ANALYSIS OF XMM-Newton DATA

Edited by Elena Franciosini

(For more details, see also the SAS User Guide at http://xmm.esac.esa.int/external/xmm_user_support/documentation/sas_usg/USG/ and the XMM-Newton ABC Guide at http://heasarc.gsfc.nasa.gov/docs/xmm/abc/)

1. INITIALIZE SAS AND SET SAS ENVIRONMENT VARIABLES

To use SAS you need first to set the environment variables ***SAS_DIR*** and ***SAS_CCFPATH***, defining the location of the SAS installation and of the calibration files (CCF data), and to set some parameters using the script sas-setup.csh or sas-setup.sh in the $SAS_DIR directory. This is done by executing (for csh/tcsh and bash, respectively):

```
source ~/bin/sas.csh    or     . ~/bin/sas.sh
```

You also need to set the variables SAS_ODF and SAS_CCF, as explained in the following section.

These settings must be done each time you start a new session.

2. PREPARE THE DATA FOR ANALYSIS

Create a new directory for data analysis (e.g. ANALYSIS) and move into it to perform the following steps.

When starting to analyse the data for the first time, you need to create the CIF file ***ccf.cif*** and the summary file ***rrrr_nnnnnnnnnn_SCX00000SUM.SAS*** (*rrrr* = revolution number, *nnnnnnnnnn* = observation ID).

First, set the path to the directory containing the ODF files for the observation you are analysing, using:

```
setenv SAS_ODF /path/to/ODF_dir    or    export SAS_ODF=/path/to/ODF_dir
```

Then, to create the CIF file, do:

```
setenv SAS_CCF $SAS_CCFPATH    or    export SAS_CCF=$SAS_CCFPATH
```

and then:

```
cifbuild calindexset=ccf.cif withccfpath=no fullpath=yes
```

Point the variable SAS_CCF to the CIF file:

```
setenv SAS_CCF ccf.cif    or    export SAS_CCF=ccf.cif
```

Then create the summary file (if it exists, first delete it):

```
odfingest outdir=. usecanonicalname=yes writepath=yes
```

When running odfingest you might receive a series of warnings like:

```
** odfingest: warning (TranslationNotAvailable), No look up possible [EPN
Filter]It is not possible to translate FILTER. Its raw value is nan
** odfingest: warning (HousekeepingParameterMissing), PN/U001 NoAvailableRows:
No rows match the expression...
```

This happens when the ODF directory contains files referring to exposures with no valid science data (e.g. like *PNU00201DLI.FIT). You can safely ignore these warnings.

Finally, point the variable SAS_ODF to the summary file:

```
setenv SAS_ODF rrrr_nnnnnnnnnn_SCX00000SUM.SAS
```

or

```
export SAS_ODF=rrrr_nnnnnnnnnn_SCX00000SUM.SAS
```

2a. REPROCESS THE DATA (Not needed for the school)

This step is only needed if the observation is old and new calibration files or a new SAS version are available after the data release. After the creation of the files ccf.cif and *SUM.SAS, create a new subdirectory for reprocessing (e.g. PROC), move into it and rerun the pipelines:

```
emchain       # for MOS
epchain       # for PN
```

The pipelines produce the event files for each instrument, with the following names:

```
PnnnnnnnnM1SeeeMIEVLI0000.FIT        for MOS1
PnnnnnnnnM2SeeeMIEVLI0000.FIT        for MOS2
PnnnnnnnnPNSeeeMIEVLI0000.FIT        for PN
```

(where *nnnnnnnn* is the Observation ID and *eee* is the esposure number). Moreover the pipeline produces the attitude file

```
PnnnnnnnnnnBX000ATTTSR0000.FIT
```

and a number of additional files not required for the analysis (for details see the XMM guides).

