xcolorcod

August 16, 2005

Abstract

Generate three-color coded spatial image from scalar event attribute

1 Instruments/Modes

Instrument	Mode

2 Use

pipeline processing	no
interactive analysis	yes

3 Description

xcolorcod generates from a given input event list a spatial image in which the value of a specified scalar event attribute (e.g. energy) is three-color coded. The coloring process is driven by three fundamental color curves (red, green, blue) that are constructed from the contents of an input color table.

The following examples demonstrates how the color of a pixel in the final image is computed. It is assumed there are only two events and the image should be color-coded using the event energy as scalar attribute: One pixel receives one photon of 1 keV and the other receives two photons of 1 keV. Assume that in the color table 1 keV corresponds to the color $(r, g, b) = (255, 128, 0)$. After adding all events the first pixel gets color $(255, 128, 0)$ and the second $(510, 256, 0)$. Then the image is normalized: Divide by the largest value in the image and multiply by 255. Now the first pixel has color $(128, 64, 0)$ and the second $(255, 128, 0)$. Note that they have the same color but a different intensity. So the color indicates the distribution of energy, the intensity indicates the flux.

Some more examples using the following color table:

1 keV: (255,128,0)
8 keV: (0,50,100):

photons times energy	corresponding color	normalized against a global max of 1800
1x1keV	255 128 0	36 18 0
2x1keV	510 256 0	72 36 0
1x8keV	0 50 100	0 7 14
1x1keV,1x8keV	255 178 100	36 25 14
2x1keV,2x8keV	510 356 200	72 50 28

Please note: The actual generation of the red, green, and blue component images is done through the task **evselect**. **xcolorcod** therefore inherits all of **evselect**'s image extraction parameters which allows to control the image generation process, e.g., binning, windowing, etc.

3.1 Color tables

xcolorcod provides a list of pre-defined color tables that is identical to the internal color maps of the image displayer Ds9. These tables can simply be selected by their names, e.g. `heat`, `cool`, etc., given as value of the task parameter `colortable`. For a complete list of the available color tables invoke the task in dialog mode (`xcolorcod -d`) and select the choice widget labeled `colortable`.

It is also possible to provide an external color tables in the form of a data set. The set needs to consist of a table named `COLORTABLE` with the following four scalar columns:

name	type	allowed range	remark
`LEVEL`	`E` (real32)	$0 \leq$ `LEVEL` ≤ 1	the scalar value
`RED`	`E` (real32)	$0 \leq$ `RED` ≤ 1	red component
`GREEN`	`E` (real32)	$0 \leq$ `GREEN` ≤ 1	green component
`BLUE`	`E` (real32)	$0 \leq$ `BLUE` ≤ 1	the blue component

Each rows defines a discrete point in the R/G/B color space that is to be associated with the scalar value `LEVEL`. The table can also be regarded as defining three separate curves for the three basic colors red, green, and blue and piecewise linear interpolation is carried out for intermediate `LEVEL` values.

As an example, here is the contents of the color table `heat`:

LEVEL	RED	GREEN	BLUE
0.00	0.0	0.00	0
0.34	1.0	0.34	0
0.65	1.0	0.65	0
0.98	1.0	0.98	0
1.00	1.0	1.00	0

So a pixel with a scalar value of e.g. 0.17 will be assigned a color with 50% red, 17% green, and no blue intensity.

The table can be generated from scratch with e.g. `fv`.

3.1.1 Conversion from Ds9 format

Another way to generate color tables is to convert Ds9 color maps to **xcolorcod** format with the two auxiliary converter tools `sao2xcolorcod/lut2xcolorcod`. It is possible to write Ds9 color tables to

an output files in either `SAOimage` (extension `.sao`) or `SAOtng/XImtool` (extension `.lut`) format (please consult the Ds9 documentation for more details on this). `sao2xcolorcod/lut2xcolorcod` read these data from *stdin* and take the name of the **xcolorcod** color table data set to be generated as only argument, e.g.

```
sao2xcolorcod rainbow.ds <rainbow.sao
```

4 Parameters

This section documents the parameters recognized by this task (if any).

Parameter	Mand	Type	Default	Constraints
table	yes	table		name of existing table

The table that contains the event data - need to at least contain the columns named with the `xcolumn`, `ycolumn`, `ecolumn` parameters.

colortablechoice	no	string	internal	internal|external

Choice parameter determining source of the color table that is to be used for the color coding. The value *internal* signifies that an internal table is to be selected via the `colortable` parameter. *external* activates the `colortableset` parameter.

colortable	no	string	heat	name of existing internal color table

If `colortablechoice=internal` the parameter signifies an internal color table that is the be used in the color-coding of the scalar event attribute. For a complete list of the available color tables invoke the task in dialog mode (`xcolorcod -d`) and select the choice widget labelled `colortable`.

colortableset	no	data set		name of exisiting data set

If `colortablechoice=external` the name of an external data set representing a color table that is to be used in the color coding of the scalar event attribute. The format must adhere to the specifications in Sect. 3.1.

invertcolortable	no	boolean	false	true|false

With this boolean parameter the color curves in the color table can be inverted, i.e., a component that is associated with the scalar value *level* is associated with 1-*level* in the inverted case.

