The *Detect* Reference Manual

CIAO Software Release v3.4

December 2006

http://cxc.harvard.edu/ciao/

Contents

List of Figures

List of Tables

Chapter 1

A Tour of Detect

1.1 Introduction

The detection of significant features in 2-D images is at the heart of much of astronomy. X-ray astronomy poses some special challenges (*e.g.* large sparse arrays, Poisson statistics) that mandate specially-developed methods for source detection. Moreover, while the extremely small beam area of the *Chandra* mission can resolve tight clusters of sources, it also means that more diffuse sources will be resolved, making those clusters more difficult to detect.

The *CXC* has responded to these needs by providing a suite of *Detect* tools within the Chandra Interactive Analysis of Observations (*CIAO*) software package: *celldetect*, *vtpdetect*, and *wavdetect*. These tools comprise three types of algorithms, each tailored to address the different source detection problems that the user of *Chandra* data may encounter. Table 1.1 gives a concise listing of and introduction to the *Detect* tools.

In this section, we provide a brief description of each tool and information on how to obtain and use the *CIAO* software.

1.1.1 celldetect

The best known detection algorithm for X-ray images is the "sliding cell" method developed for use with the *EINSTEIN* Observatory images and also employed by the standard processing of *Rosat* data. *celldetect* searches for sources by summing counts in square "detect" cells in the dataset, and comparing the summed counts to those of "background" frames. At each point where a cell is placed, a signal-to-noise (S/N) ratio of source counts to background counts is computed. If this ratio is above the detection threshold, a candidate source is recorded.

This method was tailored to optimize the detection of unresolved sources and had two variants, "local detect" and "map detect". In the former, the background is estimated in a frame around the detect cell; the latter requires a background map. The observed data may be cleaned of strong sources to generate the background map or a background map may be constructed from a collection of other data. Each of these variants has limitations, the most obvious perhaps is the difficulty of detecting resolved sources and estimating their

Table 1.1: Introduction to *Detect* Tools

Method	**Strengths**	**Weaknesses**
celldetect	1. Long heritage (*EINSTEIN*, *Rosat*); well understood. 2. Good for faint point sources outside crowded fields.	1. Requires fine tuning of parameters for extended sources. 2. Divides extended sources into multiple point sources. 3. Has difficulty separating closely-spaced point sources.
vtpdetect	1. Finds faint features with low surface brightness. 2. Extended sources found as single source in visually "correct" way, regardless of their actual shape.	1. Combines closely-spaced point sources. 2. Combines diffuse emission with embedded point sources. 3. Slow for more than $\approx\ 10^5$ photons.
wavdetect	1. Separates closely-spaced point sources. 2. Finds extended sources well.	1. Slower than celldetect.

parameters (*e.g.* size, intensity). *celldetect* is good at detecting unresolved sources for a wide variety of data, such as images that are oversampled, those with dominating background levels, and those covering a very large area compared to the size of the resolution element.

The *CXC* version of this method is based on the *EINSTEIN* and *Rosat* codes, but contains enhancements to improve performance for *Chandra* data. For example, at any given position the size of the detect cell may be determined by the size of the point spread function (PSF) at that point and the encircled energy percentage specified by the user. So for *Chandra* data, the cells are small in the center of the field, and become larger as you go off-axis. In addition, one or more exposure maps may be supplied in order to minimize spurious detections along detector edges.

Chapter 6 contains further information about *celldetect* theory.

1.1.2 vtpdetect

A quite different approach is used by *vtpdetect*, which is based on the Voronoi Tessellation and Percolation (VTP) method. Tessellation is the process of dividing the total area of the image into Voronoi cells by constructing a cell about each event position. The area of each cell is a measure of the density of events. Percolation follows by creating bonds between the cells that are above the noise level and which are not farther away from each other than a given distance. This "friends-of-friends" algorithm establishes the group of cells which comprise a source detection.

The advantage of *vtpdetect* is that no assumptions are made about the geometrical properties of the sources. Also, very extended sources are detected as significant without a single pixel in the image needing to feature a photon count significantly different from the background. Therefore, *vtpdetect* is particularly well suited for resolved sources of low surface brightness and potentially irregular shape. The main disadvantage of the algorithm is that it tends to produce blended detections when run with a low flux threshold on a crowded field.

Chapter 9 contains further information about *vtpdetect* theory.

1.1.3 wavdetect

In its most general form, the method used by *wavdetect* correlates the data with a wavelet function that has limited spatial extent and zero overall normalization. Each pixel's correlation value is compared with the expected distribution of values (computed from the estimate of the background); if the value is an extreme outlier within this distribution, the pixel is assumed to be associated with a source. A point source is most easily detected when a wavelet function of similar size to the PSF is used; since the PSF size varies markedly over the field of view, putative source lists are made at a number of scales, which are then combined to yield a final source list with estimated locations, count rates, *etc.* The implementation provided in *wavdetect* uses the so-called "Mexican Hat" function (see Chapter 13), which has a positively-valued quasi-Gaussian core surrounded by a negatively-valued annulus. This is a reasonable function for mirrors and detectors which are characterized by a quasi-Gaussian PSF, and will work effectively even for abnormal PSFs.

There are two parts to the *wavdetect* code. The first, *wtransform*, convolves the data with the wavelet function for however many scales are chosen. The resulting correlation maps are used by the second part, *wrecon*, to construct a final source list and estimate various parameters for each source. Each part may be run separately (see Chapter 12), but the recommended procedure is to use *wavdetect*, which runs them sequentially (see Chapter 11).

Chapter 13 contains further information about *wavdetect* theory.

1.2 Getting Started with CIAO

The *CIAO* software package, of which *Detect* is a part, is available for download from:

```
http://cxc.harvard.edu/ciao/
```

Once *CIAO* has been installed, the user may confirm that the *Detect* tools are available by doing:

```
unix% which celldetect
/soft/ciao/bin/celldetect

unix% which wavdetect
/soft/ciao/bin/wavdetect

unix% which vtpdetect
/soft/ciao/bin/vtpdetect
```

1.3 Running the Detect Tools

The *Detect* tools are executed by typing the tool name at the *UNIX* command line within *CIAO*. For example, to run *celldetect*:

```
unix% celldetect
Input file (): events_img.fits
Output source list (): celldetect_out.fits
```

There are a number of *CIAO* threads which explain how to run the *Detect* tools:

- Running *celldetect*

  `http://cxc.harvard.edu/ciao/threads/celldetect/`

  Chapter 4 also contains examples of running *celldetect*.

- Running *vtpdetect*

  `http://cxc.harvard.edu/ciao/threads/vtpdetect/`

  Chapter 8 also contains examples of running *vtpdetect*.

- Running *wavdetect*

  `http://cxc.harvard.edu/ciao/threads/wavdetect/`

  Chapter 11 also contains examples of running *wavdetect*.

1.3.1 The Detect Parameter Files

The *Detect* tools make use of parameter files, which are stored by default in the user's `$HOME/cxcds_param` subdirectory. For an introduction on working with *CIAO* parameter files, read the parameter interface help file (type "`ahelp parameter`" within *CIAO*). There is also a "Using Parameter Files" thread, which illustrates how to do many common tasks:

`http://cxc.harvard.edu/ciao/threads/param_files/`

1.3.2 Autonaming

The *CIAO*autonaming feature can be used with many of the *Detect*tool parameters. If the parameter value is set to "." (dot), autonaming automatically determines the output file based upon the name of an input file. The output file name is typically of the form `"<infile_root>_src"`, where the "`src`" extension changes depending on the data product being created. "`ahelp autoname`" has more information on autonaming.

Not all parameters support autonaming. Check the help file for each tool to see when this option is available.

1.3.3 Data Model Filtering

The *Detect* tools are able to take advantage of the Data Model library, which enables data to be filtered dynamically when it is input to one of the tools. This allows the user to select regions of the data (spatial, temporal, *etc.*) without having to create an intermediate file on disk. For example, to specify a box in sky(x,y) coordinates:

```
unix% celldetect infile="events_img.fits[x=4096.5:4213.5,y=4142.3:6120.1]"
```

For a detailed description of the DM syntax, see the help file (“`ahelp dm`”) or the “Introduction to the Data Model” thread:

```
http://cxc.harvard.edu/ciao/threads/dm_intro/
```

1.4 Displaying a Region File in ds9

The most straightforward way to examine the detections in a region file is to overlay it on the data used to create it. This is simple to do in ds9, the imager packaged with *CIAO*.

First, open the event file or image in ds9:

```
unix%  ds9 s3\_evt2.fits &
```

Then load the region file from the ds9 menu ”Regions” -¿ ”Load Regions...”.

If the regions are stored in a FITS file, highlight the file and then add the extension to the filename before clicking the “Ok” button, *e.g.* `s3_src.fits[SRCLIST]`. “`[SRCLIST]`” indicates in which block of the file the region information is stored; the extension name is the same for all three detect tools.

No extension exists for for ASCII files; just select the filename and click “Ok”.

The regions will then be displayed on the data in the imager window. For more information on working with region files, see the “Using the Output of Detect Tools” thread:

```
http://cxc.harvard.edu/ciao/threads/detect_output/
```

1.5 Online Help and Other Resources

The *CIAO* software package includes an extensive help system. Further information about this help system is available by typing:

```
unix% ahelp
```

To obtain help regarding a particular *CIAO* tool, type '`ahelp <toolname>`'. For example:

```
unix% ahelp celldetect
```

In addition, the *CXC* has prepared many processing recipes, called "threads". These documents are designed to teach users by leading, step-by-step, through a procedure that utilizes the *CIAO* software package. These are available from,

```
http://cxc.harvard.edu/ciao/threads/index.html
```

and include further information about *CIAO*, parameter manipulation, and *Detect* tools.

Chapter 2

Description of Sample Datasets

The sample datafiles used in the examples in this manual may be downloaded from:

`http://cxc.harvard.edu/ciao/manuals.html`

2.1 Dataset A. Simulated Chandra Data

The *Detect* tools were designed primarily for the analysis of *Chandra* X-ray Observatory data, so the *CXC* has tested them using, in part, simulated ACIS and HRC data produced with the *MARX* [1] simulator.

We use simulations so that we can seed the image with noise background plus a number of sources of known characteristics and then see how well various tools recover sources and their properties. The field constructed for ACIS is shown in Figures 2.1 and 2.2.

The simulation contains a background corresponding to a 30 ks exposure and has 6 repetitions of a linear array of sources. The basic array consists of 6 unresolved sources of the same intensity. From the first source, the following 5 are spaced at 0.5, 4, 5, 10 and 30 arcsec. The pattern is repeated $60''$ from the last of these 6 sources, with the addition of an extended source at the final position. That extended source is a Gaussian with a sigma of $25''$. At right angles to this line of sources, at a distance of $60''$ and $120''$, the pattern is repeated in two additional rows. In the first of the three rows, the unresolved sources have approximately 200 counts each (depending on the detector and off-axis distance) and the extended source has about 2500 counts. For the second row, the unresolved sources are much weaker with $\approx$30 counts each; the extended source intensity does not change. The last row is the same as the middle row, except that the unresolved sources are changed to discs with a diameter of $1''$.

[1] *MARX* is a suite of programs created by the MIT/CXC group and designed to enable the user to simulate the on-orbit performance of the *Chandra* X-ray Observatory. Further information is available at `http://space.mit.edu/CXC/MARX/`.

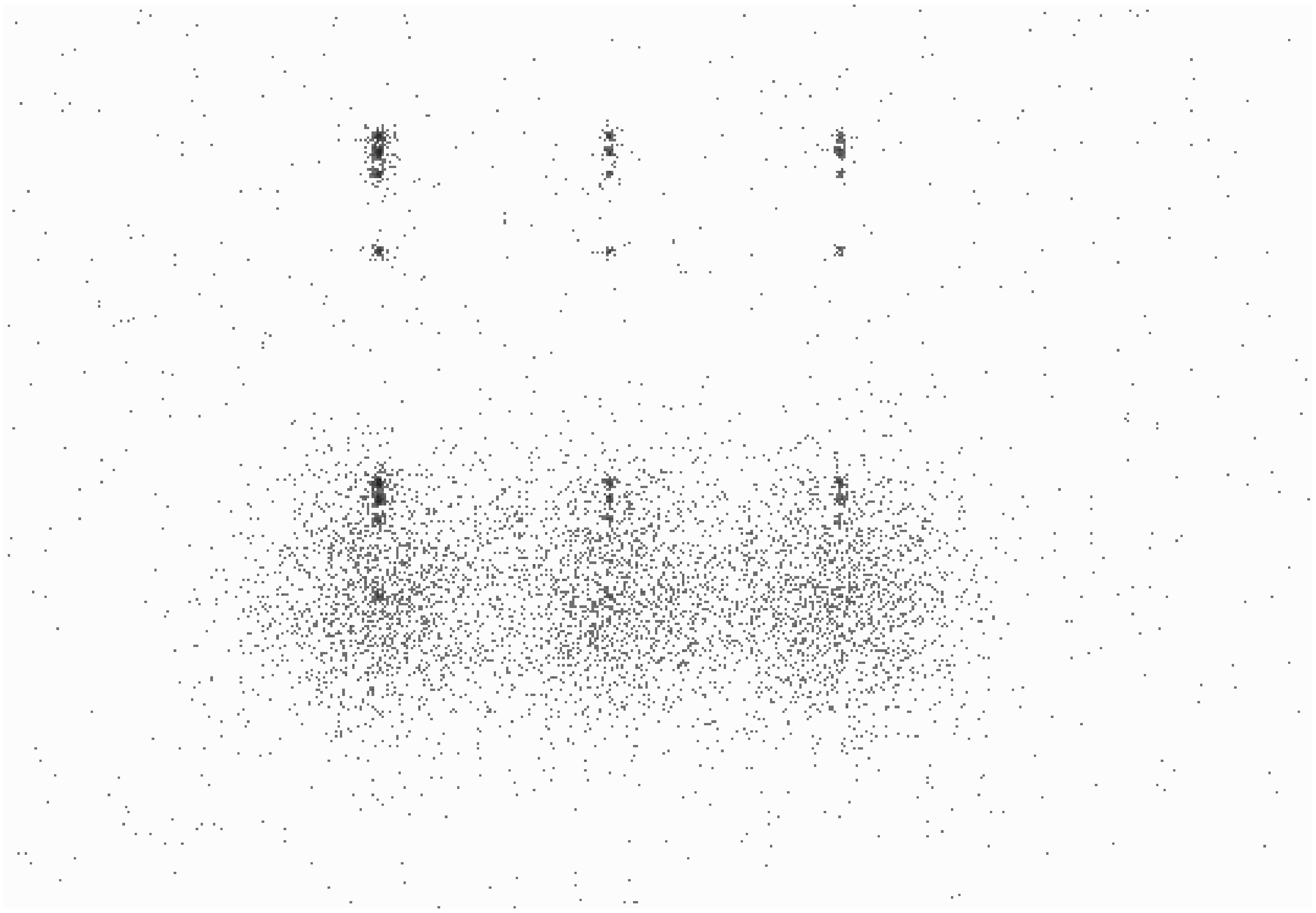

Figure 2.1: A simulated ACIS field, binned by a factor of 4.

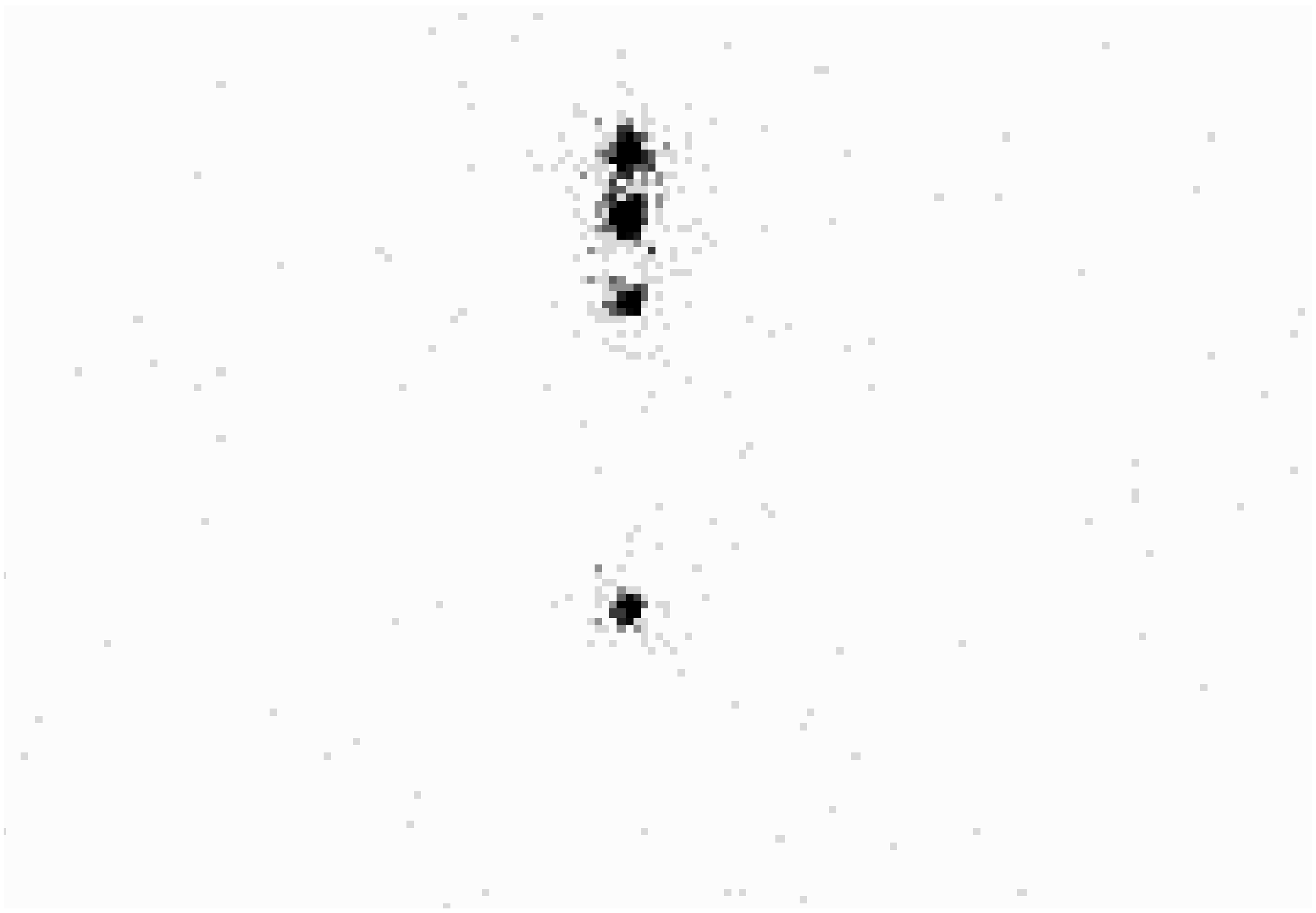

Figure 2.2: At full ACIS resolution, the grid of sources is shown for the simulated field in Fig. 2.1. The distance between the first and last sources is 30″.

2.2 Dataset B. 3C 295 on ACIS-S: a radio galaxy in a cluster of galaxies

3C 295 was observed with ACIS-Searly in the mission as a calibration observation. The target has several features useful for demonstrating and comparing the *Detect* tools. The radio galaxy itself has a heavily absorbed core which might be slightly resolved, but is close to being a point source. Within a few arcsec to the north and south are emission regions associated with the radio hotspots, thereby presenting a test case for the separation of close sources. All of these radio-associated sources are embedded in the extended emission arising from the hot intracluster gas, for 3C 295 is within a cluster of galaxies at $z = 0.45$. Hence the extended emission provides further tests: detecting extended sources and separating unresolved sources within extended emission.

An adaptively-smoothed version of the field center is shown in Figure 2.3. Most of the cluster emission is shown, and the insert gives a magnified view of the emission associated with the radio galaxy at the center.

This dataset was obtained from the *Chandra* Archive (ObsID 578). The examples focus on the chip S3 (`ccd_id = 7`) data, which contains over 100,000 events.

2.3 Dataset C. 3C 273 on HRC-I: a quasar with a bright jet

3C 273 is probably the brightest nearby quasar in the radio band. A prominent jet consisting of several brightness enhancements ("knots") begins about $12''$ from the intense unresolved X-ray core. The knots decrease in intensity with radial distance. The faintest detectable emission is about $20''$ from the core. The closely-spaced features in the jet provide a good test of the *Detect* tools' ability to separate close features. An HRC-I image of this source is shown in Figure 2.4.

This dataset was obtained from the *Chandra* Archive (ObsID 461), and contains over 800,000 events.

2.4 Dataset D. The Chandra Deep Field South on ACIS-I

Having a very long (*i.e.* deep) exposure, such as the *Chandra* Deep Field South (CDFS) data, allows us to run *Detect* on real data with very conservative parameters and see how many real and spurious detections result.

For the CDFS, we have an long exposure of 991.45 ksec containing 1,927,235 counts. The short observation (ObsID 2405) is roughly 100 ksec and has 106,940 events. Each of these two datasets was filtered with a large circular region (radius = 8 arcmin) to avoid the edges which were not covered uniformly by all the merged ObsIDs that comprise these data.

To locate all real sources which might be detected by tools operating on the short segment, we ran *celldetect* with a S/N threshold of 5; *wavdetect* with 5 scales (1, 2, 4, 8, 16) and a signal threshold of E-6; and *vtpdetect* with a limit of E-7 false source events per number of background events. The resulting detections are shown in Figure 2.5.