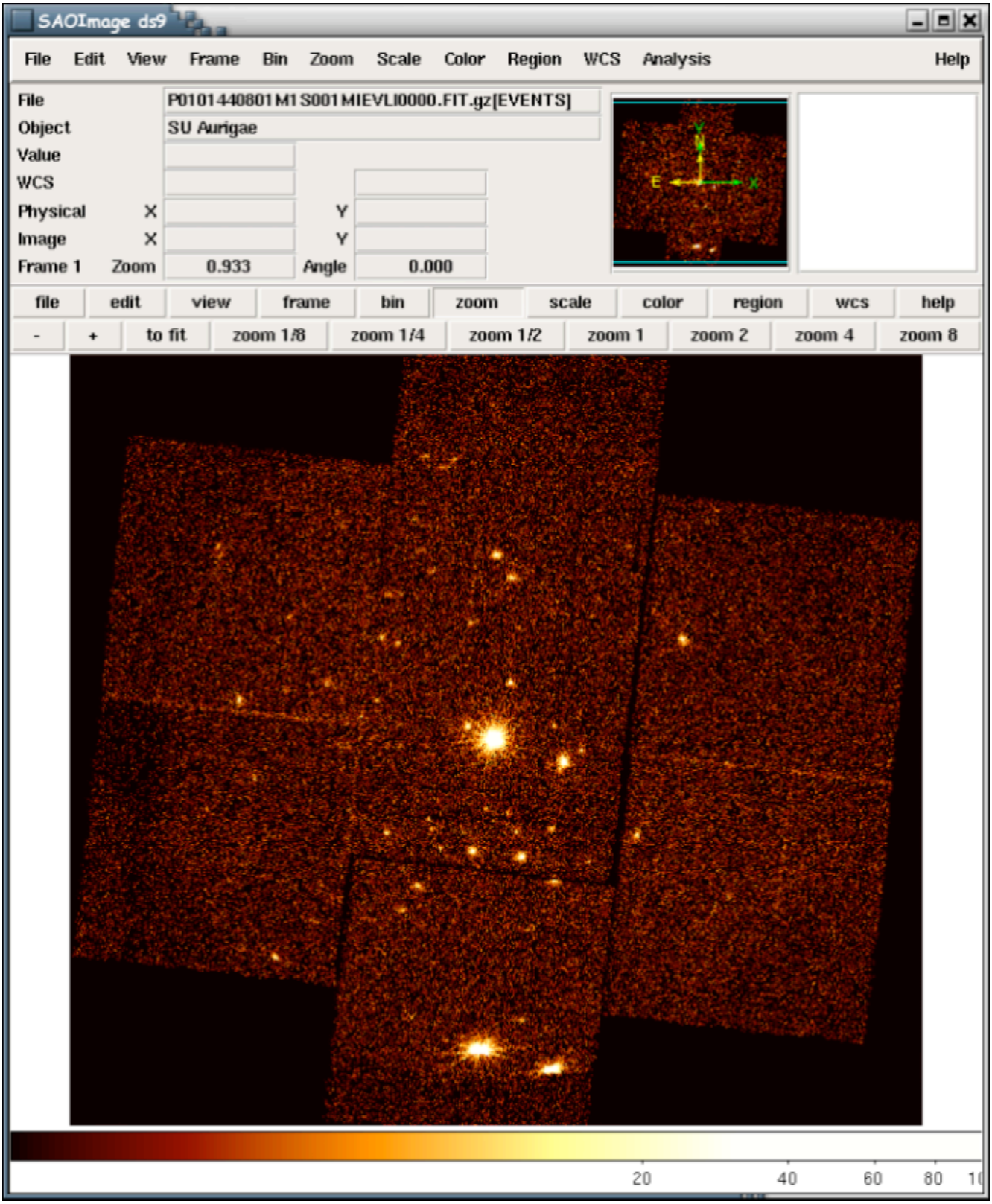
3. FILTER THE EVENT FILE

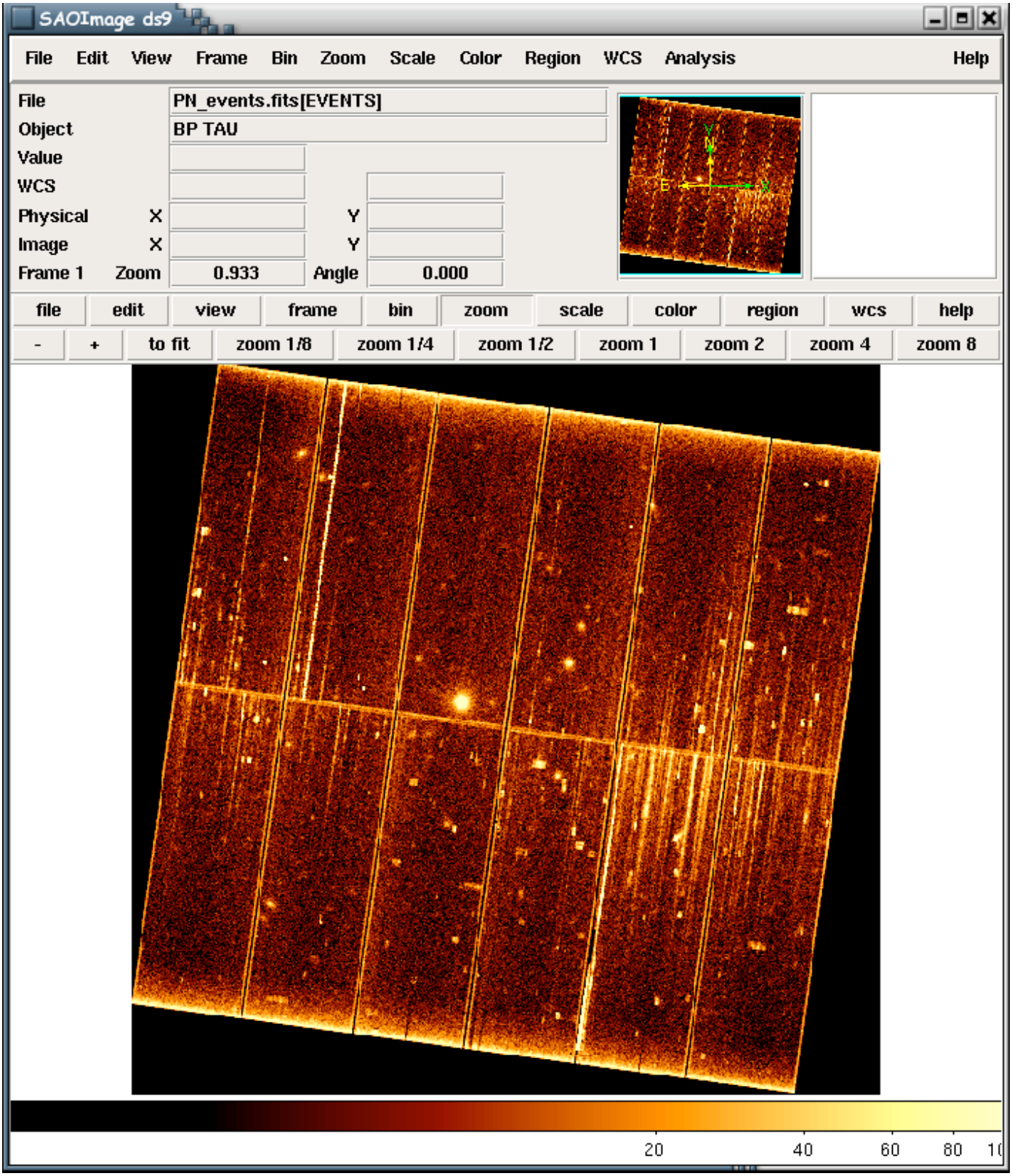
Copy the event files ***P*EVLI*.FIT*** from the directory containing the reprocessed data (e.g. PROC from Sect.2a above, if you reprocessed the data yourself, or PIPEPROD, if you are using the standard pipeline data provided by the XMM Data Archive). It is suggested to rename them with a more intuitive name, e.g.

```
M1_events.fits            for the MOS1 event file
M2_events.fits            for the MOS2 event file
PN_events.fits            for the PN event file
```

