fxta-clean.fits
xsl:EP-FXTA-FF > exit
> Save this session? >[no]
```

The "copyall=yes" is needed to contain all the extension of the events file. This equals `HEASoft` command below:

```
ftselect "fxta-grade.fits[events]" fxta-clean.fits "grade >= 0 && grade <=12 \
    && status==b0 && gtifilter('fxta.gti')" copyall=yes clobber=yes
```

Images, spectra, and light curves can be extracted from this filtered events file and intermediate products can be deleted.

These steps can also be replaced by a single batch command *fxtchain*:

```
fxtchain indir=[Input directory] outdir=[Output directory]
```

GTI Checking

In order to avoid sudden changes in the background noise level, it is still necessary to check the light curve over the entire field. The light curve of the field can be obtained by the following operation:

```
xselect
> Enter session name >[EP] EP  # session name, will be the prefix of later output
> set datadir .
> read events fxta-clean.fits
> set datamode FF/PW/TM   # when the mode are not recognized correctly.
> set binsize 20
> show status
> extract curve
> plot curve
PGPLOT file/type: /xw        # set the display device as X window
PLT> device lc.eps/CPS       # switch the output device to a color PS file: lc.eps
PLT> plot
PLT> quit
> save curve lc.fits clobber=yes    # save curve data with a fits file
> quit
```

More details could be found at the "Timing Analysis" section of the "ROSAT Xselect Guide"[2]. If there is any flares detected in the light curve, it must be removed before further analysis.

[2]https://heasarc.gsfc.nasa.gov/docs/rosat/ros_xselect_guide_v1.1/node5.html

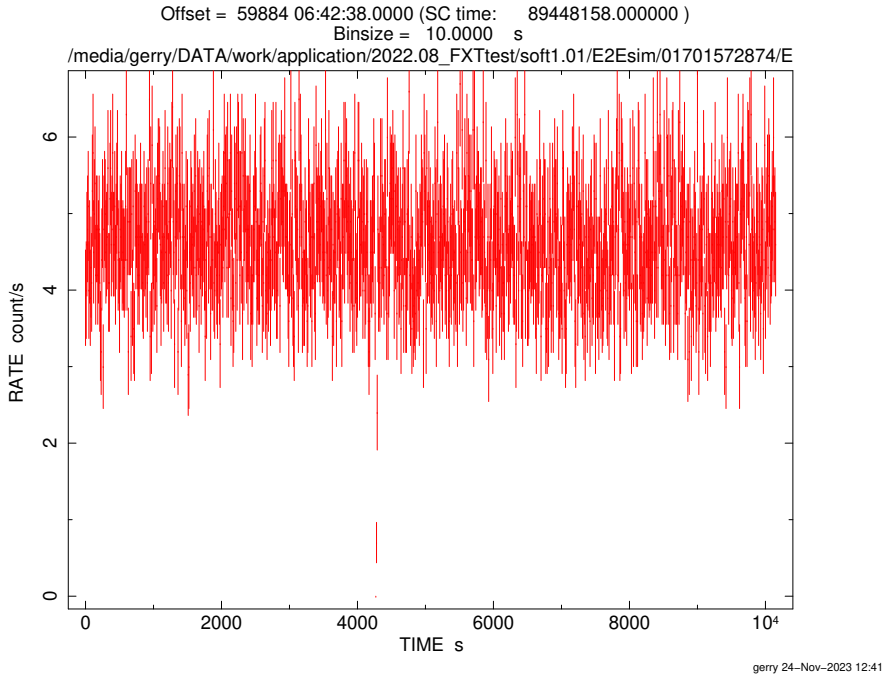

Figure 4.1: light curve of the field

4.2 Point source

After the initial calibration and the screening, the light curve and the spectrum can be extract from the cleaned event file (fxta-clean.fits) with the mission-generic tool — XSELECT. The output scientific products can be analyzed with the most commonly used XANADU software — XRONOS, XIMAGE and XSPEC. The following example illustrates a simple analysis session which aims to extract an image, spectrum and light curve of a point source within the field of view of FXT.

There are 5 main steps to create high-level products:

1. Extract images, spectra, and light-curves.
2. Pileup estimation.
3. Generate the exposure map.
4. Create RMF and ARF files for this source.
5. Barycentric Correction.

4.2.1 FF & PW modes

For the imaging modes, i.e. the FF and PW modes, the initial data processing procedures are similar. In such modes, photons are not only registered during the

actual integration interval but also during the readout of the CCD (shift of charges along a column toward the readout node). These so called Out-of-Time (OoT) events get a wrong RAWY value assigned and thus finally a wrong energy correction (the correction for CTI depends on the distance from the readout node). The fraction of OoT events are depended on the observing mode, and the correction for OoT events is described in Section 4.3.

- `Extracting light curve & spectrum`

  XSELECT allows the user to set the different type of filters (e.g. energy, region, time) using command 'filter', and remove a filter with the command 'clear'. Then, user can create the high-level products with the command 'extract', and write them to files with command 'save'.

  Energy grade. The event files are screened for either the recommended grades (0–12) or other specific grade settings.

  Region filtering. An image is extracted from event file read in XSELECT using the `'extract image'` command. Plot the image with the `'plot image'` command which invokes DS9. Both the source and the background region files are created with DS9. The source photons are extracted from a circular shape and its size is depended on the brightness of the source. Typically, the 90% of the PSF at 1.5 keV is enclosed by a $\sim$ 1 arcmin radius circle (suggesting the radius of region r >= 1 arcmin). The region filter in XSELECT is processed with the command `'filter region source.reg'`. Alternatively, the background photons should be extracted from a source-free region away from the strip associated with the OoT events.

  Energy filtering. Light curves or images for specific energy bands can be extracted by setting an energy filter. The FXT event files do not have an energy column, but a PHA and PI channel is assigned for every photon. The energy-to-channel conversion can be done with the task `fxte2pi`. The current tool can accept more than one energy value in a single command call, either as space-separated positional arguments or as a comma-separated `energy=` list. For instance, the channels for 2 and 10 keV can be calculated as follows:

```
$fxte2pi 2.0 10.0
$-PI value for fxt at 2.0 keV: 274
$-PI value for fxt at 10.0 keV: 925
$fxte2pi energy=2.0,10.0
$-PI value for fxt at 2.0 keV: 274
$-PI value for fxt at 10.0 keV: 925
```

  Therefore, the energy selection can be achieved by the command, e.g. `'filter pha_cutoff 274 925'`, which filters all data within the channel range 274–925,

corresponding to 2–10 keV. **Caution: do not apply the PHA filter in the spectrum extraction.**

Time filtering. There are three ways to set the time filter in XSELECT: 1. Specifying the time interval with the mouse cursor, using the command '`filter time cursor`'; 2. Entering the start and the stop times at the keyboards, using command '`filter time UT`' or '`filter time SCC`' or '`filter time MJD`'; 3. Selecting the time intervals with a GTI file containing a series of desired start and stop times, using command '`filter time file filename`'.

The default time resolution of light curve is 1 second. The net source light curve can be created with the task `lcmath`, while the tools for a more detailed correction (e.g. exposure-correction) will be provided in the future.

The image, spectrum and light curve of the field can be obtained by the following operation:

```
$xselect
> Enter session name >[xsl]
xsl:SUZAKU > read event fxta-clean.fits
> Enter the Event file dir >[./]
> Reset the mission ? >[yes]
xsl:EP-FXTA-FF > extract all
xsl:EP-FXTA-FF > plot image
xsl:EP-FXTA-FF > filter region source.reg
xsl:EP-FXTA-FF > extract spectrum
xsl:EP-FXTA-FF > save spectrum source.pha
xsl:EP-FXTA-FF > clear region
xsl:EP-FXTA-FF > filter region back.reg
xsl:EP-FXTA-FF > extract spectrum
xsl:EP-FXTA-FF > save spectrum back.pha
xsl:EP-FXTA-FF > clear region
xsl:EP-FXTA-FF > filter region source.reg
xsl:EP-FXTA-FF > filter pha_cutoff 6 551
xsl:EP-FXTA-FF > extract curve
xsl:EP-FXTA-FF > save curve source.lc
xsl:EP-FXTA-FF > clear region
xsl:EP-FXTA-FF > filter region back.reg
xsl:EP-FXTA-FF > extract curve
xsl:EP-FXTA-FF > save curve back.lc
xsl:EP-FXTA-FF > exit
> Save this session? >[no]
```

In some cases, count rates erratically and sharpely increase during the start or the end of GTI due to the Earth contamination. Users should manually subtract these time intervals themselves.

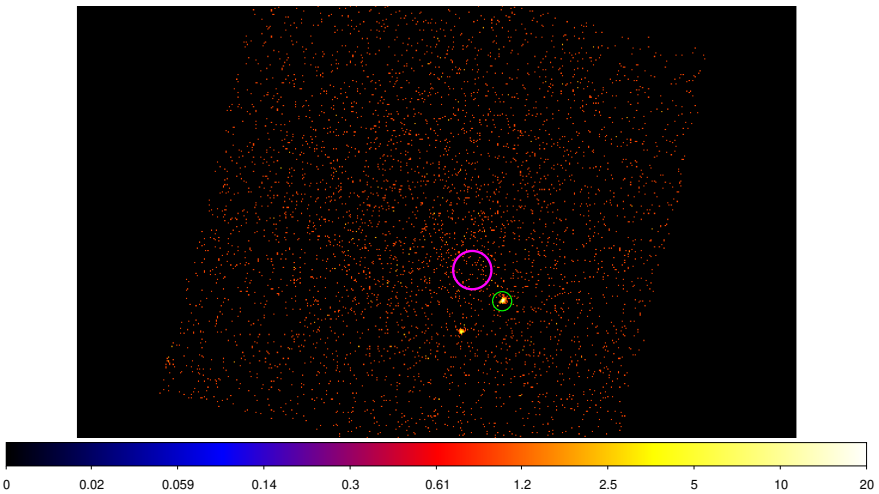

Figure 4.2: Image plot with the source (green circle) and background (cyan circle) region selected for the FF mode observation.

- `Pile-up estimation`

  Point sources with a count rate above the critical value produce severe pile-up in the central region of the Point Spread Function (PSF). In this case, there is a significant probability that two or more soft X-ray photons are registered as a single higher-energy photon (i.e. energy pile-up) of higher patter type (i.e. pattern pile-up). A quick way to estimate the pile-up effect can be done with the online simulation tools (`Filter & Window Mode Evaluation`[3]). There are another two methods to assess whether an observation is affected by pile-up based on the pattern migration and photon loss, respectively. The former method is similar to those used in *XMM-Newton* EPIC data analysis [4]), and the latter is similar to those used in *Swift*/XRT data analysis.

  1. Apply the task `fxtplotgrade` on the filtered source event file, source.evt, to produce a PNG file displaying the observed versus expected pattern distribution. In the figure, the spectral distributions as function of PI channel for different grades are plotted. Figure 4.4 shows an example of observation affected by pile-up. In order to reduce pile-up, the filtered event file is extracted from an annular region. As can be seen in Figure 4.5, the pile-up effect becomes negligible after excising the core of the PSF.

  2. The PSF file, fxt_psf.dat, and the model file, fxt_psf.cod are copied from the `CALDB` directory to current directory by running the task of `fxtpsf`. Then, the subsequent analysis are almost same as the processing steps in Swift/XRT data analysis[5]).

[3]http://epfxt.ihep.ac.cn/simulation
[4]https://www.cosmos.esa.int/web/xmm-newton/sas-thread-epatplot
[5]https://www.swift.ac.uk/analysis/xrt/pileup.php

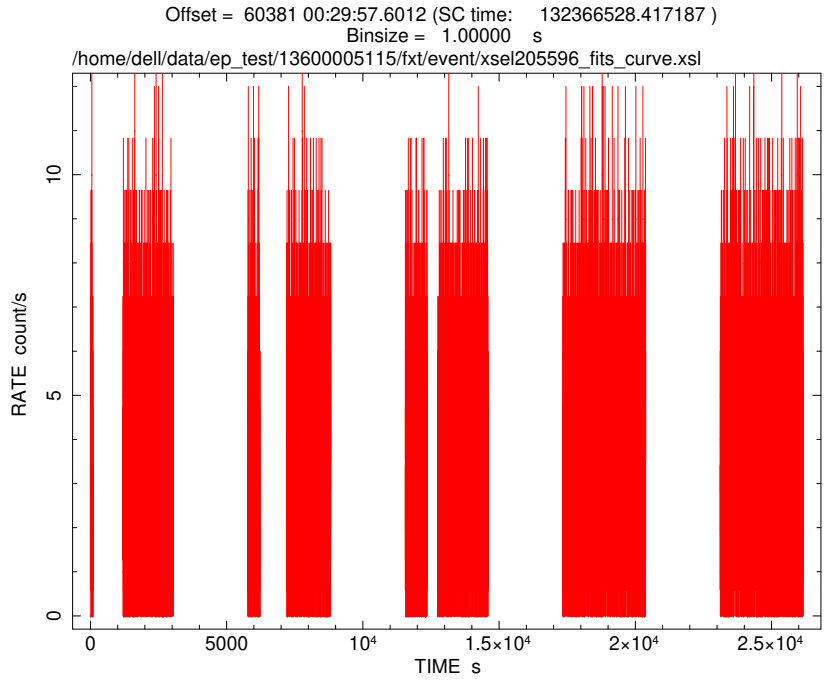

Figure 4.3: Light curve extracted from the source region

