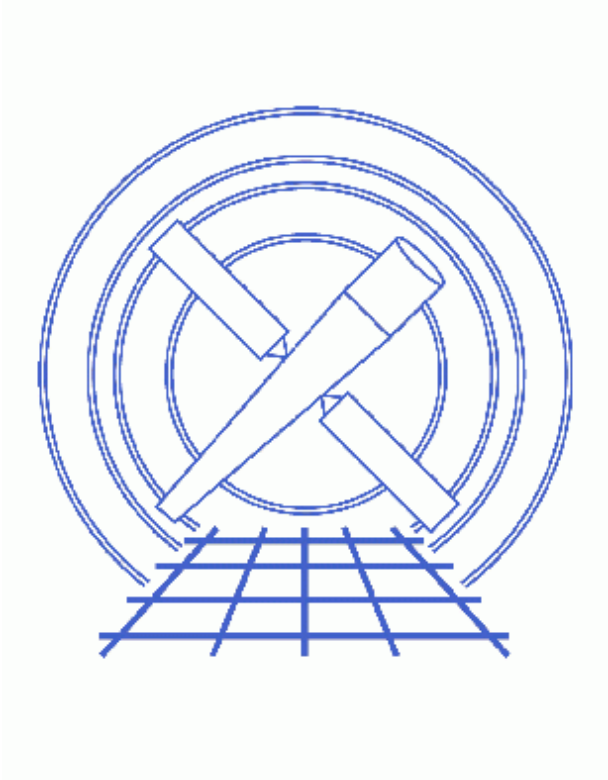
Compute Multiple Chip ACIS Exposure Map and Fluxed Image Step−by−Step



CIAO 3.4 Science Threads

Table of Contents

Compute Multiple Chip ACIS Exposure Map and Fluxed Image Step−by−Step

CIAO 3.4 Science Threads

Overview

Last Update: 6 Mar 2007 − added ACIS dead area correction section and example of setting the `pbkfile` and `dafile` parameters

Synopsis:

`mkexpmap` generates an exposure map which may be used to convert a counts image of a source to an image in flux units. The computed exposure map is essentially an image of the effective area at each sky position, accounting for the effects of dither motion which are especially important near the edges of the detector.

Purpose:

To build an exposure map for an entire ACIS chip array, create a fluxed image, and find an approximation for the source flux.

Read this thread if:

you are working with an ACIS observation and would like to and would like to create an exposure map for ***all*** of the chips. If only one chip is being used, follow the Compute an ACIS Exposure Map (Single Chip) and Build Fluxed Image thread instead.

Calibration Updates:

- ***CALDB v3.0.0*** ***(15 Dec 2004):*** The ACIS QE degradation model has been enhanced to account for spatial variations in the contamination on the ACIS optical blocking filters. The contamination is now expressed as a function of time, energy, and ACIS chip coordinate. The ACIS QE Degradation why topic contains further details.
- ***CALDB v2.26*** ***(2 Feb 2004):*** The ACIS contamination files were added to the CALDB. The Correcting Responses for ACIS Contamination thread describes how this is taken into account in the data analysis.

Related Links:

- Analysis Guide: Extended Sources
- An Introduction to Exposure Maps (PS, 12pp): a general discussion of exposure maps.
- Calculating Spectral Weights thread: if you are interested in apply spectral weights to your data at the Calculate the Instrument Map step, run this thread first.

Proceed to the HTML or hardcopy (PDF: A4 | letter) version of the thread.

Get Started

Sample ObsID used: 1838 (ACIS−S, G21.5−09)

File types needed: evt2; asol1; msk1

In this thread, we assume that all relevant files are in the same working directory.

To create an exposure map, we will use an aspect histogram file − which contains information on the aspect motion during the observation − and an instrument map − which is essentially the product of the detector quantum efficiency and the mirror effective area projected onto the detector surface.

Please ensure that you have set up ardlib to use the bad pixel file for your observation before following this thread.

In this example, the energy range was restricted from 0.3 keV to 10 keV:

```
unix% dmcopy "acisf01838N001_evt2.fits[energy=300:10000]" acis_1838_evt2.fits
```

The ACIS dead area correction

There is a fractional area loss per unit time due to cosmic ray flux incident on the ACIS detector. Calibration to account for this ACIS "dead area" was included in CALDB 3.3.0 on 15 December 2006. The correction is non−zero for the 8 front−illuminated ACIS chips; the effect is not detectable for the BI chips, so the nominal calibration value is 0.0. The resulting chipy−dependent reduction in the EA will be approximately 2.2% at the readout, and 4.0% at the top of the chip. Refer to the ACIS Dead Area Correction why topic for technical details.

In CIAO 3.4, the application of the dead area correction is ***turned off*** by default. However, users may opt to include it in the analysis by setting the `pbkfile` and `dafile` parameters in the `mkinstmap` step. Refer to the `mkinstmap` help file for details on these parameters.