For the school, the reprocessed data are available in the DATA directory under each user's home, already renamed as above. Only PN data will be used in the exercises.

You can display the event files using ds9 to examine the data: for best viewing, bin the data (Bin button or menu) using Block 32 or 64, and use a log scale.


SAOImage ds9
File Edit View Frame Bin Zoom Scale Color Region WCS Analysis Help
File P0101440801M1S001MIEVLI0000.FIT.gz[EVENTS]
Object SU Aurigae
Value
WCS
Physical X Y
Image X Y
Frame 1 Zoom 0.933 Angle 0.000
file edit view frame bin zoom scale color region wcs help
- + to fit zoom 1/8 zoom 1/4 zoom 1/2 zoom 1 zoom 2 zoom 4 zoom 8
20 40 60 80



Images of MOS (left) and PN (right) event files resulting from the pipeline, displayed in ds9

Now you need to apply a series of filters to clean the data. It is suggested to create a script file (eg. filter.csh or filter.sh) in which to write the following commands, for ease of use and debugging. The script can then be run with

```
source filter.csh    or    sh filter.sh
```

IMPORTANT NOTE: in the following, for the sake of generality, it is assumed that the name of the event file produced by the pipeline is ***events.fits***. Substitute it with the appropriate name you have given to the event file in the step above. Also, the output files have general names. Substitute them with specific appropriate names (in particular when extracting files related to individual sources) to avoid confusion.
It is highly recommended to prepend a prefix to the file names as above (M1, M2 or PN) to distinguish the files from different instruments.

After each step, you can see the effect of the applied filter using ds9.

3.1. Filtering for bad pixels, bad columns, cosmic rays, events outside the FOV, etc.

This filtering requires different filtering expressions for MOS and PN.

For ***MOS data:***

```
evselect table=events.fits withfilteredset=y \
filteredset=filtered_events.fits \
expression='(PATTERN <= 12)&&(#XMMEA_EM)' \
updateexposure=y destruct=y writedss=y
```

For ***PN data:***

```
evselect table=events.fits keepfilteroutput=y withfilteredset=y \
filteredset=filtered_events.fits \
expression='(PATTERN <= 4)&&((FLAG & 0xfb0825) == 0)' \
updateexposure=y destruct=y writedss=y

evselect table=filtered_events.fits keepfilteroutput=y \
expression='(!(CCDNR==5&&RAWX==11)&&!(CCDNR==11&&RAWX==27&&RAWY>=187))' \
updateexposure=n destruct=y writedss=n
```

where events.fits is the name of the input unfiltered event file and ***filtered_events.fits*** is the name of the output filtered event file.

The ***PN filter*** ((FLAG & 0xfb0825) == 0) is a combination of the standard filter #XMMEA_EP and additional filters to exclude also invalid patterns (>12), bright CCD borders and events outside the field of view.
In addition, some bright columns remain in the PN on CCDs #5 and 11. The second evselect command excludes these columns, overwriting the file filtered_events.fits with the new filtered data.
NOTE: be careful to set the parameters ***updateexposure*** and ***writedss*** to ***n*** in the second filtering command: this prevents the selection expression stored in the file "Data Subspace" (DSS) to become too long, causing problems for subsequent filtering. Since this is just a geometric filtering, saving the filtering information and updating the exposure keyword is not necessary.
NOTE 2: since in the second filtering for PN the output file is overwritten, if you need to redo the above filtering, remember to repeat both commands (i.e. start again from events.fits).

