emosaic

January 12, 2017

Abstract

Mosaic EPIC images.

1 Instruments/Modes

Instrument	Mode
EPIC MOS+pn	Imaging

2 Use

pipeline processing	yes
interactive analysis	yes

3 Description

This task accepts an arbitrary number (maximum 999 in practice) of input images and combines them to produce a single mosaiced image. Exposure images may optionally be supplied in parallel with the raw event images, in which case the output image will be exposure corrected.

The task will only handle images which are a tangent plane projection of the sky, for example those created from the `X` and `Y` columns of EPIC event lists using `evselect`. Images from the optical monitor, which in their raw form contain a significant amount of distortion, could in principle be mosaiced after reprojection to the tangent plane projection.

The output image is the additive mosaic of the input images, in other words no average is performed. If the input image list is matched with a corresponding list of exposure maps, these exposure maps are additively mosaiced in parallel, then the output image is divided by the mosaiced exposure map. This approach is preferable to mosaicing exposure corrected images, as it helps avoid problems of poor counting statistics in areas of low exposure. Binary masks can be simulated by providing exposure images which have a pixel values of zero or one.

The output pixel size (in degrees) and world-coordinate system (WCS) reference coordinate for the output image are set to those of the first input image.

The task can read images in any of the **dal**-supported numeric data types. If all the input images have the same data type (and if `--withexposure` is not set), the output image also has that type; otherwise the output is written in real64. If `--withexposure` is set, this task also looks up the input `exposuresets` as well, then the datatype with a larger bytesize one (e.g., FLOAT rather than INTEGER) between input images and `exposuresets` is used for the type for the output image, providing that all the input sets have the same data type; otherwise the output is written in real64.

The task support FITS images which have their image array in the **PRIMARY** extension of the file. It also, exceptionally, supports XMM-Newton mask files, which hold the mask array in an extension called **MASK**. In the latter case, the output file also contains an extension **MASK** which holds the mosaiced mask array.

4 Parameters

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

Parameter	Mand	Type	Default	Constraints
imagesets	yes	dataset list		

List of input image datasets to be mosaiced

mosaicedset	yes	dataset		

Name of output mosaiced image dataset

withexposure	no	boolean	no	

Handle exposure images in parallel

exposuresets	no	dataset list		

List of exposure image datasets.

sampling	no	string	point	point—bilinear

Sampling method

forceuniformkwds	no	boolean	no	point—bilinear

Whether to use the same keyword forms in the case of 1 input image as are used when there is more than 1.

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.

wrongNumberOfImages *(error)*

Number of images and exposure images differs

badDims *(error)*
Only 2-D images are supported

sizesDiffer *(error)*
Image and corresponding exposure image differ in size

badType *(error)*
Image is of unsupported data type

tooMany *(error)*
More than 999 input images were supplied

6 Input Files

1. List of image files (image in primary header)
2. List of exposure images (image in primary header)

7 Output Files

1. Output image

8 Algorithm

```
Read parameters

Read each input image
Read corresponding exposure image (if required)

Set output projection (reference coordinate, pixel size) from the
   first input image

Project corners of each input image into output frame to find the
   bounds of the output image

Calculate size of output image in pixels from bounds and pixel size

Create output image

Loop over input images
   Loop over pixels in output image
      Project coordinate of output pixel into frame of input image
      Sample input pixel, add sample to output image
      Sample input exposure image pixel, add to output exposure
         image (if required)
   End loop
End loop
```

```
Divide output image by output exposure image (if required)

Write output image
```

9 Comments

- Should we allow the reference coordinate and pixel size of output image to be defined by input parameters?
- Current implementation of the re-projection is a little inefficient.
- Current implementation loads all image data into memory simultaneously. This could cause problems if mosaicing a large number of images.
- The task currently assumes that exposure images are listed in the same order as their raw photon twin, and the WCS keywords of exposure images are taken from the raw image.
- Task should be capable of writing output image as any (sensible) data type, chosen by input parameter.
- Currently only point-sampling is supported.
- Limits should be placed on the size of the output image.

References