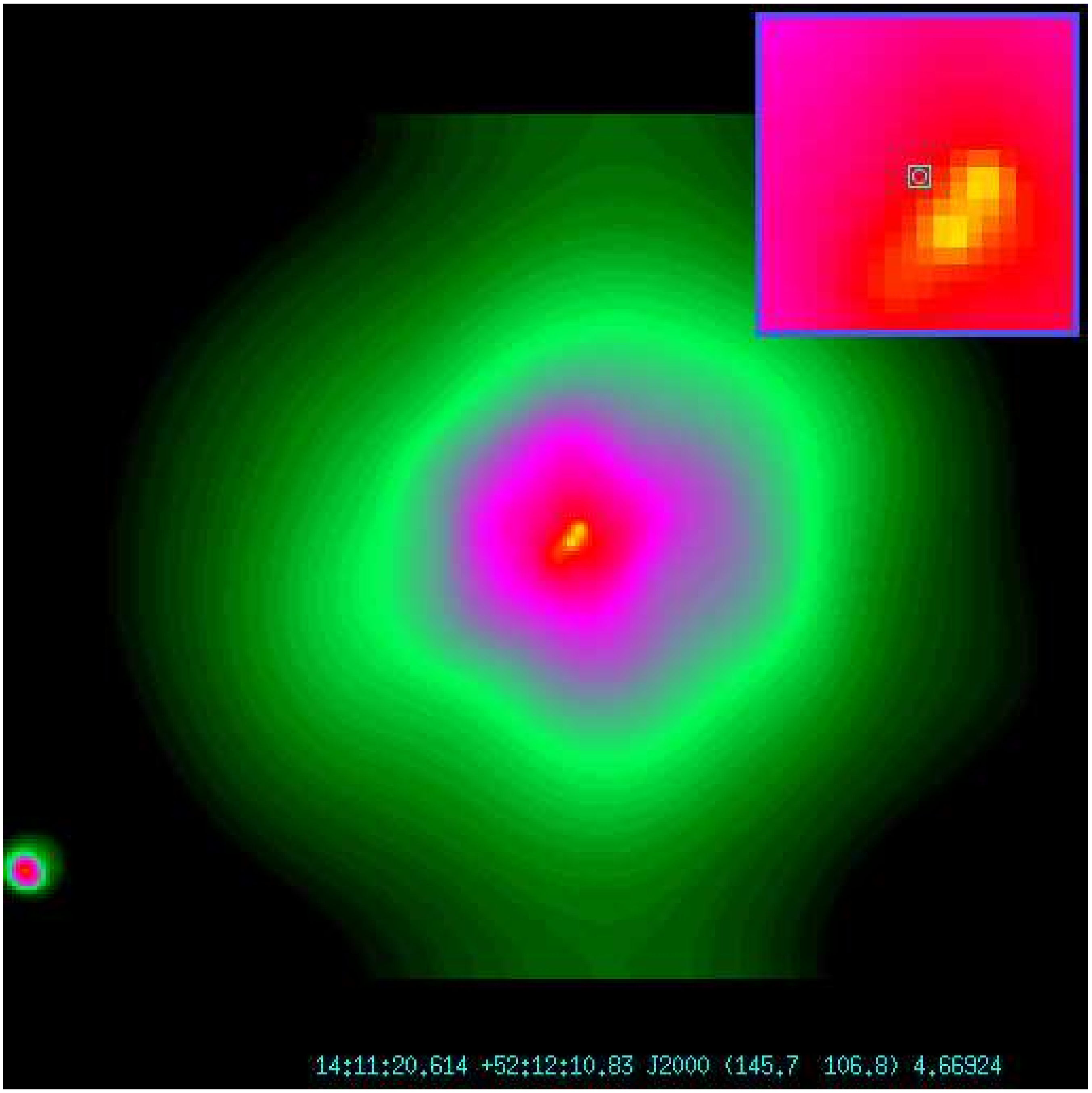

Figure 2.3: The ACIS-S data (adaptively smoothed) of the radio galaxy 3C 295 and the surrounding emission from the hot intracluster gas. The insert is an enlargement of the center, showing the emission associated with the radio galaxy nucleus and its hotspots.

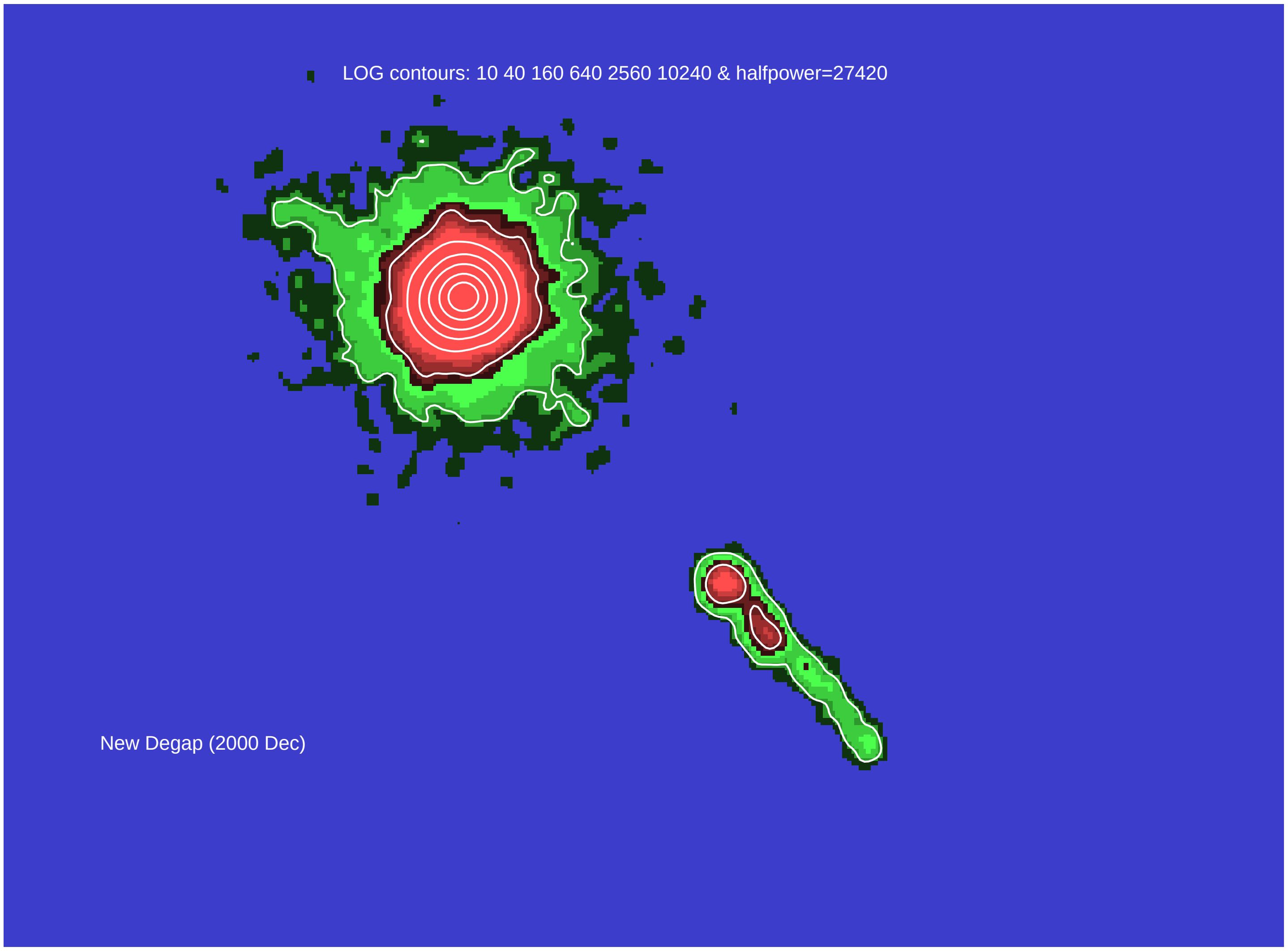

Figure 2.4: HRC-I imaging data of the quasar 3C 273 and its jet.

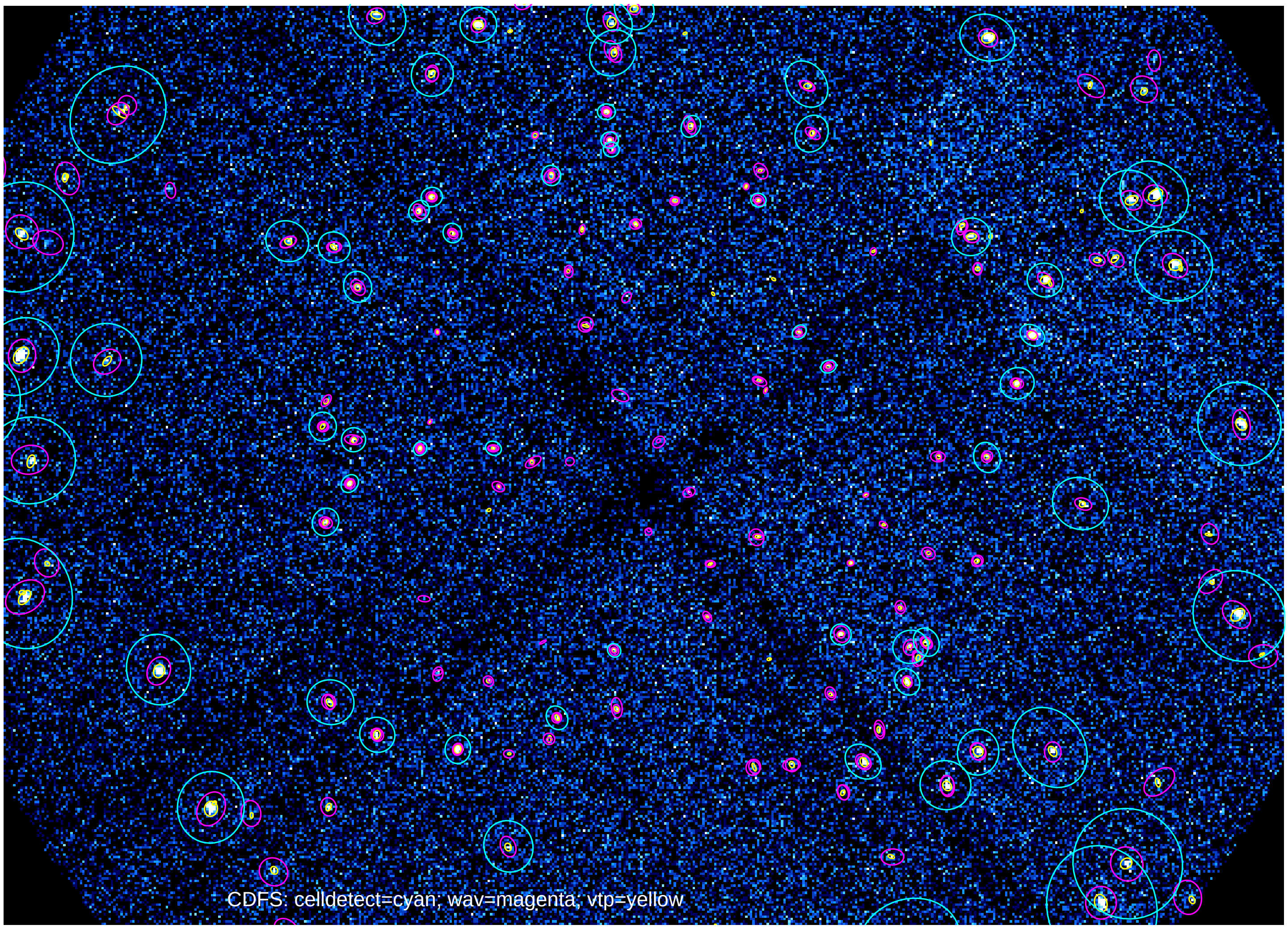

Figure 2.5: The Chandra Deep Field South with ACIS-S. Only data within 8′ of the center were accepted. *celldetect* results are shown by cyan; magenta is for *wavdetect*; and yellow for *vtpdetect*.

Chapter 3

Comparison of Detect Methods

3.1 Factors Affecting Performance and Runtime

3.1.1 celldetect

The primary factor that influences the runtime of *celldetect* is the size of the dataset. Two secondary factors are the number of cell sizes to examine and the actual number of detected sources.

3.1.2 wavdetect

The three most important factors that affect the runtime of *wavdetect* are: (1) the size of the dataset, (2) the number of wavelet scales, and (3) the number of background cleaning iterations. The first two factors influence both *wtransform* and *wrecon*, the third only affects *wtransform.* An additional factor is the size of the wavelet: for large scales the dataset needs to be padded, so the size of the data gets augmented artificially.

In general, *wavdetect* uses a lot of computer memory; 1024×1024 pixels should be considered the practical limit for the size of the input dataset.

3.1.3 vtpdetect

While the runtime of *vtpdetect* does *not* depend on the spatial size of the dataset, it does depend on many other factors and is rather unpredictable. The most important factors are the number of unique photon locations and the overall number of events.

vtpdetect will run quickly if the number of photons is low and there is a high contrast between background and sources. In the opposite scenario (*i.e.* a large number of photons and a large number of faint sources),

Size [pixel]	$N_{\rm evt}$	Approximate run time [min:sec] *celldetect*	*wavdetect*	*vtpdetect*
Short exposure, bkg+sources				
512 × 512	400	0:25	3:45 (3:00+0:45)	0:03
1024 × 1024	1300	0:50	11:40 (9:30+2:10)	0:05
2048 × 2048	3800	2:00	—	0:16
32768 × 32768	110000	8:50	—	9:40
Long exposure, bkg only				
512 × 512	1800	0:25	3:35 (2:55+0:40)	0:15
1024 × 1024	7300	0:50	11:30 (9:30+2:00)	0:50
2048 × 2048	28000	2:00	—	1:40
32768 × 32768	1100000	8:50	—	—

Table 3.1: Sample runtimes for the three tools on a Sun Ultra 1 with a 167 MHz processor and 124 MB of RAM. For *vtpdetect* and *wavdetect*, actual runtimes will normally be longer than indicated in the table because only one scale size was used in *wavdetect* and only one background iteration was used in both *vtpdetect* and *wavdetect*. The "recursive blocking scheme" (see section 4.1) was used in *celldetect* runs for 32k×32k data. Values in parentheses for *wavdetect* indicate the time used by *wtransform* and *wrecon*.

the *vtpdetect* run can be very long, since fitting background becomes an arduous task. In these situations, tests have shown that *vtpdetect* becomes very slow if the observation contains more than $\sim 10^5$ photons.

3.2 Sample Runtimes

To give the reader a rough idea on the performance, we ran all three tools on various subsets of two simulated HRC-I observations. One simulation contained a set of sources on top of a flat background. The second contained only a flat background, but had an exposure 10 times longer than the first and therefore contained roughly 10 times as many background events.

Table 3.1 shows the runtimes on a Sun Ultra 1 with a 167 MHz processor and 124 MB of RAM. The same runs on a Sun SPARCstation 5 with 70 MHz CPU and 64 MB of RAM were ∼3 times slower. The runtime of *celldetect* and *wavdetect* are not affected by the increase in the number of events; the major factor is the size of the dataset. In the case of *vtpdetect*, however, the number of events is the primary factor; *vtpdetect* slows down considerably when the number of events is large.

In the runs reported in Table 3.1, only one wavelet scale was analyzed for *wavdetect*, and only one background iteration was performed for both *wavdetect* and *vtpdetect*. To a first approximation, the time required for execution of *wrecon* and *vtpdetect* increases linearly with the number of iterations. Also, each additional scale in *wavdetect* increases the runtime by roughly the amount of time required for one scale.

3.3 Comparative Results for the Chandra Deep Field South

As described in section 2.4, we chose to use ObsID 2405 as a test field since we have determined all real sources on the full exposure. ObsID 2405 is approximately 10% of the size of the full CDFS. For *celldetect*, we used a S/N threshold of 3, an encircled energy of 0.8, and the PSF library of April 2001. For *wavdetect*, we used scales of `1,2,4,8,16` and `sigthresh = E-6`. For *vtpdetect* we used `scale = 1` and a maximum probability of being a false source (limit) equal to E-5 and E-6. The results are shown in Table 3.2.

Tool:	Total	Edge	Spurious	Valid	Percent Spurious:
celldetect	36	1	1	34	3%
wavdetect	75	19	0	56	0%
vtpdetect(E-5)	61	0	6	55	10%
vtpdetect(E-6)	48	0	2	46	4%

Table 3.2: Detections for ObsID 2405

The distinction between "spurious" and "edge" is somewhat subjective. In the case of *wavdetect*, the edge detections were not actually on the edge, but were close to the edge. The percent spurious assumes that the edge detections don't count (*i.e.* if we had used an exposure map, they probably would not have been detected). The values presented here are indicative, but each tool can be made to perform differently by adjusting the parameter values.

Chapter 4

Running celldetect

4.1 Key celldetect Parameters

This section highlights some frequently used *celldetect* parameters. Similar parameters and their description have been grouped together. Chapter 7 has information on each *celldetect* parameter in alphabetical order.

1. Input file (`infile`)

   The input FITS file can an event list or an image. *CIAO* Data Manipulation (DM) syntax may be used in the input file specification, as explained in Chapter 1, Section 1.3.3.

   The current maximum size allowed for an image is 2048×2048, but an event list may occupy a larger spatial region. For event list datasets larger than 2048×2048 (as given by `TLMIN, TLMAX`), *celldetect* may use the "recursive blocking scheme": First the inner 2048×2048 pixel region of the dataset is searched for sources, then the inner 4096×4096 pixel region (excluding the area already analyzed) is blocked by 2 and searched for sources, then the inner 8192×8192 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 in higher resolution is skipped, *i.e.* only one blocking factor is used for each region of the dataset. [1] For the recursive blocking scheme to operate, `fixedcell` must be set to zero and point spread function (PSF) information must be supplied via `psftable` (since *celldetect* will not examine a dataset past the radius for which PSF information is available). By default, both `fixedcell` and `psftable` are properly set for recursive blocking of *Chandra* data.

2. Output source list (`outfile`)

   The `outfile` parameter is used to provide a name for the output file. The `kernel` parameter controls the output format; the default is for the output file to have the same format as the input file. If autonaming is used (*i.e.* `"outfile=."`), then the output file will be named `"<infile_root>_src"`.

3. Source threshold (`thresh`)
   Find local peaks (`findpeaks`)
   Compute source centroids (`centroid`)

[1] Since a source could be missed if it straddles the border between regions of different blocking factors, each *celldetect* 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 Chapter 7).

The detection threshold, `thresh`, is the signal-to-noise (S/N) cutoff for source detection, and should be set to a value of 3 (the default) or 4, particularly for the initial run.

If the `findpeaks` parameter is set to `no`, then the source list will consist of all detection cells which exceeded the S/N threshold (often resulting in single source being identified multiple times). If `findpeaks` is set to `yes` (the default), then adjacent detections are recognized as a single source and the cell with the largest S/N appears in the source list.

If `centroid` is set to `yes` (the default), then the source list will be based on the source centroid. In addition, the major and minor axes of the events distribution are calculated as well as the position angle of the major axis. However, if `centroid` is set to `no`, then the center pixel of a source's detect cell is reported as the location of the source.

4. ASCII region file (`regfile`)
   Size of output source ellipses (`ellsigma`)

   In addition to the output source list (`outfile`), the user may choose to have an ASCII source region file written by specifying a filename in `regfile` parameter. If autonaming is used (*i.e.* `"regfile=."`), then the source regions output file will be named `"<infile_root>_reg"`.

   This region file is useful for subsequent input into ds9, particularly to overlay region markers over each detection; see Chapter 1, Section 1.4 for details.

   The size of the elliptical source regions in both of the source region files is controlled via the `ellsigma` parameter. This parameter is a multiplicative factor applied to sigma, the standard deviation of the distribution, to scale the major and minor axes of the ellipses for each source detection. This feature is included so that the graphics overlay may be made more visible; it is does not affect the detections themselves, only the display of the regions.

   By default the value for `ellsigma` is 3, but a larger value is often helpful.

5. Background file name (`bkgfile`)
   Background count/pixel (`bkgvalue`)
   and Background error (`bkgerrvalue`)

   In *celldetect*, there are three options for computing the background count for a detect cell:

   (a) By default, it will be estimated from a background frame which surrounds the detect cell. This is essentially the *Rosat* `LDETECT` (aka "local detect") procedure.
   (b) The user may specify the background as an input image file, using the `bkgfile` parameter. If a reliable background map is available, then the user will be able to better detect extended sources which are devoid of significant brightness gradients. The error in the supplied background map may be given using the `bkgerrvalue` parameter; if unknown, this parameter should be left at the default value (0).
   (c) The user may instead specify the background as a fixed value of counts/pixel, using the `bkgvalue` parameter. Be aware that if this value is too low, spurious sources will be detected on the periphery of the dataset. The error in the specified background value may be given using the `bkgerrvalue` parameter; if unknown, this parameter should be left at the default value (0).

   For local detect (option 5), the size of the background frame is calculated by multiplying the detect cell size by $\sqrt{2}$, and then rounding it up or down to the closest number of the same parity as the detect cell size. This acheives two goals: the area of the background frame is roughly the same as the area of the detect cell, and the background frame is centered on the same location as the detect frame.

   For options 5b and 5c, the background frame is the same as the detect cell. If a background map is provided, then the background value will be obtained from that map, using the same pixels as the detect cell.

6. Table of PSF size data (`psftable`)
   Energy band (`eband`)
   Encircled energy of PSF (`eenergy`)
   Offset of x axis from data center (`xoffset`)
   Offset of y axis from data center (`yoffset`)
   Output cell size image stack name (`cellfile`)

   celldetect compensates for the larger *Chandra* PSF off-axis by using a "variable cell size" technique, which increases the size of the detect cell with off-axis distance. The PSF information used by *celldetect* is contained in a *Chandra* calibration file distributed with *CIAO*; the location is given by the value of the `psftable` parameter, which points by default to the appropriate *Chandra* calibration PSF file.

   The *Chandra* HRC and ACIS detectors are the sole combinations in the PSF library file used by *celldetect*; information for other missions is not provided. *celldetect* can only utilize the variable cell size procedure if the following keywords are present in the input *Chandra* data file (`infile`): `TELESCOP`, `INSTRUMENT`, `DETNAM`, `GRATING`, `RA_NOM`, and `DEC_NOM`. If any of the keywords are missing, the tool cannot calculate the appropriate cell size and `fixedcell` needs to be specified instead.

   The selection of a particular PSF size is governed by the energy band `eband` parameter [keV], since the *Chandra* PSF is a function of energy. By default the energy band is set to 1.4967 keV.

   celldetect also uses an encircled energy percentage value, given by the `eenergy` parameter, in concert with the calibration PSF file. This value is the percentage of PSF energy to be encircled by the detect cell, expressed as a fraction of 1.0 (default is 0.8). The `eenergy` is a key parameter for detecting off-axis sources, and users are urged to experiment with other values.

   By default, *celldetect* calculates the off-axis angle using the nominal pointing of the data file as the origin. Users may change this behavior by setting offsets to any numerical values (using the `xoffset` and `yoffset` parameters); these offsets provide the location of the optical axis with respect to the center of the data file. A typical scenario in which the user may select to use offsets is when the data file is a sum of two or more observations with different pointings. The nominal pointing in such a data file is usually poorly defined and the off-axis angle could be calculated from an undesired origin, leading to suboptimal selection of the detect cell size.

   If a filename is provided for the `cellfile` parameter, then one or more images are created which contain the detect cell size at each pixel location (see Example 4.2.2). If autonaming is used (*i.e.* `"cellfile=."`), then the output stack file will be named `"<infile_root>_cell"`.

7. Fixed cell size to use (`fixedcell`)

   celldetect has a fixed cell size option, where the detect cell size is kept constant for the entire dataset. Choose the cell size in pixels by setting the `fixedcell` parameter; allowed values are 1 or an integer divisible by 3. Setting `fixedcell` to a non-zero value effectively disables the parameters related to *celldetect*'s variable cell size functionality: `psftable`, `eenergy`, `eband`, `xoffset`, and `yoffset` parameters.

   If *celldetect* is to be run on non-*Chandra* data, supply a value for `fixedcell`.

   Fixed cell size is also useful for extended source detection. It is well known that local detect methods are not very useful for significantly extended sources. To overcome this limitation, the user may either set `fixedcell` to a size sufficient to match the expected size of an extended source, or employ a background map (or fixed value for the background) instead of using local detect.

8. List of exposure map files (`expstk`)
   Cutoff ratio for source cell exposure variation (`expratio`)

   To minimize spurious detections along detector edges, one or more exposure maps may be supplied. The `expstk` parameter takes a stack of exposure map filenames; sse "`ahelp stack`" for more information. Each of the exposure map files should have appropriate binning to match *celldetect*'s recursive blocking (*i.e.* binning by 2, 4, 8, ...). Exposure maps only work with the local detect procedure. In addition, they do not work with the convolution option.

