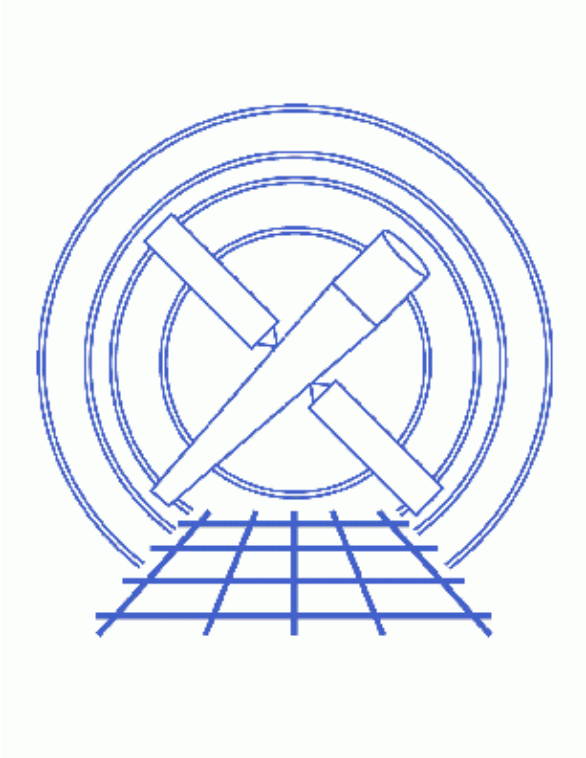
Create a Color Spectrum



CIAO 3.4 Science Threads

Table of Contents

Create a Color Spectrum

CIAO 3.4 Science Threads

Overview

Last Update: 1 Dec 2006 − reviewed for CIAO 3.4: no changes

Synopsis:

The tool `dmimg2jpg` may be used to create color images of grating spectra. Two different ways of displaying the spectrum are available; the result depends on whether the event file is filtered on sky or grating angular coordinates.

Purpose:

To create color spectra of grating data.

Read this thread if:

you are working with any grating observation and would like to image the spectrum.

Related Links:

- Analysis Guide for Chandra High Resolution Spectroscopy: an in−depth discussion of grating analysis.

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

Get Started

Sample ObsID used: 1451 (ACIS−S/HETG, II PEG)

File types needed: evt2

Filter the Event File

In both of the following examples, we want to create a spectrum that contains events from all orders of the observation (+/−1, 2, 3 for HEG and MEG) for the brightest source. To do so, we need to filter on the `tg_srcid`, `tg_part`, and `tg_m` columns:

```
unix% dmcopy \
      "acisf01451N002_evt2.fits[tg_srcid=1,tg_part=0,1,2,tg_m=-3,-2,-1,0,1,2,3]" \
      1451_spec_evt2.fits
```

Since this is an HETG observation, there is no need to add the LEG component to the `tg_part` filter. For an LETG observation, that portion of the command would be `tg_part=0,3`.

The following explanation of the columns is from the Chandra Grating Analysis Page:

- `tg_srcid`: the source identifier index. Up to ten sources can be resolved simultaneously, but the pipeline only detects the brightest source.
- `tg_part`: the spatial part of the spectrum. 0 − zero−order; 1 − HEG; 2 − MEG; 3 − LEG; 99 − unresolved.
- `tg_m`: the signed diffraction order that is resolved if the detector has sufficient energy resolution (i.e. is the ACIS detector). Unresolved photons are assigned to order 99.

Create a Color Spectrum

A. In Sky Coordinates

This example creates a color spectrum in the familiar cross−dispersion pattern of Chandra grating observations.

First we need to create the input image files for `dmimg2jpg`. Here `dmcopy` is used to create 3 image files corresponding to different energy bands:

```
unix% punlearn dmcopy
unix% pset dmcopy infile="1451_spec_evt2.fits[energy=000:999][bin sky=2000:7000:8]"
unix% pset dmcopy outfile=sky_red.fits
unix% dmcopy
Input dataset/block specification (1451_spec_evt2.fits[energy=000:999][bin sky=2000:7000:8]):
Output dataset name (sky_red.fits):
```

This command takes the events in the given energy range and bins them on the sky coordinates. A blocking factor of 8 was used; see ahelp dmcopy for information on blocking.