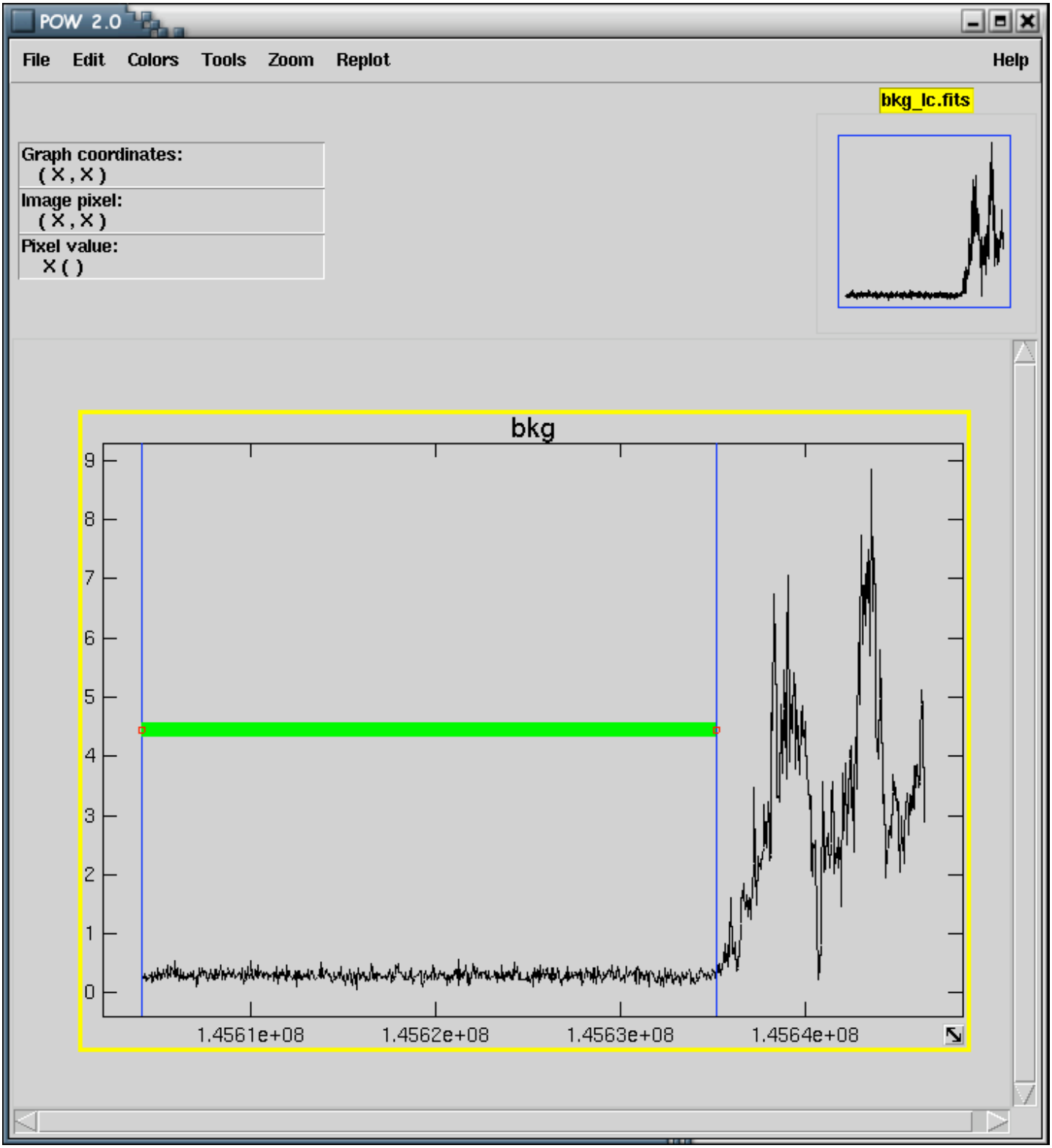
3.2 Filtering for high-background periods

In many observations there are periods showing a very high background, due to proton flares. These periods need to be excluded to achieve a higher signal-to-noise for detection of faint sources. To do this you need to create a GTI (Good Time Interval) file that contains only good times, and use it to filter the event files.
First create a light curve of the background of the entire field of view selecting only events with pattern=0 and energy >10 keV (at these energies the contribution of X-ray sources is generally negligible). Use as input the filtered event file created in the previous step:

```
evselect table=filtered_events.fits \
expression='(PI>=10000)&&(PATTERN==0)' \
withrateset=y rateset=bkg_lc.fits maketimecolumn=y \
timecolumn=TIME timebinsize=50 makeratecolumn=yes
```

Display the resulting light curve ***bkg_lc.fits*** using **fv** (see the fv_and_simbad_tutorial.pdf on how to do this). Identify the time intervals where the background count rate is approximately constant and low, following the instructions in the exercises (in most cases below ***0.2-0.5 cts/s for MOS and 0.5-1.0 cts/s for PN***; the true values depend on the observation)

NOTE: If the background is constant throughout the whole observation, this filtering is not necessary, and you can skip to the next section.



Background light curve displayed with fv, showing the selected time interval of low background.

If there are high-background periods, create the GTI file ***gti.fits***, using the task tabgtigen. Assuming the interval of low background identified above is 133309680 - 133322680 sec, write:

```
tabgtigen table=bkg_lc.fits gtiset=gti.fits \
expression='(TIME in [133309680:133322680])'
```

If you need to concatenate two (or more) time intervals, use the following syntax for the expression:

```
tabgtigen table=bkg_lc.fits gtiset=gti.fits \
expression='(TIME in [t1:t2])||(TIME in [t3:t4])'
```

If the high-background period is at the beginning or the end of the observation, you can also use the following expressions:

```
tabgtigen table=bkg_lc.fits gtiset=gti.fits expression='TIME > t2'

or

tabgtigen table=bkg_lc.fits gtiset=gti.fits expression='TIME < t1'
```

respectively.

