DETECT/LOCAL - technical documentation

DOE/MPE Dec 99

1. Overview

DETECT/LOCAL is the first step in the EXSAS source detection, and employs a sliding window technique to detect significant enhancements in an image. The sliding window consists of a 3x3-pixel detection cell (9 pixel source window), surrounded by a 5x5-pixel-frame for background estimation (16 pixel background frame).

The window is slid pixel-by-pixel over the image. For each position of the sliding window, the likelihood that the counts within the 3x3 source window do **not** come from the same statistical distribution as the 16-pixel background counts is calculated. If this likelihood exceeds a given treshold, it is stored into an internal array, the likelihood map. True sources ususally give a positive signal for more than one window position and therefore will show up as extended isolated isles of positive values in the likelihood map.

In a second step the central position of each of those isles is calculated by a center-of-mass algorithm and stored in a source list.

To account for extended sources or sources at the edge of the field with large point spread functions, the likelihood map calculation and the determination of positions is usually repeated for a specified number of runs. In each run the image is successively compressed by a compression factor of two. To save execution time and avoid spurious identifications, for each run the user can specify that only a part of the image is to be searched. These areas (search boxes) are properly defaulted for each experiment.

For the sources from the resulting source list approximate source parameters (like source counts, etc.) are determined, and the result is output to disk.

With a single call of DETECT/LOCAL up to six input images can be treated at once, resulting in up to six output source lists. The normal case and default, however, has been to handle only one image per call.

A 0/1 mask image can optionally be specified to exclude certain parts of the image.

Alternatively to the mask, an exposure map can be employed. With the exposure map information the counts from the 5x5 background frame are corrected for exposure differences compared to the 3x3 source window. This functionality has only recently been implemented due to a suggestion by W. Voges, to treat images with large exposure variations (e.g. survey images).

2. Parameter file

DETECT/LOCAL is called with a parameter file, containing the names of the input/output files and other parameters relevant for steering the process. Parameters which are seldom modified are only accessible when creating the parameter file with the *expert* option (extended parameters). Otherwise they are properly defaulted.

Example of a template parameter file for ROSAT PSPC:

```
!
! parameter file for      DETECT/LOCAL           TASK: DETELOC
!
 MISSION,DETECTOR   ROSAT,PSPC                            ! mission and detector
 OBS_MODE           POINTING                              ! pointing or survey
 NUM_IMAGES           1                                   ! number of images
 IMAGE_1            image1.bdf                            ! first  image
 IMAGE_2            image2.bdf                            ! second image
 IMAGE_3            image3.bdf                            ! third  image
 IMAGE_4            image4.bdf                            ! fourth image
 IMAGE_5            image5.bdf                            ! fifth  image
 IMAGE_6            image6.bdf                            ! sixth  image
 MIN_ML                   8.00                            ! minimum of ML
 MASK_FLAG           T                                    ! flag to use mask
 MASK               EXSAS_CAL:mask057.bdf                 ! mask or exposure map
 LDETECT_LIST_1     lslst1.tbl                            ! first  LDET source list
 LDETECT_LIST_2     lslst2.tbl                            ! second LDET source list
 LDETECT_LIST_3     lslst3.tbl                            ! third  LDET source list
 LDETECT_LIST_4     lslst4.tbl                            ! fourth LDET source list
 LDETECT_LIST_5     lslst5.tbl                            ! fifth  LDET source list
 LDETECT_LIST_6     lslst6.tbl                            ! sixth  LDET source list
!
!  extended list of parameters
!
 NUM_RUNS             4                                   ! number of search runs
 SEARCH_BOXES        61    76   160 9999    0    0        ! half box size [ima_pix]
 IMAGE_PRIORITY      16    4    1    0    0    0          ! priorities for images
```

MISSION,DETECTOR: mission and detector identifier, separated by comma. Internal variable *DLEXPD*.

OBS_MODE: keywords **POINTING** or **SURVEY** Internal variable *DLOBSM*.

NUM_IMAGES: number of images to be treated in one call. 1 to 6 possible. Internal variable *ICT_ECH*. If more than one image is specified, the images must have the same dimensions.

IMAGE_n: filename of image **n**, **n** = 1..**NUM_IMAGES**. Internal variables *IMAGE*(n).

MIN_ML: threshold for likelihood map. Likelihoods below this threshold are not entered in the likelihood map. Consequently, this is also a threshold for the resulting source list. Internal variable *DL_SNRM*.

