eootepileupmask

June 2, 2019

Abstract

This task makes a mask image of the out-of-time event streaks and/or RGA spikes affected by the piled-up pixels, which include those piled-up pixels themselves.

1 Instruments/Modes

Instrument	Mode
EPIC	Imaging

2 Use

pipeline processing	yes
interactive analysis	yes

3 Description

3.1 Out-of-time events and pile-ups

Out-of-time (OOT) events are associated with any CCD instrument. In CCD instruments, events in each pixel are read by transferring those *laterally* from pixel to pixel (*i.e.*, frame-transfer) before they reach either the read-out node or storing region, which has no sensitivity for photons in the energy band of interest.

In most of the cases, this transfer (into a reading region) takes much shorter time than the actual exposure. However it still may take a significant time. If there is a bright enough source(s), a significant number of the photons from the source reach pixels during this frame-transfer stage. These happen in the following procedure. Suppose there is a single (bright) source with a flux of R [count/frame] (in the no-pileup situation) at a Y-axis position of $y = y_{\rm S}$, where the frame transfer is carried out in the direction of negative Y-axis[1] (towards $y = 0$) and the Y-axis in the imaging area has $y_{\rm Max}$ pixel. This frame is transferred into the reading (non-exposed) area, which takes a time of $t_{\rm F}$. Now a frame in the imaging area on the CCD is exposed for a given exposure, $t_{\rm E}$. It is then transferred to the reading area. During

[1]This is the definition of the RAWX/RAWY coordinates.

this transfer pixels at Y-axis positions between $y_{\rm S} < y < y_{\rm Max}$ are inevitably exposed to the source with the total exposure of

$$\frac{t_{\rm F}}{t_{\rm E}} \times \frac{y_{\rm Max} - t_{\rm S}}{y_{\rm Max}}. \quad (1)$$

In this frame transfer, new blank pixels are also transferred to the imaging area for the next exposure. In this transfer the pixels at Y-axis positions between $0 < y < y_{\rm S}$ are inevitably exposed to the source with the total exposure of

$$\frac{t_{\rm F}}{t_{\rm E}} \times \frac{t_{\rm S}}{y_{\rm Max}}. \quad (2)$$

Consequently all the pixels along the row are affected by the OOT events with the total rate of count per frame of

$$R \times \frac{t_{\rm F}}{t_{\rm E}}. \quad (3)$$

In the standard source detection scheme for EPIC in SAS (**emldetect**), this effect is taken into account, based on the detected count rates. However if a source is too bright and causes a significant pile-up, then the count rate of the source (R in the above notation) is underestimated, because the current source detection scheme works on the basis that there is no pile-up. As a result any estimate of parameters of the sources which are located at (or very close to) an OOT event streak is likely to be wrong[2].

This task creates a mask image, which represents the OOT event streaks affected by significant pile-ups, if `foroote`='yes'.

3.1.1 Note for the case where there is a too bright source

When a source itself is too bright, there will be a depression at and near the peak position of the source in the image proceesed in the standard way, because the pixels around the peak are so heavily piled up that most or all the events are detected with a bad grade (such as, Pattern 36 in MOSs) and have been filtered out before the image or event is created. In extreme cases those pixels may not contain hardly any event, hence are not regarded as piled-up pixels by this task, since the count per frame in those pixels would not exceed the threshold.

However even in those cases this task should work nicely, because (1) somewhere away from the peak position of the source, there must be pixels which are not as badly piled-up as the peak position, but show a large enough the count per frame rate to be recognised as piled-up, (2) the OOT event streaks are then defined based on these 'peripheral' pixels, (3) the area of those recognised piled-up pixels should be distributed in a roughly concentric area from the source, therefore they overall mask out the large chunk of area, which includes the source itself. Even in the case a source is so bright and causes so much pile-up that some entire rows in a CCD chip are not recognised as piled-up, it will not matter practically, because any of those pixels will not be recognised as a source in the (standard) source detection, hence there is practically no need to mask those positions, providing the mask file is used in conjunction with the result of a source detection.

3.2 RGA spikes

In front of the MOS detectors Reflection Grating Arrays (RGA) exist. The RGA splits some of the photons into RGS. However it also produces some reflected structures, called RGA spikes, detected on MOSs, which can be seen when there is a very bright source(s) in the field of view.

[2]This includes the bright source itself, which causes the OOT event. The source causes a significant pile-up, which is not handled in the current source detection scheme.

Unfortunately no quantative study has been made about the RGA spikes, such as the surface brightness and intensity or position of bright sources. However, it is known that the RGA spikes appear across the whole MOS, rather than a single chip like the OOT events, and it is parallel to the X-axes of chip 1 or the Y-axes of the other chips.