Create An Image

First, we need to create the image which will ultimately be normalized by the exposure map. Here we decided to block the image by a factor of 8:

```
unix% dmcopy "acis_1838_evt2.fits[bin x=8,y=8]" 1838_img8.fits
```

This creates an image that is 1024 x 1024 (given that a full resolution ACIS image is 8192 x 8192); this information is used again in the Calculate the Exposure Map step. ***You may choose to use different binning***, but make sure that you change the `xygrid` appropriately in that step.

Compute Exposure Map

1. Check Which Chips Are On

The list of chips used in the observation is stored in the DETNAM keyword of the event file:

```
unix% dmkeypar acis_1838_evt2.fits DETNAM echo+
ACIS−012367
```

A description of the layout of the ACIS focal plane can be found in The Chandra Proposers' Observatory Guide.

2. Find Peak Energy Of Source

We selected a region around the central source, using ds9, and saved it as obj.reg:

```
unix% more obj.reg
# Region file format: CIAO version 1.0
circle(4072,4249,35)
```

We can use this file to extract a spectrum of the object in energy space and find the peak energy.

First, we use the CIAO tool dmextract to create a histogram of count−rate as a function of energy. Since we are not binning on pi or pha, we set opt=generic, and we use a bin size of 20 eV to improve the signal to noise:

```
unix% punlearn dmextract
unix% pset dmextract infile="acis_1838_evt2.fits[sky=region(obj.reg)][bin energy=300:10000:20]"
unix% pset dmextract outfile=obj_1838_histogram_energy.fits
unix% pset dmextract opt=generic
unix% dmextract
Input event file  (acis_1838_evt2.fits[sky=region(obj.reg)][bin energy=300:10000:20]):
Enter output file name (obj_1838_histogram_energy.fits):
```

You can check the dmextract parameter file that was used with plist dmextract.

The resulting histogram can be plotted using *Prism* or *ChIPS*. Choosing the energy and count_rate columns produces a plot like this. If you used *ChIPS* to plot the histogram, then the PICKPOINTS command can be used to find the energy corresponding to the peak in the spectrum. An alternative method involves the use of dmstat (to find the maximum count rate), followed by dmlist (to locate the corresponding energy):

```
unix% dmstat "obj_1838_histogram_energy.fits[cols count_rate]"
COUNT_RATE[count/s]
    min:        0             @:        6
    max:        0.017951569397        @:        71
   mean:        0.0037646188012
  sigma:        0.0040689885567
    sum:        1.8258401186
   good:        485
   null:        0

unix% dmlist "obj_1838_histogram_energy.fits[count_rate > 0.015][cols energy,count_rate]" data,clear
#  ENERGY               COUNT_RATE
               1710.0      0.0179515693970
               1750.0     0.01642377625683
               1770.0     0.01616914406680
               1830.0     0.01667840844686
```

```
               1950.0     0.01591451187677
```

For this dataset, the peak of the measured spectrum is ~1.7 keV.

3. Compute the Aspect Histograms

With the aspect solution file we can create a binned histogram for each chip, detailing the aspect history of the observation:

```
unix% punlearn asphist
unix% pset asphist infile=pcadf084244404N001_asol1.fits
unix% pset asphist mode=h
unix% foreach d ( 0 1 2 3 6 7 )
foreach? asphist outfile=asphist_${d}.fits evtfile="acis_1838_evt2.fits[ccd_id=$d]"
foreach? end
```

- In some cases there will be more than one asol1.fits file for an observation. ***All*** the files must be input to the `infile` parameter, either as a list or as a stack (see ahelp stack for more information).
- The `dtffile` parameter is left blank (i.e. "`none`") since the dead−time correction is read from the input event file for ACIS data.

Plotting the 2−dimensional `POS_OFFSET` vector − composed of the `X_OFFSET` and `Y_OFFSET` columns − in `asphist_7.fits` shows the dither pattern used during the observation  .

4. Calculate the Instrument Maps

Since the mirror effective area is used to create the instrument map, and that area is energy dependent, it is necessary to decide at what energy to perform the calculation (or whether to use a spectrum as weights). See An Introduction to Exposure Maps (PS, 12pp) and the Calculating Spectral Weights thread for details on weighting a map by a spectrum. In this example we are going to assume a monoenergetic distribution of source photons of 1.7 keV (`monoenergy parameter`).

