Running celldetect

CIAO 3.4 Science Threads

Table of Contents

Running celldetect

CIAO 3.4 Science Threads

Overview

Last Update: 1 Dec 2006 − reviewed for CIAO 3.4: no changes

Synopsis:

`celldetect` uses a sliding square detect cell whose size is matched to the instrument PSF to search for statistically significant enhancements over the background. At each point where a cell is placed, a signal−to−noise ratio of source counts to background counts is computed. If this ratio is above the detection threshold, a candidate source is recorded.

Purpose:

To illustrate several ways to use the Detect tool `celldetect`.

Read this thread if:

if you want to detect sources in an ACIS or HRC observation. This tool works well in simple fields with well−separated point sources.

Related Links:

- Detect manual: other examples of using this tool are available in the "Running celldetect" chapter.

Proceed to the HTML or hardcopy (PDF: A4 | letter) version of the thread.

Getting Started

Sample ObsIDs used: 578 (ACIS−S, 3C 295); 1522 (ACIS−I, Trapezium Cluster)

File types needed: evt2; asol1

The filtered event files, exposure maps, and output source lists for this thread are available in celldetect.tar.gz (28 MB tarred & gzipped; 68 MB unpacked).

For the first three examples, we are interested only in data on the S3 (ccd_id=7) chip of ObsID 578:

```
unix% dmcopy "acisf00578N002_evt2.fits[EVENTS][ccd_id=7]" s3_evt2.fits
```

The event list may be viewed in ds9:

```
unix% ds9 s3_evt2.fits &
```

Figure 1 shows the filtered dataset.

Note that the current maximum size allowed for an image is 2048x2048, but an event list may occupy a larger spatial region. See the Using Recursive Blocking section for more information.

Running celldetect

A Simple Example

To run `celldetect` with the basic parameters set, it is necessary to specify the infile and output file names only. Note that it is not required to create an ASCII version of the region file; if the `regfile` parameter is left blank, it will not be used.

```
unix% punlearn celldetect
unix% pset celldetect infile=s3_evt2.fits
unix% pset celldetect outfile=s3_src.fits
unix% pset celldetect regfile=s3_src.reg
unix% celldetect
Input file (s3_evt2.fits):
Output source list (s3_src.fits):
```

The contents of the parameter file may be checked using plist celldetect.

There are two source lists created, both containing the same sources: an ASCII file (`s3_src.reg`) and a FITS file (`s3_src.fits`). Both formats are fully described in the Detect manual.

To display the image with the source detections overlaid:

```
unix% ds9 s3_evt2.fits &
```

Load the source list from either file (Region −> Load Regions... −> `s3_src.fits[SRCLIST]` ***OR*** `s3_src.reg`). Note that FITS files ***must*** end in `.fits` for ds9 to recognize them. The results are shown in Figure 2.

A key parameter that you may wish to vary is `thresh`, the detect cell signal−to−noise (SNR) threshold; it was left at the default value of 3 in the above example. Cells whose SNR exceeds this value are identified as containing possible source candidates. Note that not all such cells will result in detected sources; if the `findpeaks` parameter is set to "yes" (the default), adjacent cells whose SNRs exceed `thresh` will be combined into a single detection. Simulations indicate that values of `thresh` as low as ~2 may be appropriate for short, on−axis observations, but it is recommended to start with a value of 3 or 4.

With An Exposure Map

False sources may be detected in the vicinity of significant exposure variations, such as detector edges or chip gaps. To mitigate this effect, run `celldetect` with an exposure map.

The exposure map used in this example is included in the `celldetect.tar.gz` file and was created by following the Compute an ACIS Exposure Map (Single Chip) thread. For the purpose of illustration, we used a monochromatic instrument map of 0.71 keV. When creating your own exposure map, be sure that the exposure map and input image are congruent (i.e. same size, image center, and binning factor).

When running the tool with an exposure map, the `expratio` parameter must be set. This is the ratio of the average exposure of the background frame around the detect cell to that in the cell. Detections for sources whose ratio falls below the specified value are omitted from the source regions file (`regfile`); all detections are present in the output source list (`outfile`) and the column `EXPO_RATIO` is created.

First, create an image to use as the input file:

```
unix% dmcopy \
      "acisf00578N002_evt2.fits[ccd_id=7][bin x=3500:4900:1,y=3650:5100:1]" \
      s3_image.fits
```

then, run the tool:

```
unix% punlearn celldetect
unix% pset celldetect infile=s3_image.fits
unix% pset celldetect outfile=s3_expmap_src.fits
unix% pset celldetect regfile=s3_expmap_src.reg
unix% pset celldetect expstk=s3_0.71keV_expmap.fits
unix% pset celldetect expratio=0.9
unix% celldetect
Input file (s3_image.fits):
Output source list (s3_expmap_src.fits):
```

The contents of the parameter file may be checked using plist celldetect.

To display the image:

```
unix% ds9 s3_image.fits &
```

Load the source list from the region file (Region −> Load Regions... −> `s3_expmap_src.reg`). Figure 3 shows the results from both runs; the blue ellipses are the results of using an exposure map and the green ellipses are the additional sources picked up when running without an exposure map.

Remember that only the results in the ASCII region file (`regfile`) are affected by the exposure map.

Using Recursive Blocking

As mentioned before, an event list may occupy a larger spatial region than the maximum image size (2048x2048). In such a case, `celldetect` *automatically* uses a "recursive blocking scheme": the inner 2048x2048 pixel region of the dataset is searched for sources, then the inner 4096x4096 pixel region (excluding the part already analyzed) is blocked by 2 and searched for sources, then the inner 8192x8192 is blocked by 4, etc., until `TLMIN, TLMAX` is reached.