Alternatively, you can exclude a high-background interval [t1:t2] using:

```
tabgtigen table=bkg_lc.fits gtiset=gti.fits \
expression='!(TIME in [t1:t2])'
```

or for two intervals:

```
tabgtigen table=bkg_lc.fits gtiset=gti.fits \
expression='!(TIME in [t1:t2])&&!(TIME in [t3:t4])'
```

It is also possible to create the gti file using a filtering expression in RATE instead of TIME. However, **note** that this may remove a lot of small intervals, leading to an event file with several time gaps, which will make it difficult to perform temporal analysis of the data. If you want to retain intervals where the background count rate is less than 0.5 cts/s, you can use the following expression:

```
tabgtigen table=bkg_lc.fits gtiset=gti.fits \
expression='RATE < 0.5'
```

When you have created the gti file gti.fits, use it to filter the event file filtered_events.fits with the command:

```
evselect table=filtered_events.fits \
expression='GTI(gti.fits,TIME)' \
withfilteredset=y filteredset=events_gti.fits \
destruct=y keepfilteroutput=y updateexposure=y writedss=y
```

This creates the file ***events_gti.fits*** with the high-background periods excluded.

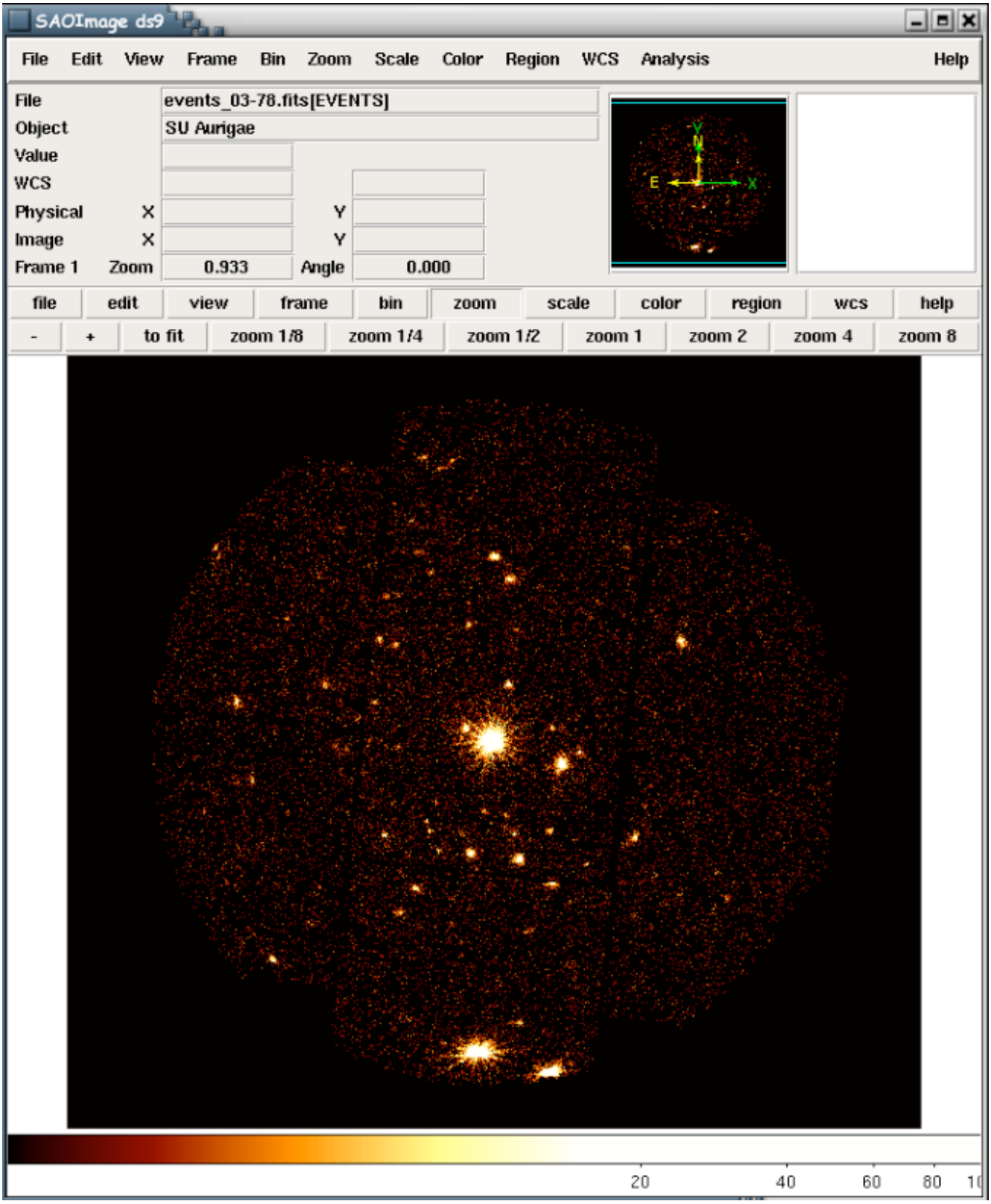
3.3. Filter events in energy

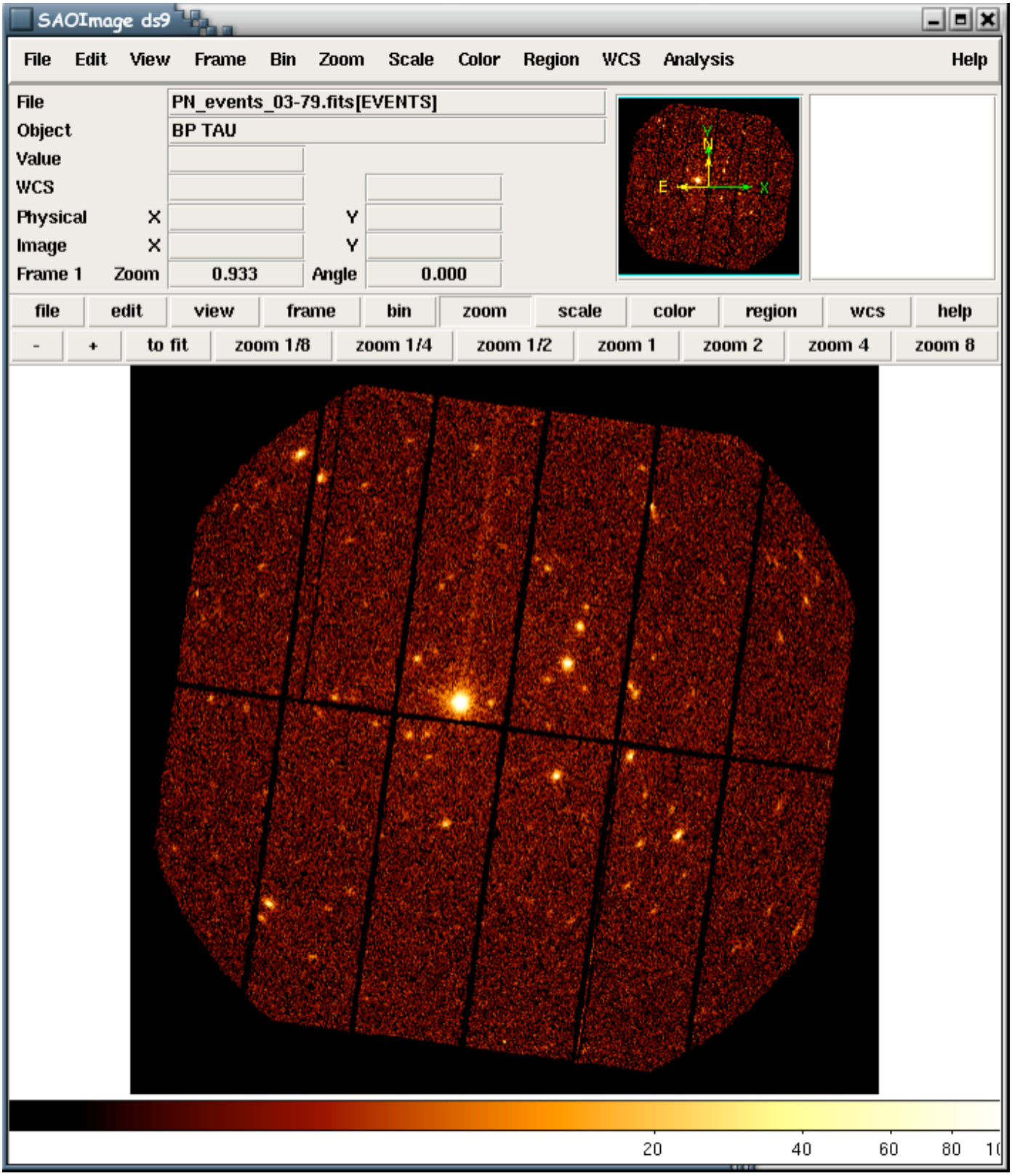
Filter the event file in the range **0.3-7.9 keV**.
Events with energy below 0.3 keV are mostly related to artifacts and noise, while above 7.9 keV almost only background is present (there is also a prominent background line emission at 8 keV).

To apply the energy filter to the gti-filtered event file do:

```
evselect table=events_gti.fits \
expression='(PI in [300:7900])' withfilteredset=y \