Note that it is not necessary for the instrument map to be congruent with the exposure map; the instrument map should describe the chip with full resolution. We will use the default values for the detector region (`pixelgrid parameter`), creating a 1024 x 1024 pixel image that covers the whole chip.

```
unix% punlearn mkinstmap
unix% pset mkinstmap pixelgrid="1:1024:#1024,1:1024:#1024"
unix% pset mkinstmap maskfile=acisf01838_000N001_msk1.fits
unix% pset mkinstmap spectrumfile=NONE monoenergy=1.7
unix% pset mkinstmap mode=h
```

If you wish to include the ACIS dead area correction (not applied in this thread), set the `pbkfile` and `dafile` parameters in `mkinstmap`:

```
unix% pset mkinstmap pbkfile=acisf084245776N001_pbk0.fits dafile=CALDB
```

Now run the tool once for each chip:

```
unix% foreach d ( 0 1 2 3 6 7 )
foreach? mkinstmap obsfile="asphist_${d}.fits[asphist]" detsubsys=ACIS−${d} \
                  outfile=instmap_1.7kev_${d}.fits
foreach? end
```

Note that the monoenergy parameter value will probably be different for your dataset (and therefore also the outfile specification).

Including the maskfile parameter is particularly important if you are interested in having an accurate exposure map at the very edge of a CCD, subarray or window. For more information, see the dictionary entry on mask files.

You can check the parameter file that was used with plist mkinstmap.

---

5. Calculate the Exposure Maps

Now we use mkexpmap and the aspect information stored in the histogram to project the instrument map onto the sky. We need to set the xygrid parameter to produce an exposure map that is the same size as the image created from the event list. The get_sky_limits script (part of the CIAO Scripts distribution) can be used to easily calculate this information from the existing image:

```
unix% get_sky_limits 1838_img8.fits verbose="1"
Checking binning of image: 1838_img8.fits
  Image has 1024 x 1024 pixels
  Lower left (0.5,0.5) corner is x,y= 0.5, 0.5
  Upper right (1024.5,1024.5) corner is x,y= 8192.5, 8192.5
  DM filter is:
    x=0.5:8192.5:#1024,y=0.5:8192.5:#1024
  mkexpmap xygrid value is:
    0.5:8192.5:#1024,0.5:8192.5:#1024
```

You can then set the xygrid parameter using the information provided by the script, either manually or via

```
unix% pset mkexpmap xygrid=")get_sky_limits.xygrid"
```

(if the latter, do not run get_sky_limits again until after running mkexmap).

Note: If you are computing a low−resolution exposure map and speed is more important than accuracy, set useavgaspect=yes. In doing so, only the average aspect pointing will be used to derive the exposure map; otherwise all points in the aspect histogram will be used. The time required to compute the exposure map is proportional to the number of bins in the aspect histogram; if the aspect histogram contains 100 bins, then the use of this option reduces the run time by a factor of 100, approximately (you may also want to set verbose to 2, since this causes mkexpmap to output percentage−completed information). Using the full aspect solution will help accurately account for chip edges, bad pixels, etc.

```
unix% punlearn mkexpmap
unix% pset mkexpmap normalize=no
unix% pset mkexpmap xygrid="0.5:8192.5:#1024,0.5:8192.5:#1024"
unix% pset mkexpmap useavgaspect=no
unix% pset mkexpmap mode=h
unix% foreach d ( 0 1 2 3 6 7 )
foreach? mkexpmap instmapfile=instmap_1.7kev_${d}.fits \
                  outfile=expmap_1.7kev_${d}.fits \
                  asphistfile=asphist_${d}.fits
foreach? end
Exposure map limits: 0.000000e+00, 3.598171e+06
Writing exposure map to expmap_1.7kev_0.fits
Exposure map limits: 0.000000e+00, 3.617616e+06
Writing exposure map to expmap_1.7kev_1.fits
Exposure map limits: 0.000000e+00, 4.168458e+06
Writing exposure map to expmap_1.7kev_2.fits
Exposure map limits: 0.000000e+00, 4.179710e+06
```

```
Writing exposure map to expmap_1.7kev_3.fits
Exposure map limits: 0.000000e+00, 4.974079e+06
Writing exposure map to expmap_1.7kev_6.fits
Exposure map limits: 0.000000e+00, 5.361629e+06
Writing exposure map to expmap_1.7kev_7.fits
```

You can check the mkexpmap parameter file that was used with plist mkexpmap.

Since we set the `normalize` parameter = ***no***, the exposure map has units of [`cm^2*s*counts/photon`]. This allows us to simply divide the image by the exposure map to derive an image in units of flux [`photons/cm^2/s/pixel`]. If the setting had been left as ***yes*** (the default), the units of the exposure map would be [`cm^2*counts/photon`]. Please see the help file for `mkexpmap` for more details on this.