The `expratio` parameter is the ratio of the average exposure of the background frame to the average exposure in the detect cell. Exposure maps supplied by the user are used in conjunction with the `expratio` parameter. For `expratio` values other than 0, exposure maps must be supplied in the `expstk` parameter. Detections for sources whose ratio falls below the specified `expratio` value will be suppressed from the source region file (`regfile`). However, all detections will be present in the output source list (`outfile`), and the column `EXPO_RATIO` will be created. Recommended `expratio` values ares between 0.9 and 0.99.

9. Use convolution (`convolve`)
   S/N output file name (`snrfile`)

   Setting `convolve` to `yes` will cause detection to be done using the convolution library instead of sliding cells (the default). This method is much slower, does not work with recursive blocking, and should really only be used when an output S/N map is needed.

   The convolve option produces a pixel-by-pixel map of the S/N, which is written to the file specified by `snrfile`. A S/N value is computed for every pixel in the input dataset, not just for source locations. The file `snrfile` is then an image of these values. If autonaming is used (*i.e.* `"snrfile=."`), then the S/N output file will be named `"<infile_root>_snr"`.

10. Overwrite exiting outputs (`clobber`)
    Log verbosity level (`verbose`)
    Make a celldetect.log file (`log`)

    celldetect does not not overwrite existing outputs unless `clobber` is set to `yes`.

    The verbosity of log information ranges from 0 (none) to 5 (maximum output) and is sent to `STDERR` (usually the screen). However, if `log` is set to `yes`, then the log information will be written to the file `celldetect.log`.

4.2 Examples Using Dataset A (Simulated ACIS Data)

4.2.1 Example 1: Running celldetect with the Default Parameters

This first example illustrates running *celldetect* using a simulated ACIS dataset as input; see Chapter 2, Section 2.1 for further information about this dataset.

Description of the Default Parameter Functionality

As required, the user specifies the following:

1. Input file (`infile = "data/datasetA_acis_img.fits"`) The input file is an image of simulated *Chandra* data named `data/datasetA_acis_img.fits`.

2. Output source list (`outfile = "example1_out.fits"`) The output file is to be named `example1_out.fits`.

None of the parameters are changed from their default values, so:

1. Background file name (`bkgfile = none`)
   Background count/pixel (`bkgvalue = 0`)
   and Background error (`bkgerrvalue =0`)

   The background will be estimated from a background frame that surrounds the detect cell, *i.e.* using the "local detect" procedure.

2. Table of PSF size data (`psftable`)
   Encircled energy of PSF (`eenergy = 0.8`)
   Energy band (`eband = 1.4967`)
   Offset of x axis from data center (`xoffset = INDEF`)
   Offset of y axis from data center (`yoffset = INDEF`)
   Fixed cell size to use (`fixedcell = 0`)
   Output cell size image stack name (`cellfile = none`)

   celldetect will use the "variable cell size" technique, increasing the size of the detect cell with off-axis distance, according to the *Chandra* calibration PSF information distributed with *CIAO*. The default encircled energy percentage value and energy band are used. The off-axis angle will be calculated using the nominal pointing of the data file as the origin. An output image file of the cell sizes is not produced.

3. Source threshold (`thresh = 3`)
   Find local peaks (`findpeaks = yes`)
   Compute source centroids (`centroid = yes`)

   The S/N cutoff for source detection will be 3. Adjacent detections will be recognized as a single source, and the source list will be based on the source centroid.

4. ASCII region file (`regfile = none`)
   Size of output source ellipses (`ellsigma = 3`)

   No ASCII region file will be produced. The FITS source list will contain a 3-sigma ellipse for each detection.

Setting Required Parameters

The `pset` tool is used to supply the required parameter values (`infile` and `outfile`):

```
unix% punlearn celldetect
unix% pset celldetect infile = data/datasetA_acis_img.fits
unix% pset celldetect outfile = example1_out.fits
```

Notice that a path may be include with the filenames. Chapter 1, Section 1.3.1 has further information on manipulating parameters.

Listing Current Parameter Settings

If desired, the user may then list the current *celldetect* parameter settings with `plist`:

```
unix% plist celldetect
Parameters for /home/username/cxcds_param/celldetect.par

#
#   celldetect parameter file
#
#
#   input
#
        infile = data/datasetA_acis_img.fits Input file
#
#   output
#
       outfile = example1_out.fits Output source list
       (expstk = )                list of exposure map files
      (regfile = none)            ASCII regions file
#
#   output options
#
       (kernel = default)         Output file format
      (clobber = no)              Overwrite exiting outputs?
#
#   output content/format options
#
       (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
#
#   detect cell size parameters
#
    (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 
     (cellfile = )                Output cell size image stack name
#
#   background parameters
#
      (bkgfile = )                Background file name
     (bkgvalue = 0)               Background count/pixel
  (bkgerrvalue = 0)               Background error
#
#   using defaults is recommended here
#
     (convolve = no)              Use convolution?
      (snrfile = )                SNR output file name (for convolution only)
#
```

```
#   run log verbosity and content
#
     (verbose = 0)              Log verbosity level
         (log = no)             Make a celldetect.log file?
#
#   mode
#
        (mode = ql)
```

Executing *celldetect*

To execute *celldetect*, type the name of the tool on the command line:

```
unix% celldetect
Input file (data/datasetA_acis_img.fits):
Output source list (example1_out.fits):
unix%
```

The parameter settings are shown in the prompt for each required parameter; these settings may be accepted by hitting the <RETURN> key or changed if desired.

Figure 4.1 shows the data with 3-sigma ellipses from the region file overlaid. Chapter 1, Section 1.4 has instructions on how to display the region file in ds9.

Examining the Results with dmlist

The *CIAO* tool `dmlist` is used to for examine the contents of the output file. To see what blocks are in the file:

```
unix% dmlist example1_out.fits opt=blocks
--------------------------------------------------------------------------------
Dataset: example1_out.fits
--------------------------------------------------------------------------------

     Block Name                          Type         Dimensions
--------------------------------------------------------------------------------
Block    1: PRIMARY                        Null
Block    2: SRCLIST                        Table        31 cols x 18       rows
```

In the `SRCLIST` block, there are 31 columns of information for each of the 18 detected sources (one row for each detected source). To list the names of these columns, along with brief column information:

```
unix% dmlist "example1_out.fits[SRCLIST]" opt=cols

--------------------------------------------------------------------------------
```

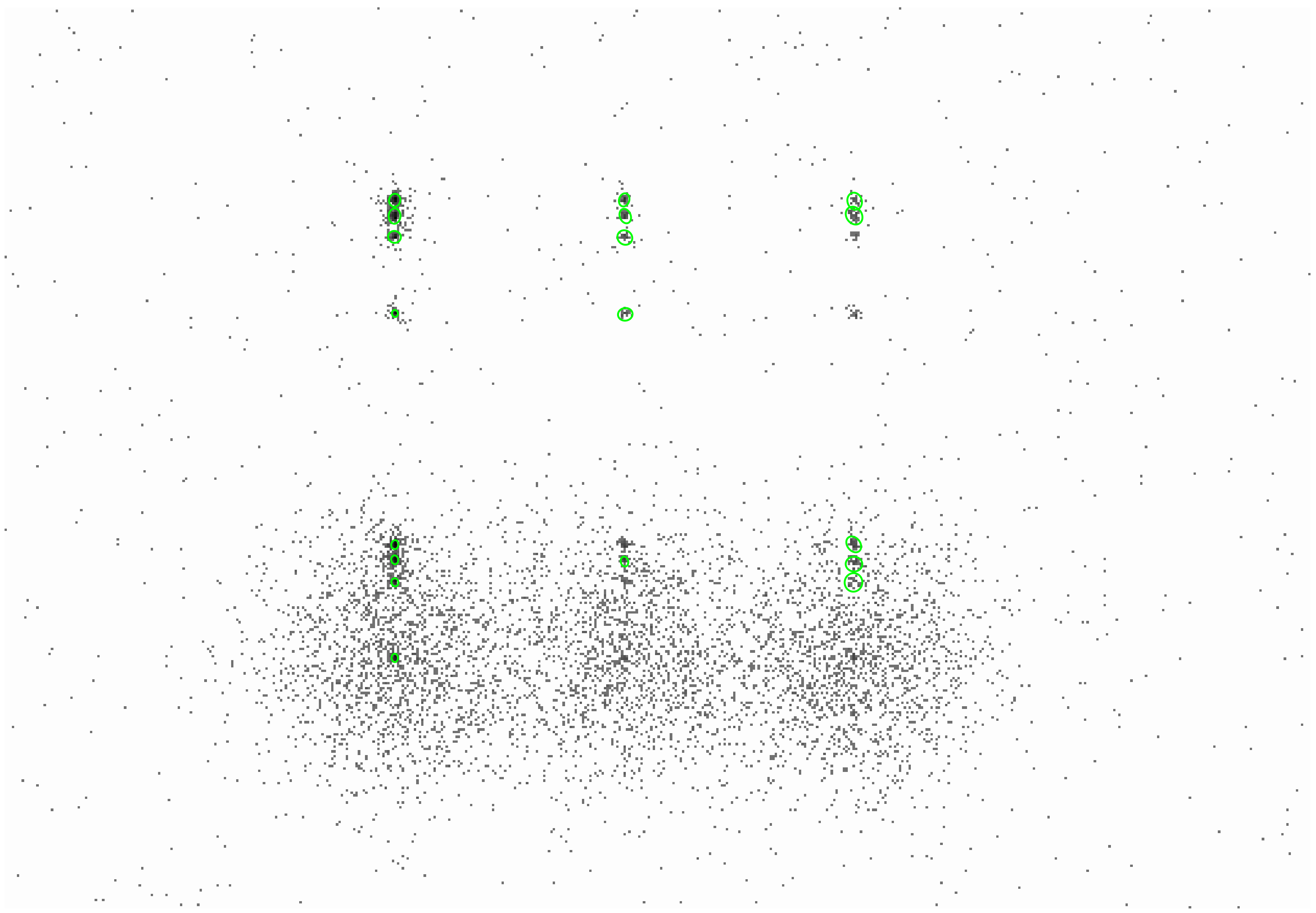

Figure 4.1: *celldetect* with default parameters. The ACIS-S simulated field with 3-sigma ellipses showing the *celldetect* results.

```
Columns for Table Block SRCLIST
--------------------------------------------------------------------------------

ColNo  Name                  Unit         Type             Range
   1   RA                    deg           Real8          0:       360.0
Source Right Ascension
   2   DEC                   deg           Real8          -90.0:       90.0
Source Declination
   3   RA_ERR                deg           Real8          -Inf:+Inf
Source Right Ascension Error
   4   DEC_ERR               deg           Real8          -Inf:+Inf
Source Declination Error
   5   POS(X,Y)              pixel         Real8          -Inf:+Inf
Physical coordinates
   6   X_ERR                 pixel         Real8          -Inf:+Inf
Source X position error
   7   Y_ERR                 pixel         Real8          -Inf:+Inf
Source Y position error
   8   CELLPOS(CELL_X,CELL_Y) pixel        Real8          -Inf:+Inf
Physical coordinates
   9   DETSIZE               pixel         Int4           -
source cell size
  10   BKGSIZE               pixel         Int4           -
background cell size
  11   NPIXSOU               pixel         Int4           -
pixels in source cell
  12   NPIXBKG               pixel         Int4           -
pixels in background cell
  13   NET_COUNTS            count         Real4          -Inf:+Inf
Net source counts
  14   NET_COUNTS_ERR        count         Real4          -Inf:+Inf
Error in net source counts
  15   BKG_COUNTS            count         Real4          -Inf:+Inf
Background counts (scaled to source cell)
  16   BKG_COUNTS_ERR        count         Real4          -Inf:+Inf
Error in BKG_COUNTS
  17   NET_RATE              count/s       Real4          -Inf:+Inf
Source count rate
  18   NET_RATE_ERR          count/s       Real4          -Inf:+Inf
Source count rate error
  19   BKG_RATE              count/s       Real4          -Inf:+Inf
Background count rate
  20   BKG_RATE_ERR          count/s/pixel Real4          -Inf:+Inf
Background count rate error
  21   EXPTIME               s             Real4          -Inf:+Inf
Effective exposure time
  22   SNR                                 Real4          -Inf:+Inf
Signal-to-Noise ratio
  23   SHAPE                               String[10]
Shape of source region
  24   R[2]                                Real4(2)       -Inf:+Inf
```

```
Radii of source region source
  25   ROTANG               deg          Real4          0:      180.0
Rotation angle of source region
  26   PSFRATIO                          Real4          -Inf:+Inf
ratio of source ellipse size to PSF size
  27   BLOCK                             Int4           -
blocking factor
  28   COMPONENT                         Int4           -
Source Number
  29   EXPO_RATIO                        Real4          -Inf:+Inf
ratio of average background and detect cell exp
  30   DOUBLE                            Logical
double detection flag
  31   DOUBLE_ID                         Int4           -
double source component

--------------------------------------------------------------------------------
4:
--------------------------------------------------------------------------------

ColNo    Name
5:    EQSRC(RA_SRC ) = (+0)[deg] +TAN[(-0.0001361)* (POS(X)-(+256.50))]
          (DEC_SRC)   (+0)           (+0.0001361)  (    (Y) (+256.50))
8:    EQSRC_CELL(RA_CELL_X ) = (+0)[deg] +TAN[(-0.0001361)* (CELLPOS(CELL_X)-(+256.50))]
               (DEC_CELL_Y)   (+0)           (+0.0001361)  (       (CELL_Y) (+256.50))
```

To further examine the contents of the output file, use `dmlist` to show the position and net counts of each source detection:

```
unix% dmlist "example1_out.fits[SRCLIST][cols X,Y, NET_COUNTS]" opt=data

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

ROW    POS(X,Y)                                 NET_COUNTS

     1 (      251.9541284404,      268.8899082569)        80.31250
     2 (      252.0310077519,      309.0387596899)       113.250
     3 (      251.9555555556,      320.9833333333)       123.750
     4 (      252.0196078431,      329.0196078431)       217.8750
     5 (      252.0491803279,      451.7786885246)       106.81250
     6 (      373.8648648649,      320.0540540541)         28.0
     7 (      251.8198757764,      492.4347826087)       148.1428527832
     8 (      251.8598901099,      503.4862637363)       331.8571472168
     9 (      251.9226190476,      511.8779761905)       285.8571472168
    10 (      374.0357142857,      492.1071428571)        25.4285717010
    11 (      373.6935483871,      512.0806451613)        55.5714302063
    12 (      374.2647058824,      451.3235294118)         34.0
    13 (      374.2641509434,      503.4905660377)         44.0
```

```
   14        (      495.3666666667,      309.0)        24.8571434021
   15 (      495.6304347826,      318.7173913043)        33.1428565979
   16 (      495.4923076923,      329.1846153846)        53.4285697937
   17 (      495.6935483871,      503.7419354839)        54.2857131958
   18          (      495.9750,      511.3750)        32.2857131958
```

The graphical user interface of the *CIAO* tool *Prism*is an alternative to the command-line interface of `dmlist`:

```
unix% prism example1_out.fits &
```

4.2.2 Example 2: Running celldetect to Create Additional Output

This next example illustrates running *celldetect* using as input the same *Chandra* dataset from the first example (see Section 4.2.1), but with a few parameters changed from their defaults. This will cause additional output files to be produced:

- `cellfile`: an output cell size image stack file.
- `regfile`: an ASCII region file.
- `log` and `verbose`: increased verbosity is written to the `celldetect.log` output file.

In addition, the `ellsigma` parameter is increased to make the display of detected regions easier to see when displayed.

Setting Parameters

The input file is the same image used in the first example, and the output file is to be named `example2_out.fits`. We begin by setting all the parameters mentioned:

```
unix% punlearn celldetect
unix% pset celldetect infile = "data/datasetA_acis_img.fits"
unix% pset celldetect outfile = example2_out.fits
unix% pset celldetect regfile = example2_out.reg
unix% pset celldetect ellsigma = 5
unix% pset celldetect cellfile = example2_cellfile.stack
unix% pset celldetect verbose = 4
unix% pset celldetect log = yes
```

A region file named `example2_out.reg` will be produced, and the size of the output source ellipses is increased to 5-sigma. The output cell size image stack file, which contains information about the detect cell sizes used, will be named `example2_cellfile.stack`. Finally, the verbosity of log information is changed to 4 and will be recorded in `celldetect.log`.

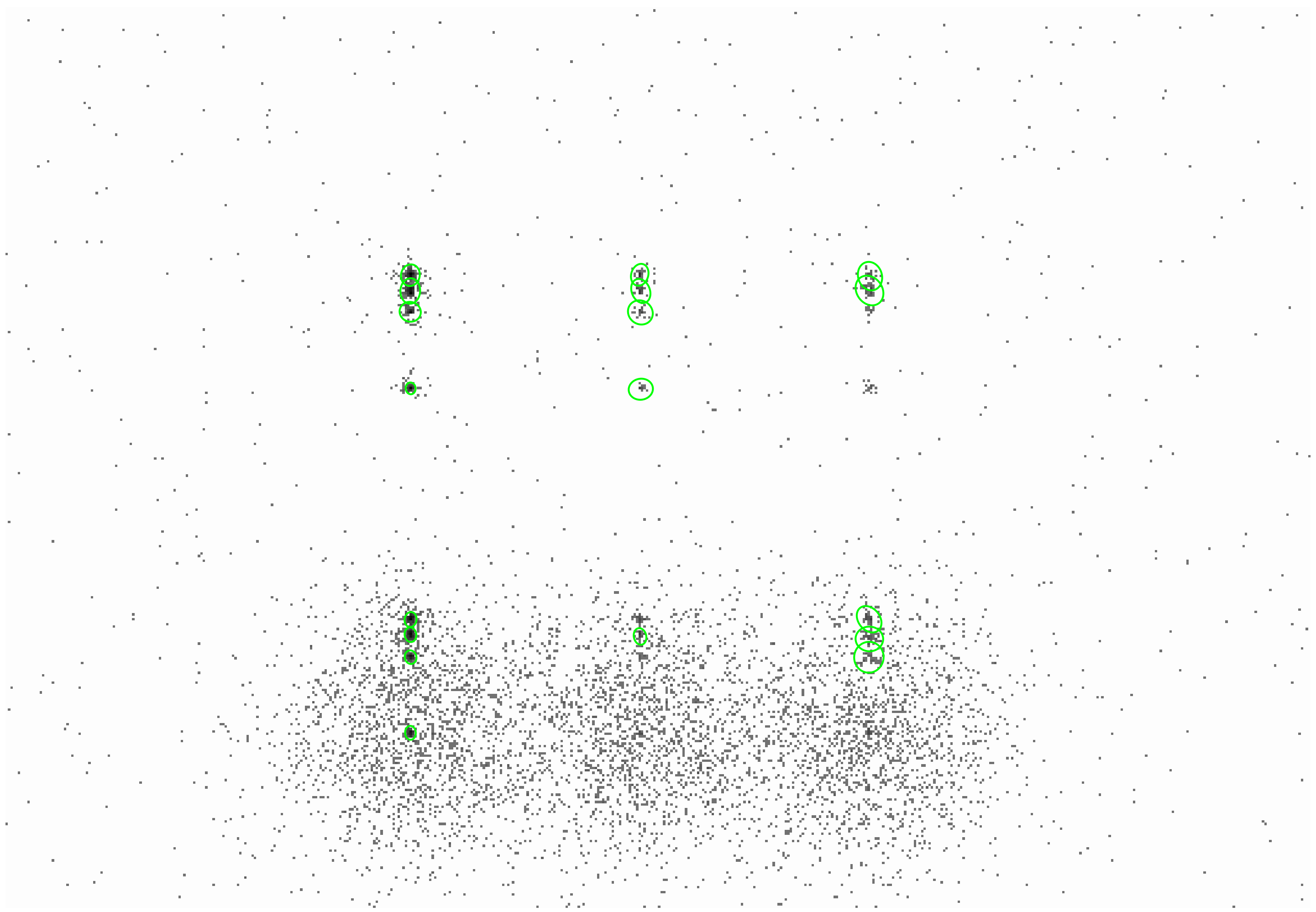

Figure 4.2: *celldetect* results with `ellsigma=5`. The ACIS-S simulated field with ellipses showing the *celldetect* results. The size of the ellipses has been increased from the default, which improves their visibility.

Executing *celldetect*

```
unix% celldetect
Input file (data/datasetA_acis_img.fits):
Output source list (example2_out.fits):
unix%
```

The results are shown in Figure 4.2. Each detection has a 5-sigma ellipse overlay, since `ellsigma` was set to 5.

Examining Results

- As in the first example, this `dmlist` command shows what blocks are in the output source list file:

  ```
  unix% dmlist example2_out.fits opt=blocks

  --------------------------------------------------------------------------------
  Dataset: example2_out.fits
  --------------------------------------------------------------------------------

       Block Name                          Type         Dimensions
  --------------------------------------------------------------------------------
  Block    1: PRIMARY                        Null
  Block    2: SRCLIST                        Table        31 cols x 18       rows
  ```

  The `SRCLIST` block looks the same as Example 4.2.1: 31 columns of information for each of the 18 detected sources.

- Since the `cellfile` parameter was set, an image was created containing the detect cell size at each pixel location. The name of this image is stored in the ASCII file `example2_cellfile.stack`:

  ```
  unix% more example2_cellfile.stack
  datasetA_acis_img_cell_1.fits
  ```

  There is only one filename in this case, but multiple files may be created, depending on how *celldetect*is run. This image may be viewed using ds9:

  ```
  unix% ds9 datasetA_acis_img_cell_1.fits &
  ```

- Since the `verbose` parameter was increased and the `log` parameter was set to `yes`, an output `celldetect.log` file was created:

  ```
  unix% more celldetect.log
  input file: data/datasetA_acis_img.fits[123:630,109:614]
  output file: example2_out.fits
  ASCII source regions file: example2_out.reg
  creating cell size table
  reading data image
  creating candidate source list
  cell size 3
  cell size 6
  cell size 9
  cell size 12
  Writing out cell file: datasetA_acis_img_cell_1.fits
  ```

4.2.3 Example 3: Running celldetect with a Lower Detection Threshold

This next example illustrates running *celldetect* using as input the same *Chandra* dataset from the first two examples (see Section 4.2.1 and Section 4.2.2). The following parameters will be used:

- `ellsigma`: increased to make the detected regions easier to see.
- `regfile`: an ASCII region file.
- `thresh`: controls the S/N cutoff for source detection; here it will be set to 2.5 (the default value is 3).

Setting Parameters

We begin by setting all the parameters just mentioned:

```
unix% punlearn celldetect
unix% pset celldetect infile = "data/datasetA_acis_img.fits"
unix% pset celldetect outfile = example3_out.fits
unix% pset celldetect regfile = example3_out.reg
unix% pset celldetect ellsigma = 5
unix% pset celldetect thresh = 2.5
```

All other parameters are left at their default values.

Executing *celldetect*

```
unix% celldetect
Input file (data/datasetA_acis_img.fits):
Output source list (example3_out.fits):
unix%
```

Figure 4.3 shows the data overlaid with 5-sigma ellipses.

4.3 Examples Using Dataset B (3C 295 on ACIS-S)

4.3.1 Example 1: Detecting Sources within Extended Emission

This first example illustrates running *celldetect* using a *Chandra* ACIS dataset of 3C 295 as input; see Chapter 2, Section 2.2 for further information about this dataset.

All parameters are kept at defaults, except for:

- `ellsigma`: increased to make the display of detected regions easier to see.
- `regfile`: an ASCII region file.