For each consecutive search, the portion of the data that has already been searched for sources is excluded from the analysis. Therefore, only one blocking factor is used for each region on the dataset. Since a source could be missed if it straddles the border between regions of different blocking factors, each pass actually extends a bit into the previously analyzed region. Occasionally this leads to two detections of the same source, but such cases are automatically identified (see the output source list columns `DOUBLE` and `DOUBLE_ID`, discussed in the "`celldetect` Input Parameters & Data Products Reference Chapter" of the Detect manual).

For the recursive blocking scheme to operate, the `fixedcell` parameter must be set to zero and point−spread−function (PSF) information must be supplied via `psftable` (the tool will not examine a dataset past the radius for which PSF information is available). By default, both parameters are correctly set for recursive blocking of Chandra data.

Here we use the level=2 event file of ObsID 1522. The `DETNAM` keyword shows that the four ACIS−I chips (0−3) and two ACIS−S chips (6−7) were on for this observation:

```
unix% dmkeypar acisf01522N002_evt2.fits DETNAM echo+
ACIS−012367
```

A Data Model filter is added to the input datafile so that only the ACIS−I chips are used in the analysis:

```
unix% punlearn celldetect
unix% pset celldetect infile="acisf01522N002_evt2.fits[(ccd_id=0:3)]"
unix% pset celldetect outfile=acisi_block_src.fits
unix% pset celldetect regfile=acisi_block_src.reg
unix% celldetect
Input file (acisf01522N002_evt2.fits[(ccd_id=0:3)]):
Output source list (acisi_block_src.fits):
```

The contents of the parameter file may be checked using plist celldetect.

To display the event data with the source detections overlaid:

```
unix% ds9 acisf01522N002_evt2.fits &
```

Load the source list from either file (Region −> Load Regions... −> `acisi_block_src.fits[SRCLIST]` ***OR*** `acisi_block_src.reg`). Figure 4 shows the resulting image.

To see what recursive blocking was used on the event file, examine the `BLOCK` column of the FITS output file. This dmlist command gives the the position of each source and the the blocking factor used at that point:

```
unix% dmlist "acisi_block_src.fits[cols x,y,block]" data

--------------------------------------------------------------------------------
Data for Table Block SRCLIST
--------------------------------------------------------------------------------

ROW    POS(X,Y)                                 BLOCK

     1 (     3866.0912408759,     4058.8686131387)          1
     2 (     3868.6666666667,     4062.1358024691)          1
     3 (     3869.4761904762,     4150.0952380952)          1
.
.
.
   620 (     5145.5617283951,     3301.5740740741)          2
   621 (     5265.3390804598,     3343.9597701149)          2
   622 (     5282.8150684932,     3232.0205479452)          2
```

Using Recursive Blocking and Exposure Maps

Here we repeat the recursive blocking example with the addition of exposure maps.

One exposure map is needed to match each blocking factor that `celldetect` will use. There are two options for creating them:

1. Use the `acis_expmap` script. This script conveniently produces one exposure map for each blocking factor. However, it currently only produces exposure maps for a single energy of 1.5 keV.

   The most recent version of `acis_expmap` is v3.0.0 (6 June 2003):

   ```
   unix% egrep '(Version|Date)' `which acis_expmap`
   # Version:  3.3
   # Date: 27 September 2005
   ```

   Please check that you are using the most recent version before continuing. If you do not have the script installed or need to update to a newer version, please refer to the Scripts page.

   Executing the script without any input shows the syntax:

   ```
   unix% acis_expmap

    Usage:  acis_expmap evt.fits asol.list

    This script generates exposure maps for use with celldetect in
    recursive blocking mode.   It creates a stack, expmap.lis, of
    expmap_bl1.fits, expmap_bl2.fits, expmap_bl4.fits −− the 3
    blocking factors used by celldetect for centered ACIS−I images.

    acis_expmap expects two arguments:
    Event file                     (e.g. acisf00578N002_evt2.fits)
    List of aspect solution files  (e.g. asol.list)

    The script requires that a stack list be used as input, even
    if there is only one aspect solution file.  The aspect solution
    filenames are of the form 'pcad...asol1.fits'.  To generate a
    list of these files, use either of the following commands:

    % ls pcad* > asol.list
    % ls *asol1.fits > asol.list
   ```

   A. Make sure that you have set up ardlib to use the bad pixel file for your observation.
   B. The script requires as input the event file and the corresponding aspect solution (`asol1.fits`) files. The `asol1.fits` files must be listed in an ASCII file named `asol.list`:

      ```
      unix% ls pcad*asol* > asol.list
      unix% more asol.list
      pcadf070998448N002_asol1.fits
      ```

   C. Run the script:

      ```
      unix% acis_expmap acisf01522N002_evt2.fits asol.list
      ```

      Be aware that this script may take awhile to run; in this case, it ran for 4 hours on a SunBlade 100 with 128 MB of main memory.
   D. The script produces several files:

      ```
      unix% ls −1 expmap*
      expmap.lis
      expmap_bl1.fits
      expmap_bl2.fits
      expmap_bl4.fits
      ```

where `expmap_bl1.fits` is the exposure map blocked by 1, `expmap_bl2.fits` is blocked by 2, and so forth. Also, `expmap.lis` lists the exposure maps:

```
unix% more expmap.lis
expmap_bl1.fits
expmap_bl2.fits
expmap_bl4.fits
```

2. Alternatively, create the exposure maps by following the Compute Multiple Chip ACIS Exposure Map and Fluxed Image Step−by−Step thread. This option allows the user to produce exposure maps either for a monochromatic energy or by using spectral weights. The thread will need to be repeated for each blocking factor (typically 1, 2, and 4).