6. Combine the Exposure Maps

The individual exposure maps are combined into a single, binned exposure map with the CIAO tool `dmregrid`. First we need a list of files to combine:

```
unix% ls expmap_1.7kev*.fits > expmap_1.7kev.lis
unix% cat expmap_1.7kev.lis
expmap_1.7kev_0.fits
expmap_1.7kev_1.fits
expmap_1.7kev_2.fits
expmap_1.7kev_3.fits
expmap_1.7kev_6.fits
expmap_1.7kev_7.fits
```

Now we can use this list by passing it into dmregrid as a `stack`:

```
unix% dmregrid infile=@expmap_1.7kev.lis outfile=expmap_1.7kev.fits \
               bin="1:1024:1,1:1024:1" rotangle=0 npts=1 \
               xoffset=0 yoffset=0 rotxcenter=0 rotycenter=0
```

The `bin` parameter value will need to be changed if you used a different binning specification when creating the exposure maps. The exposure map can be displayed in ds9  .

You can check the parameter file that was used with plist dmregrid.

Normalize the Image by the Exposure Map

The exposure map is in units of [`cm^2*s*counts/photon`] since it was created by projecting the instrument map (in [`cm^2*counts/photon`]) onto the tangent plane of the observation. To create an image in units of [`photon/cm^2/s/pixel`], we simply need to divide by the exposure map. This can be performed in one step with `dmimgcalc`:

```
unix% punlearn dmimgcalc
unix% pset dmimgcalc infile=1838_img8.fits
unix% pset dmimgcalc infile2=expmap_1.7kev.fits
unix% pset dmimgcalc outfile=1838_img8_norm.fits
unix% pset dmimgcalc operation=div
unix% dmimgcalc
Input file #1 (1838_img8.fits):
Input file #2 (expmap_1.7kev.fits):
```

```
output file  (1838_img8_norm.fits):
arithmetic operation (div):
warning: CONTENT has 1 different values.
warning: DETNAM has different value...Merged...
```

The messages are related to how the tool merges the header information in the input files. The `merging_rules ahelp file` explains the rules and how they affect the output file header.

The units of `1838_img8_norm.fits` are `photon/cm^2/s/pixel`.

You can check the parameter file that was used with plist dmimgcalc.

Alternatively, one may use `dmimgthresh` for a cleaner final product. The strongly variable exposure near the edge of a dithered field may produce "hot" pixels when divided into an image. While technically proper, these hot pixels can be an eyesore, drawing attention to a noisy, uninteresting portion of the image. The `dmimgthresh` tool may be used to make a "threshold cut" before dividing the image by the exposure map, thus removing the hot pixels:

```
unix% punlearn dmimgthresh
unix% pset dmimgthresh infile=1838_img8.fits
unix% pset dmimgthresh outfile=1838_img8_clean.fits
unix% pset dmimgthresh expfile=expmap_1.7kev.fits
unix% pset dmimgthresh cut=1.5%
unix% pset dmimgthresh value=0.0
unix% dmimgthresh
Input dataset/block specification (1838_img8.fits):
Output dataset/block specification (1838_img8_clean.fits):
```

Here we set our threshold at `1.5%` of the maximum value of the exposure map. All image pixels with values of exposure ***less than*** this value will be set to `0.0` in the output file. The next step would be to divide `1838_img8_clean.fits` by the exposure map for a final, fluxed image. You may want to adjust these values for your own observation.

You can check the parameter file that was used with plist dmimgthresh.

Calculate the Source Flux

Since the units of the fluxed image are [`photon/cm^2/s/pixel`], adding up the pixel values around a source results in the source flux in [`photon/cm^2/s`]. Note that this flux is an *approximation* − as discussed in An Introduction to Exposure Maps (PS, 12pp) − since a spectral shape was assumed when using mkinstmap (in this example, a monochromatic source).

Using the source region "`flux.reg`":

```
unix% more flux.reg
# Region file format: CIAO version 1.0
circle(4070.5,4250.5,112)
```

the flux can be calculated with either `dmstat`:

```
unix%  dmstat infile="1838_img8_norm.fits[sky=region(flux.reg)]" centroid=no
1838_img8_norm.fits
    min:        0            @:        ( 4060.5 4140.5 )
```

```
    max:        0.00025930177071            @:        ( 4068.5 4252.5 )
   mean:        7.3050461797e-06
  sigma:        1.8384800798e-05
        sum:        0.0045145185391
   good:        618
   null:        223
```

or dmextract:

```
unix% dmextract infile="1838_img8_norm.fits[bin sky=@flux.reg]" outfile="source_flux.fits"

unix% dmlist source_flux.fits"[cols COUNTS,ERR_COUNTS]" data

--------------------------------------------------------------------------------
Data for Table Block HISTOGRAM
--------------------------------------------------------------------------------

ROW    COUNTS               ERR_COUNTS

     1     0.00451451853905     0.06719016698189
```

Since the input to dmextract was a fluxed image, not an event list, the COUNTS column actually reports the total flux (in [photon/cm^2/s]) for the source region. While slightly more involved, the dmextract method can be used on multiple sources in a single command, and the results are conveniently stored in a table.