xcolumn	no	column	X	name of existing column in `table`

The name of the column that specifies the x position of the event; passed to **evselect** as parameter `xcolumn`.

ycolumn	no	column	Y	name of existing column in `table`

The name of the column that specifies the y position of the event; passed to **evselect** as parameter `ycolumn`.

ecolumn	no	column	PI	name of existing column in `table`

The name of the column that specifies the scalar event attribute that is to be color-coded; passed to **evselect** as parameter `zcolumn`.

ecolumnminmax	no	string	actualminmax	

actualminmax|legalminmax|explicitminmax

In determining the actual color of a pixel via the color curves the value range of the scalar column `ecolumn` is mapped onto the color table's *level* column ($0 \leq level \leq 1$). The choice parameter `ecolumnminmax` selects one of three possible ways in which the dynamic value range $[min, max]$ of `ecolumn` is determined:

actualminmax : from the data itself

legalminmax : from the minimum/maximum legal values of the column given as column attributes `TLMIN/TLMAX`

explicitminmax : by explicit parameters `min` and `max`

The `ecolumn` data value range $[min, max]$ will be linerly mapped onto the *level* range $[0, 1]$.

min	no	real	0	

Explicit minimum value of `ecolumn` column if `ecolumnminmax`=*explicitminmax*

max	no	real	12000	

Explicit maximum value of `ecolumn` column if `ecolumnminmax`=*explicitminmax*

imagebinning	no	string	imageSize	imageSize|binSize

passed to **evselect** as parameter `imagebinning`

squarepixels	no	boolean	false	true|false

passed to **evselect** as parameter `squarepixels`

ximagesize	no	integer	600	> 0

passed to **evselect** as parameter `ximagesize`

ximagebinsize	no	real	1	> 0

passed to **evselect** as parameter `ximagebinsize`

withxranges	no	boolean	false	true|false

passed to **evselect** as parameter `withxranges`

ximagemin	no	real	1	> 0

passed to **evselect** as parameter `ximagemin`

ximagemax	no	real	600	> 0

passed to **evselect** as parameter `ximagemax`

yimagesize	no	integer	600	> 0

passed to **evselect** as parameter `yimagesize`

yimagebinsize	no	real	1	> 0

passed to **evselect** as parameter `yimagebinsize`

withyranges	no	boolean	false	true|false

passed to **evselect** as parameter `withyranges`

yimagemin	no	real	1	> 0

passed to **evselect** as parameter `yimagemin`

yimagemax	no	real	600	> 0

passed to **evselect** as parameter `yimagemax`

scale	no	string	log	log|lin

Whether the intensity should be displayed linearly or logarithmically.

decades	no	real	4.0	$[-10, 20]$

Number of decades to be used if `scale`=*log*.

outputchoice	no	string	dataset	dataset|ppmfile

If set to *dataset* image is written to a data set whose name is given via parameter `colorset`. Otherwise, image is written in PPM format to standard file named via `ppmfile`.

colorset	no	string	colimg.ds	name of data set

The name of the data set the color image shall be written to if `outputchoice`=*dataset*. Depending on the value of `ascube` the data will either be written to three separate arrays corresponding to the red, green, and blue components or three slices of a 3-D data cube in the primary array, respectively. The data set can be read and the contents displayed with Ds9.

ppmfile	no	string	stdout	name of file

If `outputchoice`=*ppmfile* the name of a PPM data file that the color image shall be written to. If `ppmfile`=*stdout* the data shall be written to standard output.

ascube	no	boolean	false	false|true

Boolean parameter determining whether the red, green, and blue component images are to be written as three separate array extensions to the data set designated with `colorset` or as three slices of a single 3-dimensional data cube in the primary array.

5 Errors

This section documents warnings and errors generated by this task (if any). Note that warnings and errors can also be generated in the SAS infrastructure libraries, in which case they would not be documented here. Refer to the index of all errors and warnings available in the HTML version of the SAS documentation.

MinMaxEqual *(error)*
The minimum and maximum values of the scalar attribute data are equal. This must not happen because in this case the mapping onto the color table *level* range $[0, 1]$ is undefined.

In addition all error and warning messages of the task **evselect** and package **dal** can occur.

6 Input Files

1. event file containing x and y position and scalar attribute of a set of events.
2. color set with a table that gives the intensities of red, green and blue as a function of a scalar value.

7 Output Files

1. PPM (portable pixmap) file or
2. data set containing either
   - three arrays with R/G/B components respectively
   - R/G/B images as slices in primary array

The data set format is readable by the image viewer Ds9 in version 2.3 or later. Generated PPM pixel maps can be visualized with the display program `xv`.

8 Algorithm

```
read RGB color curves
setup temporary table with columns red/green/blue
foreach event
  red = linearInterpolate(redcurve,energy)
  green = linearInterpolate(greencurve,energy)
  blue = linearInterpolate(bluecurve,energy)
foreach {red, green, blue}
  construct component image with evselect
combine partial images
if (log)
  foreach pixel
```

```
    r,g,b = max ( log(r,g,b) - log(maxValue) + decades, 0)
normalize to 255
if (withcolorset)
   write image to data set
else
   write image in PPM format
```

9 Comments

References