Once you have created the exposure maps, run `celldetect` on the ACIS−I chips:

```
unix% punlearn celldetect
unix% pset celldetect infile="acisf01522N002_evt2.fits[(ccd_id=0:3)]"
unix% pset celldetect outfile=acisi_block_expmap_src.fits
unix% pset celldetect regfile=acisi_block_expmap_src.reg
unix% pset celldetect expstk="@expmap.lis"
unix% pset celldetect expratio=0.99
unix% celldetect
Input file (acisf01522N002_evt2.fits[(ccd_id=0:3)]):
Output source list (acisi_block_expmap_src.fits):
```

The contents of the parameter file may be checked using plist celldetect.

DM syntax was once again used to specify only the chips of interest. The exposure maps are input to the `expstk` parameter; note the stack syntax (**@**) used with the list file.

The purpose of the `expratio` parameter is explained in the With an Exposure Map section.

Display the event data with the source detections overlaid:

```
unix% ds9 acisf01522N002_evt2.fits &
```

Load the source list from either file (Region −> Load Regions... −> `acisi_block_expmap_src.fits[SRCLIST]` ***OR*** `acisi_block_expmap_src.reg`). The results are shown in Figure 5. Placing these results on top of those from the recursive blocking without an exposure map example (Figure 6) shows that using an exposure map helps to eliminate spurious detections along the chip edges.

```
Parameters for /home/username/cxcds_param/celldetect.par


(comment lines have been omitted)

        infile = s3_evt2.fits     Input file
       outfile = s3_src.fits Output source list
       (expstk = )                list of exposure map files
      (regfile = s3_src.reg) ASCII regions file
       (kernel = default)         Output file format
      (clobber = no)              Overwrite exiting outputs?
       (thresh = 3)               Source threshold
```

```
    (findpeaks = yes)             Find local peaks?
     (centroid = yes)             Compute source centroids?
     (ellsigma = 3)               Size of output source ellipses (in sigmas)
     (expratio = 0)               cutoff ratio for source cell exposure variation
    (fixedcell = 0)               Fixed cell size to use (0 for variable cell)
      (xoffset = INDEF)           Offset of x axis from data center
      (yoffset = INDEF)           Offset of y axis from data center
        (eband = 1.4967)          Energy band
      (eenergy = 0.8)             Encircled energy of PSF
     (psftable = ${ASCDS_CALIB}/psfsize20010416.fits -> /soft/ciao/data/psfsize20010416.fits) Table
     (cellfile = )                Output cell size image stack name
      (bkgfile = )                Background file name
     (bkgvalue = 0)               Background count/pixel
  (bkgerrvalue = 0)               Background error
     (convolve = no)              Use convolution?
      (snrfile = )                SNR output file name (for convolution only)
      (verbose = 0)               Log verbosity level
          (log = no)              Make a celldetect.log file?
         (mode = ql)
```

```
Parameters for /home/username/cxcds_param/celldetect.par


(comment lines have been omitted)

        infile = s3_image.fits    Input file
       outfile = s3_expmap_src.fits Output source list
       (expstk = s3_0.71keV_expmap.fits) list of exposure map files
      (regfile = s3_expmap_src.reg) ASCII regions file
       (kernel = default)         Output file format
      (clobber = no)              Overwrite exiting outputs?
       (thresh = 3)               Source threshold
    (findpeaks = yes)             Find local peaks?
     (centroid = yes)             Compute source centroids?
     (ellsigma = 3)               Size of output source ellipses (in sigmas)
     (expratio = 0.9)             cutoff ratio for source cell exposure variation
    (fixedcell = 0)               Fixed cell size to use (0 for variable cell)
      (xoffset = INDEF)           Offset of x axis from data center
      (yoffset = INDEF)           Offset of y axis from data center
        (eband = 1.4967)          Energy band
      (eenergy = 0.8)             Encircled energy of PSF
     (psftable = ${ASCDS_CALIB}/psfsize20010416.fits -> /soft/ciao/data/psfsize20010416.fits) Table
     (cellfile = )                Output cell size image stack name
      (bkgfile = )                Background file name
     (bkgvalue = 0)               Background count/pixel
  (bkgerrvalue = 0)               Background error
     (convolve = no)              Use convolution?
      (snrfile = )                SNR output file name (for convolution only)
      (verbose = 0)               Log verbosity level
          (log = no)              Make a celldetect.log file?
         (mode = ql)
```

```
Parameters for /home/username/cxcds_param/celldetect.par
```

```
(comment lines have been omitted)

        infile = acisf01522N002_evt2.fits[(ccd_id=0:3)] Input file
       outfile = acisi_block_src.fits Output source list
       (expstk = )                list of exposure map files
      (regfile = acisi_block_src.reg) ASCII regions file
       (kernel = default)        Output file format
      (clobber = no)             Overwrite exiting outputs?
       (thresh = 3)              Source threshold
    (findpeaks = yes)            Find local peaks?
     (centroid = yes)            Compute source centroids?
     (ellsigma = 3)              Size of output source ellipses (in sigmas)
     (expratio = 0)              cutoff ratio for source cell exposure variation
    (fixedcell = 0)              Fixed cell size to use (0 for variable cell)
      (xoffset = INDEF)          Offset of x axis from data center
      (yoffset = INDEF)          Offset of y axis from data center
        (eband = 1.4967)         Energy band
      (eenergy = 0.8)            Encircled energy of PSF
     (psftable = ${ASCDS_CALIB}/psfsize20010416.fits -> /soft/ciao/data/psfsize20010416.fits) Table of PSF
     (cellfile = )               Output cell size image stack name
      (bkgfile = )               Background file name
     (bkgvalue = 0)              Background count/pixel
  (bkgerrvalue = 0)              Background error
     (convolve = no)             Use convolution?
      (snrfile = )               SNR output file name (for convolution only)
      (verbose = 0)              Log verbosity level
          (log = no)             Make a celldetect.log file?
         (mode = ql)
```