Analysis Caveats

Users should be cautious about analyzing the data for sources near the edges of the ACIS CCDs.

1. For X−rays passing through the mirrors, the very bottom of each CCD is obscured by the frame store. As a result, some of the events in rows with CHIPY <= 8 are not detected. (The set of rows affected varies from CCD to CCD.) Since the CIAO tools do not compensate for this effect, the ARFs and exposure maps for sources in these regions may be inaccurate.
2. For sources within about thirty−two pixels of any edge of a CCD, the source may be dithered off the CCD during part of an observation. The aspect histogram, which is used to create ARFs and exposure maps, is designed to compensate for this effect.
3. A contaminant has accumulated on the optical−blocking filters of the ACIS detectors, as described in the ACIS QE Degradation why topic. Since there is a gradient in the temperature across the filters (the edges are colder), there is a gradient in the amount of material on the filters. (The contaminant is thicker at the edges.) Within about 100 pixels of the outer edges of the ACIS−I and ACIS−S arrays, the gradient is relatively steep. Therefore, the effective low−energy (' 1 keV) detection efficiency may vary within the dither pattern in this region. The ARF and instrument map tools are designed to read a calibration file which describes this spatial dependence.

```
Parameters for /home/username/cxcds_param/dmextract.par


#-------------------------------------------------------------------
#
# DMEXTRACT -- extract columns or counts from an event list
```

```
#
#------------------------------------------------------------------------
        infile = acis_1838_evt2.fits[sky=region(obj.reg)][bin energy=300:10000:20] Input event file
       outfile = obj_1838_histogram_energy.fits Enter output file name
          (bkg = )                Background region file or fixed background (counts/pixel/s) subtra
        (error = gaussian)        Method for error determination(poisson|gaussian|<variance file>)
     (bkgerror = gaussian)        Method for background error determination(poisson|gaussian|<varian
      (bkgnorm = 1.0)             Background normalization
          (exp = )                Exposure map image file
       (bkgexp = )                Background exposure map image file
      (sys_err = 0)               Fixed systematic error value for SYS_ERR keyword
          (opt = generic)         Output file type: pha1
     (defaults = ${ASCDS_CALIB}/cxo.mdb -> /soft/ciao/data/cxo.mdb) Instrument defaults file
         (wmap = )                WMAP filter/binning (e.g. det=8 or default)
      (clobber = no)              OK to overwrite existing output file(s)?
      (verbose = 0)               Verbosity level
         (mode = ql)
```

```
Parameters for /home/username/cxcds_param/mkinstmap.par


       outfile =                  Output File Name
#------------------------------------------------------------------------
# Energy Band Info
#------------------------------------------------------------------------
# Currently, this file is a simple ascii file with two columns
  spectrumfile = NONE             Energy Spectrum File (see docs)
    monoenergy = 1.7              Energy for mono-chromatic map [keV]
#
     pixelgrid = 1:1024:#1024,1:1024:#1024 Pixel grid specification x0:x1:#nx,y0:y1:#ny
       obsfile =                  Name of fits file + extension with obs info
     detsubsys =                  Detector Name
       grating = NONE             Grating for zeroth order ARF
      maskfile = acisf01838_000N001_msk1.fits NONE, or name of ACIS window mask file
       (mirror = HRMA)            Mirror Name
#
      (pbkfile = NONE)            NONE, or the name of the parameter block file
       (dafile = NONE)            NONE, CALDB, or name of ACIS dead-area calibration file
#
(ardlibparfile = ardlib.par)      name of ardlib parameter file
      (geompar = geom)            Parameter file for Pixlib Geometry files
#
      (verbose = 0)               Verbosity
      (clobber = no)              Overwrite existing files?
         (mode = h)               Enter mode for parameter file.
```

```
Parameters for /home/username/cxcds_param/mkexpmap.par


   asphistfile =                  Aspect Histogram File
       outfile =                  Output File Name
   instmapfile =                  Name of Instrument Map
#
        xygrid = 0.5:8192.5:#1024,0.5:8192.5:#1024 grid specification syntax x0:x1:#nx,x0:x1:ny
  useavgaspect = no               Use Average Aspect Pointing
#------------------------------------------------------------------------
```

```
#  Aspect Histogram Parameters
#  If UseAvgAspect is set to yes, then only the average pointing derived from
#  the many pointings in the aspect histogram will be used.
#------------------------------------------------------------------------
#asphistfile,f,a,"../data/aciss_asphist.fits",,,"Aspect Histogram File"
    (normalize = no)              Normalize exposure map by exposure time
#
      (geompar = geom)            Parameter file for Pixlib Geometry files
      (verbose = 0)               Verbosity
      (clobber = no)              Overwrite existing files?
         (mode = h)               Enter mode for parameter file.
```

```
Parameters for /home/username/cxcds_param/dmregrid.par


##
## DMREGRID -- regrid image
##
        infile = @expmap_1.7kev.lis Input image
       outfile = expmap_1.7kev.fits Enter output file name
           bin = 1:1024:1,1:1024:1 Binning specification
      rotangle = 0                 CCW rotation angle in degrees about rotation center
    rotxcenter = 0                 x coordinate of rotation center
    rotycenter = 0                 y coordinate of rotation center
       xoffset = 0                 x offset
       yoffset = 0                 y offset
          npts = 1                 Number of points in pixel (0='exact' algorithm)
    (coord_sys = logical)          Coordinate system of bin parameter
      (clobber = no)               OK to overwrite existing output file(s)?
      (verbose = 0)                Verbosity level (0 = no display)
         (mode = ql)
```

```
Parameters for /home/username/cxcds_param/dmimgcalc.par


        infile = 1838_img8.fits        Input file #1
       infile2 = expmap_1.7kev.fits Input file #2
       outfile = 1838_img8_norm.fits   output file
     operation = div              arithmetic operation
      (weight = 1)                weight for first image
     (weight2 = 1)                weight for second image
    (lookupTab = ${ASCDS_CALIB}/dmmerge_header_lookup.txt -> /soft/ciao/data/dmmerge_header_lookup.txt) loc
      (clobber = no)               delete old output
      (verbose = 0)                output verbosity
         (mode = ql)
```

```
Parameters for /home/username/cxcds_param/dmimgthresh.par


        infile = 1838_img8.fits        Input dataset/block specification
       outfile = 1838_img8_clean.fits  Output dataset/block specification
      (expfile = expmap_1.7kev.fits) Exposure map file
          (cut = 1.5%)             Threshold value
```