Although we do not know the quantative nature of the RGA spikes, the pixels which show significant pile-ups are likely to cause significant RGA spikes. Hence, this task creates, if `forrgaspike`='yes', a mask image, which represents the RGA spike streaks caused by heavily piled-up pixels.

If there is a too bright source, the same argument as discussed in subsection 3.1.1 applies.

3.3 Relation between this task and epileupmask.

The algorithm to estimate pile-ups is basically the same as **epileupmask** (see its document for detail of the algorithm; that also describes the limitation in the estimate of pile-ups).

This task **eootepileupmask** offers some command-line options so that a user can get a piled-up pixel mask file etc., in addition to the OOTE mask etc. so that users do not have to run **epileupmask** again just to get them.

3.4 Behaviours and command-line arguments.

This task reads an event file as an input, estimating the piled up pixels, and outputs either the sky coordinate image or 3-dimensional chipcube (controlled by `outputstyle`).

In short, we recommend to specify and the sky coordinate image as an output, of which the reasons are described below.

1. if the output is the chipcube, the process runs as follows:
   The task calculates the normalised count-per-frame-per-pixel (=rate) map for each chip, taking account of the energy (PI) of each event. Then when a pixel is found to show a rate exceeding the given threshold, the entire row along either or both X and/or Y-axis of the pixel is masked depending whether the areas of OOTE events and/or RGA spikes are specified to be masked or not, and it is output.

2. if the output is the sky-coordinate image, the process runs as follows:
   The core of the routine is exactly the same as the above case. However, the coordinate conversion into sky-coordinates is not a trivial job, as the pile-up phenomenon is basically processed on the chip(raw) coordinates. Hence a user is allowed to specify the binned-attitude file. If this (`binnedattset`) is set, the task splits the event files into the given time bins (described in the file `binnedattset`), performs the above-described calculation (of the normalised rate map) for each of these time bin, converts each normalised rate map and mask map into the sky coordinates[3]. Finally the task calculate the logical sum of the mask file. In other words, all the pixels that experience the rate above the threshold in one of the time bin are masked. Note that frames with too short duration are not used in this calculation of rate and mask images (Controlled by `avcnttoprocess`) in this case. If a user specifies so (controlled by `attstyle`), the entire observation is used as one set — in that case, the spatial fluctuation of the spacecraft is not taken into account (the median attitude value is simply used for the coordinate conversion).

[3]By definition, the attitude of the spacecraft during the time-bin is stable *enough*. Therefore the produced sky coordinate image should be accurate *enough*.

4 Parameters

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

Parameter	Mand	Type	Default	Constraints
ratethreshold	yes	real	0.008	0 <=`ratethreshold`

The threshold of the rate, where the pixels having the rate larger than this value are regarded as 'bad'.

eventset	no	dataset		

Name of the input event list dataset used to construct the mask image. This parameter is read if `inputstyle`='evlist'.

masksset	no	dataset	ootemask.ds	

The name of the output mask image in sky coordinates is written to this file name. If `outputstyle`='raw', the output mask cube image in chip coordinates is written to this file name.

outputstyle	no	string	sky	sky—raw

If 'sky', the OOTE map is output in sky coordinates, to the file referred to by parameter `ooteimageset`. In this case a template set (`templateset`) is needed and the `attstyle` parameter is also read. If `outputstyle`='raw' on the other hand the output is written to a cube (in the `expcubeset` format) to the file pointed to by `ootecubeset`.

templateset	yes	dataset		

This parameter is read if `outputstyle`='sky'. This file should contain an image in the primary extension, which is used to define the pixel dimensions and World Coordinates of the output image.

attstyle	no	string	binnedset	binnedset—template

This parameter is read if `outputstyle`='sky'. To convert from chip to sky coordinates it is necessary to know the spacecraft attitude. However the attitude is never completely stable and may vary significantly during an exposure. In this case the net sky image must be a mosaic of components from different values of the attitude. A time series of attitude values (such as that made either by **attbin** or **evproject**) can be supplied to parameter `binnedattset` if `attstyle` is set to 'binnedset' (default). If it is judged that the attitude wander during the exposure does not exceed some small fraction of the image pixel dimensions, or if the binned attitude set is not available, then the user may choose to set `attstyle` to 'template' instead. In this case a single fixed value of attitude is read from *_PNT keywords in the template image header.

binnedattset	yes	dataset		

If `attstyle`='binnedset' the user should supply to the present parameter the name of a dataset which contains a time series of the spacecraft attitude variation during the exposure. The user should be aware of the fact that if an attitude (time) bin (specified in this file) is too small, the time bin is unlikely to be processed (see the description of `avcnttoprocess` below). Therefore it is recommended to create the `binnedattset`, allowing a relatively large attitude fluctuation, such as 1 arcsec, so that attitude bins are not too finely split.