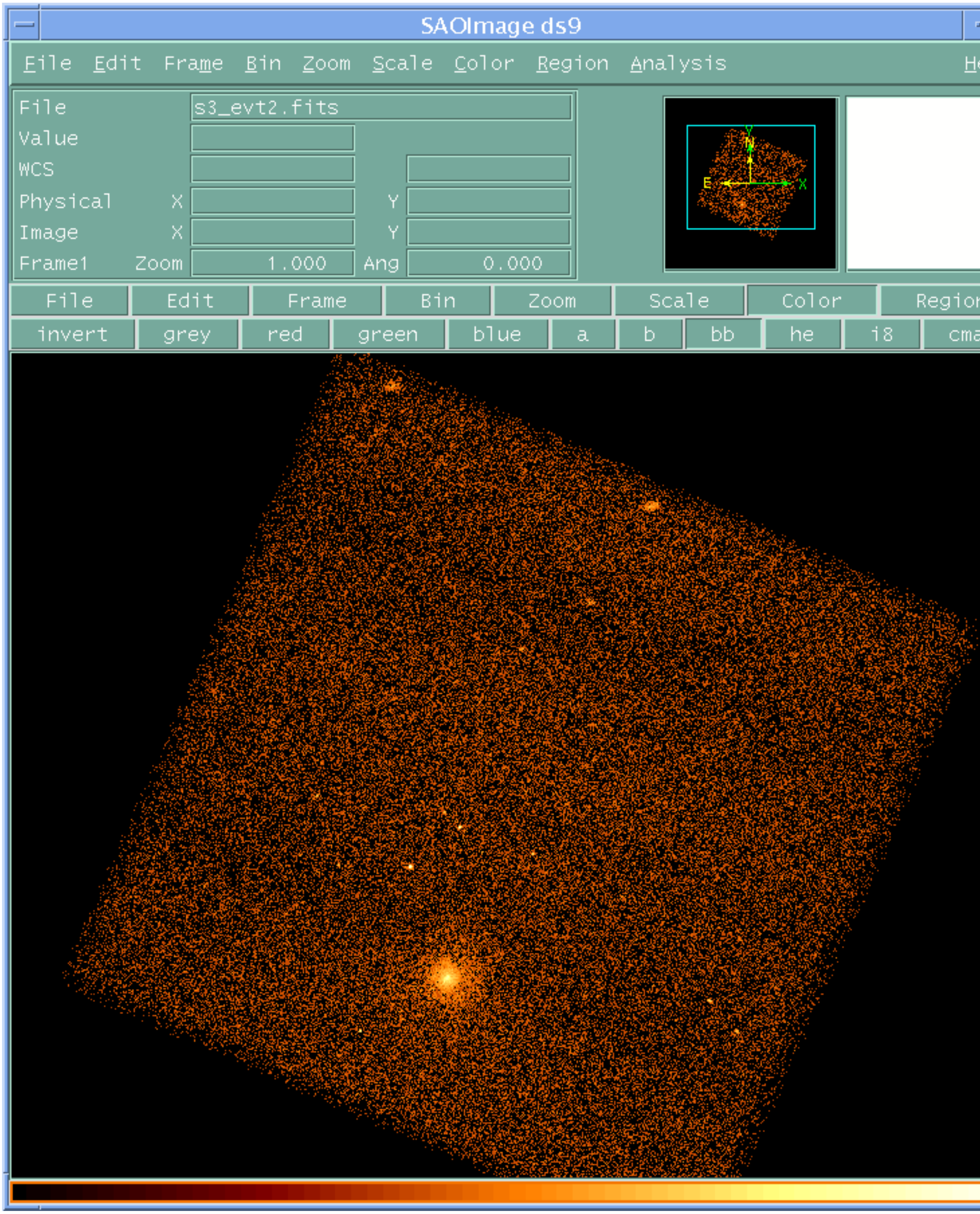
```
Parameters for /home/username/cxcds_param/celldetect.par


(comment lines have been omitted)

        infile = acisf01522N002_evt2.fits[(ccd_id=0:3)] Input file
       outfile = acisi_block_expmap_src.fits Output source list
       (expstk = @expmap.lis)     list of exposure map files
      (regfile = acisi_block_expmap_src.reg) ASCII regions file
       (kernel = default)        Output file format
      (clobber = no)             Overwrite exiting outputs?
       (thresh = 3)              Source threshold
    (findpeaks = yes)            Find local peaks?
     (centroid = yes)            Compute source centroids?
     (ellsigma = 3)              Size of output source ellipses (in sigmas)
     (expratio = 0.99)           cutoff ratio for source cell exposure variation
    (fixedcell = 0)              Fixed cell size to use (0 for variable cell)
      (xoffset = INDEF)          Offset of x axis from data center
      (yoffset = INDEF)          Offset of y axis from data center
        (eband = 1.4967)         Energy band
      (eenergy = 0.8)            Encircled energy of PSF
     (psftable = ${ASCDS_CALIB}/psfsize20010416.fits -> /soft/ciao/data/psfsize20010416.fits) Table of PSF
     (cellfile = )               Output cell size image stack name
      (bkgfile = )               Background file name
     (bkgvalue = 0)              Background count/pixel
  (bkgerrvalue = 0)              Background error
     (convolve = no)             Use convolution?
      (snrfile = )               SNR output file name (for convolution only)
      (verbose = 0)              Log verbosity level
          (log = no)             Make a celldetect.log file?
         (mode = ql)
```

History

03 Jan 2005	reviewed for CIAO 3.2: no changes
03 Jun 2005	updated links for CIAO 3.2 version of Detect manual
19 Dec 2005	updated for CIAO 3.3: the `acis_expmap` script has been updated to version 3.3 for the new `asphist` tool syntax; updated files in detect data tarfile
05 Jan 2006	update to `acis_expmap` usage message (script functionality is unchanged)
01 Dec 2006	reviewed for CIAO 3.4: no changes