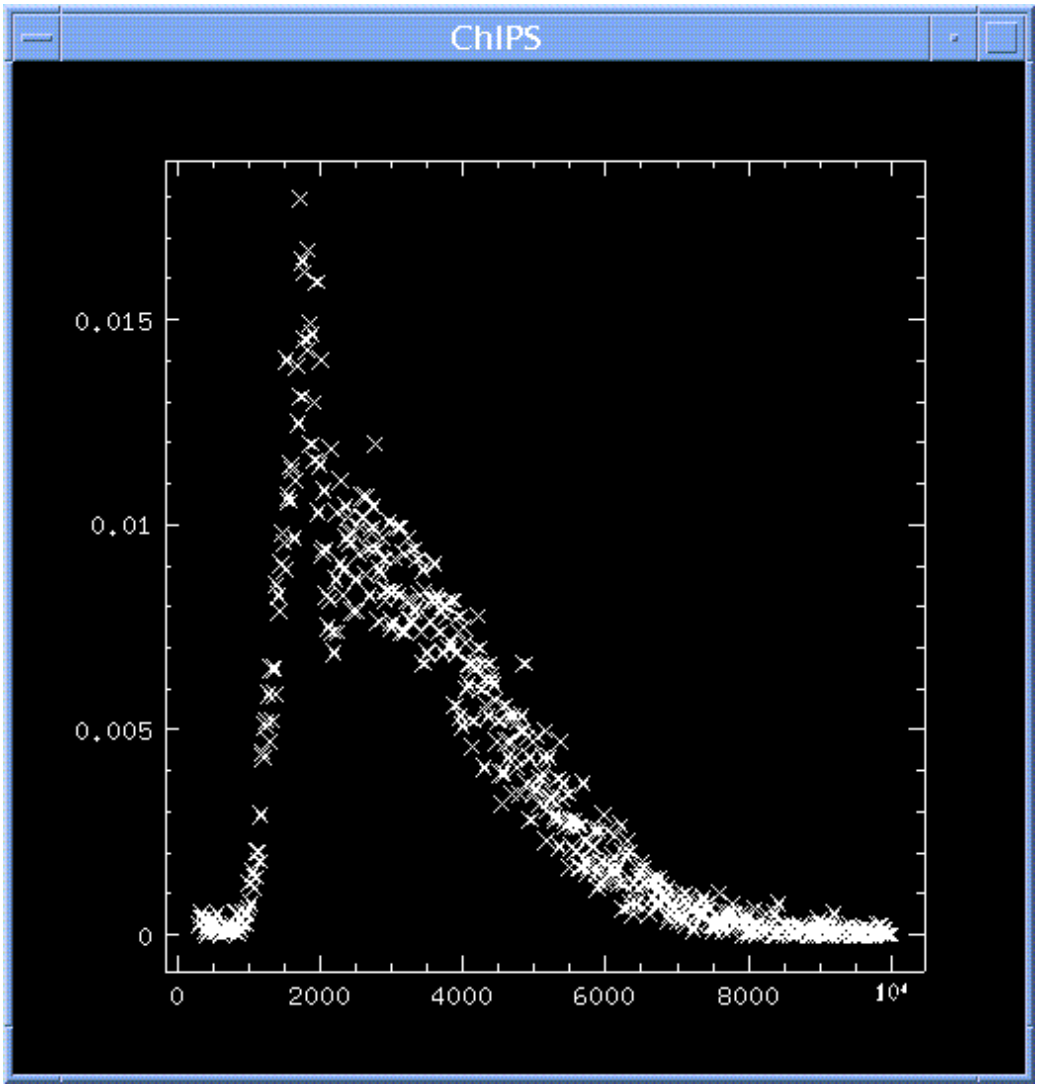
```
  (value = 0)              Replacement value
(verbose = 0)              Debug Level(0−5)
(clobber = no)             Clobber existing file
   (mode = ql)
```