filteredset=events_03-79.fits keepfilteroutput=y \
destruct=y updateexposure=y writedss=y
```

The resulting file ***events_03-79.fits*** is the final event file to use for subsequent analysis.


SAOImage ds9
File Edit View Frame Bin Zoom Scale Color Region WCS Analysis Help
File events_03-78.fits[EVENTS]
Object SU Aurigae
Value
WCS
Physical X Y
Image X Y
Frame 1 Zoom 0.933 Angle 0.000
N
E
file edit view frame bin zoom scale color region wcs help
- + to fit zoom 1/8 zoom 1/4 zoom 1/2 zoom 1 zoom 2 zoom 4 zoom 8
20 40 60 80



Images of the resulting MOS (left) and PN (right) event files after all the filtering steps

4. SOURCE DETECTION AND IDENTIFICATION

To perform source detection you first need to create an exposure map. To do this the first step is to create an image (***image.fits***) from the filtered event file with a pixel size of 2 arcsec (required by the detection program), corresponding to a binsize of 40 pixels (the pixel size in the event file is 0.05 arcsec):

```
evselect table=events_03-79.fits withimageset=y \
imageset=image.fits squarepixels=y imagebinning=binSize \
ximagebinsize=40 yimagebinsize=40 xcolumn=X ycolumn=Y
```

Then create the exposure map ***expimage.fits***:

```
eexpmap eventset=events_03-79.fits imageset=image.fits \
attitudeset=attitude.fits pimin=300 pimax=7900 \
expimageset=expimage.fits
```

The file ***attitude.fits*** is the attitude file. This file is generated by the pipeline (with name PnnnnnnnnnnBX000ATTTSR0000.FIT, see Sect.2a). If it does not exist, it can be created with:

```
atthkgen atthkset=attitude.fits
```

Source detection is performed with the program **pwxdetect_xmm**, which is not part of SAS.