The `dmcopy` command may be condensed to a single line; for the second band:

```
unix% dmcopy "1451_spec_evt2.fits[energy=800:2399][bin sky=2000:7000:8]" \
      sky_green.fits
```

And finally:

```
unix% dmcopy "1451_spec_evt2.fits[energy=1700:9999][bin sky=2000:7000:8]" \
      sky_blue.fits
```

Now we can use `dmimg2jpg` to combine the files into a single RGB image:

```
unix% punlearn dmimg2jpg
unix% pset dmimg2jpg infile=sky_red.fits
unix% pset dmimg2jpg greenfile=sky_green.fits
unix% pset dmimg2jpg bluefile=sky_blue.fits
unix% pset dmimg2jpg outfile=sky_1451.jpg
unix% pset dmimg2jpg showgrid=no
unix% pset dmimg2jpg showaimpoint=no
unix% dmimg2jpg
Input file name (red image if true color) (sky_red.fits):
Green color channel file name (sky_green.fits):
```

```
Blue color channel file name (sky_blue.fits):
Output jpg file name (sky_1451.jpg):
```

The resulting spectrum is shown in Figure 1  .

You can check the parameter file that was used with plist dmimg2jpg.

The parameters `showgrid` and `showaimpoint` control whether a WCS grid and the location of the aimpoint, respectively, are marked on the spectrum. More about these options can be found in the ahelp file.

B. In Grating Angular Coordinates

This example creates a color spectrum in the grating angular coordinates; the image will look like a traditional line spectrum.

Again, we need to create input image files for dmimg2jpg. In this case, however, we both filter and bin on `tg_d` and `tg_lam`. From the Chandra Grating Analysis Page:

- `tg_d`: the cross−dispersion angle in degrees, using the Rowland spacing as the focal−length.
- `tg_lam`: if the photon is resolved, then this is the wavelength in Angstroms. `tg_lam` is non−negative; unresolved photons are assigned wavelengths of 0.0.

In this example, we choose the angles between +/− 0.000663889 degrees and wavelengths up to 25.005 Å. Due to a Data Model bug, the filtering and binning must be done in separate `dmcopy` commands.

For the first band:

```
unix% dmcopy \
      "1451_spec_evt2.fits[tg_d=−0.000663889:0.000663889,tg_lam=1:25.005,energy=000:999]" \
      red_filter.fits

unix% dmcopy \
      "red_filter.fits[bin tg_lam=1:25.00:0.04,tg_d=−1.0e−3:1.0e−3:2.0e−5]" \
      ga_red.fits
```

Note that the binning command here is not as simple as in the previous example − the two variables are blocked by different factors.

For the second band:

```
unix% dmcopy \
      "1451_spec_evt2.fits[tg_d=−0.000663889:0.000663889,tg_lam=1:25.005,energy=800:2399]" \
      green_filter.fits

unix% dmcopy \
      "green_filter.fits[bin tg_lam=1:25.00:0.04,tg_d=−1.0e−3:1.0e−3:2.0e−5]" \
      ga_green.fits
```

And finally:

```
unix% dmcopy \
      "1451_spec_evt2.fits[tg_d=−0.000663889:0.000663889,tg_lam=1:25.005,energy=1700:9999]" \
      blue_filter.fits

unix% dmcopy \
      "blue_filter.fits[bin tg_lam=1:25.00:0.04,tg_d=−1.0e−3:1.0e−3:2.0e−5]" \
      ga_blue.fits
```

Use `dmimg2jpg` to combine the files into a single RGB image; here we supply all the parameters on the command line:

```
unix% dmimg2jpg infile=ga_red.fits greenfile=ga_green.fits \
      bluefile=ga_blue.fits outfile=ga_1451.jpg showgrid=no \
      showaimpoint=no
```

The resulting spectrum is shown in Figure 2  .