History

23 Dec 2004	updated for CIAO 3.2: minor changes to parameter files, added Caveats section
19 Dec 2005	updated for CIAO 3.3: default value of `dmextract` `error` and `bkgerror` parameters is "gaussian"; updated syntax for `asphist` (GTI filter is associated with the event file rather than the aspect solution); corresponding changes to screen output
24 May 2006	changed "`det`" abbreviation to full parameter name ("`detsubsys`") in `mkinstmap` call
01 Dec 2006	updated for CIAO 3.4: parameter file update for `mkinstmap`; minor change to `dmstat`/`dmextract` screen output
06 Mar 2007	added ACIS dead area correction section and example of setting the `pbkfile` and `dafile` parameters

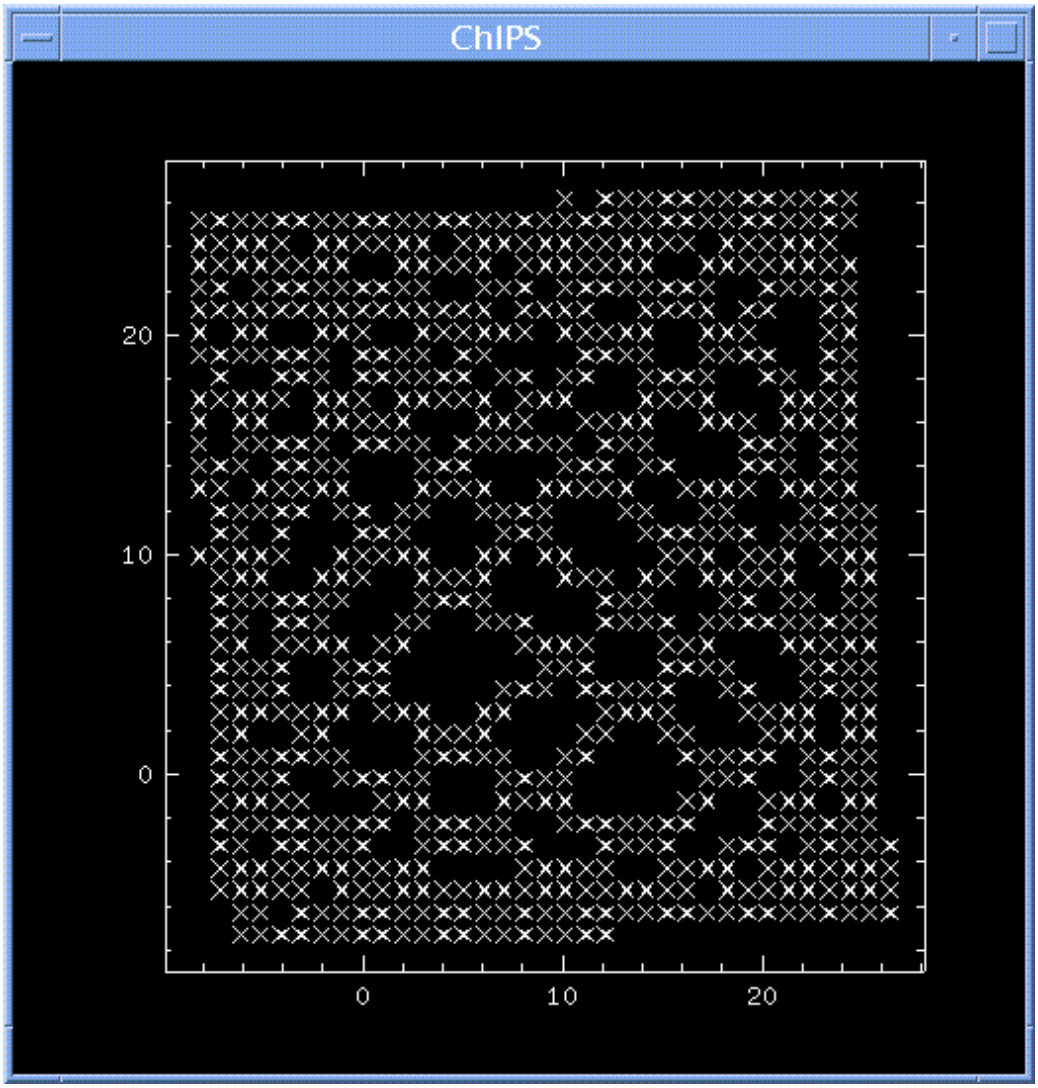
URL: http://cxc.harvard.edu/ciao/threads/expmap_acis_multi/ Last modified: 6 March 2007