```
$ximage
> CHMDB/ADD/TEL=EP/INST=FXT/DETNAM=FXTA/KEY=psffile/VALUE=fxt_psf.dat
> read fxta-clean.fits                          % read the event/image file
> cpd /xtk
> disp                                          % plot image
> back                                          % estimate background level
> psf/cur
> col off 1 2 3 4 6
> rescale x 30
> model fxt_psf.cod                             % load the PSF model
> fit                                           % fit with the PSF model
> rescale
> p
```

The software XIMAGE is called to read and plot the filtered event or image file generated by XSELECT. The mean background level (counts per pixel) over the entire image is estimated with the task `back`. The user should zoom into the source of interest, run the command `psf/usr`, click in the centre of the source and where the PSF fades into the background. This will open an interactive plot window, with panels showing both the Encircled Energy Function and the PSF. Then, running the command `col off 1 2 3 4 6` to turn off all datasets except the cyan points, which are further fitted by the

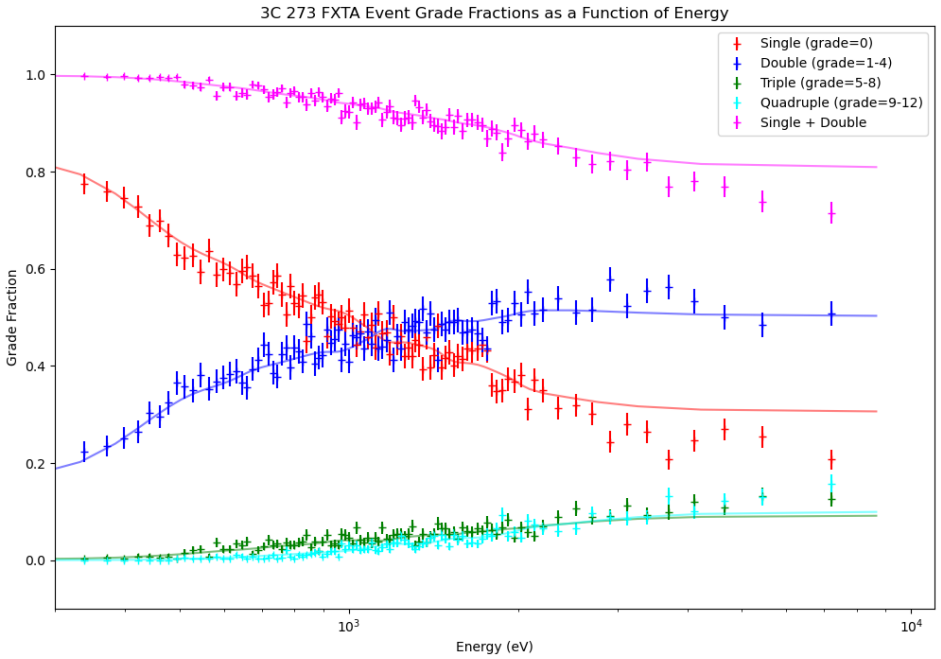

Figure 4.4: Plot of pattern distribution by the task `fxtplotgrade`, which in this case indicates the presence of pile-up.

PSF model. Because the pile-up mostly affects the centre the PSF, the PSF model should be fitted to the outer wings and then extrapolated to the inner region to compare with the data points, determining the point at which the data and model diverge (Figure 4.6). In this way, the user can estimate the size of bright core which is significantly affected by pile up and should be excluded.

Currently, these methods are applicable to the imaging mode (FF and PW) data analysis only. The loss of counts caused by using the annulus can be estimated with the PSF correction. It can be accomplished by setting the parameter `'psfcor=1'` in the task `fxtarfgen` when generating the Ancillary Response File.

- `fxtexpogen`

Generates the exposure map, which is used to correct for the loss of flux caused by some of the CCD pixels not being used to collect data. This task requires the event file and the `mkf` filter file, which can be found in the `auxil` directory.

This command generates two exposure maps: one without vignetting correction and one with vignetting correction applied. The exposure map without vi-

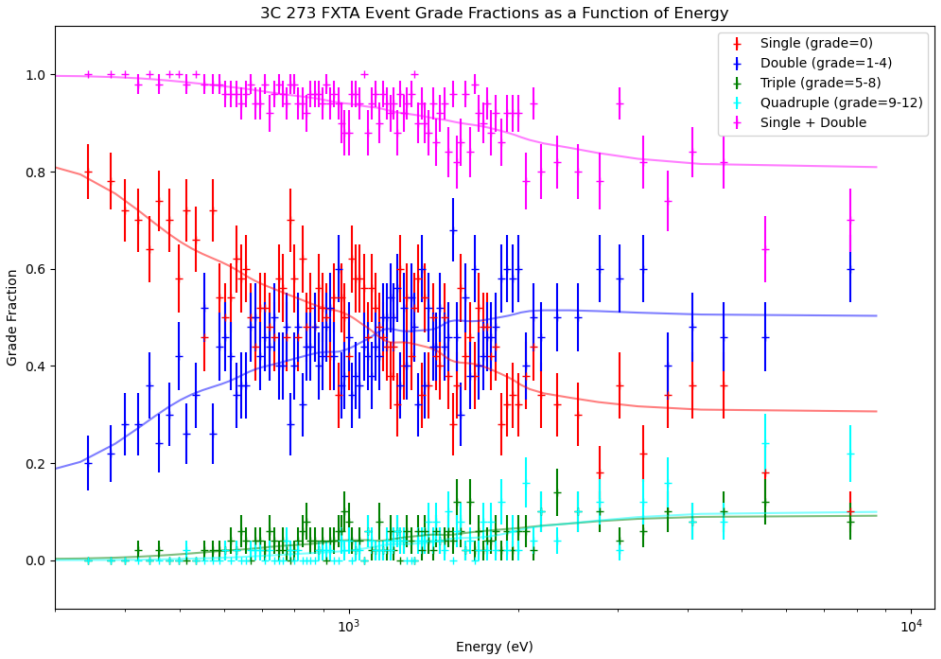

Figure 4.5: Plot of pattern distribution by the task `fxtplotgrade`, which in this case indicates a negligible pile-up fraction after excluding the bright core.

gnetting correction can be identified by the addition of the field `-without_vign` in the output file name. In v1.30, `fxtexpogen` also documents optional effective-area scaling (`area_scale=yes`) and clustering parameters (`use_clustering`, `ra_threshold`, and `pa_threshold`) that can accelerate long calculations while preserving the exposure-map geometry.

Example:

```
$fxtexpogen mkffile=../hk/mkf.fits evtfile=fxta-clean.fits

outfile=fxta-expo.fits
```

- `fxtpsfmap`

This new v1.30 task generates a PSF-size map directly from an exposure map for a given energy and enclosed energy fraction (EEF). The output image has the same geometry as the exposure map and records the PSF radius in arc-seconds at each position. If the filter is not supplied explicitly, the task can

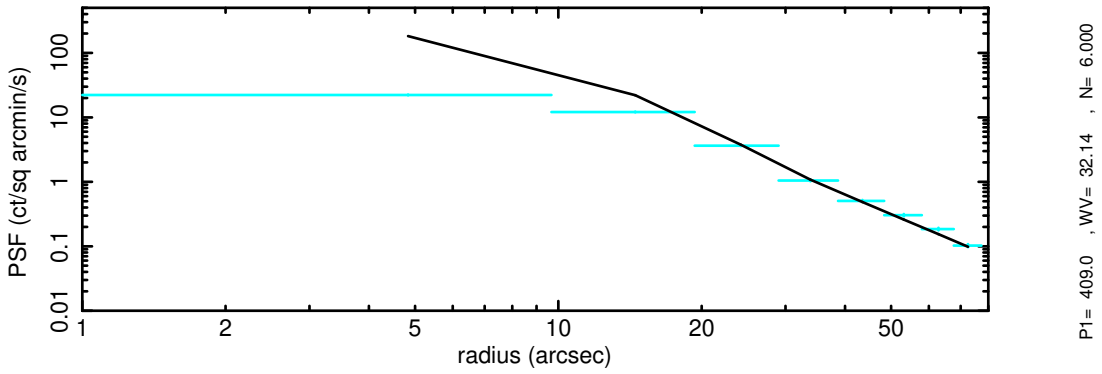

Figure 4.6: The PSF plot. The model and data diverge below about 20 arcsec.

read it from the FITS header.

Example:

```
$fxtpsfmap expmap=fxta-expo.fits outfile=fxta-psfmap.fits energy=2.0
eef=0.5
```

- `fxtarfgen`

  This command creates the Ancillary Response File (ARF) for the extracted spectrum, taking into account the exposure map generated by the task `fxtexpogen` and the Point Spread Function (PSF) correction. In the current release, `fxtarfgen` applies a dual-beta PSF model for point sources within about 3 arcmin of the optical axis when `psfcor=1`; at larger off-axis angles it falls back to interpolated EEF calibration curves. By default, the parameter `psfcor` is set to 0, which means that no PSF correction is applied. PSF correction is only necessary when the source exhibits pileup and the selected region is an annulus (to exclude the piled-up central area). In such cases, the `psfcor` parameter should be set to 1 to ensure accurate ARF generation. When there is no pileup, it is recommended to use a circular region with a radius greater than 1 arcminute to capture as many source photons as possible while minimizing the loss of flux. The input exposure map should normally be the file with suffix `-without_vign`.

  Example:

```
$fxtarfgen specfile=source.pha expfile=fxta-expo-without_vign.fits
     psfcor=0 outfile=source.arf
```

- `fxtrmfgen`

  creates the Redistribution Matrix Files (RMF) for a given spectrum.

  Example:

  ```
  $fxtrmfgen specfile=source.pha outfile=source.rmf
  ```

  In the final step, the source spectrum, the background spectrum, the RMF and the ARF files are passed through `grppha` to make it suitable for spectral fitting in XSPEC. In particular, the spectrum is usually grouped (e.g. to have at least 10 photons per bin) to enable the use of χ^2 statistics.

  ```
  $ grppha infile=source.pha outfile=source.pi
  comm=' group min 10 & chkey RESPFILE source.rmf & chkey ANCRFILE source.arf
  & chkey BACKFILE back.pha & exit' clobber=yes
  ```

  ```
  $xspec
  >cpd /xw
  >data source.pi
  >setplot energy
  >ignore bad
  >ignore 0.0-0.5
  >ignore 10.0-**
  >show rate
  >plot ldata
  >model tbabs*powerlaw
  >fit 100
  >plot ldata del
  >flux 0.5 10.0
  >quit
  ```

4.2.2 TM mode

The TM mode is designed to observed the bright sources and for high time resolution. The following steps are run in sequence to analyze the cleaned event file (fxta-clean.fits):

- `Extracting light curve & spectrum`

  XSELECT allows the user to set the different type of filters (e.g. energy, region, time) using command 'filter', and remove a filter with the command 'clear'. Then, user can create the high-level products with the command 'extract', and write them to files with command 'save'.

Energy grade. The event files are screened for either the recommended grades (0–12) or other specific grade settings.

Region filtering. An image is extracted from event file read in XSELECT using the `'extract image'` command. Plot the image with the `'plot image'` command which invokes DS9. For observations made in the TM mode, only one dimensional imaging is preserved. Therefore, a rectangular box of ~ 3 arcmin long and ~ 1 arcmin wide can be adopted to extract the source photons, and a source-free rectangular region of the same width is adopted for the background. The roll angle of observation can be found in the header of the event file ('PA_PNT', or 270°-'PA_PNT'). Caution: for the TM mode observations, do not use a circle/annulus region to extract photons. Both the source and the background region files are created with DS9. The region filter in XSELECT is processed with the command `'filter region source.reg'`. The default time resolution of light curve is 1 second, while higher time resolution is also available. The net source light curve can be created with the task `lcmath`, while the tools for a more detailed correction (e.g. exposure-correction) will be provided in the future.

Energy filtering. Light curves or images for specific energy bands can be extracted by setting an energy filter. The FXT event files do not have an energy column, but a PHA and PI channel is assigned for every photon. The energy-to-channel conversion can be done with the task `fxte2pi`. The current tool can accept more than one energy value in a single command call, either as space-separated positional arguments or as a comma-separated `energy=` list. For instance, the channels for 2 and 10 keV can be calculated as follows:

```
$fxte2pi 2.0 10.0
$-PI value for fxt at 2.0 keV: 274
$-PI value for fxt at 10.0 keV: 925
$fxte2pi energy=2.0,10.0
$-PI value for fxt at 2.0 keV: 274
$-PI value for fxt at 10.0 keV: 925
```

Therefore, the energy selection can be achieved by the command, e.g. `'filter pha_cutoff 274 925'`, which filters all data within the channel range 274–925, corresponding to 2–10 keV. Caution: do not apply the PHA filter in the spectrum extraction.

Time filtering. There are three ways to set the time filter in XSELECT : 1. Specifying the time interval with the mouse cursor, using the command `'filter time cursor'`; 2. Entering the start and the stop times at the keyboards, using command `'filter time UT'` or `'filter time SCC'` or `'filter time MJD'`; 3. Selecting the time intervals with a GTI file containing

a series of desired start and stop times, using command ‘`filter time file filename`’.

The image, spectrum, light curve and source event of the field can be obtained by the following operation:

```
$xselect
> Enter session name >[xsl]
xsl:SUZAKU > read event fxta-clean.fits
> Enter the Event file dir >[./]
> Reset the mission ? >[yes]
xsl:EP-FXTA-FF > extract all
xsl:EP-FXTA-FF > plot image
xsl:EP-FXTA-FF > filter region source.reg
xsl:EP-FXTA-FF > extract spectrum
xsl:EP-FXTA-FF > save spectrum source.pha
xsl:EP-FXTA-FF > clear region
xsl:EP-FXTA-FF > filter region back.reg
xsl:EP-FXTA-FF > extract spectrum
xsl:EP-FXTA-FF > save spectrum back.pha
xsl:EP-FXTA-FF > clear region
xsl:EP-FXTA-FF > filter region source.reg
xsl:EP-FXTA-FF > filter pha_cutoff 75 338
xsl:EP-FXTA-FF > extract curve
xsl:EP-FXTA-FF > save curve source.lc
xsl:EP-FXTA-FF > clear region
xsl:EP-FXTA-FF > filter region back.reg
xsl:EP-FXTA-FF > extract curve
xsl:EP-FXTA-FF > save curve back.lc
xsl:EP-FXTA-FF > exit
> Save this session? >[no]
```

- `fxtexpogen`

  generates the exposure map, which is used to correct for the loss of flux caused by some of the CCD pixels not being used to collect data. This task requires the event file and the `mkf` filter file, which can be found in the `auxil` directory. Example:

  ```
  $fxtexpogen mkffile=../hk/mkf.fits evtfile=fxta-clean.fits
  outfile=fxta-expo.fits
  ```

- `fxtarfgen`

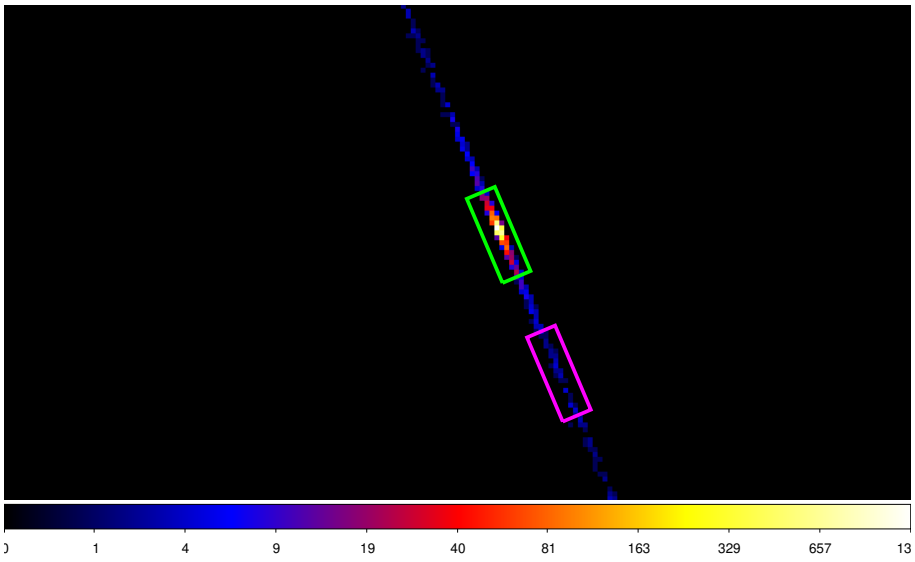

Figure 4.7: Image plot with the source (green rectangular) and background (cyan rectangular) region selected for the TM mode observation.

creates the Ancillary Response File (ARF) for the extracted spectrum, taking into account the exposure map generated by the task `fxtexpogen`.
Example:

```
$fxtarfgen specfile=source.pha expfile=fxta-expo-without_vign.fits \
    outfile=source.arf psfcor=0
```

Currently, the pile-up/PSF correction is unavailable for the TM data, so here do not set the parameter 'psfcor' to 1.

- `fxtrmfgen`

  creates the Redistribution Matrix Files (RMF) for a given spectrum.
  Example:

  ```
  $fxtrmfgen specfile=source.pha outfile=source.rmf
  ```

  In the final step, the source spectrum, the background spectrum, the RMF and the ARF files are passed through `grppha` to make it suitable for spectral fitting in XSPEC. In particular, the spectrum is usually grouped (e.g. to have at least 10 photons per bin) to enable the use of χ^2 statistics.

- `fxtbary`

  applies barycenter correction to FXT event data files before a precise timing analysis.
  Example:

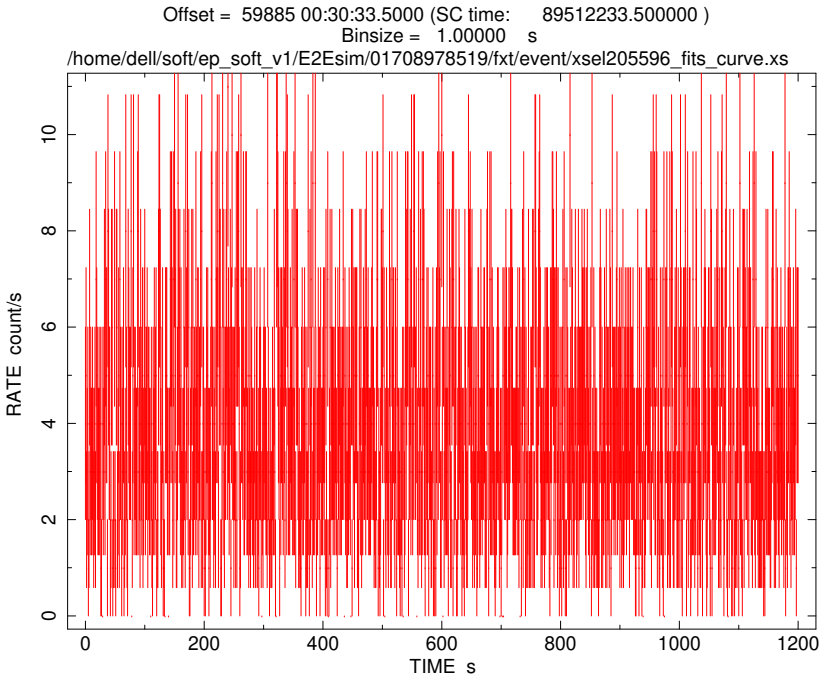

Figure 4.8: light curve of the field

```
$ fxtbary evtfile=fxta-source.evt outfile=fxta-source-bary.evt
orbitfile=../../auxil/orb.fits
ephemfile=/fxtsoftv1.10/fxt/tasks/fxtbary/de421.bsp
ra=342.1721004 dec=-51.1657098
```

4.3 Extended Sources Imaging

With a large field of view of 1° and a low instrument background, EP/FXT is ideal for observing diffuse objects with a large angular size including supernova remnants, superbubbles, diffuse Galactic plasma, nearby galaxies, and nearby galaxy clusters. Compared to point-like sources, extended sources commonly have a low surface brightness and may even cross several pointings. Therefore, imaging analysis on diffuse emissions is far more than image extraction from the events file. Various corrections should be applied before the image can be used for science. This manual will introduce the threads of imaging analysis from scratch, including image extraction, pixel masking, vignetting correction, and finally adaptive smoothing or mosaic if needed.

The FXT data analysis software `FXTDAS` provides little scripts for general image processing, but scripts provided by `HEASoft` can be used along with `FXTDAS`. Other packages with similar functions, like `CIAO`[6], `IRAF`[7], and `AstroPy`[8], can also help as well.

4.3.1 Image extraction

Different from optical data, the origin X-ray data is named "event file", a data list recording information of every photon detected, which includes the incident time, energy (column `channel`, `PI`, and `ENG` for FXT), location, and other labels like grades. Once filters like energy and grades are selected, make a 2-D histogram in space and an image can be obtained. For FXT data, the location of a photon can be defined in 3 planes given by 6 columns in total. Column `RAWX` and `RAWY` define the `RAW` plane, where photon electrons are read out. The detector (`DET`) plane, defined by `DETX` and `DETY`, is equal to the `RAW` plane as there is only one CCD for an individual FXT module. In fact, the relation between these two planes can be described as:

$$\texttt{DETX} = \texttt{RAWX} + 1; \texttt{DETY} = \texttt{RAWY} + 1. \tag{4.1}$$

The other one is the `SKY` plane defined by the columns `X` and `Y`, reflecting the location in the equatorial coordinate system. The final products for science analysis should be in this plane in most cases. The size of the `RAW/DET` plane is 384×384 pixels, while for the `SKY` plane, the default size is 600×600 pixels. Each pixel corresponds to an angular size of $\approx 9.66''$. Figure 4.9 gives an example of images in `RAW/DET` and `SKY` coordinates.

For each FXT module (A/B), once the event file has been reprocessed through *fxtchain* or step by step, the event file is ready for image extraction. The column `ENG` is the energy of the events in units of keV and can be used to split different energy band for imaging. (**Caution: column `ENG` cannot be directly used for**

[6]https://cxc.cfa.harvard.edu/ciao4.16/
[7]https://iraf-community.github.io
[8]https://www.astropy.org

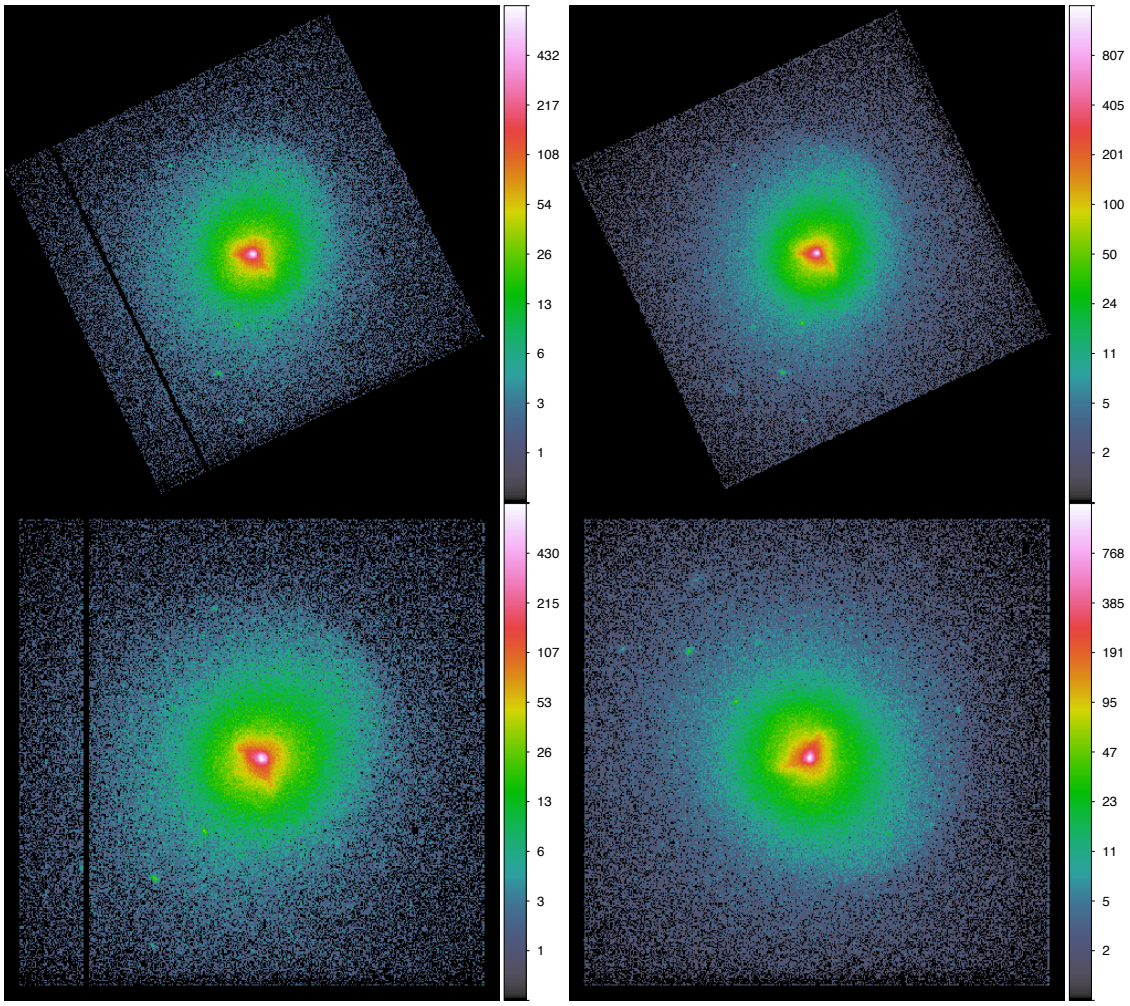

Figure 4.9: EP/FXT first light on M87 of FXTA (*left panels*) and FXTB (*right panels*) in `SKY` (*top panels*) and `RAW/DET` (*bottom panels*) coordinates. The bad pixels have been masked.

spectral analysis! Extract `PI` column instead!) Assuming the filtered event file is named *fxta-clean.fits*, here is an example of extracting an image in the 0.4–7.0 keV band with *ftcopy* of `HEASoft`[9]:

```
ftcopy "fxta-clean.fits[eng=0.4:7.0][bin x=0.5:600.5,y=0.5:600.5]" \
    fxta-0.4-7.0-cts.fits clobber=yes
```

Equivalent scripts include `Xselect` package and *ftcopy* from `FTOOLS`. The extracted image can then be visualized by software like `SAOImage ds9`.

[9]https://heasarc.gsfc.nasa.gov/lheasoft/ftools/ftools_subpacks.html

4.3.2 Exposure map: pixel masking and vignetting correction

An exposure map records the effective exposure time of every single pixel in the field of view, whose generation only depends on the technical characteristics and observation configuration. Similar to the flat images in optical observations, it applies some imaging correction, including the nodding of the pointing, pixel masking, and vignetting correction, but can also help to estimate the intensity of a source in a given energy without spectroscopy at the same time.

Pixel masking

Some pixels or areas in the detector cannot reflect the real incident photons and cannot be correctly analyzed for various reasons, which would lead to image distortion. Part of them may be well corrected based on further calibration in the future but currently, the easiest way is simply masking these pixels. Here, we mainly consider such types of pixels below:

Bad pixels: Bad pixels are those with anomalously high read-out electrons. Currently, there is one bad column in FXTA and several bad pixels in both FXTA and FXTB. The bad pixels can be identified when reprocessing the event file with *fxtbadpix* and their location is written into the event file as a FITS extension. Noticeably, as the photon events are reconstructed based on a 3×3 "event island", the incidence of these pixels actually extends to a radius of two extra pixels. The events from bad pixels would be abandoned in the filtered event file given by the *fxtchain* pipeline.

Detector edge: Due to electronic design, no events can be read out in rows at the edge of the detector (i.e. `RAWX=0` and `RAWX=383`). Meanwhile, in the event reconstruction, the rate of events would be overestimated at the edge of the field of view (`RAWX`=1, 382 and `RAWY`=0, 383, see Figure 4.9). For the lack of an entire 3×3 event file, the event grades of nearby two rows or columns are not accurate either. Strictly speaking, these pixels should thus be masked as well.

Shading: There is a light-blocking board at the read-out end of the detector to prevent photons from entering the storage area. However, this also shades some columns nearby as well. The current study found that the effective area begins to drop at `RAWY`$\approx$20, and drops to almost zero at `RAWY`=1 for both FXT modules (see bottom panels of Figure 4.9 as an example). Correcting this shading requires future calibration study and currently masking these pixels is the easiest way to handle them.

Vignetting correction

Due to the X-ray optics, the effective area always decreases with the distance from the optical axis increases and the degree of this effect differs for X-ray photons with

different energy. In general, the center of the count image always looks "brighter" than the edge and therefore the vignetting correction is needed.

In default, the CCD edge and bad pixels are automatically masked and exposure maps with and without vignetting correction (filename with a suffix "-without_vign") are created at the same time. Such an example is Figure 4.10. Applying the parameter *edgecovermask*, pixels with `RAWY`<20 affected by the shading, and two rows/columns on the detector edge will be masked in the exposure map as well. The exposure map can be created by the script *fxtexpogen* with a given effective area:

```
fxtexpogen evtfile=fxta-clean.fits mkffile=../hk/mkf.fits \
    outfile=fxta-0.4-7.0-exp.fits edgecovermask=1
```

In v1.30, the parameter *area_scale* can be used to multiply the on-axis effective area into the exposure map if it is set as "yes". This changes the unit of the exposure map from "s" to "cm^2 s", and also takes the difference in the effective area between FXTA and FXTB into consideration. For long observations, `fxtexpogen` can also speed up the calculation by grouping similar attitudes with `use_clustering=yes`; the parameters *ra_threshold* and *pa_threshold* control the clustering tolerance. Then a threshold should be applied to abandon data with low exposure time. Here is an example to only keep the pixels with an exposure over 15% of the maximum exposure time.

```
exp=‘ftstat fxta-0.4-7.0-exp.fits | grep "maximum" | awk ’{print $3}’‘
locut=‘echo "0.15*$exp" | bc -l‘
fimgtrim fxta-0.4-7.0-exp.fits $locut INDEF 0 INDEF \
    fxta-0.4-7.0-expthresh.fits clobber=yes
farith fxta-0.4-7.0-expthresh.fits fxta-0.4-7.0-expthresh.fits \
    fxta-0.4-7.0-mask.fits DIV clobber=yes
fimgtrim fxta-0.4-7.0-mask.fits 0 INDEF 0 INDEF fxta-0.4-7.0-mask.fits clobber=yes
farith fxta-0.4-7.0-cts.fits fxta-0.4-7.0-mask.fits \
    fxta-0.4-7.0-thresh.fits MUL clobber=yes
```

Dividing the count map by the exposure map produces a vignetting-corrected count rate map in units of count per second:

```
farith fxta-0.4-7.0-thresh.fits fxta-0.4-7.0-expthresh.fits \
    fxta-0.4-7.0-rate.fits DIV clobber=yes
fimgtrim fxta-0.4-7.0-rate.fits 0 INDEF 0 INDEF fxta-0.4-7.0-rate.fits clobber=yes
```

4.3.3 Background Noise

Some events may not have a astrophysical origin (e.g. instrument background) or cannot correctly reflect the source characteristics (e.g. stray light and OoT) but

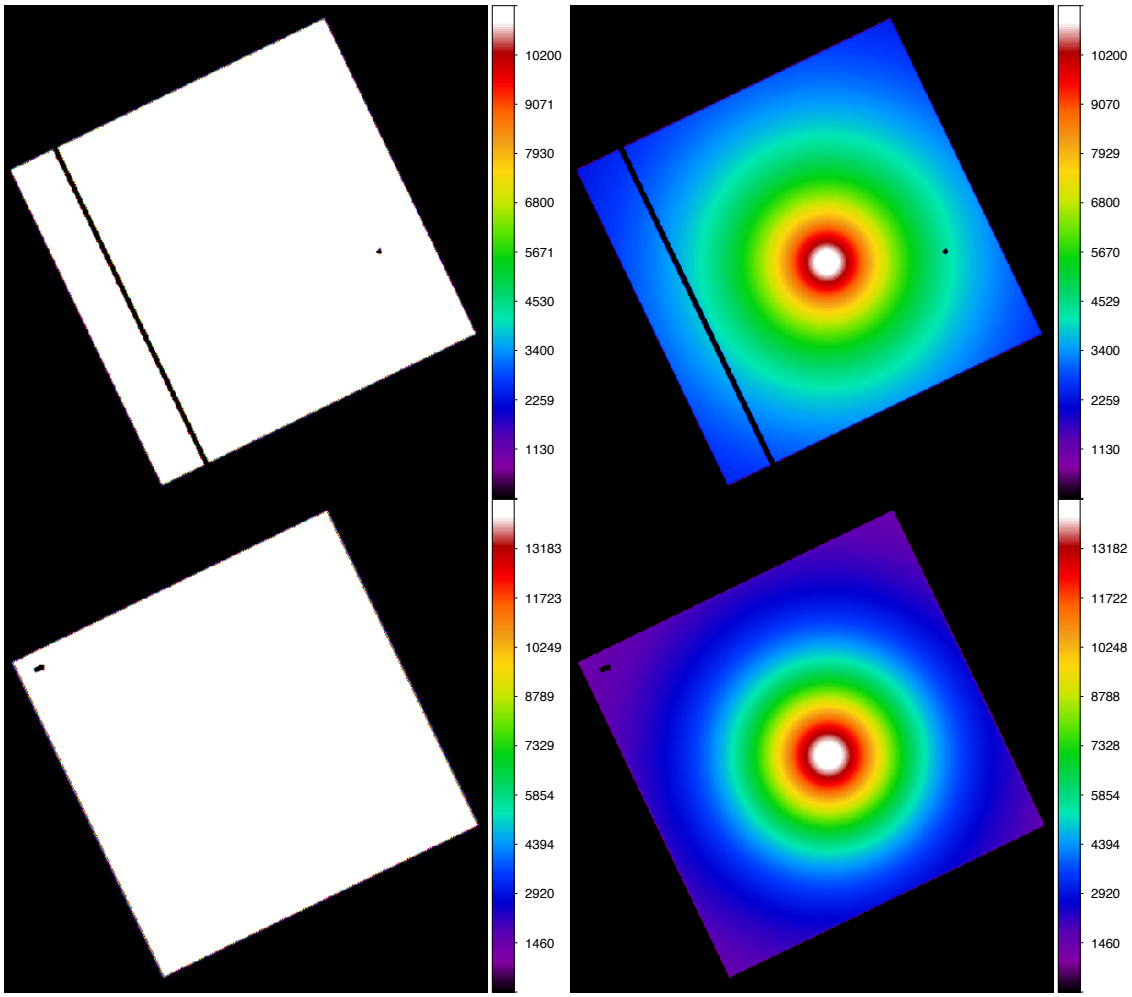

Figure 4.10: FXTA (*top panels*) and FXTB (*bottom panels*) exposure maps with an effective energy of 4 keV. Vignetting effect is considered in *right panels*. It is noticeable that vignetting is different between two modules and depends on the effective area. Bad pixels are masked here and pixels with lower values on the edge of the field of view in *left panels* reflect the nodding of the pointing.

are still recorded. They cannot be simply distinguished like cosmic rays and even difficult to be effectively modeled and subtracted. However, these effect can be relieved or avoided if handled with caution when proposing and analyzing.

OoT events

It takes time for the pn CCD to read out (e.g., ~ 0.1152 ms for Full Frame mode, see Chapter 2 for details). When X-ray photons arrive in the readout time, named out of time events (OoT), incorrect `RAWY` values would be given. Luckily, as the readout time is quite short compared to the frame time, the fraction of OoT is only 0.23% for the Full Frame mode. Therefore, this effect only matters when there is a vary

bright source in the field of view and only contaminates columns the bright source is located. The best way to avoid this scenarios is to check the rolling angle of the observation in the proposal phase, ensuring that the diffuse source is in different columns from the bright source. `FXTDAS` also provides a tool *fxtootset* to model the OoT events for subtraction:

```
fxtootest evtfile=fxta-clean.fits outfile=fxta-oot.fits \
    attfile=../../auxil/att.fits
ftcopy "fxta-oot.fits[eng=0.4:7.0][bin x=0:600,y=0:600]" \
    fxta-0.4-7.0-oot.fits clobber=yes
ftimgcalc fxta-0.4-0.7-suboot.fits 'A-0.0023*B' a=fxta-0.4-7.0-cts.fits \
    b=fxta-0.4-7.0-oot.fits clobber=yes
```

This subtraction should be done before vignetting correction, as OoT events are not affected by vignetting. It should be noted that the OoT event can only be modeled rather than accurately identified. If the extending source is fiercely contaminated by OoT events of a nearby sources, it is not recommended to apply a spectral analysis even the OoT portion of the spectra can be modeled and subtracted, because errors are incorrectly calculated in the subtraction.

Meanwhile, bright sources, especially point-like sources, may suffer from pile-up as discussed in Section 4.2. Therefore, simulated OoT events are based on the observations with pile-up. Differently, OoT events are immediately read out when they arrive at the detector, indicating their pile-up fraction would be much small and they actually reflect the source without pile-up. This would lead to an underestimation of the OoT count rate and thus the correction is not accurate. This is another reason for not recommending spectroscopy on diffuse sources affected by OoT.

Stray light

Commonly, X-ray photons experience two reflections and then are focused on the CCD. However, some photons can also arrive at the detector with only one reflection. Therefore very luminous X-ray sources can affect the imaging even it is located up to several degrees out of the field of view even from sources out of the field of view. Arc- and knot-like patterns named "stray light" would cross the field of view (see Figure 4.12), which should be avoided especially in the spectral analysis. Different from OoT events, the stray light cannot be even modelled accurately for subtraction in the current stage.

Instrument background

The spectrum of FXT instrument background is quite flat, especially above 2 keV, which probably dominates the X-ray emissions. As the instrument background is almost uniformly distributed in the detector, if not subtracted, the vignetting correction will result in an artificially enhanced brightness towards the image edges,

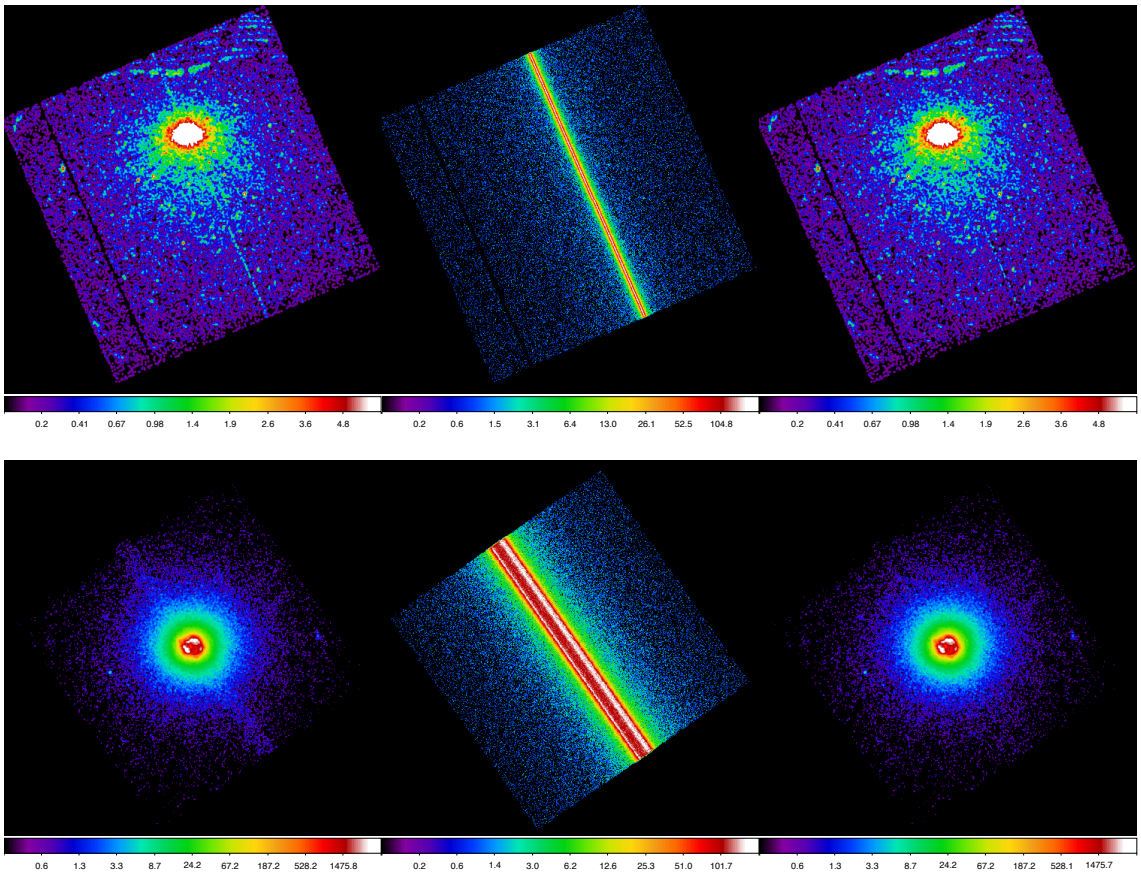

Figure 4.11: From *left* to *right*: raw count maps, simulated OoT images, and OoT-subtracted count maps of a point-like source (*upper panels*) and extending source (*lower panels*). In *upper panels*, arc-like structures may be the stray light of the point-like sources and there remains some residuals probably due to pile-up.

as illustrated in Fig. 4.13. The instrumental background spectrum of each FXT module could be estimated using the FSA data with *fxtbkggen*:

```
fxtbkggen infile=fxta_fsa.pi outfile=fxta_bkg.pi
```

4.3.4 Smoothing

Smoothing is widely used in X-ray data analysis to enhance the morphology of extending sources for better display. `HEASoft` provides several smoothing scripts including *fgauss*, *fboxcar*, and *fadapt*. The first two smooth the picture in the same scale while the latter one applies adaptive smoothing. It should be noted that smoothed data are not recommended for data analysis.

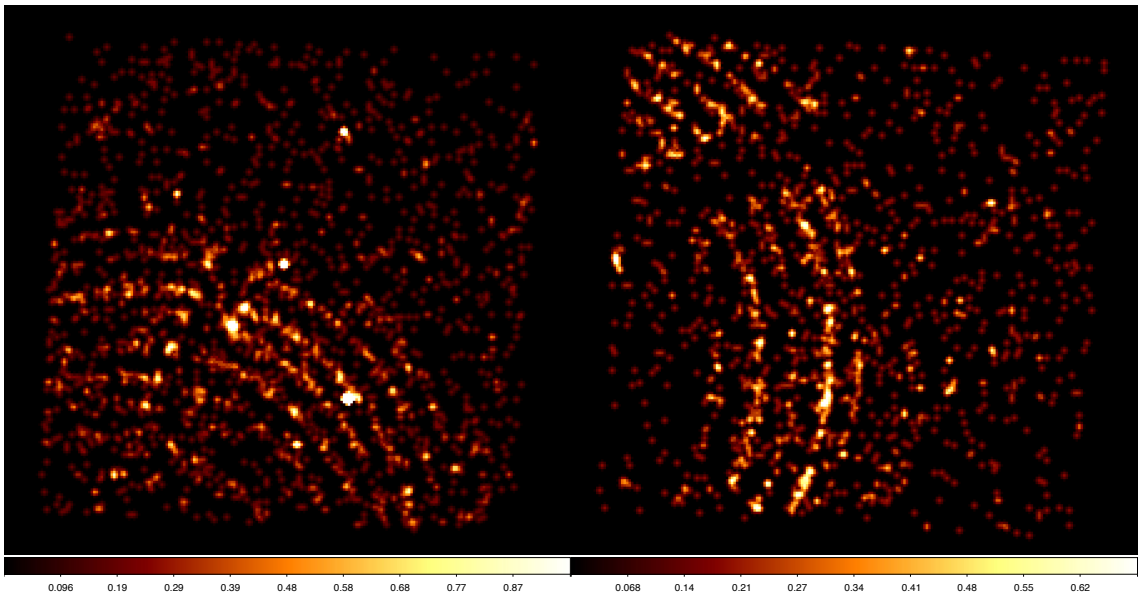

Figure 4.12: Examples of stray light from Crab nebula out of the field of view. The images are smoothed to underline the arc- and knot-like patterns of the stray light, which are not real stellar objects. The energy band is 0.4–7.0 keV.

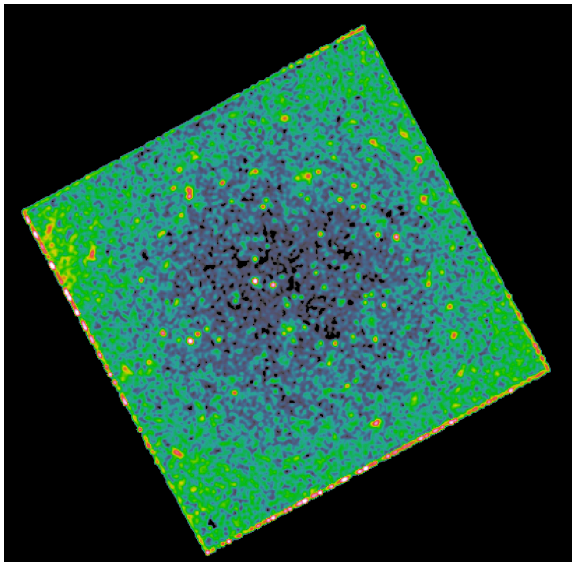

Figure 4.13: A vignetting corrected blank field without subtracting instrument background.

4.3.5 RGB image

To check the photon distribution in different energy ranges, we can assign different colors to the photons of different energies for easy viewing. Here we withdraw three images from different band ranges: 0.3-1 keV as the soft band, 1-3 keV as the middle

band, while 3-5 keV as the hard band.

```
$xselect
> Enter session name >[EP] EP  # session name, will be the prefix of later output
> set datadir .
> read events evt-a-cl.fits
> set datamode  FF/PW/TM   # when the mode are not recognized correctly.
> filter time file gti.fits
> set binsize 1.
> filter pha_cutoff 30 100   # channel number, with the width of 10eV
> extract image xysize=601 xybinsize=1 xcenter=300.5 ycenter=300.5 copyall=yes
> save image evt_soft.im
> clear pha_cutoff
> filter pha_cutoff 100 300
> extract image xysize=601 xybinsize=1 xcenter=300.5 ycenter=300.5 copyall=yes
> save image evt_mid.im
> clear pha_cutoff
> filter pha_cutoff 300 500
> extract image xysize=601 xybinsize=1 xcenter=300.5 ycenter=300.5 copyall=yes
> save image evt_hard.im
> quit
```

With the images of different bands, the false-color image could be created and tuned with ds9:

```
$ ds9 -rgb -red evt_soft.im -green evt_mid.im -blue evt_hard.im
```

Figure 4.14 shows a tri-color, vignetting-corrected, and adaptively smoothed image of M87 stacked by 4 observations with different filter. It is worth noting that the count rate is associated to the effect area of the instrument. Thus it should be cautious to combine images from different modules and filters where some artificial patterns might appear near the bad column of `FXTA` or the CCD edge. However, it always works to stack data for a higher signal-to-noise ratio.

4.3.6 Mosaic

Usually, more than one observation would be mosaicked or stacked to extend the imaging area or raise the signal-to-noise ratio. The `CIAO` script *reproject_image_grid*[10] can easily reproject and combine images based on the input image center and sampling rate. The count images (including background maps if needed) and exposure maps should be combined respectively and then the vignetting correction:

$$rate = \frac{\Sigma cts - (\Sigma bkg)}{\Sigma exp} \tag{4.2}$$

[10]https://cxc.cfa.harvard.edu/ciao/ahelp/reproject_image_grid.html

Figure 4.14: Vignetting corrected, adaptively smoothed, stacked image of X-ray lobes and halo of M87 based on the data of EP/FXT first light (*red:* 0.4–0.7 keV; *green:* 0.7–1.1 keV; *blue:* 1.1–2.3 keV). Some weak artificial patterns are still visible in the lower left and lower right parts of the picture, probably due to the difference in the effective area of the two modules. The colorbar is on a logarithmic scale to show faint structures.

It should be noted that for different filters and different modules, the effective area is different for different filters and modules, and thus, the count rate for the same source is also different. Therefore, it is recommended to use *area_scale=yes* to change the count rate to the "flux" in units of $\mathrm{cts\,s^{-1}\,cm^{-2}}$, which can effectively improve the imaging quality of the mosaic.

4.3.7 Source Detection

All point sources in the field should be identified and removed before analysis of extended sources. The simplest way to do this is to use XIMAGE, which includes a sliding-cell detection algorithm. With the command detect, all sources above the given signal-to-noise threshold (2 by default) will be labeled (as shown by Fig. 4.15) and saved.