Real vs. Virtual Image Files

It is also possible to use virtual files as input to `dmimg2jpg`, bypassing the dmcopy steps. To repeat the sky example with virtual file syntax:

```
unix% punlearn dmimg2jpg
unix% pset dmimg2jpg infile="1451_spec_evt2.fits[energy=000:999][bin sky=2000:7000:8]"
unix% pset dmimg2jpg greenfile="1451_spec_evt2.fits[energy=800:2399][bin sky=2000:7000:8]"
unix% pset dmimg2jpg bluefile="1451_spec_evt2.fits[energy=1700:9999][bin sky=2000:7000:8]"
unix% pset dmimg2jpg outfile=virtual_1451.jpg
unix% pset dmimg2jpg showaimpoint=no showgrid=no
unix% dmimg2jpg
Input file name (red image if true color) (1451_spec_evt2.fits[energy=000:999][bin sky=2000:7000:8]):
Green color channel file name (1451_spec_evt2.fits[energy=800:2399][bin sky=2000:7000:8]):
Blue color channel file name (1451_spec_evt2.fits[energy=1700:9999][bin sky=2000:7000:8]):
Output jpg file name (virtual_1451.jpg):
```

Using this method is useful if disk space is an issue, since the intermediate files aren't written out.

```
Parameters for /home/username/cxcds_param/dmimg2jpg.par


#-------------------------------------------------------------------------
# dmimg2jpg.par                 Parameter file for dmimg2jpg tool
#-------------------------------------------------------------------------
# Input files.  infile = redfile for true color output
#-------------------------------------------------------------------------
        infile = sky_red.fits     Input file name (red image if true color)
     greenfile = sky_green.fits   Green color channel file name
      bluefile = sky_blue.fits    Blue color channel file name
#-------------------------------------------------------------------------
# Output files.  Currently to get a postscript file you need
# to make the jpeg file.
#-------------------------------------------------------------------------
       outfile = sky_1451.jpg     Output jpg file name
#-------------------------------------------------------------------------
# In greenfile == 'none'  use a lookup table
#-------------------------------------------------------------------------
      (lutfile = )lut.grey -> /soft/ciao/data/grey.lut) Colormap file
 (colorstretch = 1)               Color lookup strech factor
   (colorshift = 0)               Color lookup table shift
       (invert = no)              Invert colors
#-------------------------------------------------------------------------
# Scaling functions.  Use INDEF to get min and max of data
#-------------------------------------------------------------------------
(scalefunction = log)             Scaling function(log|linear|power)
   (scaleparam = 3)               Scaling parameter (for non-linear scalefunction)
       (minred = INDEF)           Minimum value for the red color channel
     (mingreen = INDEF)           Minimum value for the green color channel
      (minblue = INDEF)           Minimum value for the blue color channel
       (maxred = INDEF)           Maximum value for the red color channel
     (maxgreen = INDEF)           Maximum value for the green color channel
```

```
      (maxblue = INDEF)           Maximum value for the blue color channel
#-------------------------------------------------------------------------
# Regions, aimpoint and grid overlay
#-------------------------------------------------------------------------
   (regionfile = )                Region overlay file
  (regioncolor = )colors.green -> 0 1 0) Region color triple
    (regionopt = individual)      Option of region shape drawing method (individal|combine)
 (showaimpoint = no)              Put crosshair at aimpoint
    (showlabel = no)              Label the contours?
     (showgrid = no)              Show grid on image
    (gridcolor = )colors.white -> 1 1 1) Grid color triple
     (gridsize = 120)             Gridsize [arcsec]
     (fontsize = 2)               Font label size
       (psfile = )                Optional post script file name
      (verbose = 0)               Level of verbose output
      (clobber = no)              Clobber existing outputs?
         (mode = ql)
```

History

03 Jan 2005	reviewed for CIAO 3.2: no changes
05 Dec 2005	updated for CIAO 3.3: workaround for a Data Model bug was added to the Grating Angular Coordinates section
01 Dec 2006	reviewed for CIAO 3.4: no changes

URL: http://cxc.harvard.edu/ciao/threads/grating_color/ Last modified: 1 Dec 2006