MASK_FLAG: **T** (true) or **F** (false). If set to **T**, the image specified in parameter **MASK** is used as mask or exposure map. Internal variable *DL_LMAP*.

MASK: filename of optional mask or exposure map. Only active if **MASK_FLAG** = **T**. Must have the same dimensions as the input images. Internal variable *MEXBM*.

LDETECT_LIST_n: filenames of resulting output lists, **n** = 1..**NUM_IMAGES**. Internal variables *LSLST*(n).

NUM_RUNS: Number of runs. In each run the input image and mask will be compressed by a factor of 2. Up to six runs possible. Internal variable *DL_NRUN*.

SEARCH_BOXES: Up to six values, specifying the search areas for each run. Unit is (uncompressed) image pixels. The search area consists of a box, centered on the sky center (not necessarily on the image center), with a half box size of **SEARCH_BOXES** image pixels. The **SEARCH_BOXES** should be set taking the size and the variation of the PSF over the field into account. If set to a sufficiently large value (e.g. 9999 image pixels), the sliding window search will cover the whole image. This parameter replaces the formerly used parameters **DEAD_X_ROWS** and **DEAD_Y_ROWS**, which were defined as distances from the image borders(!) in units of compressed(!) image pixels. Therefore they were not applicable anymore for variable image sizes. Internal variables *DL_SBOX*(m).

IMAGE_PRIORITY: source list ID flags for each image, **n** = 1..**NUM_IMAGES**. This parameter is a remnant of the old SASS code. The **IMAGE_PRIORITY** value is added as flag column to each output source list, allowing identification of the original source list after merging of source lists. It also serves as priority flag for the order of merging of source lists in later commands like MERGE/SOURCES. As a rule of thumb – use high **IMAGE_PRIORITY** values for good quality data, low values for worse data. Usually the **IMAGE_PRIORITY** values are powers of two. Internal variables *DL_PRIO(n)*.

Template parameter files for each of the possible missions can be found in the *deteloc* source code directory.

The default name for the parameter file is *deteloc.epf*. Other names are possible though. If at execution time no parameter file is found, a default one for ROSAT,PSPC will be created.

To ensure, that each parameter is defaulted properly, even if missing in the parameter file, the routine *dlpari.for* reads the first two parameters **MISSION,DETECTOR** and **OBS_MODE**. Based on this, the other parameters are initialized in routine *dlini.for*. After that, *dlpari.for* looks for the other keywords in the parameter file and overwrites the default values for the found parameters.

3. I/O

The primary input file is the parameter file, either specified on the command line or by default *deteloc.epf*. The other I/O files are specified in the parameter file.

The parameter file is read with routine *dlpari.for*. The input of images and mask is handled in routine *exdlin.for*. The output of the source list is done in routines *edlout.for* and *tdllis.for*.

Input files

IMAGE_n: Up to 6 images. Images must be uncorrected. Any size, even non-square, is allowed. The images must have the usual EXSAS header.

MASK: Optional 0/1 mask or exposure map. Must be of the same size as the images **IMAGE_n** and must have the usual EXSAS header. Only active, if **MASK_FLAG** is set to **T**.

Output files

LDETECT_LIST_n: Up to 6 output source lists, depending on the number of input images. The format and content of the source lists is described in section 4.6.

4. Local source detection

4.1 Image compression

To account for large PSFs at the edge of the field and for extended sources, the detection is done in several runs with increasing image compression. The number of runs is defined in the parameter file **NUM_RUNS**. The first run is done on the uncompressed image; in the following runs the image and also the mask is subsequently compressed by a factor of two (2x2 pixels compressed into one) (see routine *ldetec.for* for the general implementation). The actual compression factor is retained in the variable **NCOM**. If the dimension of the image before compression is an odd number, the remaining pixel is thrown away.

The actual compression of the integer*4 image is done with subroutine *rcmpi2.for*, and the compression of the real*4 mask or exposure map is done in *rcmpr4.for*. Note that the name of routine *rcmpi2.for* (of library *xspagen*) is misleading – it actually compresses integer*4 arrays.

4.2 Sliding Windows

The sliding windows algorithm uses the compressed image and mask/exposure arrays.