URL: http://cxc.harvard.edu/ciao/threads/celldetect/ Last modified: 1 Dec 2006

Image 1: Event list filtered on chip S3 (ObsID 578)


SAOImage ds9
File Edit Frame Bin Zoom Scale Color Region Analysis
File
Value
WCS
Physical X Y
Image X Y
Frame1 Zoom Ang
File Edit Frame Bin Zoom Scale Color
invert grey red green blue a b bb he i8
E X Y N

File: s3_evt2.fits
Frame1 Zoom 1.000 Ang 0.000

Image 1: Event list filtered on chip S3 (ObsID 578)

Image 2: A simple example − chip S3 with detections overlaid (ObsID 578)

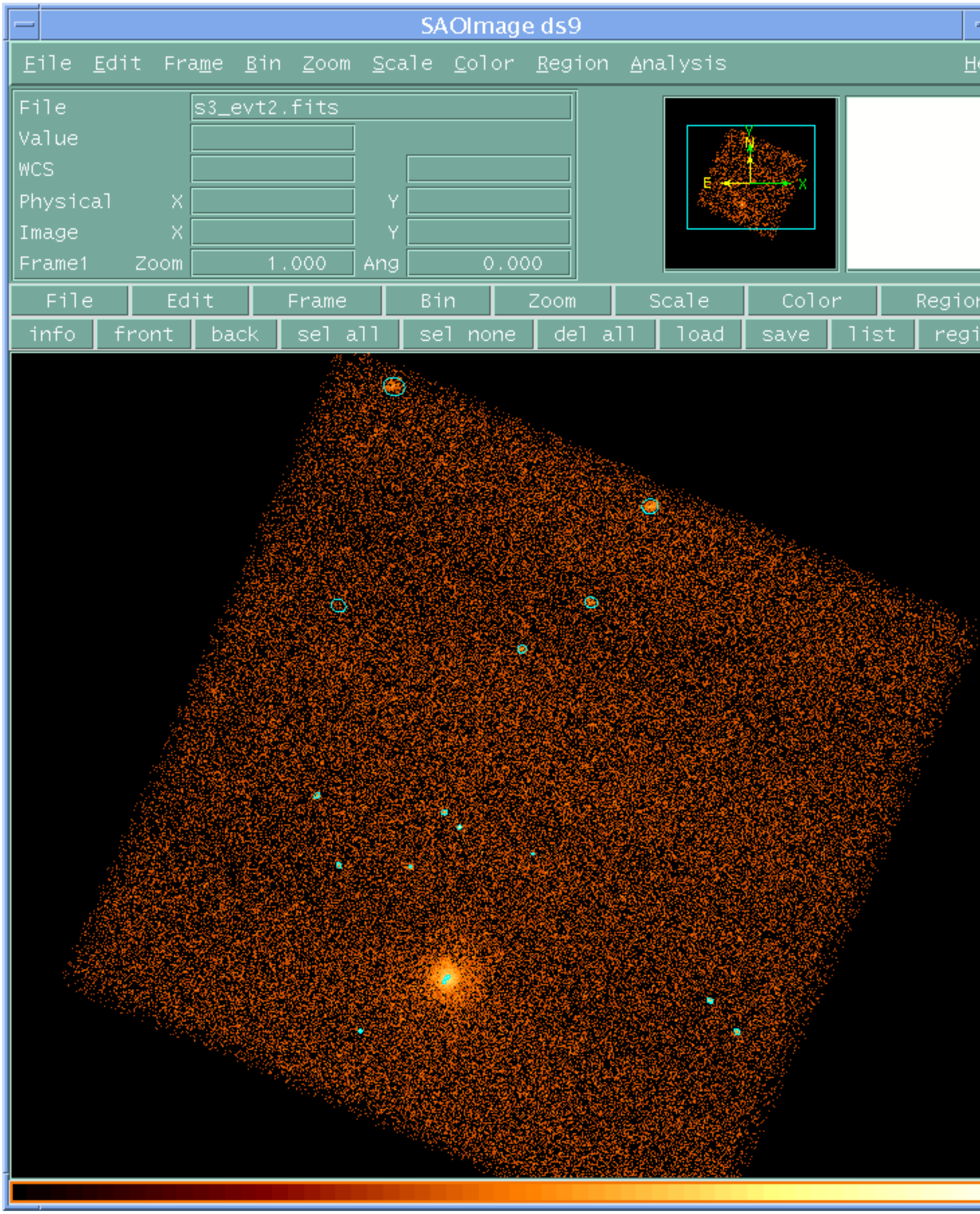
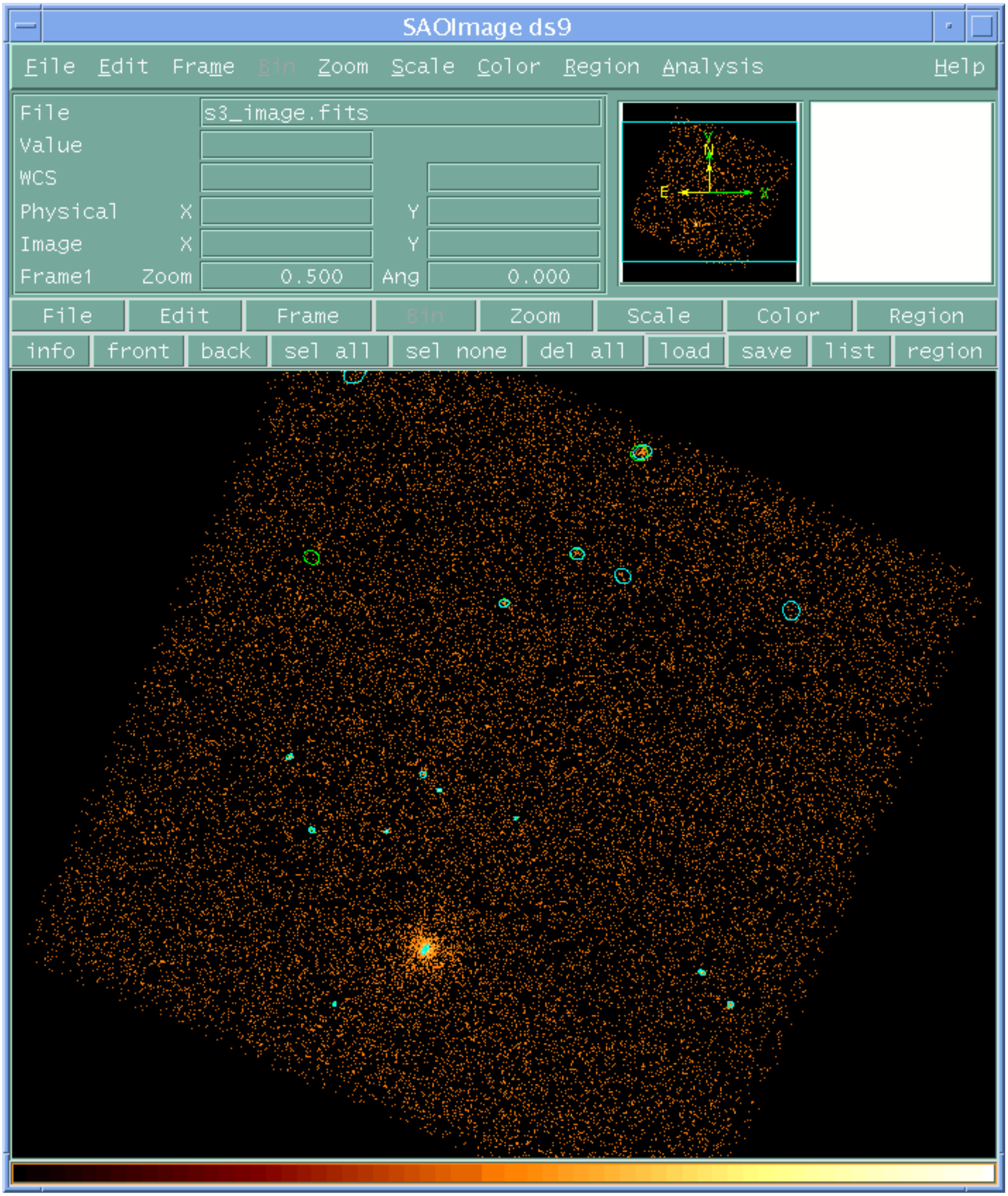