Image 1: Color spectrum in sky coordinates

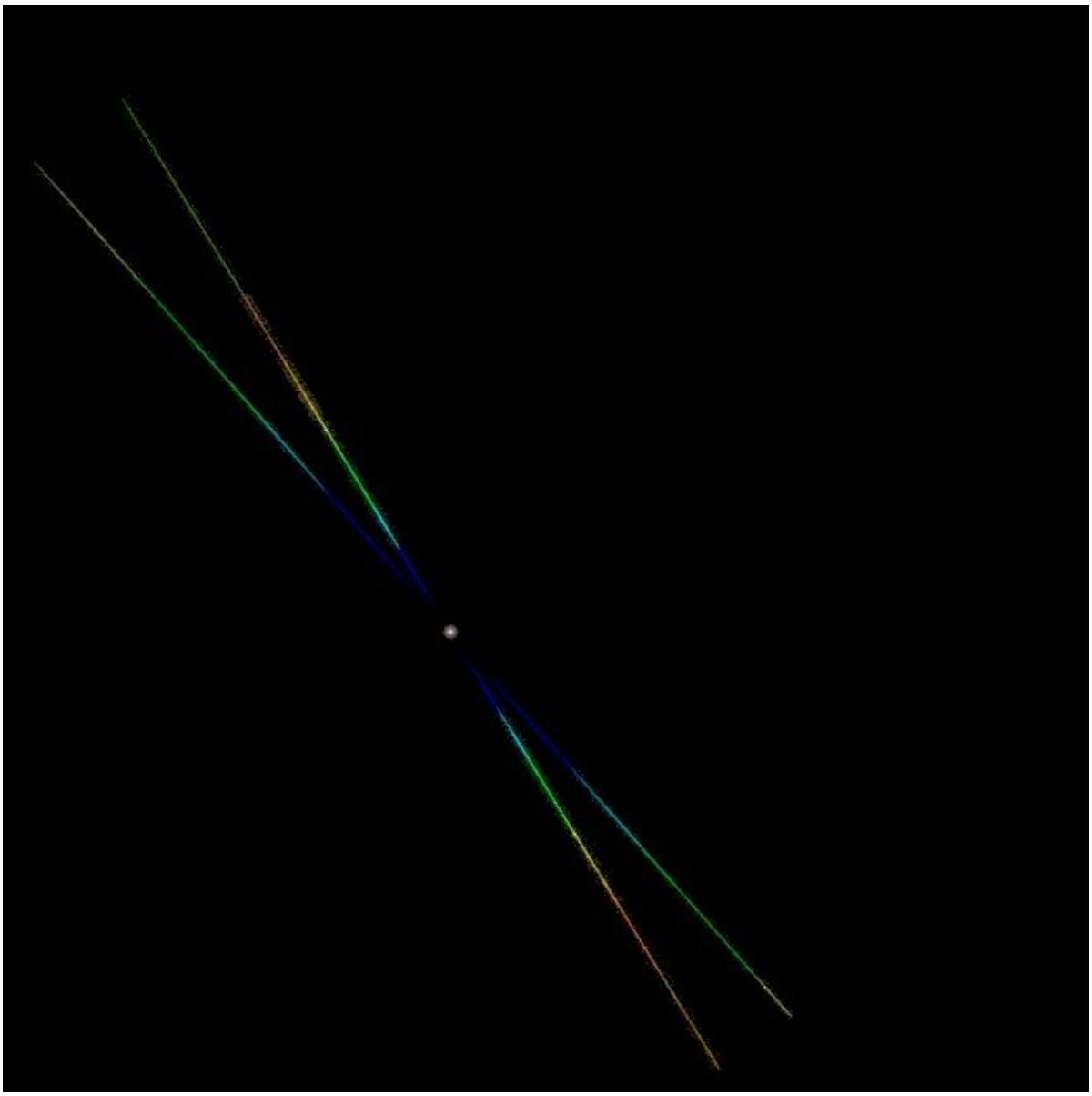


Image 2: Color spectrum in grating angular coordinates

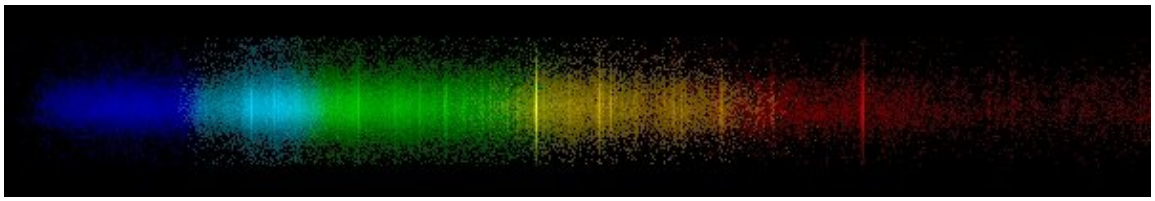


Image 2: Color spectrum in grating angular coordinates