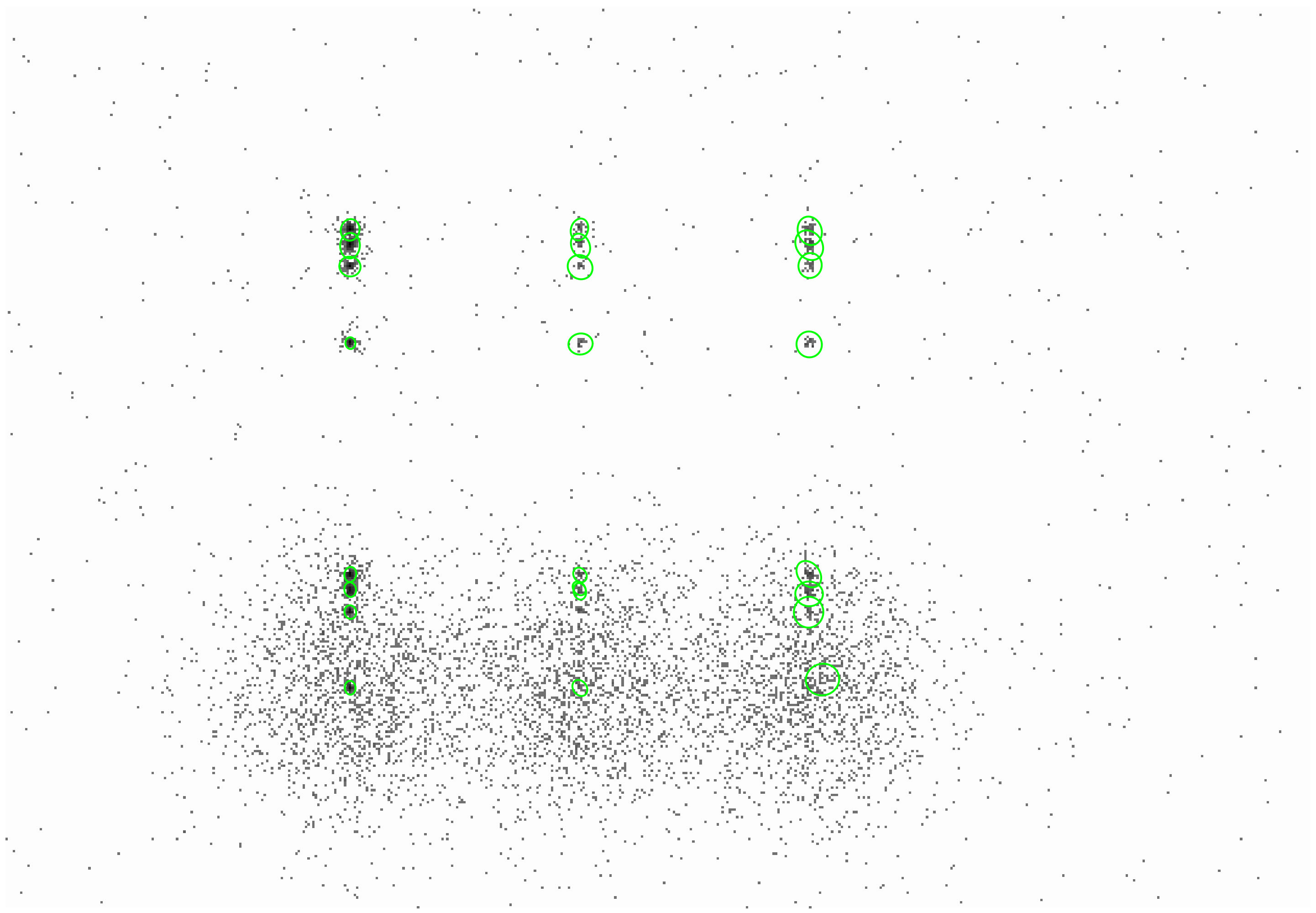

Figure 4.3: *celldetect* with lower detection threshold. The ACIS-S simulated field with ellipses showing the *celldetect* results for `thresh`=2.5. Compare these detections to those obtained using the default value of `thresh`=3 (Figure 4.2).

Setting Parameters

Set the parameters:

```
unix% punlearn celldetect
unix% pset celldetect infile = \
"data/acisf00578N002_evt2.fits[EVENTS][ccd_id=7]"
unix% pset celldetect outfile = example1_out.fits
unix% pset celldetect regfile = example1_out.reg
unix% pset celldetect ellsigma = 5
```

DM syntax is used to select the chip of interest for input (*i.e.* `[ccd_id=7]`); see Chapter 1, Section 1.3.3 for further information about DM syntax usage with *celldetect*.

Executing *celldetect*

```
unix% celldetect
Input file (data/acisf00578N002_evt2.fits[EVENTS][ccd_id=7]):
Output source list (example1_out.fits):
unix%
```

The results are shown in Figure 4.4.

4.3.2 Example 2: Eliminating Spurious Edge Detections with an Exposure Map

This next example illustrates running *celldetect* on chip S1 (`ccd_id=5`) of dataset B. In this example, *celldetect* will be run twice for comparison: with and without an exposure map.

Generating Input Data

Since we want to utilize an image exposure map, the data must be binned into an image. The `dmcopy` tool is used to create an image of chip S1 blocked by a factor of 4:

```
unix% punlearn dmcopy
unix% dmcopy "data/acisf00578N002_evt2.fits[ccd_id=5][bin x=::4,y=::4]" \
"acisf00578N002_evt2_ccdid5_imgbin4.fits"
```

An exposure map to match this image was generated following the CIAO thread, “Compute Single Chip ACIS Exposure Map”:

```
http://cxc.harvard.edu/ciao/threads/expmap_acis_single/
```

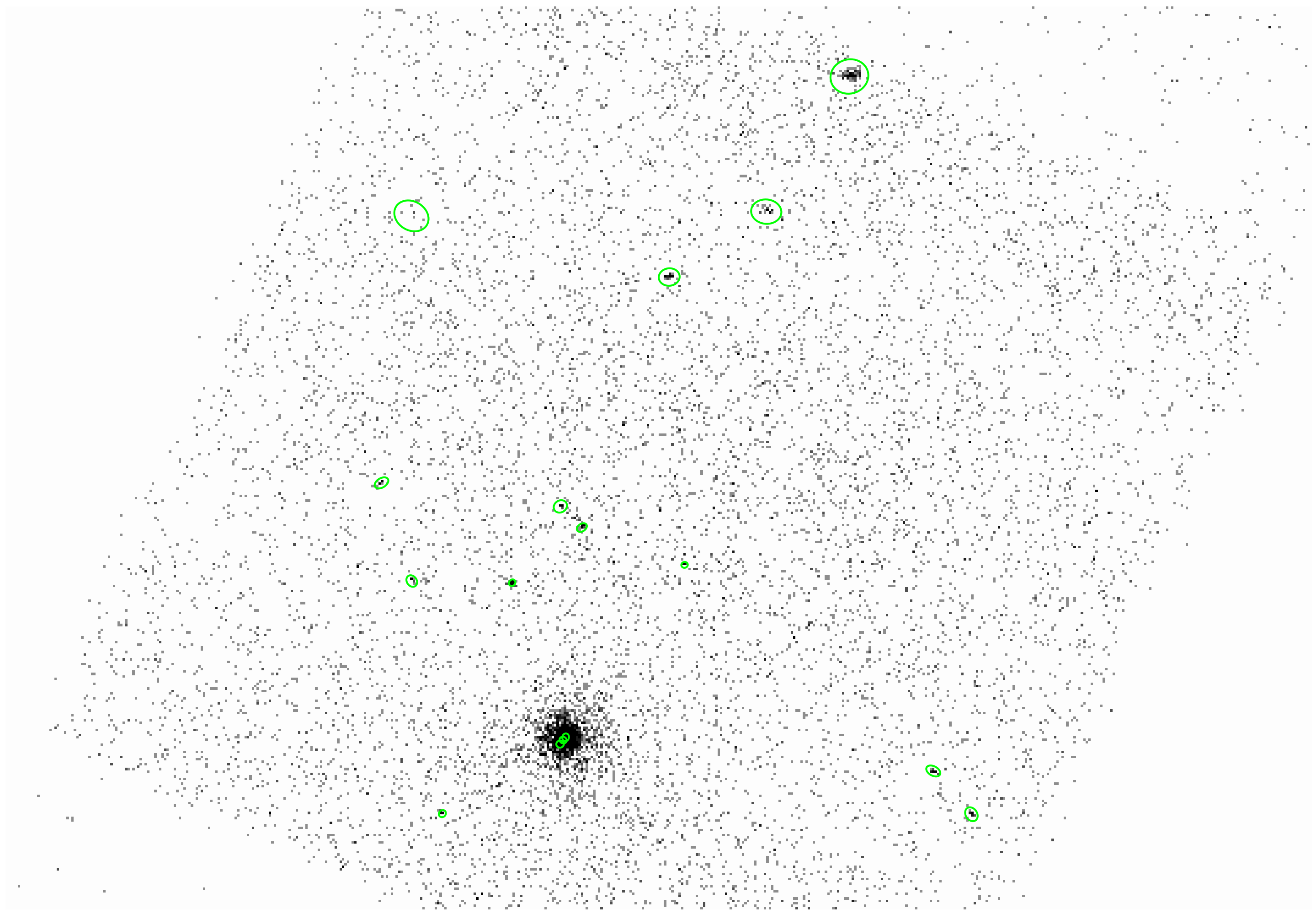

Figure 4.4: *celldetect* on extended emission. Chip S3 (`ccd_id=7`) of the ACIS-S dataset of 3C 295, displayed with binning = 2; the ellipses show the *celldetect* results. Comparing these results with those from *vtpdetect* (Figure 8.3) shows that *vtpdetect* successfully detects regions of extended emission which *celldetect* misses.

```
unix% punlearn asphist
unix% asphist \
      infile="pcadf052378346N002_asol1.fits[@data/acisf00578N002_evt2.fits[GTI5]]" \
      outfile="asphist_5.fits" evtfile="data/acisf00578N002_evt2.fits" mode="h"

unix% punlearn mkinstmap
unix% mkinstmap outfile="instmap_1.8kev_5.fits" monoenergy="1.8" \
      pixelgrid="1:1024:#1024,1:1024:#1024" obsfile="asphist_5.fits[asphist]" \
      detsubsys="ACIS-5" mode="h"

unix% punlearn mkexpmap
unix% mkexpmap asphistfile="asphist_5.fits" outfile="expmap_1.8kev_5_2048.fits" \
      instmapfile="instmap_1.8kev_5.fits" normalize="no" \
      xygrid="0.5:8192.5:#2048,0.5:8192.5:#2048" mode="h"
```

The resulting file is named `expmap_1.8kev_5_2048.fits`.

Running *celldetect* without an Exposure Map

First, *celldetect* is run without an exposure map:

- Set the necessary parameters:

  ```
  unix% punlearn celldetect
  unix% pset celldetect infile = "acisf00578N002_evt2_ccdid5_imgbin4.fits"
  unix% pset celldetect outfile = example2a_out.fits
  unix% pset celldetect regfile = example2a_out.reg
  unix% pset celldetect ellsigma = 5
  ```

  – `ellsigma`: increased to make the display of detected regions easier to see.
  – `regfile`: an ASCII region file.

  All other parameters are left at their default values.

- Execute *celldetect*:

  ```
  unix% celldetect
  Input file (acisf00578N002_evt2_ccdid5_imgbin4.fits):
  Output source list (example2a_out.fits):
  unix%
  ```

  The results are shown in Figure 4.5.

Running *celldetect* with an Exposure Map

Next, *celldetect* is run with an exposure map:

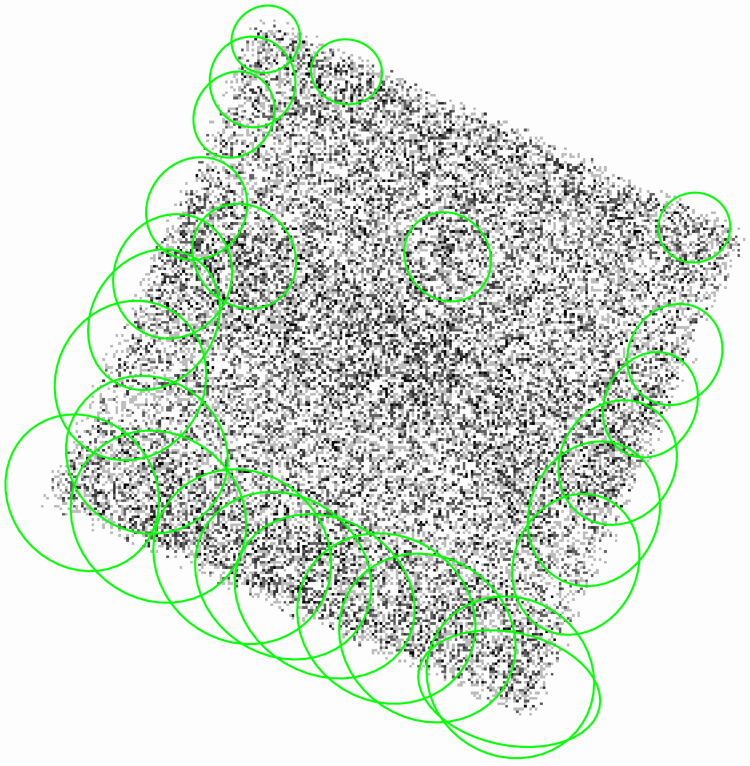

Figure 4.5: Spurious edge detections by *celldetect*. Chip S1 (`ccd_id=5`) of the ACIS-S dataset of 3C 295, binned by 4. The ellipses show the *celldetect* results achieved without the use of an exposure map. Compare these results with those obtained using an exposure map (Figure 4.6). In particular, note that using an exposure map helps to eliminate spurious detections along the chip edges.

- Set the necessary parameters:

  ```
  unix% punlearn celldetect
  unix% pset celldetect infile = "acisf00578N002_evt2_ccdid5_imgbin4.fits"
  unix% pset celldetect outfile = example2b_out.fits
  unix% pset celldetect regfile = example2b_out.reg
  unix% pset celldetect ellsigma = 5
  unix% pset celldetect expstk="expmap_1.8kev_5_2048.fits"
  unix% pset celldetect expratio=0.9
  ```

  Two additional parameters are used for the exposure map case:

  – `expratio`: the average exposure of the background frame divided by the average exposure in the detect cell. Sources whose `EXPO_RATIO` falls below this value are excluded from the ASCII source region file.
  – `expstk`: the exposure map to be used in the detection.

- Execute *celldetect*:

  ```
  unix% celldetect
  Input file (acisf00578N002_evt2_ccdid5_imgbin4.fits):
  Output source list (example2b_out.fits):
  unix%
  ```

  The results are shown in Figure 4.6.

4.4 Examples Using Dataset C (3C 273 on HRC-I)

4.4.1 Example 1: Detecting Close Extended Sources

This example illustrates running *celldetect* using a *Chandra* HRC-I dataset of 3C 273 as input; see Chapter 2, Section 2.3 for further information about this dataset.

Setting Parameters

Set the usual parameters:

```
unix% punlearn celldetect
unix% pset celldetect infile = \
"data/hrcf00461N003_evt2.fits[EVENTS][(x,y)=rotbox(16488.2,16285.8,512,512,0)]"
unix% pset celldetect outfile = example1_out.fits
unix% pset celldetect regfile = example1_out.reg
unix% pset celldetect ellsigma = 5
```

DM syntax is used to specify only a portion of the file for input (*i.e.* `[(x,y)=rotbox(16488.2,16285.8,512,512,0)]`); see Chapter 1, Section 1.3.3 for further information about DM syntax usage with *celldetect*.

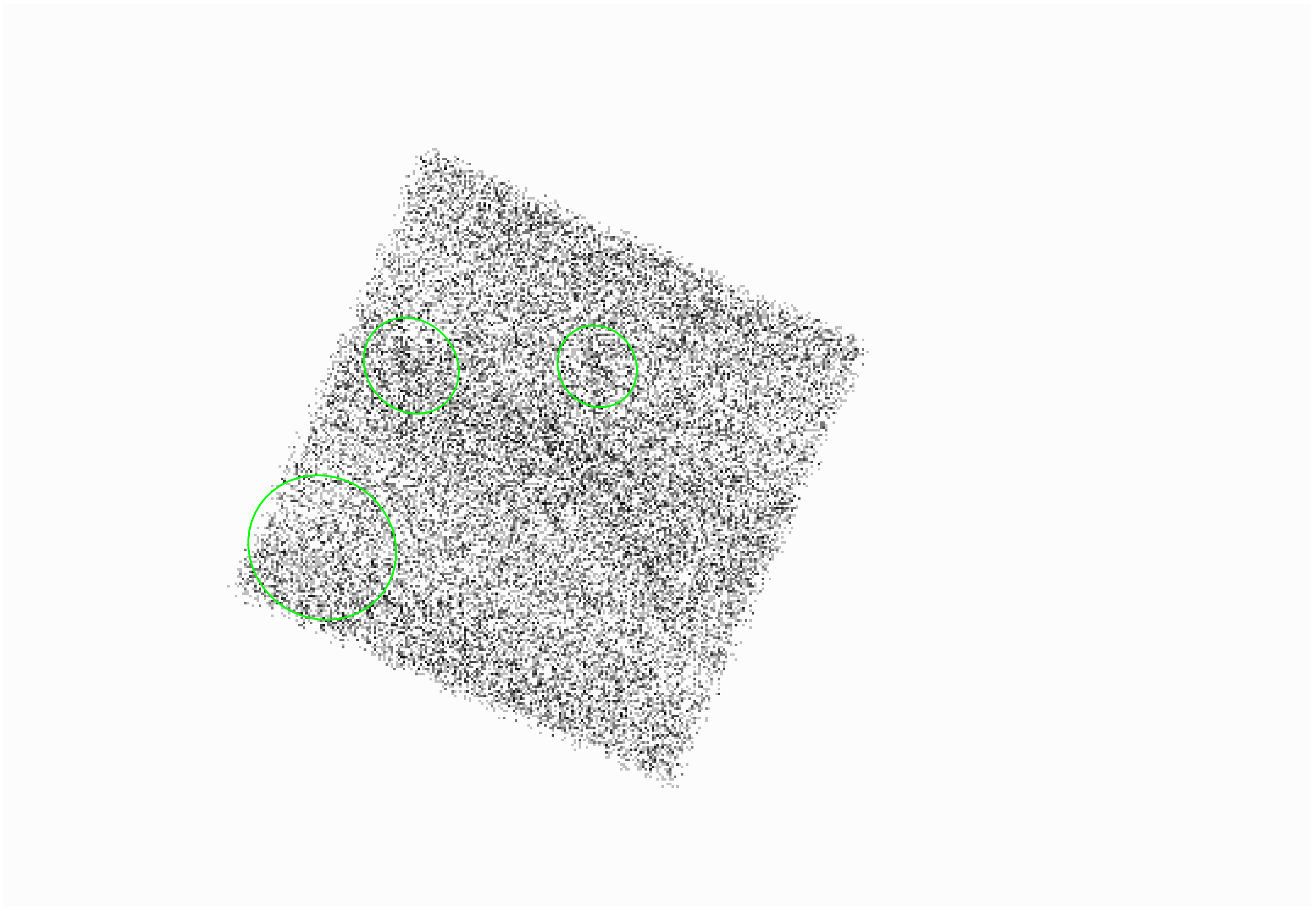

Figure 4.6: Eliminating edge detections in *celldetect* with an exposure map. Chip S1 (`ccd_id=5`) of the ACIS-S dataset of 3C 295, binned by 4. The ellipses show the *celldetect* results achieved with the use of an exposure map. Compare these results with those obtained without an exposure map (Figure 4.5). In particular, note that using an exposure map helps to eliminate spurious detections along the chip edges.

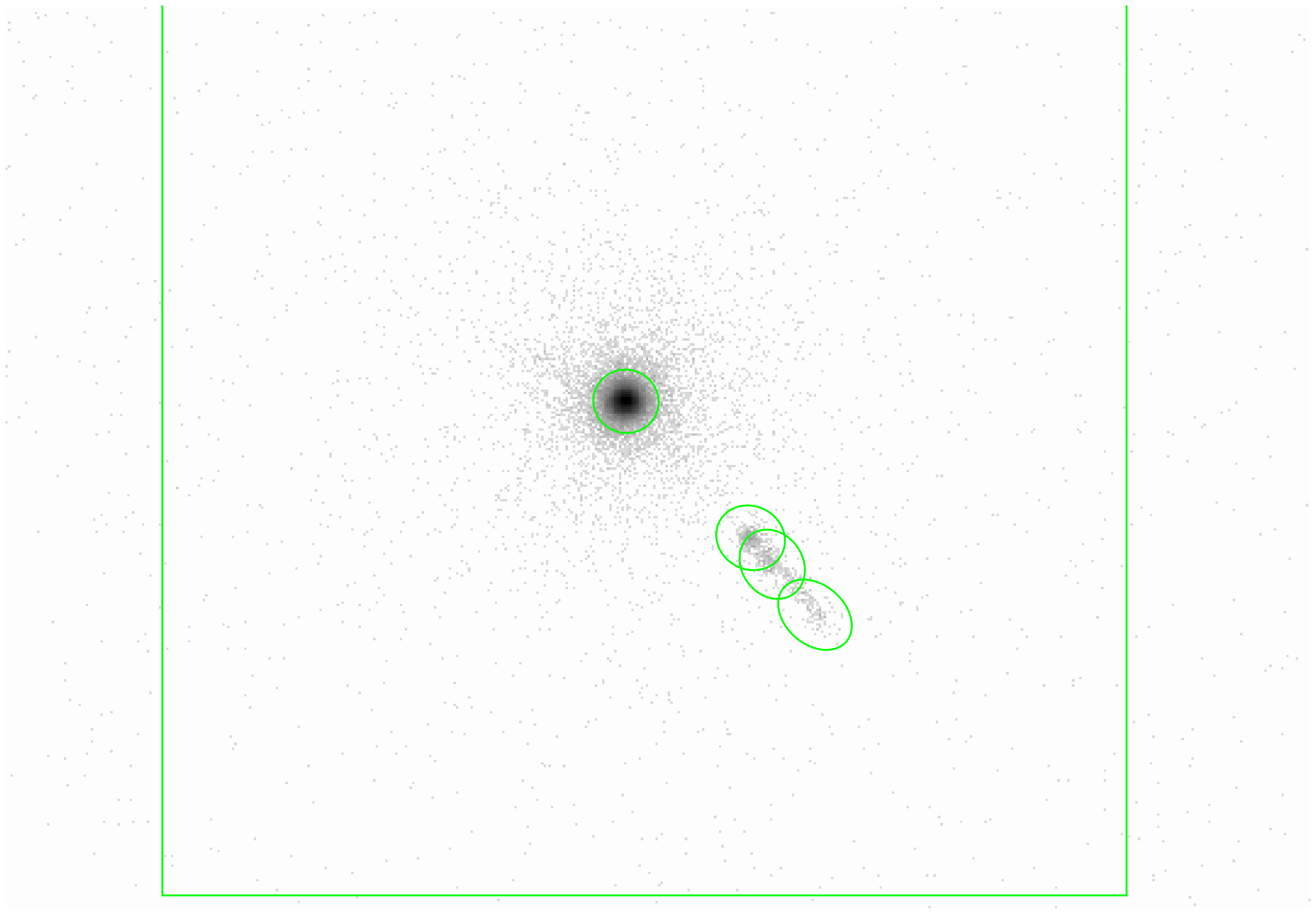

Figure 4.7: Detecting close extended sources with *celldetect*. The HRC-I dataset of 3C 273, with ellipses showing the *celldetect* results. The box shows the region of the chip searched by *celldetect*.