SAOImage ds9
File Edit Frame Bin Zoom Scale Color Region Analysis
File s3_evt2.fits
Value
WCS
Physical X Y
Image X Y
Frame1 Zoom 1.000 Ang 0.000
File Edit Frame Bin Zoom Scale Color
info front back sel all sel none del all load save list

Image 2: A simple example − chip S3 with detections overlaid (ObsID 578)

Image 3: Using an exposure map − chip S3 with detections overlaid (ObsID 578)

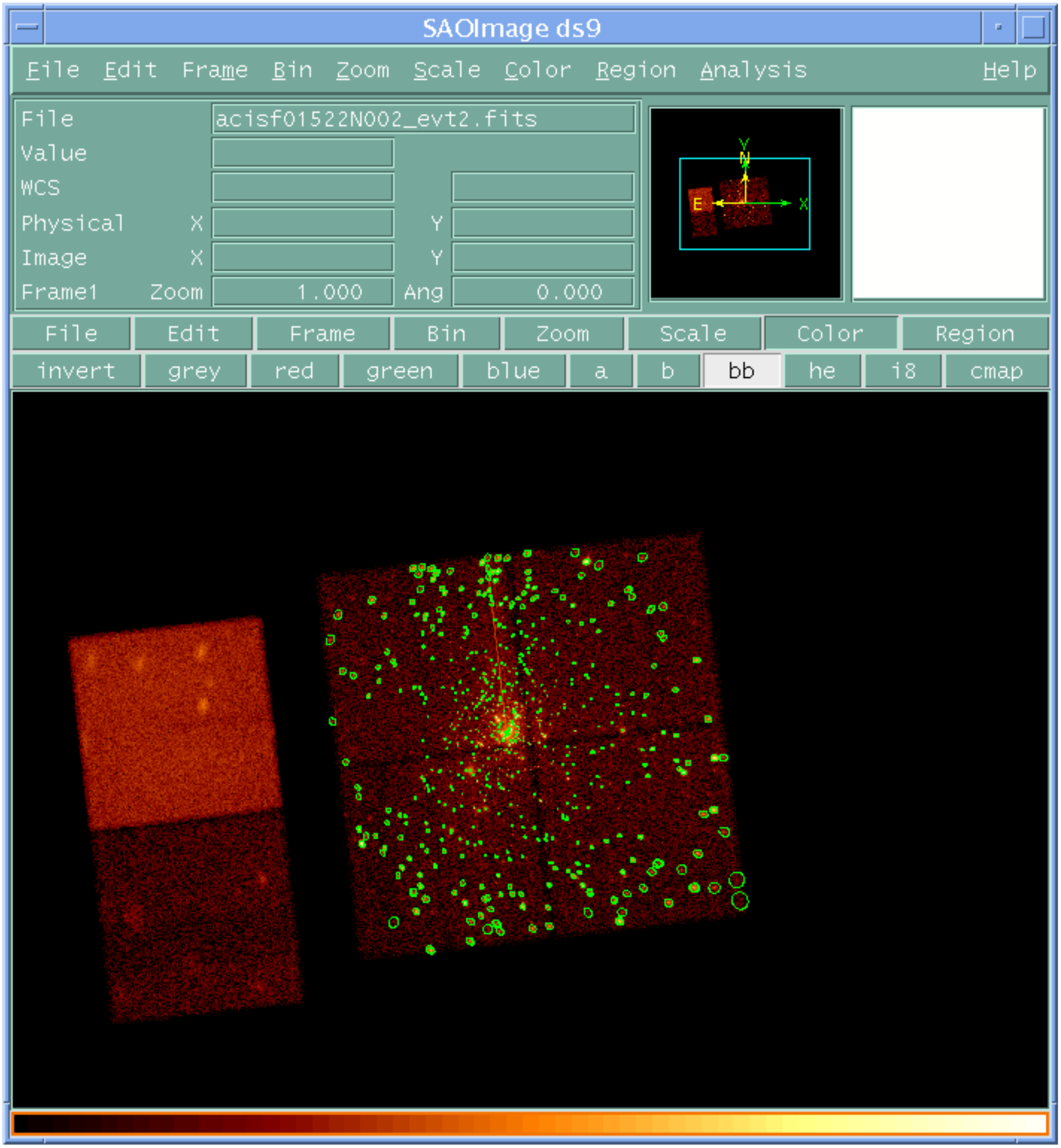
The blue ellipses are the results of using an exposure map (`s3_expmap_src.reg`) and the green ellipses are the additional sources identified from running without an exposure map (`s3_src.reg`).