Copy the file ***pwxdetect.par*** from the ~/bin directory into the ANALYSIS directory, and edit it to set the name of the filtered event file and of the esposure map, and the desired threshold significance and maximum detection scale. The final threshold significance is set by the number of spurious sources one wishes to accept, and it depends on the background level, which is approximated by the number of counts in the filtered event file. The first threshold significance should be set to a value of 1.0 less than the final threshold. For EPIC MOS and PN, typical values of the threshold to obtain at most one spurious source are around ***4.7 - 4.8*** for event files with 100000 counts, ***4.9 - 5.0*** for 150000-200000 counts, ***5.2*** for 300000 counts, and ***5.5*** for 500000 counts. To find the number of counts in the filtered event file use fv to display it and get the number of rows in the events table.
Then run

```
pwxdetect_xmm
```

This produces a FITS file called ***det_src.fits*** containing the detected sources, an ASCII version of it called ***det_src***, and a region file ***det_overlay.ds9*** that can be displayed in ds9 with the position of the detected sources (the circle radius corresponds to the scale at which the source was detected with maximum significance). For more details on the detection software and on the produced files, see the pwxdetect_xmm_guide.pdf.

The file det_overlay.ds9 does not contain the identification numbers of the sources. To create an overlay file with source numbers, run the script

```
add_src_num_xmm.com
```

which will create a file ***det_overlay_txt.ds9***.

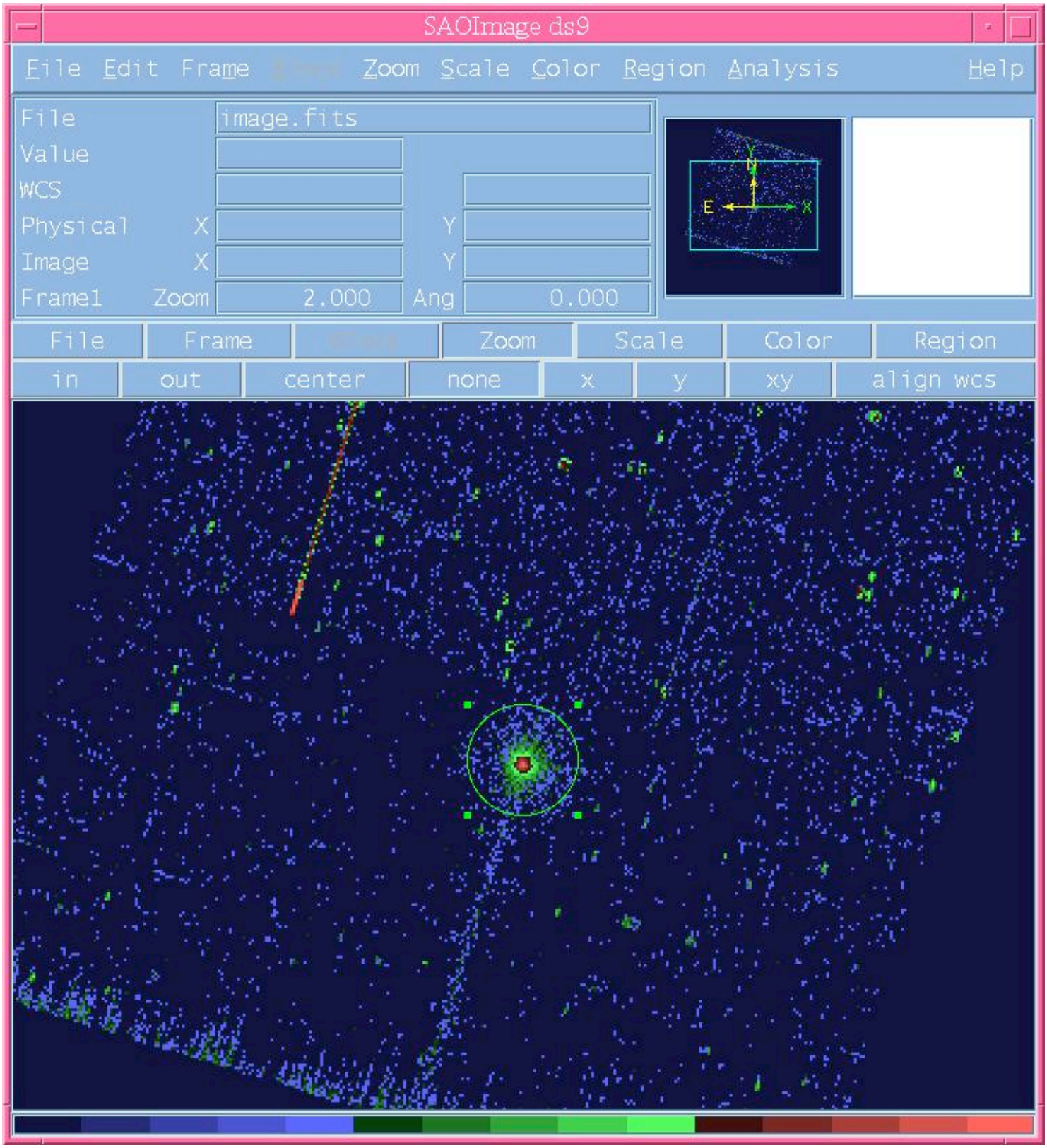
Display the image of the field with the overlayed region file using ds9. Identify the brightest sources of the field using Simbad (refer to the fv_and_simbad_tutorial.pdf for details on how to proceed).

