eident

August 15, 2005

Abstract

This detection pipeline task checks for possible position misalignments between the PN, MOS1 and MOS2 detectors. The output set will contain the offsets between the instruments and will additionally contain a merged source list based on a correlation between the two or three input source lists.

1 Instruments/Modes

Instrument	Mode
EPIC	Imaging

2 Use

pipeline processing	yes
interactive analysis	no

3 Description

The *eident* tasks reads two or three source lists, one for each instrument (PN,MOS1, MOS2) and looks for misalignments between the two or three instruments, which may be present during the initial phase of the mission. Additionally, the offsets found will be used to make a merged source list, which maybe of use for non-standard setups of the detection pipeline. The resulting output set will contain offset keywords and a merged source list compliant with the format of the *emldetect* output set. The task also works if only two source lists (say MOS1 and MOS2) are present.

The offset optimization routine is similar to the routine in used by the task *eposcorr*, which optimizes the statistic $S = \sum_{i,j} exp(-\frac{1}{2}(\frac{d_{i,j}}{\sigma_{i,j}}))$, where i, j represent the source numbers of two different sets, $d_{i,j}$ is the distance between two sources and $\sigma_{i,j}$ the distance error. A basic difference between the two routines is that *eposcorr* uses sky coordinates, whereas *eident* uses pixel coordinates. Unlike for the task *eposcorr* we do not have to worry here about chance alignments, since most of the sources are likely to be detected by all three instruments. Since after a number of observations the positional offsets between the detector systems should be known, the offsets can also be set using input parameters. The error in the offsets is estimated using the second moment in the distribution of the statistic S around the maximum value and divided by $\sqrt{N}$, with N the number of matching sources.

For the merging of the three input source lists (one for each instrument) the following scheme was used. The tasks uses a data structure containing the relevant parameters for each source. Additionally there are two extra fields called *cross_ref* and *dist*, both are arrays with dimension 2. The cross_ref elements are initiliazed to zero, but when two sources from different data sets are near each other the cross_ref fields will point to each other in the following way:

- PN set: cross_ref[1] refers to a source in the MOS1 set.
- PN set: cross_ref[2] refers to a source in the MOS2 set.
- MOS 1 set: cross_ref[1] refers to a source in the PN set.
- MOS 1 set: cross_ref[2] refers to a source in the MOS2 set.
- MOS 2 set: cross_ref[1] refers to a source in the PN set.
- MOS 2 set: cross_ref[2] refers to a source in the MOS1 set.

The *dist* field contains the associated distance between two sources linked by the cross_ref fields. Sources in two lists are taken to be one source if their distance $d_{i,j}/\sigma i, j < maxerror$, *maxerror* being an optional parameter. If two possible counterparts are found, the closest one is considered to be the actual counterpart. If the MOS1 to MOS2 reference is not consistent with the MOS1 to PN resp. MOS2 to PN reference, preference will be given to the MOS1/2 to PN references (this preference is determined in the routine *write_set*).

The pixel coordinates (called in accordance with other detection tasks X_IMA/Y_IMA) are not updated, but instead the estimated offsets between the detectors are listed in the header. The sky coordinates and other derived quantities will be averaged. The output set will be used by *eboxdetect/emldetect* to correct offsets between the instruments and make a combined detection run. However, if for some reason the combined detection using all three instruments together does not work satisfactory, *eident* can be used to merge the three individual source lists (i.e. one for each instrument) into one, which can be read by the pipeline task *srcmatch* to correlate with the OM source list and produce the final summary source list.