For each run, *ldetec.for* calculates the pixels limits of the area, over which the sliding windows algorithm will be performed, taking into account the search box areas defined in the parameter file (**SEARCH_BOXES**), and leaving at least a 2-pixel margin at the image borders so that no sliding window can exceed the image borders. This information together with the arrays themselves is passed into routine *snrmap.for*.

snrmap.for contains the loops which slide the window over the search area. The window consists of a 3x3 source window (9 pixels) and a 5x5 background frame (16 pixels). For each window position, the following sums are calculated:

- *ISO*: number of photons in 3x3 source window (only pixels summed where mask>0)
- *MSO*: number of valid pixels in 3x3 source window (number of pixels where mask>0)
- *ESO*: mask/exposure sum within 3x3 source window
- *IBK*: number of photons in background frame (only pixels summed where mask>0)
- *MBK*: number of valid pixels in background frame (number of pixels where mask>0)
- *EBK*: mask/exposure sum within background frame

To sort out spurious detections due to bad photon statistics, the following (hard-coded) criteria must be met to continue:

- $ISO \geq 6$
- $MSO \geq 3$
- $MBK \geq 12$

If every criterion has been met, the likelihood of source existence is calculated, using *ISO*, *ESO*, *IBK* and *EBK*.

4.3 Likelihood map

As described above, for each window position *ISO*, *ESO*, *IBK* and *EBK* are passed to the subroutine *snrldt.for* to calculate the likelihood of source existence for this position. The likelihood is then saved into an array with the same dimension than that of the image, the likelihood map (in the source code sometimes erroneously referred as SNR map).

The likelihood of source existence is defined as $-\ln(p)$. p is the probability, that the source counts *ISO* and the background counts *IBK* do come from the **same** Poissonian distribution, disregarding the actual expectation value of this distribution. The exact way for obtaining p is the calculation of a binomial probability. A good approximation in case of larger count values is the calculation of a χ^2 value and transforming that into a probability. Which method is taken depends on the count values. If both *ISO* and *IBK* are below 99, the binomial probability is used. In the other case, the χ^2 approach is used. Both methods relay on pre-calculated and tabulated values to save execution time.

Binomial probability (subroutine *pbinom.for*): The binomial probability (already expressed as likelihood) has been pre-calculated for *ISO* in [0..99] and *IBK* in [0..99], and available as table *pbin.tbl* in the EXSAS calibration area. This calculation has been done assuming a $\frac{9}{16}$ ratio between valid source window and background frame pixels. Masking and exposure differences change this ratio, so a correction is applied in *pbinom.for* to restore this ratio:

$$IBK2 = int(IBK * \frac{ESO}{EBK} * \frac{16.}{9.})$$

IBK2 is then used with *ISO* to get the likelihood from *pbin.tbl*. The described correction is mathematically not strictly correct since it destroys the Poissonian statistics, but it gives at least a more or less good approximation if $\frac{ESO}{EBK}$ is not far from $\frac{9}{16}$.

χ^2 **approximation** (subroutine *pchisq.for*): Within subroutine *snrldt.for* the χ^2 is calculated via:

$$\chi^2 = \frac{(ISO * EBK - IBK * ESO)^2}{ESO * EBK * (ISO + IBK)}$$

The χ^2 is then converted into a likelihood of source existence using pre-calculated values in table *pchsq.tbl*. *pchsq.tbl* contains the likelihoods corresponding to χ^2 values in [0..39.9] in steps of 0.1, and is available in the EXSAS calibration area. If χ^2 exceeds 40, the likelihood is calculated using the exponential approximation $L = 0.5 * \chi^2$.

I tested the continuity of the different methods (binomial probability, χ^2 table, χ^2 approximation) against each other, and found that they differ less than a few percent from each other, where their ranges of validity join. I did not test the values itself, however.

Now we have a value for the likelihood of existence for the relevant window position. If the likelihood exceeds the likelihood limit **MIN_ML** defined in the parameter file, the value is stored into the corresponding pixel of the likelihood map; otherwise the likelihood map pixel will be left at zero.

After a complete run over an image, mutual sources will show up as isolated likelihood islands in the likelihood map. The likelihood island for a marginal source will be made up of one pixel, the island for a brighter source can be quite extended. The next step will be to obtain source positions from the likelihood map.