Image 1: Energy versus count rate for the source



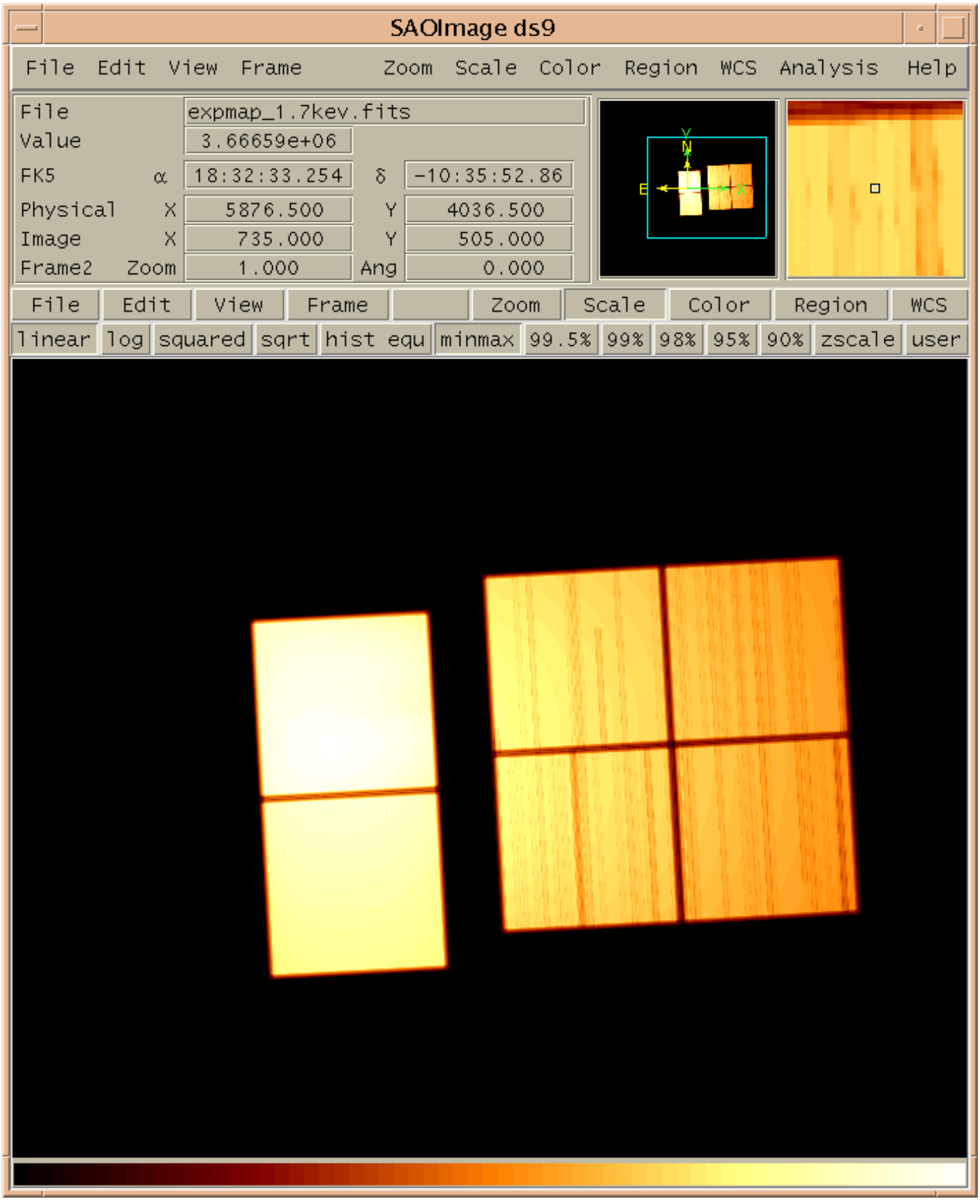
The abscissa shows the energy (in eV) and the ordinate the count_rate (count/s) columns.

Image 2: Histogram of the observation dither



The abscissa shows the `X_OFFSET` and the ordinate the `Y_OFFSET` columns of the aspect histogram.

Image 3: Exposure map for the ACIS−S3 chip



The almost−horizontal and vertical stripes are due to bad columns in the instrument maps and the chip gaps.

Image 3: Exposure map for the ACIS−S3 chip

Image 4: Fluxed image of the whole observation