```
$ximage
>read evt-a-cl.fits
>cpd /xtk
>disp
>detect/snr=5 filedet=source.txt
```

A typical output of the source detection (source.txt) is as below:

```
! Field Name     : Crab
! Instrument     : EP FXT FXTB
! No of sources  : 4
! Exposure (sec) : 2120.0000
! Input file     : evt-a-cl.fits
! Image zoom     : 1.0000
! Back/orig-pix/s: 4.5289551E-05
! Equinox        : 2000
! RA  Image Center: 342.17210
! Dec Image Center: -51.165710
! Start Time : 2022-11-02T08:05:50.00
! End Time   : 2022-11-02T08:41:10.00
!  #   count/s    err         pixel      Vig  RA(2000)   Dec(2000)   Err  H-Box
!                           x     y     corr                      rad  (sec)  prob snr
1 5.77E+00+/-5.3E-02 300.54980 300.44049 1.00 22 48 41.253 -51 09 57.132  -1    382 ...
2 4.69E-02+/-5.1E-03 398.15625 258.17709 1.00 22 47 00.675 -51 16 43.067  -1     63 ...
3 5.89E-02+/-5.8E-03 391.18658 239.91045 1.00 22 47 07.757 -51 19 40.047  -1     73 ...
4 4.51E-02+/-5.0E-03 213.17708 378.46875 1.00 22 50 10.658 -50 57 20.576  -1     63 ...
5
```

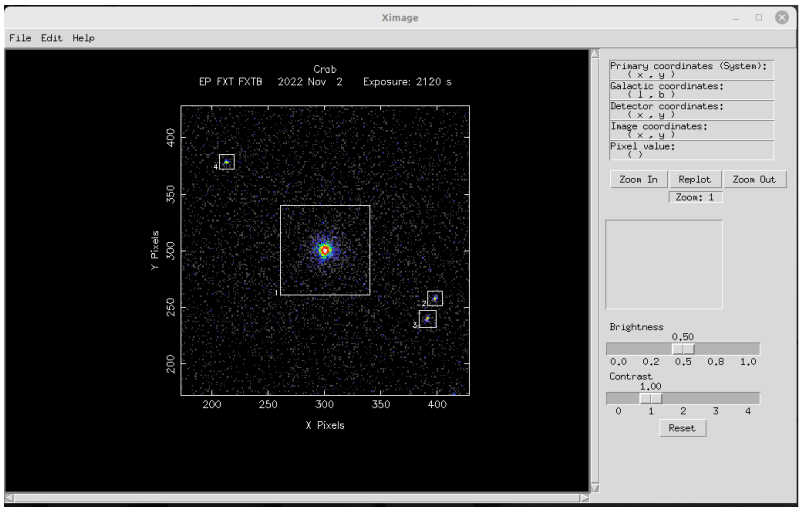

Figure 4.15: source detection with Ximage

The flux rates of these point sources can be estimated using the methods described in Section 4.2.1.

Note: the source detection tool wavdetect[11] command in the software package CIAO could also do this work.

It is possible to merge the image of FXT-A and FXT-B to increase the signal-to-noise rate. The command reproject_image[12] of CIAO could do this job.

[11]https://cxc.cfa.harvard.edu/ciao/ahelp/wavdetect.html
[12]https://cxc.cfa.harvard.edu/ciao/ahelp/reproject_image.html

4.3.8 Spectrum Fitting

Spectra extraction

Before extracting the spectra, the appropriate source and background regions need to be selected in ds9 manually with filenames as source.reg and bck.reg, respectively. With these two region files, the spectra can be withdrawn in XSELECT. CAUSTION: Don't filter the energy channel at this stage, otherwise the spectrum file will not be properly associated with the response file.

```
$xselect
> Enter session name >[EP] EP  # session name, will be the prefix of later output
> set datadir .
> read events evt-a-cl.fits
> set datamode  FF/PW/TM   # when the mode are not recognized correctly.
> filter time file gti.fits
> filter region source.reg
> extract spectrum
> save spectrum fxta.pi
> clear region
> filter region bck.reg
> extract spectrum
> save spectrum fxta_bck.pi
> quit
```

Response file generation

Then we can calculate corresponding Ancillary Response File (ARF) with the command *fxtarfgen*. For extended sources, the keyword "extend=1" is necessary.

```
fxtarfgen specfile=fxta.pi expfile=expmap-without_vign.fits
outfile=fxta_arf.fits extend=1
```

The RMF files could be generated with the command *fxtrmfgen*

```
fxtrmfgen specfile=fxta.pi outfile=fxta_rmf.fits
```

The we could link the spectrum and its response files with the command *grppha* as below.

```
grppha infile=fxta.pi outfile=fxta_gr.pi chatter=0 \
comm=' group min 10 & chkey RESPFILE fxta_rmf.fits & chkey ANCRFILE fxta_arf.fits \
& chkey BACKFILE fxta_bck.pi & exit' clobber=yes
```

Spectra fitting

Finally, we can fit the spectra with Xspec. A detailed explanation on commands and models could be found in its official manual.

```
$xspec
>cpd /xw
>data fxta_gr.pi
>show rate
>query yes
>setplot energy
>abund angr
>notice **-**
>ignore bad
>ignore 0.0-0.5
>ignore 8.0-**
>setplot rebin 3 20
>plot ldata
>statistic cstat
>model tbabs*apec
>freeze 1 4
>thaw 2 3 5
>renorm
>fit 20 1e-2
>setplot device spec.eps /cps
>setplot rebin 3 20
>plot ldata res
>quit
```

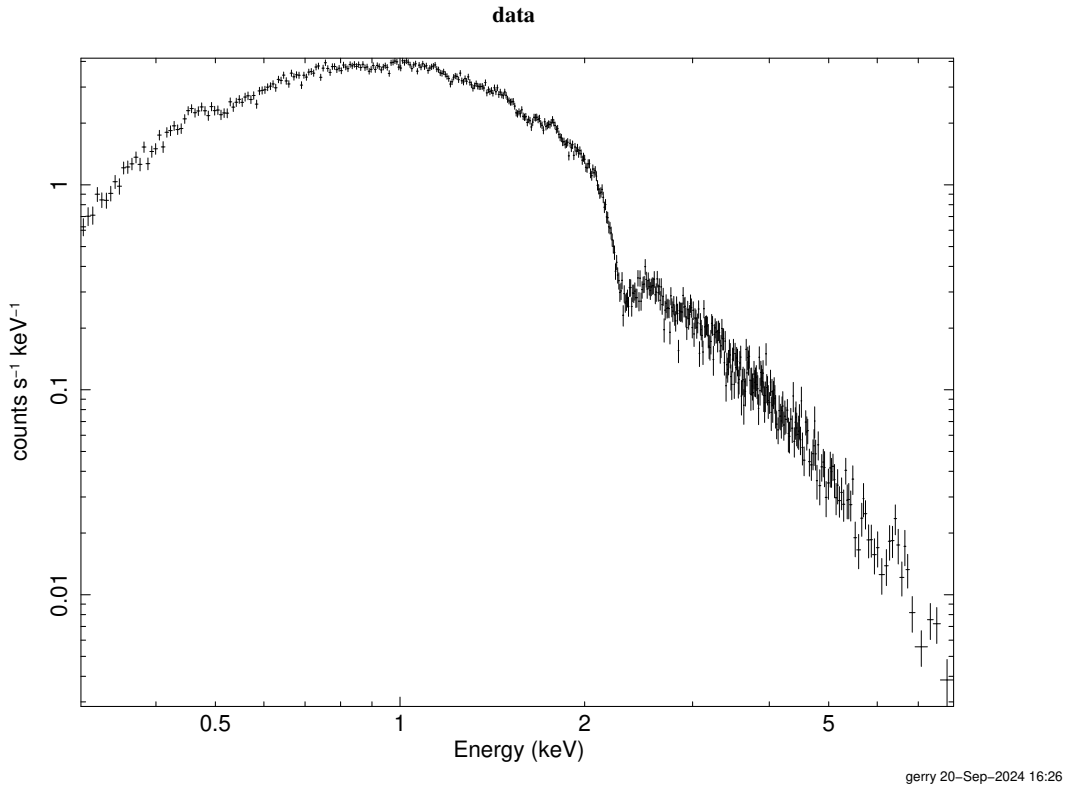

Figure 4.16: Spectrum of the PV target galaxy cluster A3571

Chapter 5

Calibration Files

5.1 Introduction

The Einstein Probe Follow-up X-ray Telescope (EP-FXT) Calibration Database (CALDB) is an essential resource for researchers working with EP-FXT data. The CALDB contains a comprehensive collection of calibration files necessary for accurate data reduction and analysis. These files provide detailed information about the properties and performance of the EP-FXT detectors, mirrors, and other instrument components. The CALDB is organized into a well-defined directory structure, making it easy for users to locate and access the calibration files required for their specific analysis tasks.

Some of the key calibration files included in the EP-FXT CALDB are:

1) Ancillary Response Files (ARF): These files contain information about the effective area of the detectors and mirrors as a function of energy.

2) Response Matrix Files (RMF): These files characterize the energy response of the EP-FXT detectors, accounting for energy dispersion and detector resolution.

3) Point Spread Function (PSF) files: These files describe the spatial distribution of X-ray photons as they are focused by the EP-FXT mirrors.

4) Energy Encircled Fraction (EEF) files: These files provide information on the fraction of energy contained within a given radius for a point source.

5) Telescope Definition (TELDEF) files: These files describe the geometry and alignment of the EP-FXT detectors and mirrors.

6) Vignetting files: These files contain information on the reduction of the telescope's effective area due to off-axis angles.

7) Filter Transmission (FTRANS) files: These files describe the transmission properties of the filters used in EP-FXT.

8) Quantum Efficiency (QE) files: These files provide information on the detectors' sensitivity to incoming photons as a function of energy.

9) Gain and Charge Transfer Inefficiency (GAIN) files: These files contain information about the gain calibration and charge transfer inefficiency of the EP-FXT detectors.

10) Mirror effective area (EFF) files: These files provide data on the reflectivity and efficiency of the EP-FXT mirrors as a function of energy.

11) Background files: These files contain information on the instrumental and astrophysical background components relevant to the EP-FXT observations.

The EP-FXT CALDB plays a crucial role in ensuring that researchers can obtain the most accurate and reliable results from their analysis of EP-FXT data. By providing up-to-date calibration information, the CALDB enables scientists to fully exploit the exceptional capabilities of the Einstein Probe Follow-up X-ray Telescope and contribute to the advancement of our knowledge of the high-energy universe.

5.2 Calibration files naming and listing

The calibration database is stored in the $CALDB environment, and all contents are stored in the data/EP/fxt/ directory, divided into index, bcf and cpf directories. 'index' stores the calibration index files, 'bcf' stores the base calibration files, and 'cpf' stores the calibration product files.

TELESCOP	INSTRUME	INDEX/CCLS	CCNM
$CALDB/data/EP/	fxt/	index/	(index files)
		bcf/	badpix/, teldef/, effarea/, ftrans/, qe/, gain/
		cpf/	psf/, arf/, rmf/, eef/, vignetting/, bkg/

Table 5.1: CALDB directory structure

The naming convention for calibration files stored in FXT's calibration database follows this format:

`FXT_module_mode_datatype_[date]_v<version>.ext`

Here are the components of this naming convention:

Module: Represents different modules, labeled as 'a' or 'b'.

Mode: Describes the observation mode. 'cal' stands for calibration, 'ff' for full frame, 'pw' for partial window, and 'tm' for timing mode.

Datatype: Indicates the type of data. For example, 'gain' corresponds to a gain file, and 'psf' corresponds to a PSF (Point Spread Function) file.

Date: The applicable date of the calibration file, formatted as YYYYMMDD.

Version: Represents the version information.

Ext: Represents the extension name. By default, it is 'fits'. Exceptions are 'RMF' for response matrix files, 'ARF' for effective area files (also known as auxiliary response files), and 'teldef' for telescope definition files.

For details on the design of EP-FXT Calibration database, please refer to the study by Li et al. 2025 [1].

Chapter 6

FXT Pipeline: fxtchain tool

An easy way to perform data reduction of FXT is to utilize the fxtchain tool, which searches for EVT, MKF, ORB, and ATT files in the input directory, and then executes tasks fxtcoord, fxttimecorr(for tm mode only), fxtpical, fxtparticleidentify, fxtbadpix, fxthotpix, fxtgrade, fxtgtigen, xselect, fxtexpogen, fxtarfgen and fxtrmfgen on each EVT file, all results of the tasks will be saved to the output directory after the process is completed.

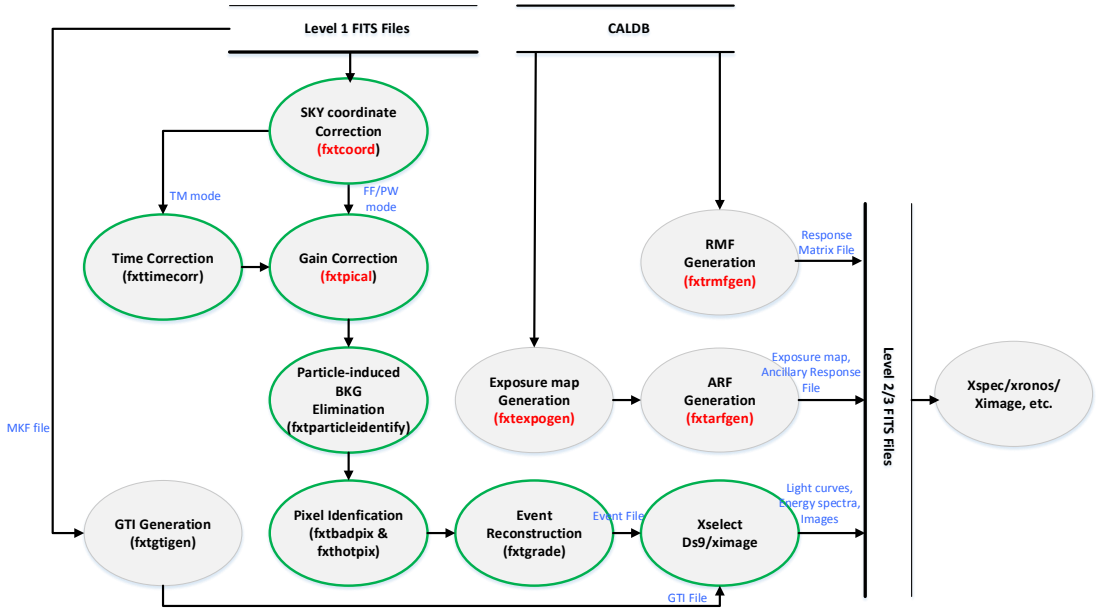

Figure 6.1: The data reduction flow of fxtchain

Figure 6.1 shows the execution process of the tool, let's introduce it step by step below:

- fxtcoord

fxtcoord is the first task to be executed, and the command executed in fxtchain is as follows:

```
$fxtcoord  evtfile={your_input_dir}/fxt/event/fxt_<module> \
    _<obsID>_<datamode>_<filter>_po_uf_evt_<ver>.fits \
    attfile={your_input_dir}/auxil/ep_<obsID>_att_<ver>.fits \
    outfile={your_output_dir}/fxt_<module>_<obsID>_<datamode> \
    _<filter>_po_evt_coord_<ver>.fits
```

If user specifies the parameters module and datamode, all the tasks will only execute for the specified EVT files; otherwise, the tasks will be executed for all EVT files by default. This is a general guideline for executing tasks in fxtchain, and it will not be repeated in the introduction of subsequent tasks.

- fxttimecorr
  This task is only applicable to EVT data in TM mode, and the command to execute in fxtchain is:

```
$fxttimecorr ra={RA} dec={DEC} \
    evtfile={your_output_dir}/fxt_<module>_<obsID>_<datamode> \
    _<filter>_po_evt_coord_<ver>.fits \
    attfile={your_input_dir}/auxil/ep_<obsID>_att_<ver>.fits \
    outfile={your_output_dir}/fxt_<module>_<obsID>_<datamode> \
    _<filter>_po_evt_timecorr_<ver>.fits
```

  Please note that the parameters ra and dec can be passed through the parameters of fxtchain, but both ra and dec must be assigned values simultaneously; otherwise, they will be considered invalid and ignored. In this case, fxtchain will read from the EVT file.

- fxtpical
  For EVT data in TM mode, the command to execute is:

```
$fxtpical evtfile={your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_evt_timecorr_<ver>.fits \
    outfile={your_output_dir}/fxt_<module>_<obsID>_<datamode> \
    _<filter>_po_evt_pi_<ver>.fits
```

  While for data in FF and PW mode, the command to execute is:

```
$fxtpical evtfile={your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_evt_coord_<ver>.fits \
    outfile={your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_evt_pi_<ver>.fits
```

The only difference between the two is the input EVT file, as an additional fxttimecorr task is executed in TM mode.

- fxtparticleidentify
  The command to execute in fxtchain is:

```
$fxtparticleidentify evtfile={your_output_dir}/fxt_<module> \
    _<obsID>_<datamode>_<filter>_po_evt_pi_<ver>.fits \
    outfile={your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_evt_particle_<ver>.fits \
    x_length={X_LENGTH} y_length={Y_LENGTH}
```

  fxtchain does not provide user input for the parameters x_length and y_length. These two parameters take different values based on the data mode. When the mode is TM, both x_length and y_length are assigned a value of 35; in other modes, they are assigned a value of 11.

- fxtbadpix
  The command to execute in fxtchain is:

```
$fxtbadpix evtfile={your_output_dir}/fxt_<module> \
    _<obsID>_<datamode>_<filter>_po_evt_particle_<ver>.fits \
    outfile={your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_evt_badpix_<ver>.fits \
    userbpfile={USERBPFILE}
```

  If the command for fxtchain includes the parameter userbpfile, then that parameter will be passed to fxtbadpix; otherwise, fxtbadpix only requires the two necessary parameters: evtfile and outfile.

- fxthotpix
  The command to execute in fxtchain is:

```
$fxthotpix evtfile={your_output_dir}/fxt_<module> \
    _<obsID>_<datamode>_<filter>_po_evt_badpix_<ver>.fits \
    outfile={your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_evt_hotpix_<ver>.fits
```

- fxtgrade
  The command to execute in fxtchain is:

```
$fxtgrade evtfile={your_output_dir}/fxt_<module> \
    _<obsID>_<datamode>_<filter>_po_evt_hotpix_<ver>.fits \
    outfile={your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_evt_grade_<ver>.fits \
    pithresh=0
```

- fxtgtigen
  The command to execute in fxtchain is:

  ```
  $fxtgtigen mkffile={your_input_dir}/fxt/hk/fxt \
      _<obsID>_mkf_<ver>.fits \
      outfile={your_output_dir}/fxt_<module>_<obsID> \
      _<datamode>_<filter>_po_gti_<ver>.fits \
      module={MODULE} expr={EXPR} usergtifile={USERGTIFILE}
  ```

  It is important to note that the parameter module for fxtgtigen does not come from the parameter module of fxtchain, but rather from the currently processed EVT data. Its values are not a and b, but fxta and fxtb, which is also different from the parameters of fxtchain. As for the parameters expr and usergtifile, they are directly taken from the parameters of the same name in fxtchain.

- xselect
  The command to execute in fxtchain is:

  ```
  $xselect @{your_output_dir}/xselect_run_<module> \
      _<obsID>_<datamode>_<filter>_po_<ver>.txt
  ```

  The commands used during the xselect filtering process are written into a file, and its contents are mainly influenced by the parameters exitstage, srcffregion, srcpwregion, srctmregion, bkgffregion, bkgpwregion, and bkftmregion.
  When exitstage is unassigned or set to 1, and the parameters srcffregion, srcpwregion, srctmregion, bkgffregion, bkgpwregion, and bkftmregion are all unassigned, the contents of the file are as follows:

  ```
  EP
  set datadir {your_output_dir}
  read events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_evt_grade_<ver>.fits
  yes
  set binsize {binsize}
  filter grade {grade}
  filter time file ./fxt_<module>_<obsID>_<datamode>_<filter>_po_gti_<ver>.fits
  select event "status==b0"
  extract events copyall=yes
  save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_<ver>.fits
  no
  # exit xselect
  clear all proceed=yes
  quit
  no
  ```

The variable "your_output_dir", "binsize", and "grade" in the file are all derived from the parameters of fxtchain. If binsize is not assigned, it defaults to 1, and if grade is not assigned, it defaults to 0-12.
When exitstage equals 2 and the parameters srcffregion, srcpwregion, srctmregion, bkgffregion, bkgpwregion, and bkftmregion are all not assigned values, the contents of the file are as follows:

```
EP
set datadir {your_output_dir}
read events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_evt_grade_<ver>.fits
yes
set binsize {binsize}
filter grade {grade}
filter time file ./fxt_<module>_<obsID>_<datamode>_<filter>_po_gti_<ver>.fits
select event "status==b0"
extract events copyall=yes
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_<ver>.fits
no
extract spectrum copyall=yes
save spectrum ./fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.pha
filter pha_cutoff 38 925
extract events copyall=yes
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_tmp_<ver>.fits
no
extract curve copyall=yes
save curve ./fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.lc
extract image xysize=601 xybinsize=1 xcenter=300 ycenter=300 copyall=yes
save img ./fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.img
clear pha_cutoff
clear events
# exit xselect
clear all proceed=yes
quit
no
```

When exitstage is unassigned or equals 1, and the parameters srcffregion, srcpwregion, srctmregion, bkgffregion, bkgpwregion, and bkftmregion provide selected regions for the currently processed EVT data, the contents of the file are as follows:

```
EP
set datadir {your_output_dir}
read events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_evt_grade_<ver>.fits
```

```
yes
set binsize {binsize}
filter grade {grade}
filter time file ./fxt_<module>_<obsID>_<datamode>_<filter>_po_gti_<ver>.fits
select event "status==b0"
extract events copyall=yes
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_<ver>.fits
no
# if src region is defined
filter region {your_src_region_file}
extract events copyall=yes
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_src_<ver>.fits
no
clear region
clear events
# if bkg region is defined
filter region {your_bkg_region_file}
extract events copyall=yes
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_bkg_<ver>.fits
no
clear region
clear events
# exit xselect
clear all proceed=yes
quit
no
```

When exitstage is assigned the value 2, and the parameters srcffregion, srcpwregion, srctmregion, bkgffregion, bkgpwregion, and bkftmregion provide selected regions for the currently processed EVT data, the contents of the file are as follows:

```
EP
set datadir {your_output_dir}
read events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_evt_grade_<ver>.fits
yes
set binsize {binsize}
filter grade {grade}
filter time file ./fxt_<module>_<obsID>_<datamode>_<filter>_po_gti_<ver>.fits
select event "status==b0"
extract events copyall=yes
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_<ver>.fits
no
```

```
# if src region is defined
filter region {your_src_region_file}
extract events copyall=yes
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_src_<ver>.fits
no
clear region
clear events
# if bkg region is defined
filter region {your_bkg_region_file}
extract events copyall=yes
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_bkg_<ver>.fits
no
clear region
clear events
extract spectrum copyall=yes
save spectrum ./fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.pha
filter pha_cutoff 38 925
extract events copyall=yes
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_tmp_<ver>.fits
no
extract curve copyall=yes
save curve ./fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.lc
extract image xysize=601 xybinsize=1 xcenter=300 ycenter=300 copyall=yes
save img ./fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.img
clear pha_cutoff
clear events
# if src region is defined
filter region {your_src_region_file}
extract all copyall=yes
save spectrum ./fxt_<module>_<obsID>_<datamode>_<filter>_po_src_<ver>.pha
save curve ./fxt_<module>_<obsID>_<datamode>_<filter>_po_src_<ver>.lc
clear region
clear events
# if bkg region is defined
filter region {your_bkg_region_file}
extract all copyall=yes
save spectrum ./fxt_<module>_<obsID>_<datamode>_<filter>_po_bkg_<ver>.pha
save curve ./fxt_<module>_<obsID>_<datamode>_<filter>_po_bkg_<ver>.lc
clear region
clear events
# exit xselect
clear all proceed=yes
quit
```

```
no
```

- fxtexpogen
  The command to execute in fxtchain is:

```
$fxtexpogen evtfile={your_output_dir}/fxt_<module> \
    _<obsID>_<datamode>_<filter>_po_cl_<ver>.fits \
    outfile={your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_<ver>.expo \
    mkffile={your_input_dir}/fxt/hk/fxt \
    _<obsID>_mkf_<ver>.fits
```

  The evtfile here is the cleaned event file obtained after executing xselect.

- fxtarfgen
  The command to execute in fxtchain is:

```
$fxtarfgen {your_output_dir}/fxt_<module> \
    _<obsID>_<datamode>_<filter>_po_<ver>.pha \
    {your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_<ver>_without_vign.expo \
    {your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_<ver>.arf
```

- fxtrmfgen
  The command to execute in fxtchain is:

```
$fxtrmfgen specfile={your_output_dir}/fxt_<module> \
    _<obsID>_<datamode>_<filter>_po_<ver>.pha \
    outfile={your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_<ver>.rmf
```

- powspec
  The command to execute in fxtchain is:

```
$powspec norm=1 \
    cfile={your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_cl_tmp_<ver>.fits \
    window=- dtnb=<DTNB> nbint=<NBINT> nintfm=INDEF \
    rebin=-1.005 plot=no \
    outfile={your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_<ver>.pds
```

Parameter	FF	PW	TM
dtnb	0.05	0.0022	0.001953125/4.0
nbint	2048	4096*8	4096*32

Table 6.1: The values of parameters.

fxtchain does not support user input for the parameters dtnb and nbint, these two parameters take different values based on the mode of the currently processed EVT data:

The fxtchain tool provides various parameters, most of which are optional, and some of them having default values(detailed parameter information will be provided in the next section). In addition, this tool also supports interactive mode. When the user's input command lacks necessary parameters, it automatically switches to interactive mode and waits for the user to complete all necessary parameters before starting the data reduction process.

PARAMETER

The parameters of this tool are listed below:

- indir [string]
  The input file directory should contain all necessary files (EVT, MKF, ORB, ATT). The directory hierarchy must adhere to strict requirements; otherwise, the fxtchain process will terminate prematurely due to missing files.

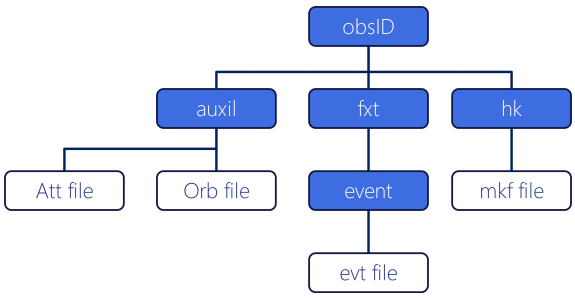

Figure 6.2: The input file directory hierarchy

This parameter must be provided, otherwise fxtchain will run into interactive mode and wait for user input.

- outdir [string]
  The output directory is used to store the results of data reduction. Users can

choose any directory as the output directory, as long as they have write permissions for it. If the specified directory does not exist, fxtchain will attempt to create it. The files that may be saved in the output directory include EVT files, GTI files, light curve files and their images, spectrum files and their images, etc. The output results are dependent on the parameters provided by the user, and their relationship is illustrated in the table below.

FileName	Parameter(The relationship between multiple parameters is "AND")					
	exitstage=	module=	datamode=	datatype=	src<datamode>region=	bkg<datamode>region=
fxt_<module>_<obsID>_<datamode>_<filter>_po_gti_<ver>.fits	1 or 2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_<ver>.fits	1 or 2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.img	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.img.png	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.lc	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.lc.png	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.pha	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.pha.png	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.expo	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.expo.png	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.pds	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.pds.png	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>-without_vign.expo	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.arf	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.rmf	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_src_<ver>.fits	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	{src region file}	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_src_<ver>.lc	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	{src region file}	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_src_<ver>.pha	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	{src region file}	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_src_<ver>.img	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	{src region file}	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_src_<ver>.expo	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	{src region file}	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_src_<ver>-without_vign.expo	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	{src region file}	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_src_<ver>.arf	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	{src region file}	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_src_<ver>.rmf	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	{src region file}	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_bkg_<ver>.fits	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	{bkg region file}
fxt_<module>_<obsID>_<datamode>_<filter>_po_bkg_<ver>.lc	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	{bkg region file}
fxt_<module>_<obsID>_<datamode>_<filter>_po_bkg_<ver>.pha	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	{bkg region file}
fxt_<module>_<obsID>_<datamode>_<filter>_po_bkg_<ver>.img	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	{bkg region file}
fxt_<module>_<obsID>_<datamode>_<filter>_po_bkg_<ver>.expo	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	{bkg region file}
fxt_<module>_<obsID>_<datamode>_<filter>_po_bkg_<ver>-without_vign.expo	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	{bkg region file}
fxt_<module>_<obsID>_<datamode>_<filter>_po_bkg_<ver>.arf	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	{bkg region file}
fxt_<module>_<obsID>_<datamode>_<filter>_po_bkg_<ver>.rmf	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	{bkg region file}
fxt_<module>_<obsID>_<datamode>_<filter>_po_fsa_cl_<ver>.fits	1 or 2	<module> or unassigned	<datamode> or unassigned	fsaevt	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_fsa_<ver>.pha	2	<module> or unassigned	<datamode> or unassigned	fsaevt	-	-

Figure 6.3: The relationship between output files and input parameters of fxtchain

This parameter must be provided, otherwise fxtchain will run into interactive mode and wait for user input.

- exitstage [int]
  The exitstage is used to control the termination of the pipeline, with optional values of 1 and 2. Value 1 signifies the termination of the pipeline after generating clean event data, while value 2 indicates the generation of higher-level products. The default parameter value is 1.

- module [string]
  The value of the module parameter is used to define a module filter. When this parameter is assigned a value, the filter is enabled. Only EVT files with modules that match the parameter value will be processed. The optional values for the parameter are 'a' and 'b', for example, module=a.

- datamode [string]
  The parameter is used to define a datamode filter. When this parameter is assigned a value, the filter is enabled. Only EVT files with datamodes that match the parameter value will be processed. The optional values for the

parameter are FF, PW, and TM (case-insensitive). Multiple values should be separated by commas, for example, datamode=FF,PW.

- datatype [string]
  The parameter is used to specify the file types that need to be processed. The default value is "evt", meaning only evt files will be processed by default. Other optional values include "fsaevt". If you need to process both types of files simultaneously, please separate them with commas in the parameter, for example: datatype=evt,fsaevt

- binsize [int]
  Bin size (seconds) for the light curve, default value is 1.

- ra [float]
  The value of RA only takes effect in TM mode, and the parameter DEC must also be assigned a value at the same time, otherwise, the parameter is considered invalid.

- dec[float]
  The value of DEC only takes effect in TM mode, and the parameter RA must also be assigned a value at the same time, otherwise, the parameter is considered invalid.

- grade[string]
  The value of grade is for missions with CCD detectors whose events are assigned a grade. The grade can be specified as a single number (eg 0), a range (eg 0:2 or 0-2), an upper limit (eg <4), or a lower limit (eg >3). Specifications should be separated by commas (eg 0,2-6) and enclosed in double quotes if given on the command line instead of when prompted.

- expr[string]
  The value of selection expression is for attitude and instrument HK or DEFAULT(the default) for standard (from CALDB). To avoid parsing issues with special characters(e.g., &, >), the value should be enclosed in single quotes (e.g., expr='ELV>-10'); however, quotes may be omitted if the value contains no special characters (e.g., expr=DEFAULT).

- usergtifile[string]
  The path of the input GTI file (FITS format) or NONE for none

- userbpfile[string]
  The path of the user BADPIX file Or NONE for none

- srcffregion[string]
  This region is used to define a source filter when the currently processed EVT file is in FF mode. The region filter is in ds9 format. Region filters for the

filter region command can be created graphically via the plot image command, which spawns a ds9 session.

- srcpwregion[string]
  This region is used to define a source filter when the currently processed EVT file is in PW mode. The region filter is in ds9 format. Region filters for the filter region command can be created graphically via the plot image command, which spawns a ds9 session.

- srctmregion[string]
  This region is used to define a source filter when the currently processed EVT file is in TM mode. The region filter is in ds9 format. Region filters for the filter region command can be created graphically via the plot image command, which spawns a ds9 session.

- bkgffregion[string]
  This region is used to define a background filter when the currently processed EVT file is in FF mode. The region filter is in ds9 format. Region filters for the filter region command can be created graphically via the plot image command, which spawns a ds9 session.

- bkgpwregion[string]
  This region is used to define a source filter when the currently processed EVT file is in PW mode. The region filter is in ds9 format. Region filters for the filter region command can be created graphically via the plot image command, which spawns a ds9 session.

- bkgtmregion[string]
  This region is used to define a source filter when the currently processed EVT file is in TM mode. The region filter is in ds9 format. Region filters for the filter region command can be created graphically via the plot image command, which spawns a ds9 session.

- outputlist[string]
  This parameter is used to control the generation of the output file list. When its value is 'yes' or 'Y', the list of output product files will be generated. When its value is 'no' or 'N', the file list will not be generated. The default value is 'N'

- savetmp[string]
  This parameter is used to control whether to keep temporary files. When the value is 'Y' or 'yes', the temporary files will be retained. When the value is 'N' or 'no' or when it is unassigned, the temporary files will be removed.

6.1 Usage

1. The basic fxtchain command uses the following required input parameters:

```
$fxtchain indir={your_input_path} \
    outdir={your_output_path} \
    exitstage=1 \
    expr=DEFAULT
```

   The 'your_input_path' is the directory where you input files are saved, the 'your_output_path' is the output directory. By default, fxtchain will produce clean event files and good time intervals (GTI) files for FF, PW and TM. However, for the TM mode, the position and gain correction is always incorrect. In this version, expr can be written as expr=DEFAULT, which will invoke the latest selection criteria. In order to obtain an accurate correction, users need to set the coordinates of the source:

```
$fxtchain indir={your_input_path} \
    outdir={your_output_dir} \
    exitstage=1 \
    expr=DEFAULT \
    ra=161.265 \
    dec=-59.685
```

2. The following command shows how to obtain Level 3 data products, note that the parameter "exitstage" here is set to 2:

```
$fxtchain indir={your_input_path} \
    outdir={your_output_dir} \
    exitstage=2 \
    expr=DEFAULT
```

3. The following command shows how to obtain module a/b files:

```
$fxtchain indir={your_input_path} \
    outdir={your_output_path} \
    exitstage=2 \
    expr=DEFAULT \
    module=a
```

4. In this example fxtchain only runs a data mode:

```
$fxtchain indir={your_input_path} \
    outdir={your_output_path} \
    exitstage=2 \
    expr=DEFAULT \
    datamode=ff
```

5. Fxtchain is applied to FSAEVT files:

```
$fxtchain indir={your_input_path} \
    outdir={your_output_path} \
    exitstage=2 \
    expr=DEFAULT \
    datatype=fsaevt
```

6. Source/backgrouond pha:

```
$fxtchain indir={your_input_path} \
    outdir={your_output_path} \
    exitstage=2 \
    expr=DEFAULT \
    datamode=ff \
    srcffregion={your_ff_src_region} \
    bkgffregion={your_ff_bkg_region}
```

   The above example deals with the case of datamode=ff. If you pass in parameters srcpwregion, bkgpwregion, srctmregion, bkgtmregion, they will have no actual effect. Of course, you can also process all datamodes:

```
$fxtchain indir={your_input_path} \
    outdir={your_output_path} \
    exitstage=2 \
    expr="DEFAULT \
    srcffregion={your_ff_src_region} \
    bkgffregion={your_ff_bkg_region} \
    srctmregion={your_tm_src_region} \
    bkgtmregion={your_tm_bkg_region} \
    srcpwregion={your_pw_src_region} \
    bkgpwregion={your_pw_bkg_region}
```

7. Interactive mode only accepts four mandatory parameters:

```
$fxtchain

                    **  fxtchain 1.20  **

The input directory : {your_input_path}
The argument indir has been set to {your_input_path}

The output directory : {your_output_path}
The argument outdir has been set to {your_output_path}

The exit stage [1] :
The argument exitstage has been set to 1

The value of expression [DEFAULT] :
The argument expr has been set to DEFAULT
```

Users can input values for the parameters "indir", "outdir", "exitstage", and "expr" when prompted. It is important to note that default values are provided for both "exitstage" and "expr". If there is no need to provide other values for these two parameters, the users can simply press the Enter key to accept the default values. Of course, if a value for any of these parameters has already been provided in the command line, it will not be requested again when entering interactive mode.

Appendix A

Installation of the FXTDAS

The following conditions need to be met in order to install this software:

- Up to 5 GB free disk space (if installing all of HEASOFT):
  Actual space needed varies with system architecture and software packages selected.
- X11 / X Windows:
  If you plan to use any graphical tools (XSPEC, XIMAGE, FV, fplot, etc.), you will need X11. For more information, visit:
  https://www.x.org
  https://xquartz.macosforge.org/landing/
- python version >= 3.8
  numba package is needed.

A.1 Installation - Source Code

A.1.1 Ubuntu

The source code of FXTSoft could be compiled and installed on the Ubuntu 20.04, 22.04 and other Linux Distribution based the them, e.g. MintLinux, Deepin, etc.

1. **Prerequisite packages**

   For the installation of the first time, you will need to install a few other packages which are needed in order to build FXTSoft from the source code distribution:

   ```
   sudo apt-get -y install libreadline-dev
   sudo apt-get -y install libncurses5-dev
   sudo apt-get -y install libx11-dev
   sudo apt-get -y install libxt-dev
   ```

```
sudo apt-get -y install ncurses-dev
sudo apt-get -y install curl
sudo apt-get -y install libcurl4
sudo apt-get -y install libcurl4-gnutls-dev
sudo apt-get -y install xorg-dev
sudo apt-get -y install make
sudo apt-get -y install gcc g++ gfortran
sudo apt-get -y install perl-modules
```

2. **Unpacking**

   You can unpack the file you downloaded using e.g.:

   ```
   tar zxvf fxtsoft.tar.gz
   gunzip fxtsoft.tar
   ```

   This will create a fxtsoft/ directory containing the source code.

3. **Building**

   Configure the software for your platform. Go to the fxtsoft/BUILD_DIR directory, and run the main configure script, which will probe your system for libraries, header files, compilers, etc., and then generate the main Makefile. TargetPATH is the path where you want to put this program.

   ```
   cd fxtsoft/BUILD_DIR
   ./configure --prefix=~/TargetPATH
   make
   make install
   ```

4. **Initialization**

   For users of Bourne Shell (sh, ash, ksh, bash, zsh), input belowing commands in your terminal then your FXTSoft will be ready.

   ```
   export HEADAS=/path/to/your/installed/fxtsoft/(PLATFORM)
   . $HEADAS/headas-init.sh
   ```

   However, these commands only work for current terminal. You can put them in your .bashrc file.

   ```
   export HEADAS="~/TargetPATH/(PLATFORM)"
   alias fxtinit=". $HEADAS/headas-init.sh"
   ```

   Then you can initialize your FXTSoft with the fxtinit command.

5. **configure CALDB**

   CALDB is the calibration database of the FXT. It does not require installation and you can unzip it to any convenient location. As long as you set the correct location in the environment variables, it will work fine. You can put input belowing commands or put them in your .bashrc file.

```
export CALDB=/PATH/CALDB
export CALDBCONFIG=$CALDB/caldb.config
export CALDBALIAS=$CALDB/software/tools/alias_config.fits
```

Note: Don't try to compile or install the code into a NTFS partition. This partition format does not allow colons (:) in filenames, which can cause a CFITSIO compilation error: "Can't write-open blib/man3/Astro::FITS::CFITSIO.3pm"

A.1.2 CentOS

The source code of FXTSoft has been tested successfully with CentOS 7.9. Note that, before installing FXTSoft, you need to update Python 3.8 or later.

A.1.3 Mac OS

The source code of FXTSoft could also be compiled and installed on the MacOS Sonoma version 14.2.1 with the M3 pro chip. For other versions and chips of Mac, proper installation and operation cannot be guaranteed.

You can unpack the file you downloaded using e.g.:

```
tar zxvf fxtsoft.tar.gz
gunzip fxtsoft.tar
```

A fxtsoft directory containing the source code will be established.

Configure the software for your platform. Go to the fxtsoft/BUILD_DIR directory, and run the main configure script, which will probe your system for libraries, header files, compilers, etc., and then generate the main Makefile. TargetPATH is the path where you want to put this program.

```
arch -x86_64 /bin/bash
cd fxtsoft/BUILD_DIR
./configure --prefix=~/TargetPATH
make
make install
```

A.2 Installation - Docker

Docker is an open-source platform that enables developers to effortlessly create, deploy, and run applications in containers. Containers are lightweight, self-sufficient executable packages that contain all the necessary components to run an application, including code, libraries, system tools, and settings.

Before utilizing FXTSoft&Docker, it is essential to ensure that the Docker engine has been installed. For detailed installation instructions, please visit:
https://docs.docker.com/engine/install.
Once your Docker environment is set up, you can obtain the FXTSoft&Docker image from the FXTSoft team and then import it into your Docker environment.

```
$docker load -i {FXTSoft&Docker_IMG}
```

This may take some time, please be patient and wait for the process to finish. after that, you can view the image by :

```
$docker images
```

To run a container with the FXTSoft image, you can create a container with its <IMAGE ID> or <REPOSITORY:TAG>, and open an interactive terminal:

```
$docker run -v {HOST_CALDB}:/caldb \
    -v {HOST_INPUT_DIR}:{LOCAL_INPUT_DIR} \
    -v {HOST_OUPUT_DIR}:{LOCAL_OUTPUT_DIR} \
    -it {REPOSITORY:TAG} /bin/bash
```

The parameter "v" indicates mounting a directory from the host onto a directory inside the container, with a colon ":" separating the two directories. This allows you to directly access files from the host in the container. In the example above, we used three mount points, but this is not a strict requirement, just to demonstrate the usage of this parameter. It is strongly recommended to mount the CALDB directory onto "/caldb" inside the container, as the container already has CALDB environment set up. If you choose to mount CALDB to a different location, remember to update the relevant CALDB environment after starting the container.

The "it" is a combination of two parameters. "i" indicates interactive operation, and "t" indicates the allocation of a pseudo-terminal.

Upon container startup, it will immediately execute the "/bin/bash" command, opening a bash terminal inside the container, where you can perform various FXT data reduction tasks.

When you leave a container, you can check its CONTAINER ID and re-enter it with:

```
$docker ps -a  # view all containers
$docker start <CONTAINER ID> # start an existing container
$docker exec -it <CONTAINER ID> /bin/bash  # open a terminal for a running container
```

If you did something wrong in the container or mess it up, you can simply delete it and then start a new one:

```
$docker stop <CONTAINER ID> # stop a container
$docker rm <CONTAINER ID> # remove a container
```

Appendix B

The Parameter of the FXT tasks

B.1 CalDB Tools in v1.30

The official FXT CalDB tools distributed with `fxtsoftv1.30` are `fxtbkggen`, `fxtexpogen`, `fxtpical`, `fxtrmfgen`, `fxtarfgen`, `fxtpsf`, `fxte2pi`, `fxtplotgrade`, `fxtpsfgen`, `fxteefgen`, and `fxtpsfmap`.

Compared with the previous public release, the main user-visible updates are summarized below:

- `fxtpsfmap` is a new officially released task. It generates a PSF-size map from an exposure map for a given energy and enclosed energy fraction (EEF), and it can read the filter information directly from the input FITS header when needed.
- `fxtarfgen` has been substantially revised. The current implementation supports dual-beta PSF/EEF correction for near on-axis point sources, subpixel region-mask handling, optional circle/ellipse/rectangle regions, and more complete FITS HISTORY records.
- `fxtpical` now has a clearer calibration flow for gain, CTI, and time-dependent gain drift corrections before writing the PI column.
- `fxtpsfgen` and `fxteefgen` now use the same telescope-definition handling as `fxtpsfmap`, improving the consistency of off-axis calculations.
- `fxtexpogen` documents optional effective-area scaling and the clustering parameters that can accelerate exposure-map generation for long observations.
- `fxtplotgrade` now supports `show=yes/no` so the plot can be displayed interactively or generated in batch mode.

Several additional scripts remain in the source tree for testing and method evaluation, but they are not part of the official v1.30 release and are not described in this guide.

B.1.1 fxtcheck

This task mainly checks the input files of the FXT and the environment of HEADAS and CALDB.

PARAMETER

The parameters of this task are listed below:

- indir [file name]
  Input directory.
- outdir [file name]
  Output directory.
- clobber [boolean]
  If *clobber=yes* and outfile=filename, the file with the same name will be overwritten if it exists (DEFAULT=yes).
- history [boolean]
  If *history=yes* the parameter values and other information are written in HISTORY keywords (DEFAULT=yes).
- chatter[integer]
  Chatter Level (min=0, max=5) (DEFAULT=2). It is used to control the output message.

B.1.2 fxtcoord

This tool updates the SKY coordinate of each event using the latest calibration file and recalculates the GTI of events based on the event file and attitude file.

PARAMETER

The parameters of this task are listed below:

- evtfile [file name]
  Name of the input Event FITS file.
- outfile [file name]
  Name of the output Event FITS File or DEFAULT for standard name.
- attfile [file name]
  Name of the attitude Event FITS file.
- clobber [boolean]
  If *clobber=yes* and outfile=filename, the file with the same name will be overwritten if it exists (DEFAULT=yes).

- history [boolean]
  If *history=yes* the parameter values and other information are written in HISTORY keywords (DEFAULT=yes).

- chatter[integer]
  Chatter Level (min=0, max=5) (DEFAULT=2). It is used to control the output message.

B.1.3 fxttimecorr

This tools provides a method to correct the time stamp for Timing mode.

PARAMETER

The parameters of this task are listed below:

- evtfile [file name]
  Name of the input Event FITS file.

- outfile [file name]
  Name of the output Event FITS File or DEFAULT for standard name.

- attfile [file name]
  Name of the attitude Event FITS file.

- teldef [file name]
  Name of FXT teldef calibration file or CALDB.

- seed [integer]
  Input seed for random number generator.

- ra [real]
  Source RA (degrees).

- dec [real]
  Source DEC (degrees).

- readout [real]
  readout time interval (in microsecond).

- resolution [integer]
  half position resolution (in pixel)

- waitrd [integer]
  waiting time (in pixel).

- clobber [boolean]
  If *clobber=yes* and outfile=filename, the file with the same name will be overwritten if it exists (DEFAULT=yes).

- history [boolean]
  If *history=yes* the parameter values and other information are written in HISTORY keywords (DEFAULT=yes).

- chatter[integer]
  Chatter Level (min=0, max=5) (DEFAULT=2). It is used to control the output message.

B.1.4 fxtpical

This task calculates the Pulse Invariant (PI) values for FXT event files in FF, PW, and TM modes. It uses the CALDB gain and E2PI calibration files together with event position and observation-time information, applies the average positional gain variation (CTI) and the time-dependent gain correction, and then creates or updates the PI column in the output event file.

PARAMETER

The parameters of this task are listed below:

- evtfile [file name]
  Name of the input Event FITS file.

- outfile [file name]
  Name of the output Event FITS file.

- clobber [boolean]
  If *clobber=yes* and outfile=filename, the file with the same name will be overwritten if it exists (DEFAULT=yes).

B.1.5 fxtparticleidentify

This task can identify particle-induced background (tracks, showers and instrumental effect event), and removal these hits. It can be used for the FF and PW modes, as well as TM mode. However, for FF and PW modes, the parameters xlength and ylength should be to 11 and 11, respectively, and for TM, they should be to 35 and 35.

PARAMETER

The parameters of this task are listed below:

- evtfile [file name]
  Name of the input Event FITS file.

- outfile [file name]
  Name of the output Event FITS file.

- searchwidth [integer]
  Search width of particle cluster

- bkgnum [integer]
  Background threshold

- efrac [real]
  Mininum fraction of High Energy particles (¿12keV) to all

- xlength [integer]
  Mininum length along X caused by instrumental effect

- ylength [integer]
  Mininum length along Y caused by instrumental effect

- particlefile [file name]
  Name of the output particle-induced Event FITS File

- clobber [boolean]
  If *clobber=yes* and outfile=filename, the file with the same name will be overwritten if it exists (DEFAULT=yes).

- history [boolean]
  If *history=yes* the parameter values and other information are written in HISTORY keywords (DEFAULT=yes).

- chatter[integer]
  Chatter Level (min=0, max=5) (DEFAULT=2). It is used to control the output message.

B.1.6 fxtbadpix

This task mainly flags the bad pixels to event file and the bad pixels are from CALDB

PARAMETER

The parameters of this task are listed below:

- evtfile [file name]
  Name of the input Event FITS file.
- outfile [file name]
  Name of the output Event FITS file or NONE to overwrite the input file.
- groundbpfile [file name]
  Name of the input BADPIX file or CALDB or NONE for none.
- boardbpfile [file name]
  Name of the input on-board BADPIX file or CALDB or NONE for none.
- userbpfile [file name]
  Name of the user BADPIX file or NONE for none.
- outbpfile [file name]
  Name of the output Bad Pixel file or DEFAULT to use standard name or NONE for none.
- clobber [boolean]
  If *clobber=yes* and outfile=filename, the file with the same name will be overwritten if it exists (DEFAULT=yes).
- history [boolean]
  If *history=yes* the parameter values and other information are written in HISTORY keywords (DEFAULT=yes).
- chatter[integer]
  Chatter Level (min=0, max=5) (DEFAULT=2). It is used to control the output message.

B.1.7 fxthotpix

This task mainly search the hot pixels from event file.

PARAMETER

The parameters of this task are listed below:

- evtfile [file name]
  Name of the input Event FITS file.
- outfile [file name]
  Name of the output Event FITS file or NONE to overwrite the input file.

- bpfile [file name]
  Name of the input Bad Pixel file.

- outhpfile [file name]
  "Name of the output Hot Pixel file or NONE for none or DEFAULT for standard name.

- cellsize [integer]
  Search cell size in pixels.

- binsize [real]
  Bin size (seconds) for count image generation.

- clobber [boolean]
  If *clobber=yes* and outfile=filename, the file with the same name will be overwritten if it exists (DEFAULT=yes).

- history [boolean]
  If *history=yes* the parameter values and other information are written in HISTORY keywords (DEFAULT=yes).

- chatter[integer]
  Chatter Level (min=0, max=5) (DEFAULT=2). It is used to control the output message.

B.1.8 fxtgtigen

This task mainly generate the Good Time Intervals file (GTI).

PARAMETER

The parameters of this task are listed below:

- hkfile [file name]
  Name of the input MKF FITS file.

- outfile [file name]
  Name of the output GTI Event FITS file or DEFAULT for standard name.

- module [string]
  Name of the module of FXT (fxta/fxtb).

- usrgtifile [file name]
  Name of the user input GTI file (FITS format) or NONE for none.

- evtfile [file name]
  Name of the event file (its GTI extension will be used).

- expr [string]
  Selection Expression for attitude and instrument HK or DEFAULT for standard (from CALDB).

- hkrangefile [file name]
  Name of the input fits HKRANGE file or CALDB.

- prefr [real]
  Pre-Time Interval factor [0,1].

- postfr [real]
  Post-Time Interval factor [0,1]

- clobber [boolean]
  If *clobber=yes* and outfile=filename, the file with the same name will be overwritten if it exists (DEFAULT=yes).

- history [boolean]
  If *history=yes* the parameter values and other information are written in HISTORY keywords (DEFAULT=yes).

- chatter[integer]
  Chatter Level (min=0, max=5) (DEFAULT=2). It is used to control the output message.

B.1.9 fxtgrade

This task mainly reconstructs the split events and assigns event grade to each photon.

PARAMETER

The parameters of this task are listed below:

- evtfile [file name]
  Name of the input Calibration Event FITS file.

- outfile [file name]
  Name of the output Event FITS File or DEFAULT for standard name.

- pithresh [integer]
  Use only events with PI of pixel greater than the threshold.

- covfile [file name]
  Name of the input E2PI configure file or CALDB (default=CALDB).

- par, par1, par2 [Real]
  Fixed values, for software acceleration.

- clobber [boolean]
  If *clobber=yes* and outfile=filename, the file with the same name will be overwritten if it exists (DEFAULT=yes).

- history [boolean]
  If *history=yes* the parameter values and other information are written in HISTORY keywords (DEFAULT=yes).

- chatter[integer]
  Chatter Level (min=0, max=5) (DEFAULT=2). It is used to control the output message.

B.1.10 fxtexpogen

This task generates exposure maps for FXT data, taking into account bad pixels and columns, attitude variations, and telescope vignetting. The task produces the standard exposure map together with a companion file with suffix `-without_vign`. When `area_scale=yes`, the map is multiplied by the on-axis effective area so that the output is scaled from seconds to cm^2 s. The current version also provides clustering parameters to speed up the exposure-map calculation for long observations.

PARAMETER

The parameters of this task are listed below:

- evtfile [file name]
  Name of the input Event FITS file.

- mkffile [file name]
  Name of the input MKF FITS file.

- outfile [file name]
  Name of the output exposure map FITS file.

- energy=1.5 [float]
  The average energy (in keV) for generating the exposure map.

- area_scale=no [string]
  Whether to apply scaling to the effective area (yes/no).

- edgecovermask=0 [integer]
  Mask edge occluded range for merging.

- clobber=yes [string]
  Overwrite existing output file if it exists (yes/no).

- use_clustering=yes [string]
  Use clustering to accelerate the calculation by grouping similar attitudes (yes/no).

- ra_threshold=2.0 [float]
  RA/DEC angular distance threshold for clustering in arcseconds. This parameter is only used when *use_clustering=yes*.

- pa_threshold=0.01 [float]
  Roll-angle difference threshold for clustering in degrees. This parameter is only used when *use_clustering=yes*.

B.1.11 fxtplotgrade

This task plots the grade fraction of an input event FITS file as a function of PI. By default the PNG file is saved and the plot window is displayed; `show=no` can be used for non-interactive batch processing.

PARAMETER

The parameters of this task are listed below:

- evtfile [file name]
  Name of the input Event FITS file.

- outfile=grade_fraction.png [file name]
  Name of the output PNG File.

- plotbins=100 [integer]
  The bins for grade fraction plot, default value is set to 100.

- show=yes [boolean]
  Display the plot window after saving the PNG file. Use *show=no* for batch use.

- clobber [boolean]
  If *clobber=yes* and outfile=filename, the file with the same name will be overwritten if it exists (DEFAULT=yes).

B.1.12 fxtarfgen

This task generates an OGIP-style ARF for an extracted FXT spectrum by using the input spectrum, the exposure map, and the CALDB response files. For point sources with `psfcor=1`, the current implementation applies a PSF/EEF correction: a dual-beta PSF model is used for sources within 3 arcmin of the optical axis, while pre-computed EEF curves are interpolated at larger off-axis angles. Region handling has also been updated to support subpixel masks and common circle, ellipse, and rectangle regions.

PARAMETER

The parameters of this task are listed below:

- specfile [file name]
  Name of the input spectrum FITS file.
- expfile [file name]
  Name of the input exposure map FITS file. In normal spectral analysis this should be the map without vignetting correction, i.e. the file with suffix `-without_vign`.
- outfile [file name]
  Name of the output arf FITS File.
- extend [0 or 1]
  0 for a point-like source and 1 for an extended source (DEFAULT=0). When *extend=1*, no point-source PSF correction is applied.
- srcx [integer]
  Sky X position of the point source. If *srcx=-1*, the task uses the source position inferred from the input spectrum region information or WMAP.
- srcy [integer]
  Sky Y position of the point source. If *srcy=-1*, the task uses the source position inferred from the input spectrum region information or WMAP.
- psfcor [0 or 1]
  Apply PSF correction for point-like sources. This parameter is ignored when *extend=1*.
- clobber [boolean]
  If *clobber=yes* and outfile=filename, the file with the same name will be overwritten if it exists (DEFAULT=yes).

Some advanced region handling depends on the optional Python package `regions`. If it is not installed, `fxtarfgen` falls back to an alternate region-mask path.

B.1.13 fxtrmfgen

This task extracts the RMF corresponding to the input spectrum from the FXT CALDB. The selected RMF depends on the detector, observing mode, and the grade selection recorded in the spectrum header.

PARAMETER

The parameters of this task are listed below:

- specfile [file name]
  Name of the input spectrum FITS file.
- outfile [file name]
  Name of the output rmf FITS File.
- clobber [boolean]
  If *clobber=yes* and outfile=filename, the file with the same name will be overwritten if it exists (DEFAULT=yes).

B.1.14 fxtootest

This task gives a uniform Y (along read-out direction) from the screened event file to a new event file and this new event file can be used to estimate the spectrum of OoT events.

PARAMETER

The parameters of this task are listed below:

- evtfile [file name]
  Name of the input Event FITS file.
- outfile [file name]
  Name of the output Event FITS File or DEFAULT for standard name.
- attfile [file name]
  Name of the attitude Event FITS file.
- seed [integer]
  Input seed for random number generator.
- teldef [file name]
  Name of teldef calibration file or CALDB.
- clobber [boolean]
  If *clobber=yes* and outfile=filename, the file with the same name will be overwritten if it exists (DEFAULT=yes).

- history [boolean]
  If *history=yes* the parameter values and other information are written in HISTORY keywords (DEFAULT=yes).

- chatter[integer]
  Chatter Level (min=0, max=5) (DEFAULT=2). It is used to control the output message.

B.1.15 fxte2pi

This task prints the PI channel values corresponding to one or more input energy values in keV. The command-line interface supports both space-separated positional inputs such as `fxte2pi 0.5 1.0 6.4` and comma-separated `energy=` lists such as `fxte2pi energy=0.5,1.0,6.4`.

- energy [float]
  One or more input energy values in keV. They may be passed either as space-separated positional arguments or as a comma-separated `energy=` list.

Examples:

```
fxte2pi 2.0 10.0
fxte2pi energy=2.0,10.0
```

B.1.16 fxtpsf

This task copies the calibration files `fxt_psf.cod` and `fxt_psf.dat` from the CALDB into the current working directory. These files are mainly used by external PSF-fitting workflows such as the pile-up diagnostic procedure described in this guide.

B.1.17 fxtbary

fxtbary applies barycenter correction to FXT event data files. The task reads EVENTS and GTI extensions in event files, and overwrites the TIME column for EVENTS extension (START and STOP columns for GTI extension). While the keywords TSTART, TSTOP, TIMESYS and TIMEREF for both extension are changed.

PARAMETER

The parameters of this task are listed below:

- evtfile [file name]
  Name of the input Event FITS file.

- outfile [file name]
  Name of the output Event FITS File or DEFAULT for standard name.

- orbitfile [file name]
  The orbital filename
- ephemfile [file name]
  The JPL Ephemeris filename
- ra [string] The right ascension of source
- dec [string] The declination of source.

B.1.18 fxtpsfgen

fxtpsfgen generates a Point Spread Function (PSF) image for FXT detectors. The task reads the telescope definition from the CALDB, converts the input RAW detector position to the corresponding off-axis angle, and interpolates the calibrated PSF image at the requested detector, filter, position, and energy. The resulting PSF image can be used for further analysis or for visualizing the FXT response.

PARAMETER

The parameters of this task are listed below:

- det [string]
  Detector name (fxta/fxtb).
- filter [string]
  Filter type (thin/medium/hole).
- rawX [float]
  The rawX position of PSF center.
- rawY [float]
  The rawY position of PSF center.
- energy [float]
  The energy of PSF in eV.
- outfile [file name]
  The name of the output PSF image file.

B.1.19 fxteefgen

fxteefgen generates Encircled Energy Fraction (EEF) data for FXT detectors. The task reads the telescope definition from the CALDB, converts the input RAW detector position to off-axis angle, and writes an EEF table as a function of radius for the requested detector, filter, position, and energy. The output file provides the enclosed energy fraction at different radii from the source position.

PARAMETER

The parameters of this task are listed below:

- det [string]
  Detector name (fxta/fxtb).
- filter [string]
  Filter type (thin/medium/hole).
- rawX [float]
  The rawX position of EEF center.
- rawY [float]
  The rawY position of EEF center.
- energy [float]
  The energy of EEF in eV.
- outfile [file name]
  The name of the output EEF file.

B.1.20 fxtpsfmap

fxtpsfmap generates a PSF-size map from an exposure map for a given energy and enclosed energy fraction (EEF). The output FITS image has the same size and WCS as the input exposure map, and each pixel value gives the PSF radius in arcseconds at that detector position. If the filter is not supplied, the task attempts to read it from the `FILTER` keyword of the input FITS header.

PARAMETER

The parameters of this task are listed below:

- expmap [file name]
  Input exposure map FITS file.
- outfile [file name]
  Output PSF map FITS file.
- energy [float]
  Effective energy for the PSF calculation, in keV.
- eef [float]
  Requested enclosed energy fraction. The value must be greater than 0 and less than or equal to 1.

- filt [integer]
  Filter ID (0=open, 1=thin, 2=medium, 3=hole). If omitted, the task reads the filter from the input exposure-map header.

- clobber [boolean]
  If *clobber=yes* and outfile=filename, the file with the same name will be overwritten if it exists (DEFAULT=yes).

B.1.21 fxtbkggen

The fxtbkggen tool is used to generate the instrumental background spectrum for the pnCCD image area (IMG). Please refer to Ref. [9] for more details. The estimate relies on the spectrum of cleaned events on the frame-store area (FSA) obtained by running *fxtchain* with the parameter *datatype=fsaevt*. In the current release, this workflow is intended for cleaned FF-mode FSA spectra with Grade 0–12, DETX$\in$ $[3, 382]$, DETY$\in$ $[3, 382]$, and thin or medium filter observations, while excluding bad or hot pixels and columns. The predicted background spectrum of the IMG also corresponds to events with grades 0–12. Considering the excluded bad/hot pixels and columns, the derived IMG spectrum covers a detector region of approximately 380×375 pixels for FXTA, and 380×380 pixels for FXTB. The "backscal" keyword in the header of the output instrumental background spectrum denotes the fraction of the effective sky region.

PARAMETER

The parameters of this task are listed below:

- infile [string]
  input frame-store area spectrum file.

- outfile [string]
  output instrumental background spectrum file.

Bibliography

[1] Li, C.K., Jia, S.M., Song, L.M., et al., 2025, RDTM

[2] Zhao H.S., Li, C.K., Wang, J., et al., 2025, RDTM

[3] Zhao, X., Xu, J., Cui, W., et al., 2024, PASP

[4] Zhao, X., Cui, W., Wang, H., et al., 2025, RAA

[5] Zhao, X., Ban, H., Cai, H., et al., 2025, EA

[6] Chen, Y., Wang, J., Yang, Y., et al., 2025, RDTM

[7] Wang, J., Zhao, X., Yang, X., et al., 2025, RDTM

[8] Wang, H., Cui, W., Zhao, X., et al., 2025, RDTM

[9] Zhang, J., Chen, Y., Jia, S., et al. 2025, Research in Astronomy and Astrophysics, 25, 11, 115019. doi:10.1088/1674-4527/ae05f8