4.4 Source identification

The relevant routines for finding the source positions are *match.for* and *cmass.for*. They work on the likelihood map.

match.for loops over the likelihood map. For every non-zero pixel encountered, *match.for* looks recursively for pixels with even higher likelihood in the immediate neighbourhood, until it has arrived at a local maximum. This maximum position is entered in the center-of-mass algorithm *cmass.for*. *cmass.for* calculates the center-of-mass in a ± 2 environment around this position, and returns the resulting refined position to *match.for*. *match.for* now checks this position against all previously identified source positions (from the actual and also the previous compression runs), and if the new-found position is at least 2 pixels (hardcoded) different either in x or in y from any previously found source position, it is added as a new source position (internal variables *XMS*, *YMS*) to the output list.

At the same time, *match.for* keeps track of the highest likelihood value encountered during this process, and adds this likelihood value (internal variable misleadingly called *ISNR*) to the source list, together the actual compression factor (internal variable *IWIND*).

Now we have positions, likelihood values, and a compression identification flag for the found sources.

4.5 Source parameters

The remaining source parameters are obtained in routines *srcbkg.for*, *abssrc.for*, and *absrib.for*.

srcbkg.for uses the positions determined in the previous step to search for the highest countrate in a 3x3 pixel environment in the image. The counts are obtained within a 3x3 source window surrounded by a 5x5 (16 pixel) background frame. The source and background counts are corrected additionally for PSF effects, assuming that 78% of the source counts fall into the source window, and 17% fall into the background frame. These percentages are hardcoded. Furthermore the algorithm assumes full exposure. No masking or exposure corrections are applied. The correction of the source window counts ISO and background frame counts IBK to the resulting corrected source counts SR and background counts per compressed pixel BK is done like this:

$$SR = \frac{16 * ISO - 9 * IBK}{16 * 0.78 - 9 * 0.17}$$

$$BK = \frac{0.78 * IBK - 0.17 * ISO}{16 * 0.78 - 9 * 0.17}$$

The background counts per compressed pixel are then converted to background counts per image pixel.

The problems with this approach is that the hard-coded percentages do not take into account the varying size of the PSF over the field-of-view, and the negligence of exposure differences. Therefore the resulting source counts are not very reliable.

abssrc.for calculates the distance of a source to the nearest other source in the source list.

absrib.src calculates the distance of a source to the nearest "black" pixel in the mask or exposure map. If a mask with ribs has been used, this corresponds to the distance of a source to the nearest rib. If no mask or exposure map has been specified, the corresponding column (**RIBD**) in the output list is set to zero.

There is one problem with *absrib.for*. The algorithm apparently searches for the nearest black pixel in the mask or exposure map in growing rectangles centered on the source position, and stops at the first black pixel encountered. To work correctly however, the algorithm should search in growing circles. This flaw was recently discovered by Rodrigo.

4.6 Source list

An example of a source list (EXSAS-MIDAS environment):

```
Sequence INDEX  X_POS     Y_POS     COUNT     BACKG      SIGNA     WINDO
-------- ------ --------- --------- --------- ---------- --------- -----
       1      1    283.22    205.22   6892.42  419.76807  23963.42     1
       2      2    254.00    244.00     14.10    0.00000     11.24     1
       3      3    284.00    273.00     27.12    0.64931     23.63     1
-------- ------ --------- --------- --------- ---------- --------- -----
Sequence SOURD     RIBD      PRIO  COF EXTFL     CLUFL     DUMMY
-------- --------- --------- ----- --- --------- --------- ---------
       1     46.00    176.78    16   0      0.00      0.00      0.00
       2     33.20    181.02    16   0      0.00      0.00      0.00
       3     41.73    181.02    16   0      0.00      0.00      0.00
-------- --------- --------- ----- --- --------- --------- ---------
Sequence X_SKY     Y_SKY
-------- --------- ---------
       1    801.16  -1538.97
       2    -75.50   -375.50
       3    824.50    494.50
-------- --------- ---------
```