5. EXTRACTION OF LIGHT CURVES AND SPECTRA OF SELECTED SOURCES

In order to extract light curves and spectra for a given source, first you need to define an extraction region for the source and background.
To do this, open ds9 and find the source you are interested in (see the fv_and_simbad_tutorial.pdf).
Click on the position of the source and define a circular region of the desired width. For bright isolated sources, use a large region to include the highest possible fraction of the source counts. In the case of crowded fields however the size of the source region must be chosen in order to maximize the source counts while minimizing the contamination from nearby sources.

Select

```
"Region" -> "File Format" -> "Ciao"
"Region" -> "File Coordinate System" -> "Physical"
```
then
```
"Region" -> "Save Regions"
```
to save the region into a file (e.g. src.reg) for future reference; the circle(...) expression inside src.reg is the one to be entered in the selection command below.



Ds9 window with a marked circle around a source

Repeat this step for the background region. Choose a circular region of the same size and on the same CCD, free of sources, possibly at the same distance from the field center as the source.

Extract an event file ***src_events.fits*** containing only the source (plus background) events, using evselect. Assuming for example that the selected source region is circle(27000,27650,400), write:

```
evselect table=events_03-79.fits \
expression='((X,Y) in circle(27000,27650,400))' \
withfilteredset=y filteredset=src_events.fits keepfilteroutput=y \
destruct=y updateexposure=y writedss=y
```

In the same way, extract an event file ***bkg_events.fits*** containing only background events (substitute the

coordinates of the background region and the output file name in the above expression).

Now you can use directly the files src_events.fits and bkg_events.fits to extract the source and background light curves and spectra.

To extract the light curve, define an appropriate time bin (e.g. 600 sec in the command below), then run:

```
evselect table=src_events.fits \
withrateset=y rateset=src_lightcurve.fits maketimecolumn=y \
timecolumn=TIME timebinsize=600 makeratecolumn=yes
```

which produces a file ***src_lightcurve.fits*** containing the source light curve.
Do the same for the background. Display the light curves using fv.

To extract the source and background spectra, use again evselect. The command is different for MOS and PN due to their different energy bin size. For the source spectrum:

For MOS:

```
evselect table=src_events.fits withspectrumset=y \
spectrumset=src_spec.fits spectralbinsize=15 \
withspecranges=y specchannelmin=0 specchannelmax=11999 energycolumn='PI'
```

For PN:

```
evselect table=src_events.fits withspectrumset=y \
spectrumset=src_spec.fits spectralbinsize=5 \
withspecranges=y specchannelmin=0 specchannelmax=20479 energycolumn='PI'
```

where ***src_spec.fits*** contains the spectrum of the source plus background.
Repeat the same command for the background, generating a file ***bkg_spec.fits***.

Run backscale on the source and background spectra to compute the area of the extraction region used to extract the spectra:

```
backscale spectrumset=src_spec.fits withbadpixcorr=y \
badpixlocation=src_events.fits

backscale spectrumset=bkg_spec.fits withbadpixcorr=y \
badpixlocation=bkg_events.fits
```

Generate a response matrix ***src.rmf*** (this can take a long time) using the task rmfgen:

```
rmfgen spectrumset=src_spec.fits rmfset=src.rmf
```

Then generate the ancillary file ***src.arf*** with arfgen:

```
arfgen spectrumset=src_spec.fits arfset=src.arf \
withrmfset=y rmfset=src.rmf \
withbadpixcorr=y badpixlocation=src_events.fits
```

To perform spectral analysis, you first need to rebin the source spectrum to a minimum number of counts per bin (e.g. 20), and to associate to it the names of the response matrix, the ancillary file and the background file, that will be stored in the keywords RESPONSE, ANCRFILE and BACKFILE, respectively. To do so, use the command grppha:

```
grppha src_spec.fits src_spec_grp.fits
```

and type the following commands at the GRPPHA prompt:

```
chkey respfile src.rmf
chkey ancrfile src.arf
chkey backfile bkg_spec.fits
group min 20
exit
```

6. EXTRACT A SPECTRUM OR AN IMAGE IN A SELECTED TIME INTERVAL

If the source is variable, you might want to extract the spectrum in selected time intervals (e.g. where the count rate is constant, or at the peak of a flare). To do this, display the light curve of the source and determine the start and

stop times of the interval of interest (e.g. start=133309680, stop=133322680). Then extract the source and background spectrum in this time interval, adding the following selection expression to the command given above (for MOS or PN) for spectral extraction:

```
expression='(TIME in [133309680:133322680])'
```

Run backscale on the source and background spectra, and group the source spectrum, as in the previous section. It is not necessary to run rmfgen and arfgen if the rmf and arf files for the source have already been generated.

The above selection expression can be used also to extract images and event files limited to the desired time interval, using the appropriate evselect commands (see previous sections).