4 Parameters

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

Parameter	Mand	Type	Default	Constraints
pnset	yes	filename		

File name for the fits binary table containing the PN source positions.

usepnset	no	Boolean	yes	

Controls whether the PN source list will be used as an input source list.

m1set	yes	filename		

File name for the fits binary table containing the MOS1 source positions.

usem1set	no	Boolean	yes	

Controls whether the MOS1 source list will be used as an input source list.

m2set	yes	filename		

File name for the fits binary table containing the MOS1 source positions.

m2set	yes	filename		

File name for the fits binary table containing the MOS1 source positions.

usem2set	no	Boolean	yes	

Controls whether the MOS2 source list will be used as an input source list.

mergedset	yes	filename		

File name for the output fits binary table containing the merged source list.

findoffsets	no	boolean	no	

If findoffsets=yes, the task will try to optimize for possible positions offsets between the three lists.

maxerror	no	real32	3	2 to 6

This parameter determines at which distance (in units of σ) two sources from two separate data sets are considered to be one source.

xoffsetpnm1	no	real32	0	-5 to 5

This parameter allows one to set the x-offset between PN and MOS1. If findofsets=yes, this value will be used as a starting value to find the optimal offset.

yoffsetpnm1	no	real32	0	-5 to 5

This parameter allows one to set the y-offset between PN and MOS1. If findofsets=yes, this value will be used as a starting value to find the optimal offset.

xoffsetpnm2	no	real32	0	-5 to 5

This parameter allows one to set the x-offset between PN and MOS2. If findofsets=yes, this value will be used as a starting value to find the optimal offset.

yoffsetpnm2	no	real32	0	-5 to 5

This parameter allows one to set the y-offset between PN and MOS2. If findofsets=yes, this value will be used as a starting value to find the optimal offset.

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.

Number of rows in data set is 0. *(fatal)*

One of the data sets is empty. *(fatal)*

Parameter pnset requires a PN data set. *(fatal)*

Parameter m1set requires a MOS1 data set. *(fatal)*

Parameter m2set requires a MOS2 data set. *(fatal)*

Sources do not seem to match, check input data *(warning)*

corrective action: Proceed, but the input data sets may not belong to one and the same observation

6 Input Files

The input files have all the same format and originate from the *eboxdetect*/*emldetect* tasks. The files are in fits format and contain sources detected in different energy bands and entries summarizing the source detection. *eident* only uses the summarizing source positions, i.e. having the value 0 in the *id_band* column.

1. pnset: source list with sources found in the PN images
2. m1set: source list with sources found in the MOS1 images
3. m2set: source list with sources found in the MOS2 images

7 Output Files

1. mergedset: contains data set merged from all three epic data sets. The column names will comply to the output set of task *emldetect*, so that they can be read by the task

srcmatch. The offsets found by *eident* can be found in the header under the keywords XOFFPNM1, YOFFPNM1, XOFFPNM2, YOFFPNM2, XOFFM1M2 and YOFFM1M2, where the last 4 characters indicate the instruments. The associated errors wil have the header names OERRPNM1, OERRPNM2 and OERRM1M2. The MAXERROR keyword in the header corresponds to the input parameter *maxerror* and gives the maximum allowed source separation in units of σ. Additionally there is the keyword N_SRCS which gives the reduced number of sources.