SAOImage ds9
File Edit Frame Bin Zoom Scale Color Region Analysis Help
File
Value
WCS
Physical X Y
Image X Y
Frame1 Zoom Ang
N
E
X
Y
File Edit Frame Bin Zoom Scale Color Region
info front back sel all sel none del all load save list region

File: s3_image.fits
Zoom: 0.500
Ang: 0.000

Image 3: Using an exposure map − chip S3 with detections overlaid (ObsID 578)

Image 4: Recursive blocking − ACIS−I detections (ObsID 1522)

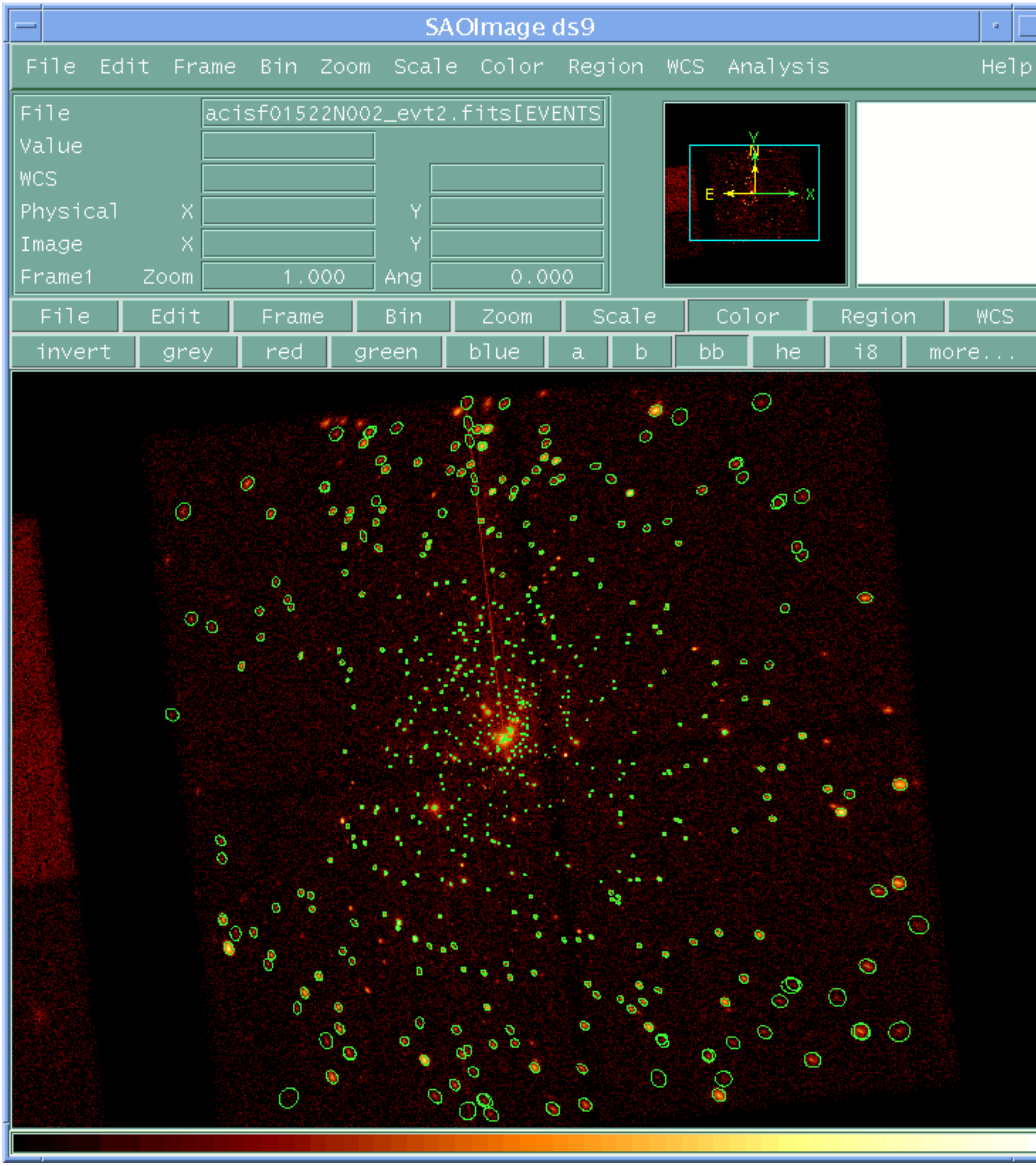

SAOImage ds9
File Edit Frame Bin Zoom Scale Color Region Analysis Help
File
Value
WCS
Physical X Y
Image X Y
Frame1 Zoom Ang
E N X Y
File Edit Frame Bin Zoom Scale Color Region
invert grey red green blue a b bb he i8 cmap