Executing *celldetect*

```
unix% celldetect
Input file (data/hrcf00461N003_evt2.fits[EVENTS][(x,y)=rotbox(16488.2,16285.8,512,512,0)]):
Output source list (example1_out.fits):
unix%
```

The results are shown in Figure 4.7.

4.4.2 Example 2: Detecting Large Extended Sources with a Fixed Cell Size

The previous example is repeated, but the fixed cell size option is used, meaning the detect cell size is kept constant for the entire dataset.

Setting Parameters

```
unix% punlearn celldetect
unix% pset celldetect infile = \
"data/hrcf00461N003_evt2.fits[EVENTS][(x,y)=rotbox(16488.2,16285.8,512,512,0)]"
unix% pset celldetect outfile = example2_out.fits
unix% pset celldetect regfile = example2_out.reg
unix% pset celldetect ellsigma = 5
unix% pset celldetect fixedcell = 60
```

The `fixedcell` parameter to match the expected size of the extended emission sources.

Executing *celldetect*

```
unix% celldetect
Input file (data/hrcf00461N003_evt2.fits[EVENTS][(x,y)=rotbox(16488.2,16285.8,512,512,0)]):
Output source list (example2_out.fits):
unix%
```

The results are shown in Figure 4.8.

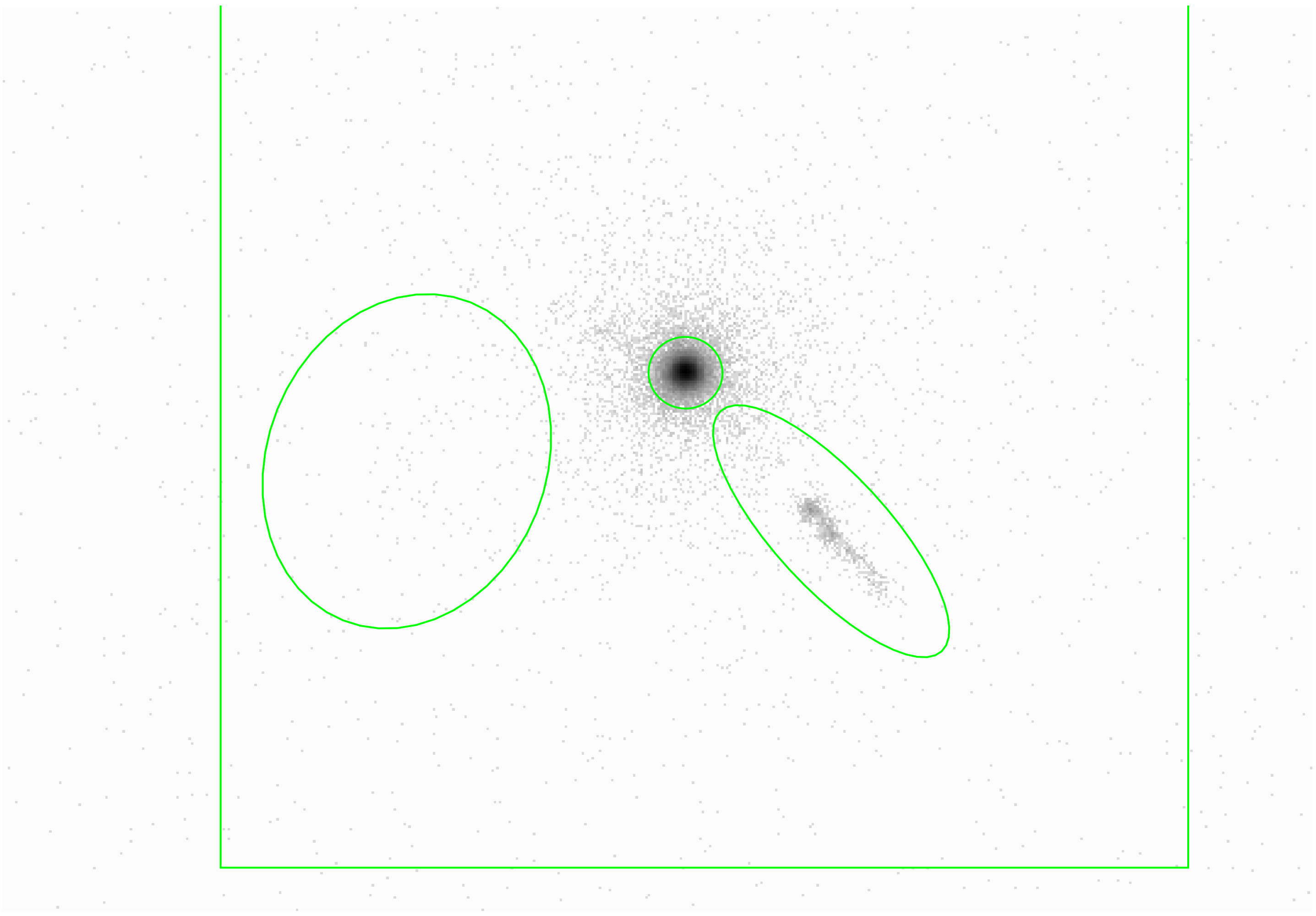

Figure 4.8: Detecting large extended sources with fixed cell size in *celldetect*. The HRC-I dataset of 3C 273, with ellipses showing the `fixedcell` *celldetect* results. The box shows the region of the chip searched by *celldetect*.

Chapter 5

False Detection Rates in celldetect

5.1 Introduction

In order to study the rates of false source detections from *celldetect*, the tool was run on sets of simulated *Chandra* data that contained only a background-like component. This chapter is provided mainly for users performing statistical analysis of of source detections; it describes the simulations, the *celldetect* parameter settings used, and the results.

5.2 Description of Simulations

The simulated data were created with *MARX* (Reference 1). Three sets of simulations, each having a different exposure time, were produced for the ACIS-I and HRC-I detectors. The 10ks and 30ks sets each contained 99 simulations. The 100ks set contained 75 simulations for ACIS-I and 50 for HRC-I. The longer exposure times require longer *celldetect* run times and greater disk space, which necessitated fewer simulations. Although the 100ks set contained fewer simulations, the statistical significance of the *celldetect* results is not diminished, as these simulations proportionally yielded more detected sources.

The simulations used a monochromatic source (1.49 keV) with a uniform disk distribution of angular extent greater than the detector field of view. The flux of the disk source was set such that the total detected count rate was $\sim 7.2 \times 10^{-7}$ events s^{-1}pixel^{-1} for ACIS-I and $\sim 3.7 \times 10^{-8}$ events s^{-1}pixel^{-1} for HRC-I. These background rate assumptions are expected on-orbit values based on the *Science Instrument Calibration Report for the AXAF CCD Imaging Spectrometer (ACIS)* (Reference 2), and the *HRC Ground Calibration Final Report* (Reference 3).

The use of a monochromatic disk as the sole source of photons is a simplified approximation to the expected true background in two ways: (1) such simulations are vignetted, but the true background will contain both a vignetted external component and a non-vignetted internal detector background component, and (2) the true background's vignetted external component will be non-monochromatic. However, we found that radial profiles of counts vs. off-axis angle matched to within ~1-2% those obtained for a number of other simulations that had both an external component with a realistic energy spectrum (based on *Rosat* data) and a flat internal component. This is expected since vignetting is a strong function of energy. Adding a flat component makes relative vignetting smaller, and these effects tend to offset one another at 1.49 keV (near which both detectors have their peak effective areas). In addition, usage of monochromatic simulations provided for simpler and faster simulation generation. Given that the predicted background count rate assumptions used for the simulations are $\pm \sim$10%, the $\sim$1-2% effects caused by this study's simplified approximations should be acceptable.

5.3 celldetect Settings

celldetect was run on these simulations using the current default settings, which were:

1. a variable detect cell size, which increases with off-axis distance as the point spread function (PSF) also increases.
2. an encircled energy value of 0.80, which indicates the encircled energy fraction of the PSF that the detect cell must contain.
3. the local detect method of estimating background, which takes the detect cell background from a region surrounding the detect cell.
4. the findpeaks=yes setting, which recognizes adjacent detections as a single source.
5. the centroid=yes setting, which calculates the source centroid for the detection source list.

Chapter 6 has further information regarding the *celldetect* algorithm, and Chapter 7 describes the *celldetect* parameter settings in detail.

5.4 Results

It is important to keep in mind that since the simulated data contained only a background-like component, all detections are false sources.

Traditionally, the false source rate is expressed per field of view. The field of view is defined here as the entire detector for HRC-I (equivalent to $\sim$0.25 deg^2) and as the four imaging chips in ACIS-I ($\sim$0.08 deg^2). The rates considered in this analysis are 0.1, 1, and 10 false sources per field of view.

The distribution of the signal-to-noise ratio (S/N) values of the false sources was examined in order to establish S/N threshold values for the given rates of false sources. For each exposure time, each false source detection rate, and each off-axis angle bin, the considered false source rate was scaled by the area of that bin and multiplied by the number of simulations. This yielded the number of "allowed" false source detections for that region in all simulations combined. The S/N values of all false source detections in that bin were sorted, and the S/N value of the last "allowed" source was identified. This is the S/N threshold value data point for that bin.

This calculation, which introduces angular dependence to the S/N threshold value for a particular false source detection rate, provides for a uniform distribution of false sources over the entire detector field of view.

We found that the S/N threshold for false sources grows with off-axis angle (see Figures 5.1, 5.2, and 5.3). This effect is expected, since the photon statistics in random simulations improve as the size of the detect cell grows with off-axis position. Note, however, that the data begin to flatten at large off-axis angles. This is due to mirror vignetting, which causes a reduction in the number of photons in the outer regions of the field of view. This in turn results in poorer photon statistics, and thus a flattening of S/N threshold values vs. off-axis angle at larger angles. The effect is especially prominent in HRC-I, which has a larger field of view than ACIS-I (see Figure 5.3).

Figures 5.1, 5.2, and 5.3 show the S/N threshold that will yield a particular false source detection rate as a function of the off-axis angle for three exposure times (10ks, 30ks, and 100ks). The tabular data for these figures may be found in Section 5.6.1.

The S/N threshold quoted here is the formal S/N as defined in Chapter 6, Eq. (6.7). It should not be mistaken for source "significance." The S/N threshold in the center of the image is quite low, especially for low exposure times. This is expected since the background in *Chandra* observations is very low and the expected number of background counts in small cells in the center of the image is effectively zero. Virtually all false sources identified in the center of the image for low exposure times have no counts in the associated background regions.

The S/N threshold dependence on the off-axis angle can be well approximated with the following function:

$$\text{S/N threshold} = (A + B\theta)(1 - C\theta^2) \tag{5.1}$$

where θ is the off-axis angle, and A, B, and C are the function coefficients. With no vignetting present, the linear component would suffice; in such a case, S/N would grow as the square root of the number of counts, while the growth of the PSF – and of the detect cell area – would be close to quadratic. With vignetting present, the equation had to be modified by a multiplicative component, which was found empirically to be parabolic.

Fitted curves using the function from Eq. (5.1) are shown in Figures 5.1, 5.2, and 5.3 as dashed lines for the rate of 0.1, as solid lines for the rate of 1.0, and as dotted lines for the rate of 10.0 false sources per field of view. The fitted values of the function coefficients A, B, and C are shown in Table 5.3 in Section 5.6.2.

Equation (5.1) works especially well for off-axis angles lower than 10 arcmin (i.e. for the region of the data where PSF sizes are small; see Figures 5.1 and 5.2). This region corresponds to the entire field of view of all four ACIS-I chips and the inner portion of HRC-I.

5.5 Summary

The data and curves in Figures 5.1, 5.2, and 5.3 should be utilized in the following way: if from a *celldetect* run the user selects only those sources that lie above the curve corresponding to N false sources per field of view, then the user can expect to have N false detections among those sources over the entire detector. These detections are distributed uniformly over the field of view; there are no preferred positions for false sources.

In addition, since no single value for the *celldetect* `thresh` parameter works universally, it is recommended that the user exercise caution when selecting the value of this parameter for their particular analysis. When S/N threshold is set too low, large numbers of spurious sources may result (which may in turn unnecessarily increase computation time). When this parameter is set too high, real source detection may be significantly impaired.

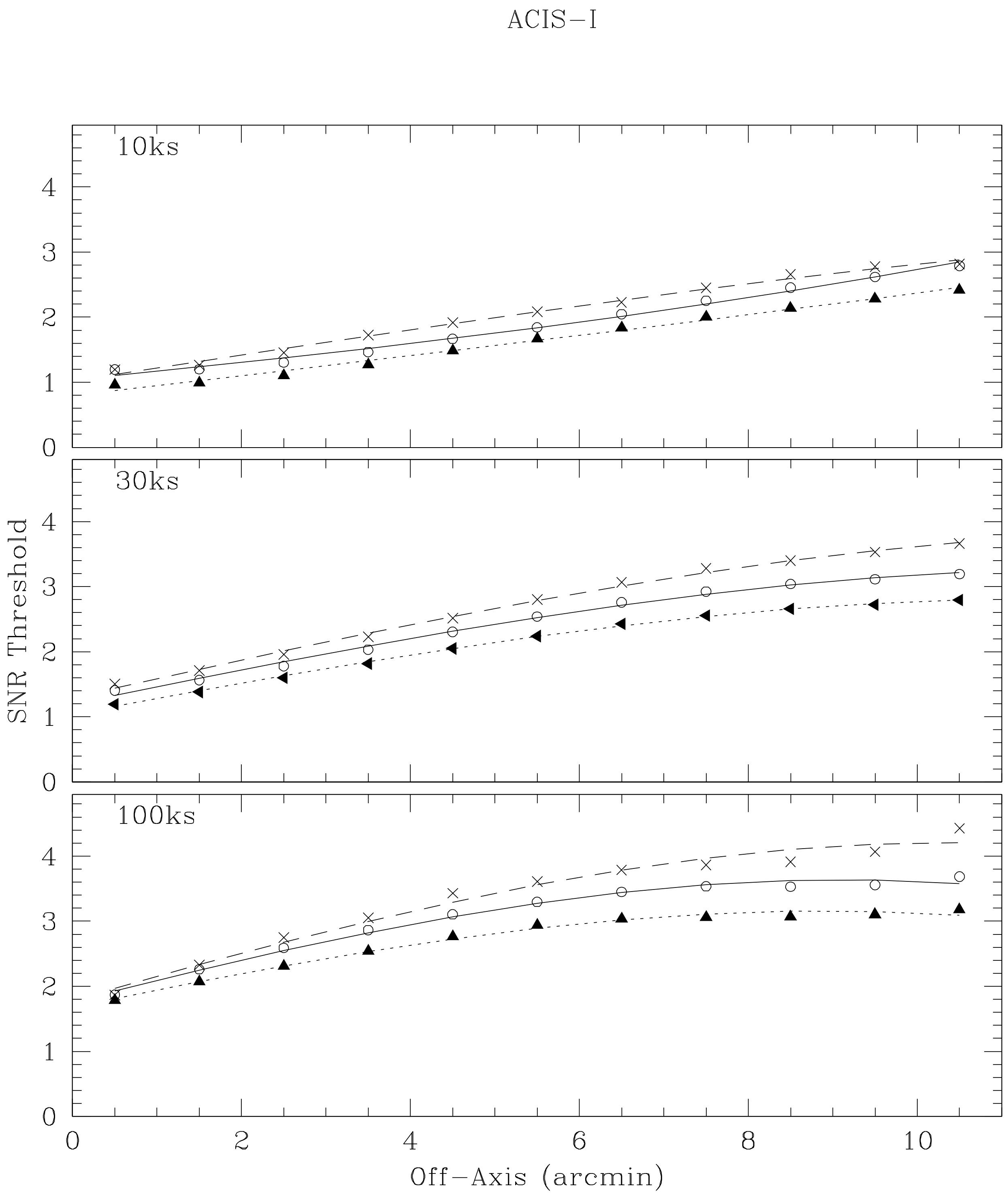

Figure 5.1: **ACIS-I Field of View.** The S/N threshold that will yield a particular false source detection rate, as a function of the off-axis angle. The rates are shown with the best-fit overlaid: 0.1 (cross, dashed line), 1 (circle, solid line), and 10 (triangle, dotted line) false sources per field of view. The data and the best fit function coefficients may be found in Section 5.6.

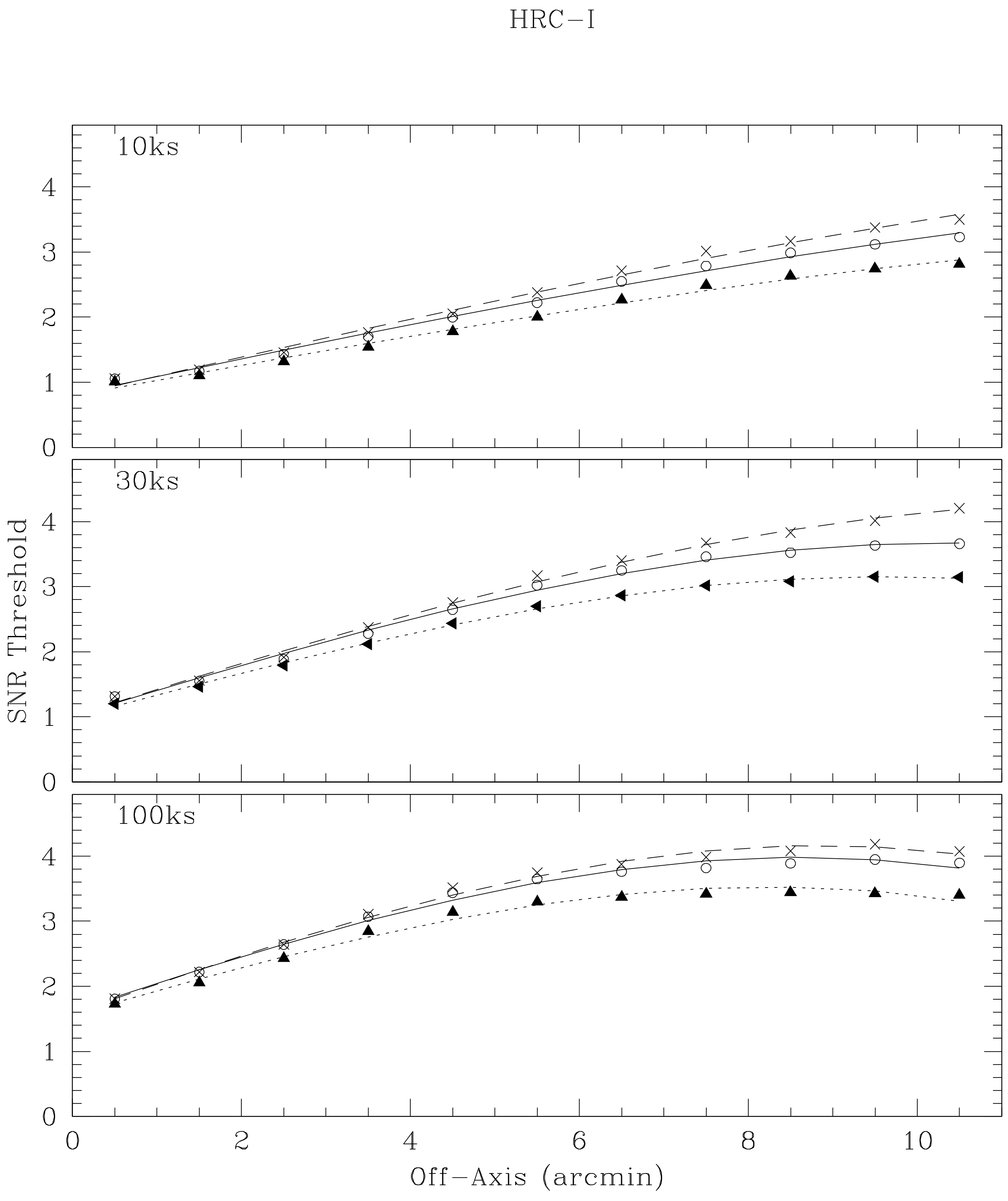

Figure 5.2: **Inner Region of HRC-I.** The S/N threshold that will yield a particular false source detection rate, as a function of the off-axis angle. The rates are shown with the best-fit overlaid: 0.1 (cross, dashed line), 1 (circle, solid line), and 10 (triangle, dotted line) false sources per field of view. The data and the best fit function coefficients may be found in Section 5.6.

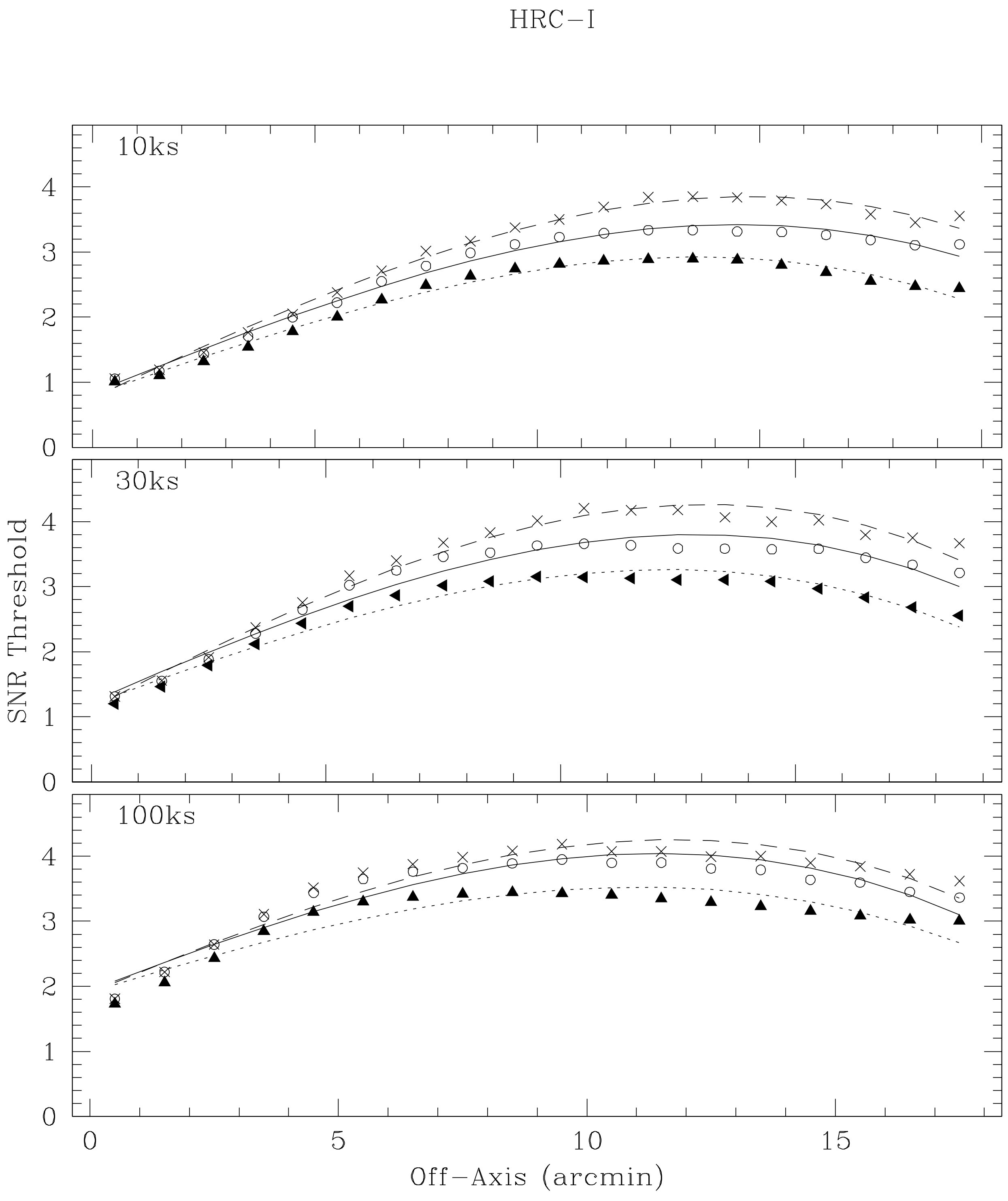

Figure 5.3: **Outer Region of HRC-I.** The S/N threshold that will yield a particular false source detection rate, as a function of the off-axis angle. The rates are shown with the best-fit overlaid: 0.1 (cross, dashed line), 1 (circle, solid line), and 10 (triangle, dotted line) false sources per field of view. The data and the best fit function coefficients may be found in Section 5.6.