8 Algorithm

```
   subroutine eident

        call parameter and i/o routines

        if( findoffsets == .TRUE. ) then
           call find_offset( pn_list ,.., mos1_list, ..)
           call find_offset( pn_list ,.., mos2_list, ..)
end if

        call compare_src(  pn_list ,.., mos1_list, .., nm1)
        call compare_src(  pn_list ,.., mos2_list, .., nm2)
        call compare_src(  mos1_list ,..,mos2_list, ..,nm3)

        ! the following lines select only those mos1/mos2 pair sources
        ! that are not found in the pn list (unlikely but nevertheless!)
        nm3b = 0
        do i=1,nmos1
          j = mos1_list(i)%cross_ref(2)
          if( j /= 0 ) then
            if( mos1_list(i)%cross_ref(1) == 0) then
               nm3b = nm3b + 1
            else
               mos1_list(i)%cross_ref(2) = 0  ! only except a mos1 mos2 link
               mos2_list(j)%cross_ref(2) = 0  ! when they are not linked to pn
            end if
          end if
        end do

        ! determine the true number of sources:
        ! number of total sources - number of (unique) pairs
        nsrcs = npn + nmos1 + nmos2 - nm1 - nm2 - nm3b

        call write_set( pn_list, mos1_list, mos2_list)

   end subroutine  eident

   subroutine find_offset( list1,.., list2,.., xoffset, yoffset )

        LOOP over xoffsets
           LOOP over yoffsets
             does this xoffset, yoffset maximize the statistic (likelihood)?
   END LOOP
END LOOP
```

```
determine center of mass xoffset, yoffset around the
previously found best fitting xoffset / yoffset

   end subroutine find_offset

   subroutine likelihood(list1,.., list2,..)


      likelihood = 0
      LOOP over list2 srcs
         correct here for offset

          LOOP over list1 srcs

    dr2 = distance( list1 src, list1 src
            likelihood = likelihood + exp( -0.5 * dr2)

       END LOOP
    END LOOP

   end subroutine likelihood

   subroutine compare_srcs( list1, n1, list2, n2, xoffset, yoffset, rotangle,
nmatches)

      LOOP over elements list1
         list1(i)%dist(ref1)= 2.0* maxerror ! initialize
         LOOP over elements list2

          newx = offset corrected x of list2
          newy = offset corrected y of list2
          xerr= sqrt( list1(i)%x_err**2 +  list2(j)%x_err**2)
          yerr= sqrt( list1(i)%y_err**2 +  list2(j)%y_err**2)
          if ( xerr /= 0.0 .AND. yerr /= 0.0) then

             dx = (list1(i)%x - newx )/xerr
             dy = (list1(i)%y - newy )/yerr
             dist1 = sqrt(dx*dx + dy*dy )
             if( dist1 < maxerror .AND. list1(i)%dist(ref1) > dist1 ) then

                ! check if another src was already associated with source "j"
                k = list2(j)%cross_ref(ref2)
                if( k /= 0 ) then

                   ! check if this source is closer than source "j"
                   dist2 = list2(j)%dist(ref2)

                   if( dist1 < dist2 ) then   ! new pair is better
                      l = list1(i)%cross_ref(ref1)
                      if( l /= 0 ) then
                         list2(l)%cross_ref(ref2) = 0
                      end if
                      list1(i)%cross_ref(ref1) = j
                      list2(j)%cross_ref(ref2) = i
```

```
                  list1(i)%dist(ref1) = dist1
                  list2(j)%dist(ref2) = dist1
                  list1(k)%cross_ref(ref1) = 0
               end if
            else
               l = list1(i)%cross_ref(ref1)
               if( l /= 0 ) then
                  list2(l)%cross_ref(ref2) = 0
               end if
               list1(i)%cross_ref(ref1) = j
               list2(j)%cross_ref(ref2) = i
               list1(i)%dist(ref1) = dist1
               list2(j)%dist(ref2) = dist1
            end if

         end if
      end if

   end do
 end do

 nmatches = 0
 do i=1,n1
    if( list1(i)%cross_ref(ref1) /= 0 ) nmatches = nmatches + 1
 end do

end subroutine  compare_srcs

! to be found in eident_io.f90 :
subroutine write_set(filename,list1, n1, list2, n2, list3, n3)

   Loop over list1 elements
     copy record to output arrays

     if( ref1 = list1(element)%cross_ref(1) /= 0 ) then
        copy list2(ref1) record to output array
     if( ref2 = list1(element)cross_ref(2) /= 0 ) then
        copy list3(ref1) record to output array

   End Loop

   Loop over list2 elements
     if( list2(element)%cross_ref(1) == 0 ) then ! not already copied
       copy record to output arrays
     if( ref2 = list1(element)cross_ref(2) /= 0 ) then
       copy list3(ref1) record to output array
   End Loop

   Loop over list3 elements
     if( list3(element)%cross_ref(1) == 0 AND
         list3(element)%cross_ref(2) == 0 ) then ! not already copied
       copy record to output arrays
     end if
   End Loop
```

```
end  subroutine write_set
```

9 Comments

10 Future developments

- At the moment offsets are determined using a grid of offset values, but the accuracy can be much better than the grid size. However, in future we may replace the grid search by a more advanced maximization algorithm.
- At the moment no role angle optimization is performed. One reason for that is that the task using the output of *eident* cannot yet work with this. The other reason is that to include role angle optimzation a more advanced (three dimensional) maximization algorithm is likely to be necessary. However, role angle corrections itself are already partly implemented.

References