File: acisf01522N002_evt2.fits

Zoom: 1.000

Ang: 0.000

Image 5: Recursive blocking with exposure map − ACIS−I detections (ObsID 1522)

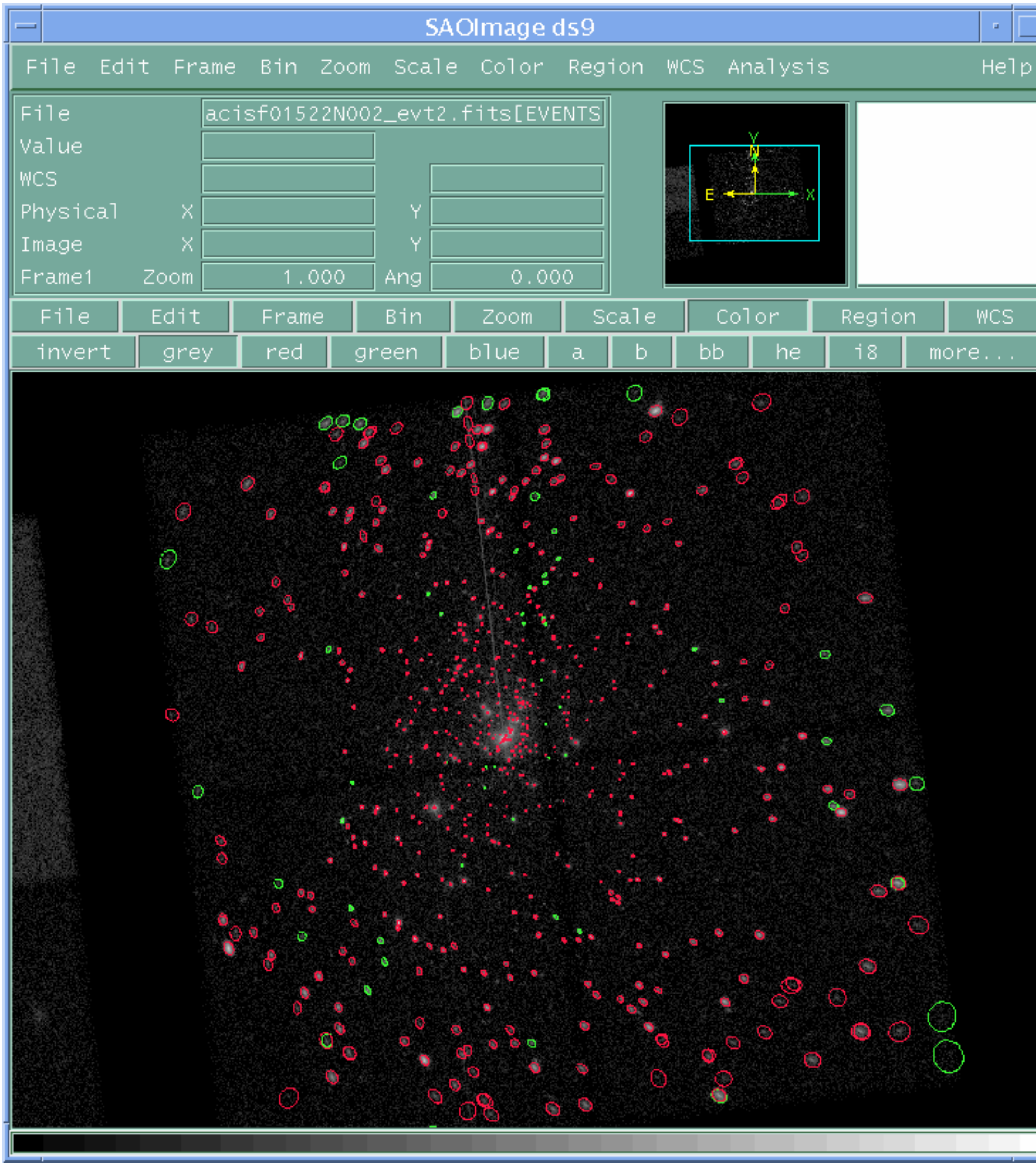

SAOImage ds9
File Edit Frame Bin Zoom Scale Color Region WCS Analysis Help
File
Value
WCS
Physical X Y
Image X Y
Frame1 Zoom Ang
File Edit Frame Bin Zoom Scale Color Region WCS
invert grey red green blue a b bb he i8 more...
Y N E X

File acisf01522N002_evt2.fits[EVENTS

Frame1 Zoom 1.000 Ang 0.000

Image 6: Comparison of recursive blocking results with and without an exposure map

The green ellipses are the results without an exposure map (`acisi_block_src.reg`) and the red ones are the results when an exposure map is used (`acisi_block_expmap_src.reg`).


SAOImage ds9
File Edit Frame Bin Zoom Scale Color Region WCS Analysis Help
File
Value
WCS
Physical X Y
Image X Y
Frame1 Zoom 1.000 Ang 0.000
File Edit Frame Bin Zoom Scale Color Region WCS
invert grey red green blue a b bb he i8 more...
Y N E X

acisf01522N002_evt2.fits[EVENTS

Image 6: Comparison of recursive blocking results with and without an exposure map

Image 6: Comparison of recursive blocking results with and without an exposure map