avcnttoprocess	no	real	7.0	0 <=`ratethreshold`

When `attstyle`='binnedset', the supplied `binnedattset` file may provide time durations with very short exposures. In that case if a pixel happens to have an event during the short duration, the value of count per frame at that pixel in that duration is large and likely exceeds the threshold `ratethreshold`, and as a result that pixel will be masked as piled-up just because the pixel has one event at a wrong time. To avoid this from happening, any time duration with too short exposure should not be processed. This parameter gives the threshold for it, *i.e.*, only when there are reasonable number of frames in the time duration, in which the averaged count, corresponding to the piled-up threshold, exceeds this parameter `avcnttoprocess`, the time duration is processed to calculate the output mask. For example, if

`ratethreshold`=0.008 and `avcnttoprocess`=7.0, each time duration should have the (maximum) number of frames larger than 875 (=7/0.008) so as to be used by the process, which corresponds to ∼2.3 ks for the standard mode in MOSs. Note that because the single-event ratio at 12 keV in MOSs is ∼0.23 (0.54 for pn), `avcnttoprocess`=7.0 corresponds to just above 2 count per frame per pixel in MOSs for very high energy photons, after the grade-branching ratio is corrected.

withrateimage	no	boolean	no	yes—no

If this is true, the rate imagesets calculated are also output with the filename of `ratesetroot` followed by an attitude sequence number in the form of "%04d" with the suffix of `.ds` . If `outputstyle`='sky', the rate image set is in sky coordinates, whereas if `outputstyle`='raw', the rate set is the cube format in chip coordinates. Note that the rate is corrected for the event at the energy of 1.49 keV.

ratesetroot	no	dataset	rateset	

This is read only if `withrateimage`=true. See the section of `withrateimage` for detail.

withpileupmask	no	boolean	no	yes—no

If this is true, the pileup mask imagesets calculated are also output with the filename of `pileupmasksetroot` followed by an attitude sequence number in the form of "%04d" with the suffix of `.ds` .

pileupmasksetroot	no	dataset	pileupmask	

This is read only if `withpileupmask`=true. See the section of `withpileupmask` for detail.

foroote	no	boolean	yes	yes—no

If true, the out-of-time (OOT) events due to piled-up pixels are masked in the output.

forrgaspike	no	boolean	no	yes—no

If true, the RGA spikes due to piled-up pixels are masked in the output.

isinverted	no	boolean	yes	yes—no

If true, the bad pixels (namely affected by pile-up) in the output mask file are zero, if not, one. For the later use of the output mask file with **dpssflag**, `isinverted`=yes is more convenient (default).

withapproxpatfrac	no	boolean	yes	yes—no

True if the approximate form of the pattern fraction parameters is used.

withboresightfudge	no	boolean	yes	yes—no

Flip the sign of the boresight euler%psi. **This parameter will be removed** after the boresight is fixed.

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.

nullTemplateImageSetName *(error)*
The command-line option `templateset` must be specified when `outputstyle`=sky.

6 Input Files

1. (Mandatory) a dataset with an exposure cube (without vignetting) in the primary image extension. The output of task **eexpchipmap** is suitable. A description of the cube format can be found in the documentation of that task.
2. (Mandatory) A calibrated event list for the relevant EPIC camera, created by either **emchain** or **epchain** or alike.
3. (Only mandatory if `outputstyle`='sky') a FITS dataset, which contains an image in its primary extension. The name of this dataset should be supplied to parameter `templateset`. The output images (`maskset` and possibly `rateset` and `pileupmaskset`) are constructed so as to match the pixel dimensions and World Coordinates of `templateset`.
4. (Only mandatory if `outputstyle`='sky' and `attstyle`='binnedset') a file output by **attbin**, containing a table `ATT_BINS` with columns `TSTOP`, `RA`, `DEC`, `PA` and `IS_GOOD`. The table should also contain a `TIMEZERO` keyword.

7 Output Files

- If `outputstyle`='sky':
  1. `maskset`: The mask image/cube is written.
  2. `rateset` (only when `withrateimage`=yes): The rate (count per frame) image/cube is written.
  3. `pileupmaskset` (only when `withrateimage`=yes): The piled-up mask (namely neither OOTE nor RGA spikes) image/cube is written.

  These datasets contain the same keywords in the primary HDU as the template image, except for DSS-related keywords. Extra extensions in the template image are not propagated.
- If `outputstyle`='raw':
  1. `ootecubeset`: an OOTE-map cube is contained in the primary image extension.

  The format of this cube is described in the task documentation of **eexpchipmap**.

8 Algorithm

**********Not yet written.

9 Comments

References