5.6 Appendix

5.6.1 S/N Threshold Values vs. Off-Axis Angle Data

Each column in the table gives the S/N threshold values, in 1 arcmin bins, above which there would be N false sources in the entire field of view. For example, if the user detected a number of sources and they all have a S/N value higher than the tabulated value for their off-axis angle for $N = 10$, then the user may expect that 10 of the sources are false detections.

	Exposure time								
	10 ks			30 ks			100 ks		
Off-axis angle	False source rate			False source rate			False source rate		
[arcmin]	0.1	1	10	0.1	1	10	0.1	1	10
0.5	1.19	1.19	0.96	1.50	1.41	1.19	1.87	1.87	1.78
1.5	1.26	1.20	0.99	1.71	1.57	1.38	2.33	2.26	2.07
2.5	1.45	1.30	1.11	1.96	1.78	1.60	2.75	2.59	2.31
3.5	1.72	1.46	1.27	2.23	2.03	1.82	3.05	2.86	2.54
4.5	1.92	1.67	1.49	2.51	2.31	2.05	3.43	3.11	2.77
5.5	2.08	1.84	1.67	2.80	2.54	2.24	3.61	3.30	2.94
6.5	2.23	2.05	1.83	3.06	2.76	2.43	3.79	3.45	3.04
7.5	2.45	2.25	2.00	3.28	2.92	2.56	3.87	3.54	3.06
8.5	2.65	2.45	2.14	3.40	3.04	2.66	3.91	3.53	3.07
9.5	2.77	2.62	2.28	3.53	3.11	2.72	4.07	3.56	3.10
10.5	2.82	2.79	2.42	3.66	3.19	2.80	4.43	3.69	3.18

Table 5.1: **ACIS-I.** S/N thresholds for three exposure times as a function of off-axis angle. Rates are over the full ACIS field of view.

5.6.2 Fitted Function Coefficients

Table 5.3 shows the fitted values of the coefficients in Equation (5.1) for all exposure times and false source rates analyzed here.

5.7 References

1. *MARXUser Guide*, `http://space.mit.edu/CXC/MARX/`.

2. *Science Instrument Calibration Report for the AXAF CCD Imaging Spectrometer (ACIS)*, `http://www.astro.psu.edu/xray/docs/cal_report/cal_report.html`.

3. *HRC Ground Calibration Final Report*, `http://hea-www.harvard.edu/HRC/calib/hrccalib_a_180998.ps`.

	Exposure time								
	10 ks			30 ks			100 ks		
Off-axis angle	False source rate			False source rate			False source rate		
[arcmin]	0.1	1	10	0.1	1	10	0.1	1	10
0.5	1.06	1.06	1.00	1.31	1.31	1.20	1.81	1.81	1.73
1.5	1.19	1.17	1.11	1.56	1.55	1.47	2.22	2.22	2.05
2.5	1.46	1.43	1.32	1.91	1.88	1.79	2.64	2.64	2.43
3.5	1.77	1.70	1.54	2.37	2.28	2.12	3.11	3.07	2.85
4.5	2.05	2.00	1.78	2.75	2.65	2.44	3.52	3.44	3.14
5.5	2.38	2.22	2.01	3.17	3.02	2.70	3.75	3.65	3.30
6.5	2.71	2.55	2.27	3.40	3.25	2.87	3.87	3.76	3.37
7.5	3.02	2.79	2.49	3.67	3.46	3.02	3.99	3.82	3.42
8.5	3.16	2.99	2.63	3.83	3.52	3.08	4.08	3.89	3.45
9.5	3.38	3.12	2.74	4.01	3.63	3.15	4.18	3.95	3.43
10.5	3.50	3.23	2.82	4.20	3.66	3.14	4.07	3.90	3.40
11.5	3.69	3.29	2.86	4.17	3.64	3.13	4.07	3.90	3.35
12.5	3.84	3.34	2.89	4.18	3.59	3.11	3.99	3.81	3.29
13.5	3.85	3.34	2.89	4.06	3.58	3.10	4.00	3.79	3.23
14.5	3.84	3.32	2.88	4.00	3.57	3.08	3.90	3.63	3.15
15.5	3.79	3.31	2.80	4.02	3.58	2.97	3.84	3.59	3.09
16.5	3.74	3.26	2.69	3.80	3.44	2.84	3.72	3.45	3.02
17.5	3.58	3.19	2.55	3.75	3.34	2.68	3.62	3.36	3.00
18.5	3.45	3.11	2.47	3.67	3.21	2.56	....	....	....
19.5	3.55	3.12	2.44	....	....	....	....	....	....

Table 5.2: **HRC-I.** S/N thresholds for three exposure times as a function of off-axis angle. Rates are over the full HRC-I field of view.

		Exposure time								
		10 ks			30 ks			100 ks		
		False source rate			False source rate			False source rate		
Detector	Coeff.	0.1	1	10	0.1	1	10	0.1	1	10
ACIS-I	*A*	1.0153	1.0378	0.7850	1.2899	1.1969	1.0385	1.7867	1.7674	1.6772
Whole	*B*	0.2035	0.1286	0.1574	0.2956	0.2675	0.2440	0.3719	0.3315	0.2700
FOV	*C*	0.0008	0.0017	0.0000	0.0015	0.0018	0.0020	0.0024	0.0029	0.0029
HRC-I	*A*	0.7909	0.8156	0.7924	1.0139	1.0139	0.9833	1.5840	1.6203	1.5541
Inner	*B*	0.3009	0.2747	0.2362	0.4097	0.3991	0.3525	0.4592	0.4347	0.3817
Region	*C*	0.0009	0.0010	0.0011	0.0020	0.0027	0.0031	0.0035	0.0035	0.0037
HRC-I	*A*	0.7611	0.8361	0.7961	1.1268	1.2286	1.1892	1.8950	1.9315	1.9064
Whole	*B*	0.3205	0.2758	0.2425	0.3706	0.3127	0.2666	0.3210	0.2969	0.2381
FOV	*C*	0.0013	0.0014	0.0015	0.0017	0.0017	0.0018	0.0018	0.00019	0.0018

Table 5.3: Coefficients for best fits to Eq. (5.1) for three exposure times. The false source rates are over the full field of view of each detector (0.08 deg^2 for ACIS-I, 0.25 deg^2 for HRC-I).

Chapter 6

celldetect Theory

6.1 Algorithm Overview

The CXC *celldetect* tool utilizes a "sliding cell" search for source candidates in an event list or an image. The tool is based on the programs from the PROS DETECT package, which was based on techniques used in the standard processing of *EINSTEIN* Observatory IPC and HRI data and in the Level 1 processing of *Rosat* HRI data.

The core of the tool is scanning an x-ray image with a square "detect cell." At each location on the image the signal-to-noise ratio (S/N) of a source assumed to be centered in the detect cell is calculated, with background estimated from a box containing the detect cell (unless a separate background image or a global background value is provided, in which case the background is estimated from the same detect cell).

If the S/N is higher than a user-defined threshold, the tool identifies the cell as the location of a source candidate.

6.2 Algorithm Details

Borrowing from the algebra derived for the *EINSTEIN* IPC Rev. 1 LDETECT programs (Harnden et al. 1984), we define:

S	=	Expected total source counts
B	=	Expected background counts in the detect cell
C	=	Total counts in a square detect cell of size d
T	=	Total counts in the detect cell plus surrounding background frame (a $b \times b$ box)
Q	=	$T - C$
α	=	Integral of the point spread function (PSF) over the detect cell
β	=	Integral of the PSF over the detect cell and surrounding background frame

There are three cases that need to be considered here:

1. Background is estimated from the background frame
2. Background is estimated from the background map or background value and background uncertainty is known
3. Background is estimated from the background map or background value and background uncertainty is negligible

We will consider these three cases in order.

6.2.1 Background Estimated from the Background Frame

We have:

$$\begin{aligned} C &= \alpha S + B && (6.1)\\ T &= \beta S + \left(\frac{b}{d}\right)^2 B. && (6.2) \end{aligned}$$

Solving for B, we get:

$$\begin{aligned} B &= \frac{(\alpha-\beta)C + \alpha Q}{\alpha b^2 d^{-2} - \beta} && (6.3)\\ \sigma_B^2 &= \frac{(\alpha-\beta)^2\sigma_C^2 + \alpha\sigma_Q^2}{(\alpha b^2 d^{-2} - \beta)^2} && (6.4) \end{aligned}$$

Solving for S, we get:

$$\begin{aligned} S &= \frac{C(b^2-d^2)d^{-2} - Q}{\alpha b^2 d^{-2} - \beta} && (6.5)\\ \sigma_S^2 &= \frac{\sigma_C^2(b^2-d^2)^2 d^{-4} + \sigma_Q^2}{(\alpha b^2 d^{-2} - \beta)^2} && (6.6) \end{aligned}$$

S/N is then:

$$\text{S/N} = \frac{S}{\sigma_S} = \frac{C(b^2-d^2)d^{-2} - Q}{\sqrt{\sigma_C^2(b^2-d^2)^2 d^{-4} + \sigma_Q^2}} \qquad (6.7)$$

Note that the estimate for S/N in Eq. (6.7) is independent of α and β.

The Gehrels' (1986) formula for uncertainty is used:

$$\sigma_N = 1 + \sqrt{N + 0.75}, \qquad (6.8)$$

where N can be either C or Q. This simple formula gives excellent approximation for uncertainty (better than 1.5% for small N and better than a fraction of a percent for large N). In this case, we have:

$$\text{S/N} = \frac{C(b^2-d^2)d^{-2} - Q}{\sqrt{(1+\sqrt{C+0.75})^2(b^2-d^2)^2 d^{-4} + (1+\sqrt{Q+0.75})^2}} \qquad (6.9)$$

6.2.2 Estimating Background with Known Uncertainty

B and σ_B are directly calculated from either the background value b:

$$B = d^2 b \qquad\qquad \sigma_B = d^2 \sigma_b$$

or from the background map:

$$B = \sum b_{ij} \qquad\qquad \sigma_B^2 = \sum \sigma_{b_{ij}}^2$$

where summing is done over the detect cell, and b_{ij} and $\sigma_{b_{ij}}$ are the values in the (i, j) pixel of the background map and background error map, respectively.

The equation for S/N becomes:

$$\text{S/N} = \frac{C - B}{\sqrt{\sigma_C^2 + \sigma_B^2}} = \frac{C - B}{\sqrt{(1 + \sqrt{C + 0.75})^2 + \sigma_B^2}} \tag{6.10}$$

6.2.3 Estimating Background with Negligible Uncertainty

The equation for source counts reduces to:

$$S = \frac{C - B}{\alpha} \tag{6.11}$$

$$\sigma_S = \frac{\sigma_C}{\alpha} \tag{6.12}$$

S/N is equal to:

$$\text{S/N} = \frac{C - B}{\sigma_C} \tag{6.13}$$

or, using Eq. (6.8):

$$\text{S/N} = \frac{C - B}{1 + \sqrt{C + 0.75}} \tag{6.14}$$

6.3 Cell Size

The *Chandra* PSF varies strongly with the off-axis angle and photon energy. Therefore, different detect cell sizes are appropriate for different spatial regions and energies.

The user specifies the basis for the calculation of the cell size by providing the reference energy band and desired encircled energy to be encompassed in the detect cell. At each position, *celldetect* calculates the radius in pixels at which PSF reaches this encircled energy for a given combination of telescope, detector and grating, and then rounds the “diameter” (i.e. radius times 2) up to the nearest multiple of 3 pixels. The rounding is done because the search for detections moves in steps of 1/3 of a detect cell.

Experience with the *EINSTEIN* and *Rosat* detection software indicates that the optimum cell size is approximately 3.6 times the sigma of the PSF (Harnden et al. 1984). For a Gaussian PSF, that corresponds to an encircled energy seting of 0.80.

Figure 6.1 shows growth of the cell size as a function of off-axis angle for the HRC-I detector.

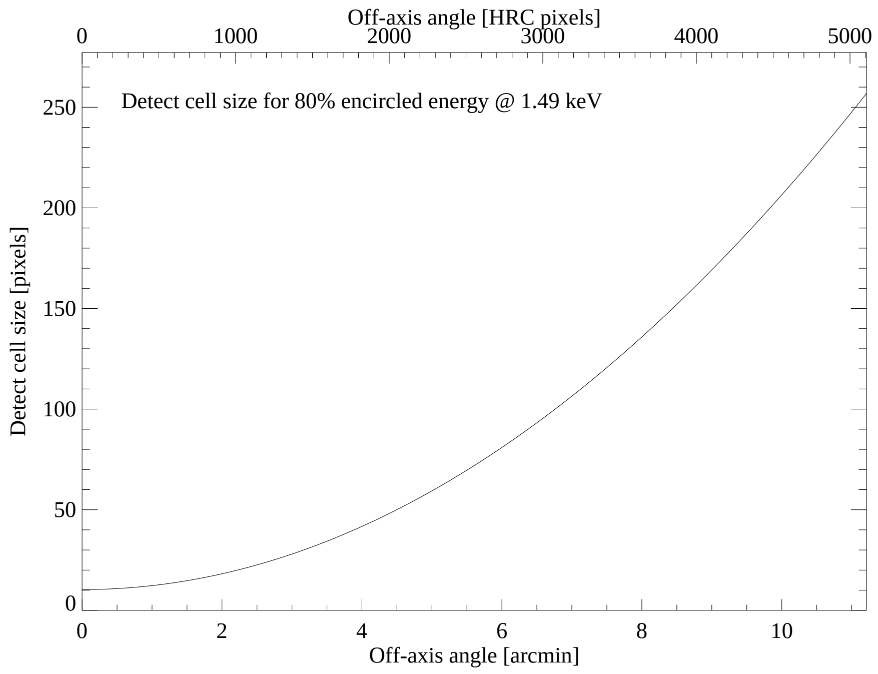

Figure 6.1: Cell Size in Pixels as a Function of Off-axis Angle for HRC-I.

6.4 Background Frame

If the local background option is used, the background counts are estimated from a frame surrounding the detect cell. The background frame size is selected to give the area of the background frame roughly twice the size of the detect cell. This makes the detect cell and the background "annulus" similar in size. The parity of the background cell size is the same as of the detect cell size, so that the background frame and detect cell are centered on the same pixel and there are no asymmetries.

6.5 Output Options

The user provides the S/N threshold value (S/N_{thr}). The tool moves across the data in steps of 1/3 of the local detect cell size searching for locations in which:

$$S/N \geq S/N_{thr} \tag{6.15}$$

The tool gives three options for detection output: all detections, local S/N peaks, and centroids of events in detect cells.

6.5.1 All Detections

celldetect can output all detect cells in which the condition from Eq. (6.15) is fulfilled. In this case, the output information should not be taken literally as a list of sources, since a single source will in all likelihood be reported several times. This option is, however, very useful if external background information is provided and the presence of extended sources is suspected, since it enables the user to identify regions in which there is an excess of events over the background.

6.5.2 Local S/N Peaks

celldetect can output detect cells which fulfill the condition from Eq. (6.15) and which are local S/N peaks, i.e. all their neighboring detect cells have lower S/N. Two cells are considered neighbors if the distance between their centers is not greater than 1/3 of the size of the smaller of the cells.

6.5.3 Centroid of Events in Detect Cells

This option is useful for isolated point sources, since the centroid gives a very accurate estimate of the position of the source. The centroids are calculated by finding the center of gravity for events in the detect cell:

$$\bar{x} = \frac{\sum x_i}{C} \qquad \bar{y} = \frac{\sum y_i}{C} \tag{6.16}$$

where x_i, y_i are coordinates of events in the detect cell, C is the total count of events in the cell, and summation is performed over the spatial extent of the cell.

In this case, the tool calculates centroid uncertainties, as well as the principal axes of the event distribution in the detect cell and the position angle of the major axis. These quantities are calculated utilizing the fact that the square of the principal axes of the distribution of events is mathematically equivalent to the principal moments of inertia for an ensemble of unit mass points.

Let us define:

$$\sigma_x = \sqrt{\frac{\sum (x_i - \bar{x})^2}{C-1}} \qquad \sigma_y = \sqrt{\frac{\sum (y_i - \bar{y})^2}{C-1}} \qquad \sigma_{xy} = \sqrt{\frac{\sum [(x_i - \bar{x})(y_i - \bar{y})]}{C-1}} \tag{6.17}$$

Centroid uncertainties are:

$$\sigma_{\bar{x}} = \frac{\sigma_x}{\sqrt{C}} \qquad \sigma_{\bar{y}} = \frac{\sigma_y}{\sqrt{C}} \tag{6.18}$$

The details of the procedure for calculating the principal axes and the position angle of the major axis can be found in any book on classical dynamics (e.g. Marion 1970). Here we will only list the final formulae for the major and minor axes - A_{maj} and A_{min}, respectively - and the position angle θ of the major axis.

$$A_{\mathrm{min}} = \sqrt{\frac{\sigma_x^2 + \sigma_y^2 - \sqrt{\left(\sigma_x^2 + \sigma_y^2\right)^2 - 4\left(\sigma_x^2\sigma_y^2 - \sigma_{xy}^4\right)}}{2}} \tag{6.19}$$

$$A_{\mathrm{maj}} = \sqrt{\frac{\sigma_x^2 + \sigma_y^2 + \sqrt{\left(\sigma_x^2 + \sigma_y^2\right)^2 - 4\left(\sigma_x^2\sigma_y^2 - \sigma_{xy}^4\right)}}{2}} \tag{6.20}$$

$$\theta = \begin{cases} \arctan\left(\frac{\sigma_x^2 - A_{\mathrm{maj}}^2 - \sigma_{xy}^2}{\sigma_y^2 - A_{\mathrm{maj}}^2 - \sigma_{xy}^2}\right) & \text{if } \sigma_y^2 - A_{\mathrm{maj}} - \sigma_{xy}^2 \neq 0 \\ 90^\circ & \text{otherwise} \end{cases} \tag{6.21}$$

6.6 References

Gehrels, N. 1996 ApJ, 303, 336

Harnden, F. R., Fabricant, D. G., Harris, D. E., & Schwartz, J. 1984, SAO Report No. 393

Marion, J. B. 1970, Classical Dynamics of Particles and Systems, Academic Press, New York

Chapter 7

celldetect Parameters & Data Products Reference

This chapter lists all the *celldetect* parameters in alphabetical order for quick reference. In addition, a guide to the data products output by *celldetect* is provided.

7.1 Default Parameter File

```
unix% plist celldetect

Parameters for /soft/ciao/param/celldetect.par

#
#   celldetect parameter file
#
#
#   input
#
        infile =                  Input file
#
#   output
#
       outfile =                  Output source list
       (expstk = )                list of exposure map files
      (regfile = none)            ASCII regions file
#
#   output options
#
       (kernel = default)         Output file format
      (clobber = no)              Overwrite exiting outputs?
#
```

```
#   output content/format options
#
      (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
#
#   detect cell size parameters
#
    (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 
     (cellfile = )              Output cell size image stack name
#
#   background parameters
#
      (bkgfile = )              Background file name
     (bkgvalue = 0)             Background count/pixel
  (bkgerrvalue = 0)             Background error
#
#   using defaults is recommended here
#
     (convolve = no)            Use convolution?
      (snrfile = )              SNR output file name (for convolution only)
#
#   run log verbosity and content
#
      (verbose = 0)             Log verbosity level
          (log = no)            Make a celldetect.log file?
#
#   mode
#
         (mode = ql)
```

7.2 Parameter Descriptions

In the following parameter descriptions, `"required=yes"` indicates that the user must provide a value for the indicated parameter (*e.g.* `infile`) before the program will run. Also, the empty string (`" "`) is equivalent to the string `none` for filenames.

- `bkgerrvalue`
  Background error
  type=real
  def=0

The error in a background map or value.

The error in the supplied background map (`bkgfile`) may be given using the `bkgerrvalue` parameter; if unknown, this parameter should be left at the default value (0).

- `bkgfile`
  Background file name
  type=string
  filetype=input

  The name of a predetermined background image for the dataset.

  Instead of using *celldetect*'s default local detect procedure, the user may specify the background as an input image file using the parameter `bkgfile`. Note that if a reliable background map is available, then the user will be able to better detect extended sources which are devoid of significant brightness gradients.

  This parameter is ignored if `bkgvalue` is nonzero.

- `bkgvalue`
  Background count/pixel
  type=real
  def=0

  A fixed value (events/pixels) to use for the background.

  The user may instead specify the background as a fixed value of counts/pixel, using the `bkgvalue` parameter. Be aware that if this value is too low, spurious sources will be detected on the periphery of the dataset.

  A non-zero value for `bkgvalue` overrides the normal background estimation and the background file (`bkgfile`) option.

- `cellfile`
  Output cell size image stack name
  type=string
  filetype=output
  autoname=yes

  The name of a stack for file names of cell size images.

  If a filename is provided for the `cellfile` parameter, then one or more images are created which contain the detect cell size at each pixel location. If autonaming is used (*i.e.* `"cellfile=."`), then the output stack file will be named `"<infile_root>_cell"`.

- `centroid`
  Compute source centroids?
  type=boolean
  def=yes

  Compute source centroid positions?

  If `centroid` is set to `yes` (the default), then the source list will be based on the source centroid. In addition, the major and minor axes of the events distribution are calculated as well as the position angle of the major axis. However, if `centroid` is set to `no`, then the center pixel of a source's detect cell is reported as the location of the source.

- `clobber`
  Overwrite exiting outputs?
  type=boolean
  def=no

If "yes", overwrite existing outputs.

celldetect does not not overwrite existing outputs unless `clobber` is set to `yes`.

- `convolve`
  Use convolution?
  type=boolean
  def=no

  If "yes", detection is done using the convolution library instead of sliding cells.

  Setting `convolve` to `yes` will cause detection to be done using the convolution library instead of sliding cells (the default). This method is much slower, does not work with recursive blocking, and should really only be used when an output S/N map is needed.

  The convolve option produces a pixel-by-pixel map of the S/N, which is written to the file specified by `snrfile`.