The contents of the source list are described below:

```
Col.#   1:INDEX           Unit:unitless         Format:I6     I*4
Col.#   2:X_POS           Unit:image_pixel      Format:F9.2   R*4
Col.#   3:Y_POS           Unit:image_pixel      Format:F9.2   R*4
Col.#   4:COUNT           Unit:unitless         Format:F9.2   R*4
Col.#   5:BACKG           Unit:1/image_pixel    Format:F10.5  R*4
Col.#   6:SIGNA           Unit:unitless         Format:F9.2   R*4
Col.#   7:WINDO           Unit:3x3_image_pixel  Format:I5     I*4
Col.#   8:SOURD           Unit:image_pixel      Format:F9.2   R*4
Col.#   9:RIBD            Unit:image_pixel      Format:F9.2   R*4
Col.#  10:PRIO            Unit:unitless         Format:I5     I*4
Col.#  11:COFL            Unit:unitless         Format:I3     I*4
Col.#  12:EXTFL           Unit:unitless         Format:F9.2   R*4
Col.#  13:CLUFL           Unit:unitless         Format:F9.2   R*4
Col.#  14:DUMMY           Unit:unitless         Format:F9.2   R*4
Col.#  15:X_SKY           Unit:sky_pix          Format:F9.2   R*4
Col.#  16:Y_SKY           Unit:sky_pix          Format:F9.2   R*4
```

INDEX: running index of the source (1..n)

X_POS: X-position of the source (c.f. section 4.4). Unit image pixels.

Y_POS: Y-position of the source (c.f. section 4.4). Unit image pixels.

COUNT: Corrected source counts (c.f. section 4.5).

BACKG: Corrected background counts per image pixel (c.f. section 4.5).

SIGNA: Likelihood of source existence (c.f. section 4.4). The name **SIGNA** is slightly misleading, since it is neither a signal nor a signal-to-noise ratio — it is a likelihood value.

WINDO: compression factor for the run, where the source was found (c.f. section 4.4).

SOURD: distance to nearest other source as calculated with *abssrc.for* (c.f. section 4.5).

RIBD: rib distance. Actually it is the distance to next "black" pixel in the mask or exposure map as calculated with *absrib.for* (c.f. section 4.5). For the problems with this routine see section 4.5.

PRIO: priority flag, as defined in the parameter file in parameter **IMAGE_PRIORITY** (c.f. section 2). Serves later as list identification flag and as priority for list merging.

COFL: continuation flag. Inactive here, but used in subsequent commands.

EXTFL: extent flag. Inactive.

CLUFL: cluster flag. Inactive.

DUMMY: dummy. Inactive.

X_SKY: X-position of the source (c.f. section 4.4). Unit sky pixels. Calculated from the image pixel position, using the descriptors **START**, **STEP**, **NPIX** of the input image. The calculation is done within *tdllis.for*, calling the general conversion routine *imasky.for*.

Y_SKY: Y-position of the source (c.f. section 4.4). Unit sky pixels. Calculated from the image pixel position, using the descriptors **START**, **STEP**, **NPIX** of the input image. The calculation is done within *tdllis.for*, calling the general conversion routine *imasky.for*.

5. Code status

The code is fully ftools-compatible. Executables for both ftools and MIDAS environment are available and running. The memory for the five internal image arrays is allocated dynamically (*dlmemo.for*), and the routines are rewritten to handle any image sizes up to hardware limits. Images can also be non-square, and the sky center is not required to coincide with the image center. The current version supports all ROSAT experiments. Lots of the code are shared with the related command DETECT/MAP.

What remains to be done:

- The method to correct source and background counts for PSF effects (routine *srcbkg.for*, c.f. section 4.5) relies on hard-coded constants and is generally unsatisfactory. It has been originally implemented for the ROSAT PSPC, using the standard image binning of 30 sky pixel per image pixel, and has never been properly adapted for other binsizes or other experiments, including the HRI. One could think of using the available point spread function routines to adapt these constants at runtime, but this is probably very programming and computation intensive. A simple solution is not in sight. As a result, the output source and background counts are unreliable.

- The routine to determine the distance to the nearest rib (*absrib.for*, c.f. section 4.5) is not optimal. It looks for the nearest black pixel in the map or exposure map in growing rectangles centered on the source, and not in growing circles as it should. A new fast algorithm has to be invented and implemented.

6. Implementation of other experiments

Required calibration files:

- 0/1 image to serve as a default mask,

Software changes:

- Prepare template parameter files with proper defaults, with special attention to parameters **NUM_RUNS** and **SEARCH_BOXES** to ensure adequate coverage of the field in the different compression runs. In case of doubt, let each compression run search the whole image (**SEARCH_BOXES** = 9999 9999 9999 ...). Note, that the search boxes are specified in units of uncompressed image pixels, and are always centered on the sky center (and not necessarily on the image center).

- Edit subroutine *dlini.for* and add the default values for the new experiment. The defaults should be the same as in the template parameter files.