- `eband`
  Energy band
  type=real
  def=1.4967
  units=keV

  The photon energy (keV) at which the point spread function (PSF) is chosen.

  The selection of a particular PSF size is governed by the energy band `eband` parameter [keV], since the *Chandra* PSF is a function of energy. By default the energy band is set to 1.4967 keV.

- `eenergy`
  Encircled energy of PSF
  type=real
  def=0.8
  max=1.0

  The percentage of PSF energy to be encircled by the detect cell, expressed as a fraction of 1.0.

  celldetect also uses an encircled energy percentage value, given by the `eenergy` parameter, in concert with the calibration PSF file. This value is the percentage of PSF energy to be encircled by the detect cell, expressed as a fraction of 1.0 (default is 0.8). The `eenergy` is a key parameter for detecting off-axis sources, and users are urged to experiment with other values.

- `ellsigma`
  Size of output source ellipses (in sigmas)
  type=real
  def=3.0

  The size, in sigmas, to make the elliptical source detection regions.

  The size of the elliptical source regions in both of the source region files (`regfile` and `outfile`) is controlled via the `ellsigma` parameter. This parameter is a multiplicative factor applied to sigma, the standard deviation of the distribution, to scale the major and minor axes of the ellipses for each source detection.

  The distribution is defined as the distribution of counts within the source cell (*i.e.* the observed counts), and is assumed to be Gaussian. Thus, if `ellsigma` is 1, then the elliptical source region will contain 39.3% of the source counts (*i.e.* the 1-sigma point for a 2-D Gaussian).

  This feature is included so that the graphics overlay may be made more visible; it is does not affect the detections themselves, only the display of the regions.

- `expratio`
  Cutoff ratio for source cell exposure variation
  type=real
  def=0

  Suppresses ASCII source regions for sources whose EXPO_RATIO falls below this value.

  The `expratio` parameter is the ratio of the average exposure of the background frame to the average exposure in the detect cell. Exposure maps supplied by the user are used in conjunction with the `expratio` parameter. For `expratio` values other than 0, exposure maps must be supplied in the `expstk` parameter. Detections for sources whose ratio falls below the specified `expratio` value will be suppressed from the source region file (`regfile`). However, all detections will be present in the output source list (`outfile`), and the column `EXPO_RATIO` will be created. Recommended `expratio` values ares between 0.9 and 0.99.

- `expstk`
  List of exposure map files
  type=string
  filetype=input
  stacks=yes

  The name of an exposure map stack or file.

  To minimize spurious detections along detector edges, one or more exposure maps may be supplied. The `expstk` parameter takes a stack of exposure map filenames; sse "`ahelp stack`" for more information. Each of the exposure map files should have appropriate binning to match *celldetect*'s recursive blocking (*i.e.* binning by 2, 4, 8, ...). Exposure maps only work with the local detect procedure. In addition, they do not work with the convolution option.

- `findpeaks`
  Find local peaks?
  type=boolean
  def=yes

  Find the local maxima of contiguous detections?

  If the `findpeaks` parameter is set to `no`, then the source list will consist of all detection cells which exceeded the S/N threshold (often resulting in single source being identified multiple times). If `findpeaks` is set to `yes` (the default), then adjacent detections are recognized as a single source and the cell with the largest S/N appears in the source list.

- `fixedcell`
  Fixed cell size to use (0 for variable cell)
  type=integer
  def=0
  units=pixels

  A fixed cell size, which *celldetect* will use over the complete field of view.

  celldetect has a fixed cell size option, where the detect cell size is kept constant for the entire dataset. Choose the cell size in pixels by setting the `fixedcell` parameter; allowed values are 1 or an integer divisible by 3. Setting `fixedcell` to a non-zero value effectively disables the parameters related to *celldetect*'s variable cell size functionality: `psftable`, `eenergy`, `eband`, `xoffset`, and `yoffset` parameters.

  If *celldetect* is to be run on non-*Chandra* data, supply a value for `fixedcell`.

  Fixed cell size is also useful for extended source detection. It is well known that local detect methods are not very useful for significantly extended sources. To overcome this limitation, the user may either set `fixedcell` to a size sufficient to match the expected size of an extended source, or employ a background map (or fixed value for the background) instead of using local detect.

- `infile`
  Input file
  type=string
  filetype=input
  required=yes

  The name of the input file.

  The input FITS file can an event list or an image. *CIAO* Data Manipulation (DM) syntax may be used in the input file specification, as explained in Chapter 1, Section 1.3.3. For a FITS event list, *celldetect* will automatically find the `EVENTS` extension; otherwise the extension should be specified using DM syntax (*e.g.* `"data.fits[EVENTS]"`).

  The current maximum size allowed for an image is 2048×2048, but an event list may occupy a larger spatial region. For event list datasets larger than 2048×2048 (as given by `TLMIN, TLMAX`), *celldetect* may use the "recursive blocking scheme": First the inner 2048×2048 pixel region of the dataset is searched for sources, then the inner 4096×4096 pixel region (excluding the area already analyzed) is blocked by 2 and searched for sources, then the inner 8192×8192 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 in higher resolution is skipped, *i.e.* only one blocking factor is used for each region of the dataset.

- `kernel`
  Output file format
  type=string
  def=default

  The format for the output file.

  The `kernel` parameter controls the output format (`default`, `fits`, or `iraf`); the default is for the output file to have the same format as the input file.

- `log`
  Make a celldetect.log file?
  type=boolean
  def=no

  Whether log information will go to a file or STDERR.

  The verbosity of log information is sent to `STDERR` (usually the screen). However, if `log` is set to `yes`, then the log information will be written to the file `celldetect.log`.

- `mode`
  def=ql

  The mode defines how to handle querying for parameters. For example, setting the mode to "h" means that no prompting will occur; this is useful when running the tool from a script. See "ahelp parameter" for more information and a list of all the available modes.

- `outfile`
  Output source list
  type=string
  filetype=output
  required=yes
  autoname=yes

The name of the output file.

The `outfile` parameter is used to provide a name for the output file. The `kernel` parameter controls the output format; the default is for the output file to have the same format as the input file. If autonaming is used (*i.e.* `"outfile=."`), then the output file will be named `"<infile_root>_src"`.

- `psftable`
  Table of PSF size data
  type=string
  filetype=ARD
  def=psfsize20010416.fits

  The path and name of the file containing PSF information.

  celldetect compensates for the larger *Chandra* PSF off-axis by using a "variable cell size" technique, which increases the size of the detect cell with off-axis distance. The PSF information used by *celldetect* is contained in a *Chandra* calibration file distributed with *CIAO*; the location is given by the value of the `psftable` parameter, which points by default to the appropriate *Chandra* calibration PSF file.

  The *Chandra* HRC and ACIS detectors are the sole combinations in the PSF library file used by *celldetect*; information for other missions is not provided. *celldetect* can only utilize the variable cell size procedure if the following keywords are present in the input *Chandra* data file (`infile`): `TELESCOP`, `INSTRUMENT`, `DETNAM`, `GRATING`, `RA_NOM`, and `DEC_NOM`. If any of the keywords are missing, the tool cannot calculate the appropriate cell size and `fixedcell` needs to be specified instead.

- `regfile`
  ASCII region file
  type=string
  autoname=yes

  The name of an ASCII source region output file.

  In addition to the output source list (`outfile`), the user may choose to have an ASCII source region file written by specifying a filename in `regfile` parameter. If autonaming is used (*i.e.* `"regfile=."`), then the source regions output file will be named `"<infile_root>_reg"`.

  This region file contains the location and size of the principal axes, for each detected source elliptical region. The elliptical region size is such that the region contains some fraction of the source counts. The parameter `ellsigma` specifies the desired source count fraction.

  If an exposure map has been used, and `expratio` is non-zero, there may be fewer entries in `regfile` than in `outfile`.

  This region file is useful for subsequent input into ds9, particularly to overlay region markers over each detection; see Chapter 1, Section 1.4 for details.

- `snrfile`
  SNR output file name (for convolution only)
  type=string
  filetype=output
  autoname=yes

  The name of a signal-to-noise (S/N) map output file.

  The convolve option produces a pixel-by-pixel map of the S/N, which is written to the file specified by `snrfile`. A S/N value is computed for every pixel in the input dataset, not just for source locations. The file `snrfile` is then an image of these values. If autonaming is used (*i.e.* `"snrfile=."`), then the S/N output file will be named `"<infile_root>_snr"`.

- `thresh`
  Source threshold
  type=real
  def=3

  The signal-to-noise (S/N) threshold for source detection.

  The detection threshold, `thresh`, is the signal-to-noise (S/N) cutoff for source detection, and should be set to a value of 3 (the default) or 4, particularly for the initial run.

- `verbose`
  Log verbosity level
  type=integer
  def=2
  min=0
  max=5

  The verbosity of log output.

  The verbosity of log information ranges from 0 (none) to 5 (maximum output) and is sent to `STDERR` (usually the screen).

- `xoffset`
  Offset of x axis from data center
  type=integer
  def=INDEF
  units=pixels

  The offsets of the x-axis for the calculation of the off-axis angle.

  By default, *celldetect* calculates the off-axis angle using the nominal pointing of the data file as the origin. Users may change this behavior by setting offsets to any numerical values (using the `xoffset` and `yoffset` parameters); these offsets provide the location of the optical axis with respect to the center of the data file. A typical scenario in which the user may select to use offsets is when the data file is a sum of two or more observations with different pointings. The nominal pointing in such a data file is usually poorly defined and the off-axis angle could be calculated from an undesired origin, leading to suboptimal selection of the detect cell size.

- `yoffset`
  Offset of y axis from data center
  type=integer
  def=INDEF
  units=pixels

  The offsets of the y-axis for the calculation of the off-axis angle.

  See the description for `xoffset`.

7.3 Data Product Descriptions

Source List File (`outfile`)

The primary output of *celldetect* is the FITS file of source detections. If the signal-to-noise (S/N) ratio in a detect cell exceeds the threshold set by the user (`thresh`), then the cell location is considered a source candidate and the following properties are recorded in the output file: [1]

[1] Items marked with "†" are not yet calculated by *celldetect*; these columns in the output file will contain zeros.

Data item	Unit	Description
RA	deg	Source right ascension
DEC	deg	Source declination
RA_ERR	deg	Source right ascension error
DEC_ERR	deg	Source declination error
POS(X,Y)	pixel	Physical coordinates
X_ERR	pixel	Source X position error
Y_ERR	pixel	Source Y position error
CELLPOS(CELL_X,CELL_Y)	pixel	Physical coordinates
DETSIZE	pixel	Source cell size
BKGSIZE	pixel	Background cell size
NPIXSOU	pixel	Pixels in source cell
NPIXBKG	pixel	Pixels in background cell
NET_COUNTS	count	Net source counts
NET_COUNTS_ERR	count	Error in net source counts
BKG_COUNTS	count	Background counts (scaled to source cell)
BKG_COUNTS_ERR	count	Error in background counts
NET_RATE†	count/s	Source count rate
NET_RATE_ERR†	count/s	Source count rate error
BKG_RATE†	count/s	Background count rate
BKG_RATE_ERR†	count/s/pixel	Background count rate error
EXPTIME†	s	Effective exposure time
SNR		Signal-to-noise ratio
SHAPE		Shape of source region
R[2]		Radii of source region
ROTANG	deg	Rotation angle of source region
PSFRATIO		Ratio of source ellipse size to PSF size
BLOCK		Blocking factor
COMPONENT		Source number
EXPO_RATIO		Ratio of average background and detect cell exp
DOUBLE		Double detection flag
DOUBLE_ID		Double source component

- RA, DEC:
  Right ascension and declination coordinates, corresponding to X and Y. Calculated only if WCS keywords are present in the data, otherwise filled with zeros.

- RA_ERR, DEC_ERR:
  RA and DEC errors, corresponding to X_ERR and Y_ERR. Calculated only if input parameter `centroid` is set to `yes`, otherwise "`INDEF`".

- POS(X,Y):
  Location of the detect cell. If input parameter `centroid` is set to `yes`, then X and Y are the centroid of photon distribution in the detect cell. If `centroid` is set to `no`, then X and Y are the location of the cell.

- X_ERR, Y_ERR:
  Uncertainties on the centroid location. Calculated only if input parameter `centroid` set to `yes`, otherwise filled with `INDEF`.

- CELLPOS(CELL_X,CELL_Y):
  The center position of the detect cell.

- DETSIZE:
  Source cell size. The final cell size must be unity or an integer which is divisible by 3. See also NPIXSOU.
- BKGSIZE:
  If either background map (`bkgfile`) or background value (`bkgvalue`) are provided then this value is equal to DETSIZE or NPIXSOU, respectively.

  If background map and background value are *not* provided then the background is estimated from the input data by calculating events in the frame surrounding the detect cell. The size of the background frame is calculated from DETSIZE by multiplying by $\sqrt{2}$ and then rounding it (up or down) to the closest number of the same parity as the detect cell size. (In that way two goals are achieved: first, the area of the background frame is roughly the same as the area of the detect cell, and second, the background frame is centered on the same location as the detect frame.)
- NPIXSOU:
  The actual number of image pixels contained in the source cell; square of DETSIZE.
- NPIXBKG:
  The actual number of pixels in the background frame. If the background frame is entirely within the dataset, then NPIXBKG is equal to $\mathrm{BKGSIZE}^2 - \mathrm{DETSIZE}^2$; it is reduced appropriately if the background frame falls outside the dataset. See also BKGSIZE.
- NET_COUNTS:
  Counts in the detect cell reduced by the number of background counts scaled to the detect cell.
- NET_COUNTS_ERR:
  Estimated uncertainty of NET_COUNTS.
- BKG_COUNTS:
  Background counts scaled to the detect cell area.
- BKG_COUNTS_ERR:
  Estimated uncertainty of BKG_COUNTS.
- NET_RATE†:
  Source count rate: the NET_COUNTS divided by EXPTIME. Not implemented at present; filled with zeros.
- NET_RATE_ERR†:
  Source count rate corresponding error: the NET_COUNTS_ERR divided by EXPTIME. Not implemented at present; filled with zeros.
- BKG_RATE†:
  Background count rate: simply the BKG_COUNTS divided by EXPTIME. Not implemented at present; filled with zeros.
- BKG_RATE_ERR†:
  Background count rate corresponding error: simply the BKG_COUNTS_ERR divided by EXPTIME. Not implemented at present; filled with zeros.
- EXPTIME†:
  Effective exposure time in the detect cell, taken from the exposure map. Not implemented at present; filled with zeros.
- SNR:
  The calculated signal-to-noise ratio in the detect cell.
- SHAPE:
  The shape of region output to the ASCII source region file. If `centroid` is `yes` then SHAPE is `ellipse`; if `centroid` is `no` then SHAPE is `box`.

- R[2]:
  The estimated semi-major and semi-minor axis lengths (*i.e.* the x and y radius lengths in pixels, before rotation). For SHAPE = `ellipse`, this is `ellsigma` times the axis lengths. For SHAPE = `box`, this is `ellsigma` times the cell side lengths.
- ROTANG:
  The counterclockwise angle between the semi-minor axis (see R) and the y-axis of the image, in degrees.
- PSFRATIO:
  PSFRATIO is defined as $\sqrt{semi\_major * semi\_minor}/psf\_radius$ where `psf_radius` is the radius at the source off-axis angle which corresponds to `eenergy`.
- BLOCK:
  For event list datasets larger than 2048×2048, *celldetect* may use the "recursive blocking scheme". The value of BLOCK provides the information on the blocking factor used at the position of the source.
- COMPONENT:
  The number assigned to the detected source. The source numbers correspond to those in the output source list. Its primary use is for filtering of the source list by CIAO tools (*e.g.* `"component=1:10"`, `"component=1,2,5"`).
- EXPO_RATIO:
  When an exposure map is input (`expstk`), *celldetect* computes the average exposure found in the detect cell and background frame. EXPO_RATIO is the ratio of these two average exposures.
- DOUBLE:
  When DOUBLE is TRUE, this detection is believed to be a duplicate of a source detected in a different recursive blocking pass.
- DOUBLE_ID:
  The component of the source for which this detection is believed to be a duplicate.

ASCII Source Region File (`regfile`)

In addition to the source list file (`outfile`), the user may also select to have a source region file (`regfile`) output.

This region file is useful for subsequent input into ds9, particularly to overlay region markers over each detection; see Chapter 1, Section 1.4 for details.

The source region file from the example in Chapter 4, Section 4.2.2:

```
unix% more example2_out.reg
ellipse(251.954128,268.889908,3.688666,2.832946,93.695023)
ellipse(252.031008,309.038760,3.638202,3.090788,103.071884)
ellipse(251.955556,320.983333,3.917530,3.097792,98.666176)
.
.
.
```

Log File (`logfile`)

If `log` is set to `yes`, then the log information will be written to the file `celldetect.log`.

The log file from the example in Chapter 4, Section 4.2.2:

```
unix% more celldetect.log
input file: data/datasetA.fits[123:630,109:614]
```

```
output file: example2_out.fits
ASCII source regions file: example2_out.reg
creating cell size table
reading data image
creating candidate source list
cell size 3
cell size 6
cell size 9
cell size 12
Writing out cell file: datasetA_cell_1.fits
```

Cell Size Image (`cellfile`)

If a filename is provided for the `cellfile` parameter, then one or more images are created which contain the detect cell size at each pixel location (see Example 4.2.2). Pixels in the cell size image have values equal to the cell size used to search for sources at those pixel locations.

Signal-to-noise Ratio Map (`snrfile`)

The `convolve` option produces a pixel-by-pixel map of the S/N, which is written to the file specified by `snrfile`. A S/N value is computed for every pixel in the input dataset, not just for source locations. The file `snrfile` is then an image of these values.

Chapter 8

Running vtpdetect

8.1 Key vtpdetect Parameters

This section highlights some frequently used *vtpdetect* parameters. Similar parameters and their description have been grouped together. Chapter 10 has information on each *vtpdetect* parameter in alphabetical order.

1. Input file name (`infile`)

   The input FITS file can an event list or an image. *CIAO* Data Manipulation (DM) syntax may be used in the input file specification, as explained in Chapter 1, Section 1.3.3.

   If the input file is an image, a pseudo event list is created; events are added with a multiplicity equal to the pixel amplitude. The FITS primary array may not be a floating point array; *vtpdetect* only knows how to convert an image that is an integer type: byte (BITPIX=8), short (BITPIX=16), or long (BITPIX=32).

2. Source list output file name (`outfile`)

   The `outfile` parameter is used to provide a name for the output file. The `kernel` parameter controls the output format; the default is for the output file to have the same format as the input file.

3. Threshold scale factor (`scale`)

   `scale` is actually a scale factor that refers to the intensity, not the spatial size. The algorithm figures out the threshold to distinguish between background and source events, based on the area of the Voronoi cell, then the `scale` parameter allows the user to scale that value. If `scale` is left to its default value of 1, the threshold remains as calculated by the code. Setting the `scale` value > 1 will tend to de-blend sources while losing faint sources. Setting it < 1 will tend to blend sources while picking out fainter sources. The recommended range for this parameter is between 0.8 and 3.

4. Maximum probability of a false source (`limit`)

   This parameter indicates the number of false detections per event. The user may wish to change the `limit`, depending on the size and population of the field. The default value is 10^{-6}; if there are significantly less than a million events, `limit` may be increased accordingly.

5. Name for ASCII output region files (`regfile`)
   Size of output source ellipses (`ellsigma`)

   In addition to the output source list (`outfile`), the user may choose to have an ASCII source region file written by specifying a filename in `regfile` parameter. If autonaming is used (*i.e.* `"regfile=."`), then the source regions output file will be named `"<infile_root>_reg"`.

   This region file is useful for subsequent input into ds9, particularly to overlay region markers over each detection; see Chapter 1, Section 1.4 for details.

   The size of the elliptical source regions in both of the source region files is controlled via the `ellsigma` parameter. This parameter is a multiplicative factor applied to sigma, the standard deviation of the distribution, to scale the major and minor axes of the ellipses for each source detection. This feature is included so that the graphics overlay may be made more visible; it is does not affect the detections themselves, only the display of the regions.

   By default the value for `ellsigma` is 3, but a larger value is often helpful.

6. Exposure map file name (`expfile`)

   vtpdetect works in fluxes, and so needs to know about significant exposure deviations in order to work properly. The provided exposure map must be a 2-D image with WCS that matches the input file (`infile`).

7. Minimum number of events per source (`coarse`)

   The minimum number of events required in order for a detection to be considered a real source. The default value is 10.

8. Maximum number of iterations to allow (`maxiter`)

   The maximum number of iterations to be allowed in estimating the background level (default is 10). *vtpdetect* may enter a state where the background estimate cycles between 2 or 3 values; setting the `maxiter` parameter to different values in this case will allow the user to control the exit level. Note that the tool may converge before reaching `maxiter`.

9. How close to the field edge to reject events (`edge`)

   The Voronoi tessellation at the edges is unbounded. This parameter controls how deep into the field to ignore events; the default is 2 tessellation cells.

10. Overwrite if file exists (`clobber`)
    Debug level (`verbose`)
    Debug file name (`logfile`)

    vtpdetect does not not overwrite existing outputs unless `clobber` is set to `yes`.

    The verbosity of log information ranges from 0 (none) to 5 (maximum output) and is sent to `STDERR` (usually the screen). However, if `log` is set to `yes`, then the log information will be written to the specified filename.

8.2 Examples Using Dataset A (Simulated ACIS Data)

8.2.1 Example 1: Running vtpdetect with the Default Parameters

This first example illustrates running *vtpdetect* using a simulated ACIS dataset as input; see Chapter 2, Section 2.1 for further information about this dataset.

Description of the Default Parameter Functionality

As required, the user specifies the following:

1. Input file (`infile = "data/datasetA_acis_img.fits"`) The input file is an image of simulated *Chandra* data named `data/datasetA_acis_img.fits`.
2. Source list output file name (`outfile = "example1_out.fits"`) The output file is to be named `example1_out.fits`.

None of the parameters are changed from their default values, so:

1. Threshold scale factor (`scale = 1`) The default threshold scale factor (1) is used.
2. Maximum probability of being a false source (`limit = 1e-06`)

   The maximum probability of a false source detection will be 1e-06.
3. Minimum number of events per source (`coarse = 10`)

   The minimum number of events per source detection is 10.
4. Maximum number of iterations to allow (`maxiter = 10`)

   The maximum number of iterations allowed is 10.
5. How close to edge of field to reject events (`edge = 2`)

   The `edge` is 2 by default.
6. Name for ASCII output region files (`regfile = none`),
   Size of output source ellipses (`ellsigma = 3`)

   No ASCII region file will be produced. The FITS source list will contain a 3-sigma ellipse for each detection.

Setting Required Parameters

The `pset` tool is used to supply the required parameter values (`infile` and `outfile`):

```
unix% punlearn vtpdetect
unix% pset vtpdetect infile = data/datasetA_acis_img.fits
unix% pset vtpdetect outfile = example1_out.fits
```

Notice that a path may be include with the filenames. Chapter 1, Section 1.3.1 has further information on manipulating parameters.

Listing Current Parameter Settings

If desired, the user may then list the current *vtpdetect* parameter settings with `plist`:

```
unix% plist vtpdetect

Parameters for /home/username/cxcds_param/vtpdetect.par

#
# parameters for vtpdetect
#
#
# inputs -- can either be an image or table
#
        infile = data/datasetA_acis_img.fits Input file name
       expfile = none             Exposure map file name
#
# output
#
       outfile = example1_out.fits Source list output file name
#
# processing parameters
#
         scale = 1                Threshold scale factor
         limit = 1e-06            Max. probability of being a false source
        coarse = 10               Minimum number of events per source
       maxiter = 10               Maximum number of iterations to allow
#
# SAOImage regions
#
      (regfile = none)            name for ASCII output region files
     (ellsigma = 3)               Size of output source ellipses (in sigmas)
         (edge = 2)               How close to edge of field to reject events
      (superdo = no)              Perform Super Voronoi Cell procedure
#
# probably use defaults for these...
#
   (maxbkgflux = 0.8)             Maximum normalized background flux to fit
   (mintotflux = 0.8)             Minimum total flux fit range
   (maxtotflux = 2.6)             Maximum total flux fit range
    (mincutoff = 1.2)             Minimum total flux cutoff value
    (maxcutoff = 3)               Maximum total flux cutoff value
       (fittol = 1e-06)           Tolerance on Possion fit
     (fitstart = 1.5)             Initial background fit starting scale factor
#
# user setable parameters
#
      (clobber = no)              Overwrite if file exists
      (verbose = 0)               Debug level
      (logfile = stderr)          Debug file name
```

```
       (kernel = default)         Output format
#
# mode
#
         (mode = ql)
```

Executing *vtpdetect*

To execute *vtpdetect*, type the name of the tool on the command line:

```
unix% vtpdetect
Input file name (data/datasetA_acis_img.fits):
Exposure map file name (none):
Source list output file name (example1_out.fits):
Threshold scale factor (0) (1):
Max. probability of being a false source (0:1) (1e-06):
Minimum number of events per source (0) (10):
Maximum number of iterations to allow (0:100) (10):
unix%
```

The parameter settings are shown in the prompt for each required parameter; these settings may be accepted by hitting the <`RETURN`> `key` or changed if desired.

Figure 8.1 shows the data with 3-sigma ellipses from the region file overlaid. Chapter 1, Section 1.4 has instructions on how to display the region file in ds9.

Examining Results

The *CIAO* tool `dmlist` is used to for examine the contents of the output file. To see what blocks are in the file:

```
unix% dmlist example1_out.fits opt=blocks
--------------------------------------------------------------------------------
Dataset: example1_out.fits
--------------------------------------------------------------------------------

     Block Name                          Type         Dimensions
--------------------------------------------------------------------------------
Block    1: PRIMARY                        Null
Block    2: SRCLIST                        Table        24 cols x 7        rows
```

In the `SRCLIST` block, there are 24 columns of information for each of the 7 detected sources (one row for each detected source). To list the names of these columns, along with brief column information:

```
unix% dmlist "example1_out.fits[SRCLIST]" opt=cols
```

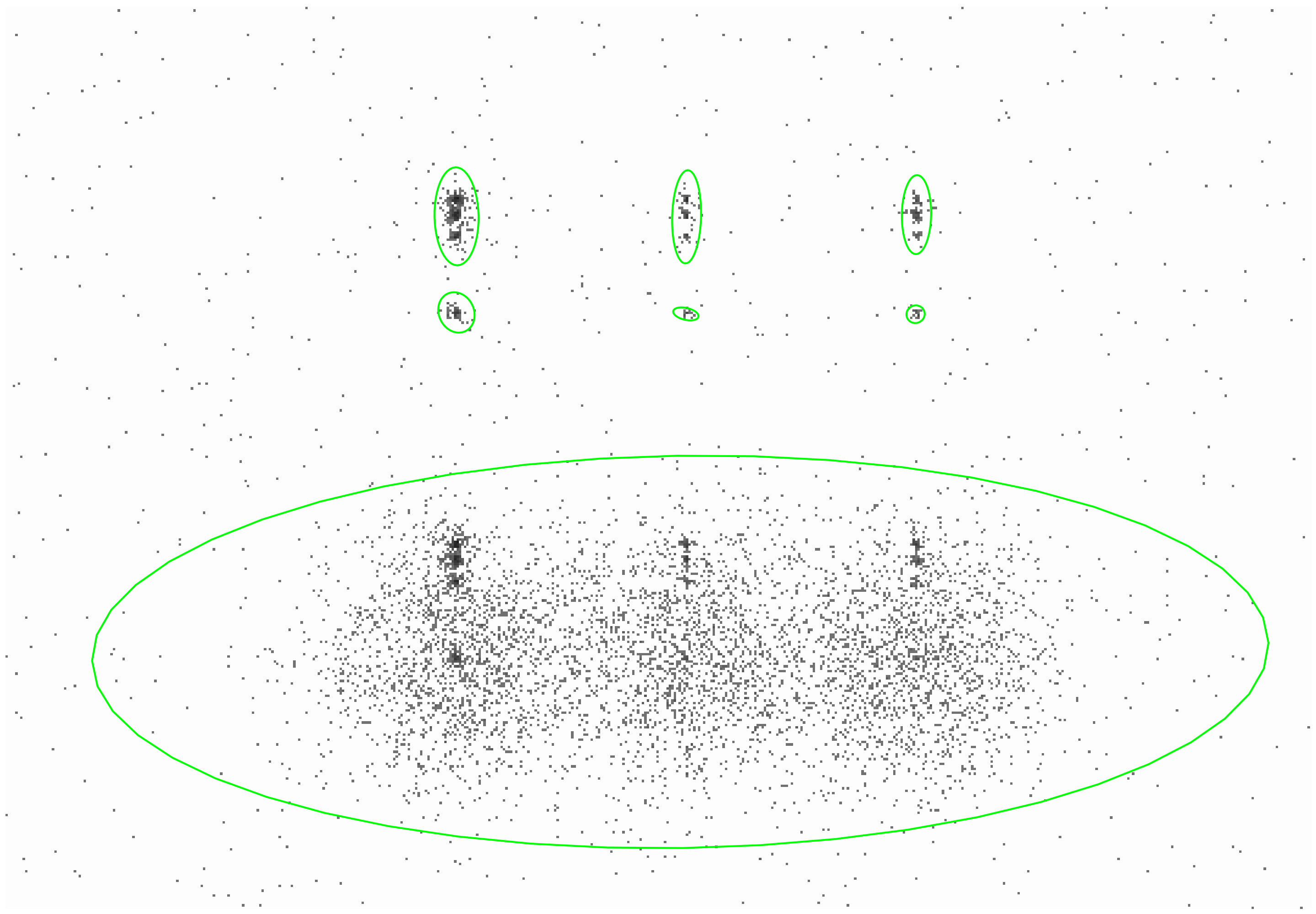

Figure 8.1: *vtpdetect* with default parameters. The ACIS-S simulated field with 3-sigma ellipses showing the *vtpdetect* results.

```
--------------------------------------------------------------------------------
Columns for Table Block SRCLIST
--------------------------------------------------------------------------------

ColNo  Name                 Unit        Type             Range
   1   RA                   deg          Real8          0:      360.0
Source Right Ascension
   2   RA_ERR               deg          Real8          -Inf:+Inf
Source Right Ascension error
   3   DEC                  deg          Real8          -90.0:       90.0
Source Declination
   4   DEC_ERR              deg          Real8          -Inf:+Inf
Source Declination error
   5   POS(X,Y)             pixel        Real8          -Inf:+Inf
physical coordinates
   6   X_ERR                pixel        Real8          -Inf:+Inf
Source X position error
   7   Y_ERR                pixel        Real8          -Inf:+Inf
Source Y position error
   8   SRC_AREA             pixel        Real4          -Inf:+Inf
Source region area
   9   NET_COUNTS           count        Real4          -Inf:+Inf
Net source counts
  10   NET_COUNTS_ERR       count        Real4          -Inf:+Inf
Error in net source counts
  11   BKG_COUNTS           count        Real4          -Inf:+Inf
Background counts (scaled to source cell)
  12   BKG_COUNTS_ERR       count        Real4          -Inf:+Inf
Error in BKG_COUNTS
  13   NET_RATE             count/s      Real4          -Inf:+Inf
Source count rate
  14   NET_RATE_ERR         count/s      Real4          -Inf:+Inf
Source count rate error
  15   BKG_RATE             count/s/pixel Real4          -Inf:+Inf
Background count rate
  16   BKG_RATE_ERR         count/s/pixel Real4          -Inf:+Inf
Background count rate error
  17   EXPTIME              s            Real4          -Inf:+Inf
Effective exposure time
  18   SRC_CUTOFF           count/s/pixel Real4          -Inf:+Inf
Source flux cutoff
  19   FSP                               Real4          0:        1.0
False source probability
  20   EDGE_OF_FIELD                     Int2           -
Edge-of-field flag
  21   SHAPE                             String[10]
Shape of source region
  22   R[2]                              Real4(2)       -Inf:+Inf
Radii of source region source
  23   ROTANG               deg          Real4          0:      180.0
```

```
Rotation angle of source region
  24   COMPONENT                         Int4          -
Source Number
```

To further examine the contents of the output file, use `dmlist` to show the position and net counts of each source detection:

```
unix% dmlist "example1_out.fits[SRCLIST][cols X,Y, NET_COUNTS]" opt=data

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

ROW    POS(X,Y)                                NET_COUNTS

     1 (      370.7111511230,      272.2136535645)      9568.4208984375
     2 (      252.0195770264,      452.2350463867)       182.9890594482
     3 (      373.6678771973,      451.4666748047)        32.6915626526
     4 (      495.4709167480,      451.2944335938)        30.5647010803
     5 (      252.0762634277,      503.2118835449)      1037.4575195312
     6 (      374.0616760254,      502.9508361816)       172.6163940430
     7 (      495.9555969238,      504.1110839844)       146.7237854004
```

The graphical user interface of the *CIAO* tool *Prism*is an alternative to the command-line interface of `dmlist`:

```
unix% prism example1_out.fits &
```

8.2.2 Example 2: Running vtpdetect to Create Additional Output

This next example illustrates running *vtpdetect* using as input the same *Chandra* dataset from the first example (see Section 4.2.1), but with a few parameters changed from their defaults. This will cause additional output files to be produced:

- `regfile`: an ASCII region file.
- `log` and `verbose`: increased verbosity is written to an output file.

In addition, the `ellsigma` parameter is increased to make the display of detected regions easier to see when displayed.

Setting Parameters

The input file is the same image used in the first example, and the output file is to be named `example2_out.fits`. We begin by setting all the parameters mentioned:

```
unix% punlearn vtpdetect
unix% pset vtpdetect infile = "data/datasetA_acis_img.fits"
unix% pset vtpdetect outfile = example2_out.fits
unix% pset vtpdetect regfile = example2_out.reg
unix% pset vtpdetect ellsigma = 5
unix% pset vtpdetect verbose = 3
unix% pset vtpdetect log = vtpdetect.log
```

A region file named `example2_out.reg` will be produced, and the size of the output source ellipses is increased to 5-sigma. The verbosity of log information is changed to 3 and will be recorded in `vtpdetect.log`.

Executing *vtpdetect*

```
unix% vtpdetect
Input file name (data/datasetA_acis_img.fits):
Exposure map file name (none):
Source list output file name (example2_out.fits):
Threshold scale factor (0) (1):
Max. probability of being a false source (0:1) (1e-06):
Minimum number of events per source (0) (10):
Maximum number of iterations to allow (0:100) (10):
unix%
```

The results are shown in Figure 8.2. Each detection has a 5-sigma ellipse overlay, since `ellsigma` was set to 5.

Examining Results

- As in the first example, this `dmlist` command shows what blocks are in the output source list file:

  ```
  unix% dmlist example2_out.fits opt=blocks

  --------------------------------------------------------------------------------
  Dataset: example2_out.fits
  --------------------------------------------------------------------------------

       Block Name                          Type         Dimensions
  --------------------------------------------------------------------------------
  Block    1: PRIMARY                        Null
  Block    2: SRCLIST                        Table        24 cols x 7        rows
  ```

  The `SRCLIST` block looks the same as Example 8.2.1: 24 columns of information for each of the 7 detected sources.

- Since the `verbose` parameter was increased and the `log` parameter was set, an output file name `vtpdetect.log` was created:

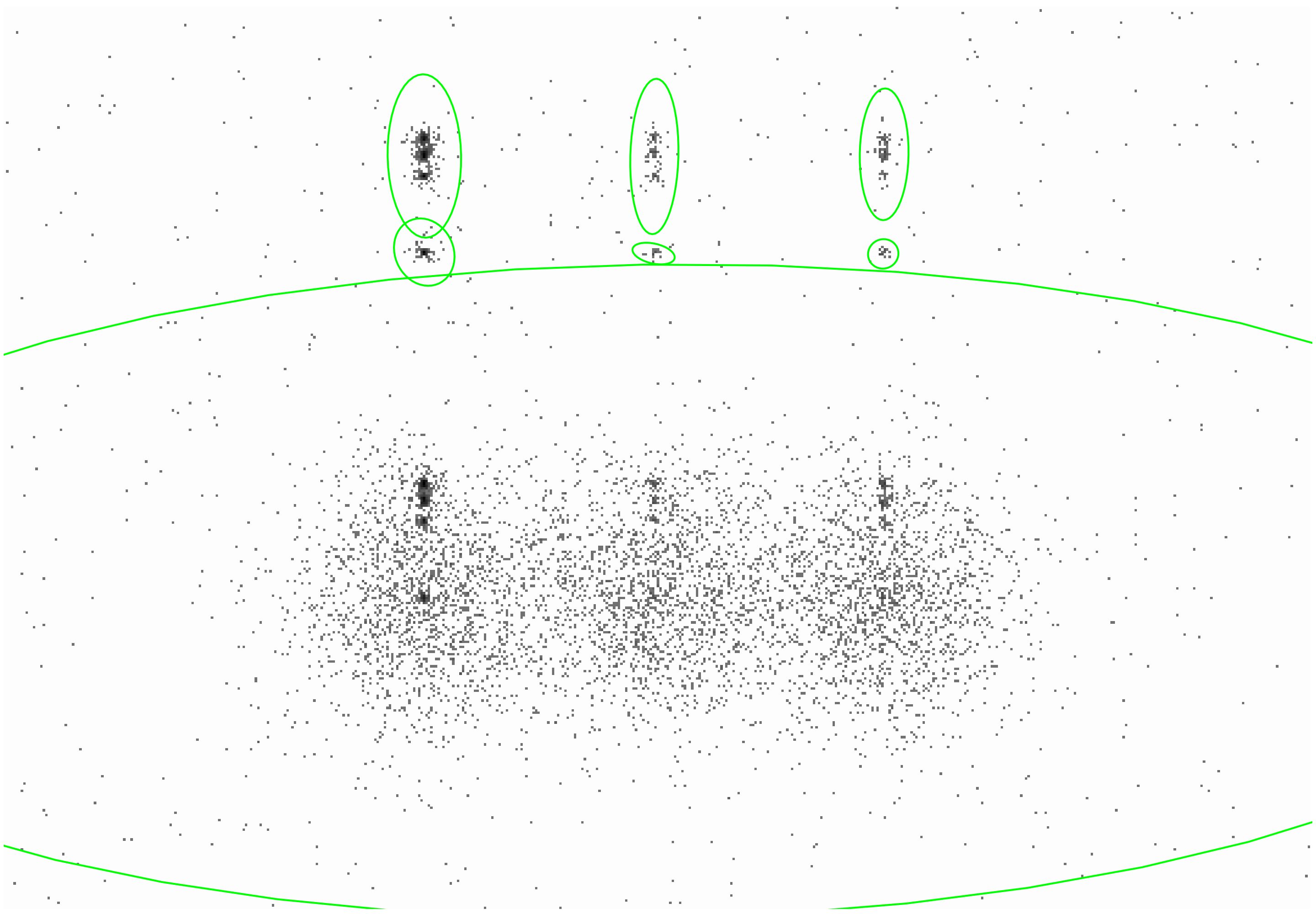

Figure 8.2: *vtpdetect* results with `ellsigma=5`. The ACIS-S simulated field with ellipses showing the *vtpdetect* results. The size of the ellipses has been increased from the default, which improves their visibility.

```
unix% more vtpdetect.log
DEBUG: Version information -
DEBUG:     vtpdetect: 0.5.1
DEBUG:     datamodel : 1.99
DEBUG:     ascfit    : 2.2
DEBUG:
DEBUG: Parameters -
DEBUG:      For input -
DEBUG:          infile    = data/datasetA_acis_img.fits
DEBUG:          expmap    = none
DEBUG:
DEBUG:      For output -
DEBUG:          outfile   = example2_out.fits
DEBUG:         (clobber   = no)
DEBUG:         (kernel    = default)
.
.
```

8.3 Examples Using Dataset B (3C 295 on ACIS-S)

8.3.1 Example 1: Detecting Extended Emission

This first example illustrates running *vtpdetect* using a *Chandra* ACIS dataset of 3C 295 as input; see Chapter 2, Section 2.2 for further information about this dataset.

All parameters are kept at defaults, except for:

- `ellsigma`: increased to make the display of detected regions easier to see.
- `regfile`: an ASCII region file.

Setting Parameters

Set the parameters:

```
unix% punlearn vtpdetect
unix% pset vtpdetect infile = "data/acisf00578N002_evt2.fits[EVENTS][ccd_id=7][cols x,y]"
unix% pset vtpdetect outfile = example1_out.fits
unix% pset vtpdetect regfile = example1_out.reg
unix% pset vtpdetect ellsigma = 5
```

DM syntax is used to specify only the chip of interest for input (*i.e.* `[ccd_id=7]`); see Chapter 1, Section 1.3.3 for further information about DM syntax usage with *celldetect*. Note that for a FITS event list, the filter `[cols x,y]` must also be specified for *vtpdetect*.

Executing *vtpdetect*

```
unix% vtpdetect
Input file name (data/acisf00578N002_evt2.fits[EVENTS][ccd_id=7][cols x,y]):
Exposure map file name (none):
Source list output file name (example1_out.fits):
Threshold scale factor (0) (1):
Max. probability of being a false source (0:1) (1e-06):
Minimum number of events per source (0) (10):
Maximum number of iterations to allow (0:100) (10):
unix%
```

The results are shown in Figure 8.3.

8.4 Examples Using Dataset C (3C 273 on HRC-I)

8.4.1 Example 1: Detecting Close Extended Sources

This example illustrates running *vtpdetect* using a *Chandra* HRC-I dataset of 3C 273 as input; see Chapter 2, Section 2.3 for further information about this dataset.

Setting Parameters

Set the usual parameters:

```
unix% punlearn vtpdetect
unix% pset vtpdetect infile = "data/hrcf00461N003_evt2_img.fits"
unix% pset vtpdetect outfile = example1_out.fits
unix% pset vtpdetect regfile = example1_out.reg
unix% pset vtpdetect ellsigma = 5
```

Executing *vtpdetect*

```
unix% vtpdetect
Input file name (data/hrcf00461N003_evt2_img.fits):
Exposure map file name (none):
Source list output file name (example1_out.fits):
Threshold scale factor (0) (1):
Max. probability of being a false source (0:1) (1e-06):
Minimum number of events per source (0) (10):
Maximum number of iterations to allow (0:100) (10):
unix%
```

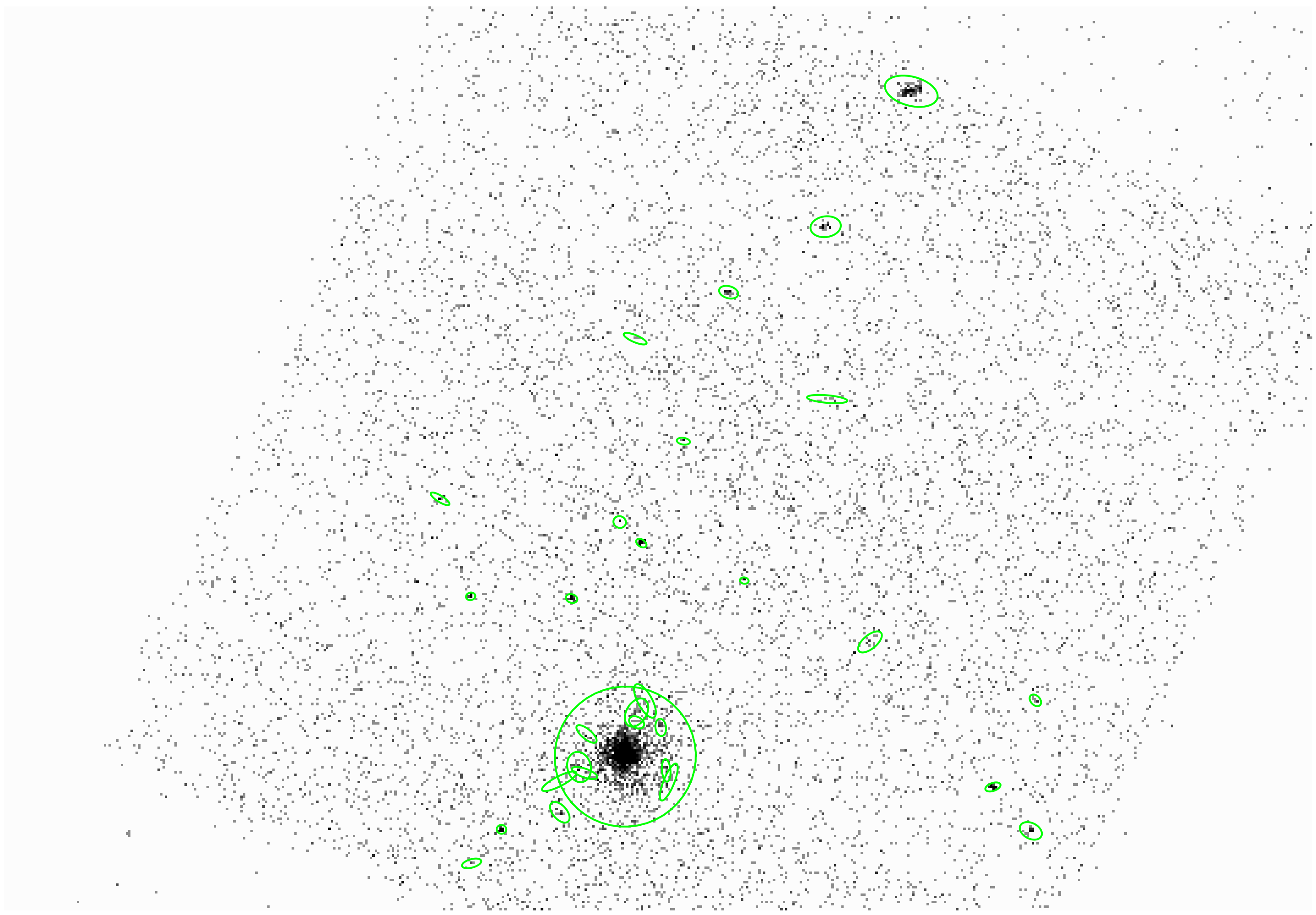

Figure 8.3: *vtpdetect* on extended emission. Chip S3 (`ccd_id=7`) of the ACIS-S dataset of 3C 295, displayed with binning = 2; the ellipses show the *celldetect* results. While it appears that some detections occur within regions of the other detections, it is just a display feature caused by choosing `ellsigma` greater than 1. Comparing these results with those obtained using *celldetect* (Figure 4.4) shows that *vtpdetect* successfully detects regions of extended emission which *celldetect* misses.

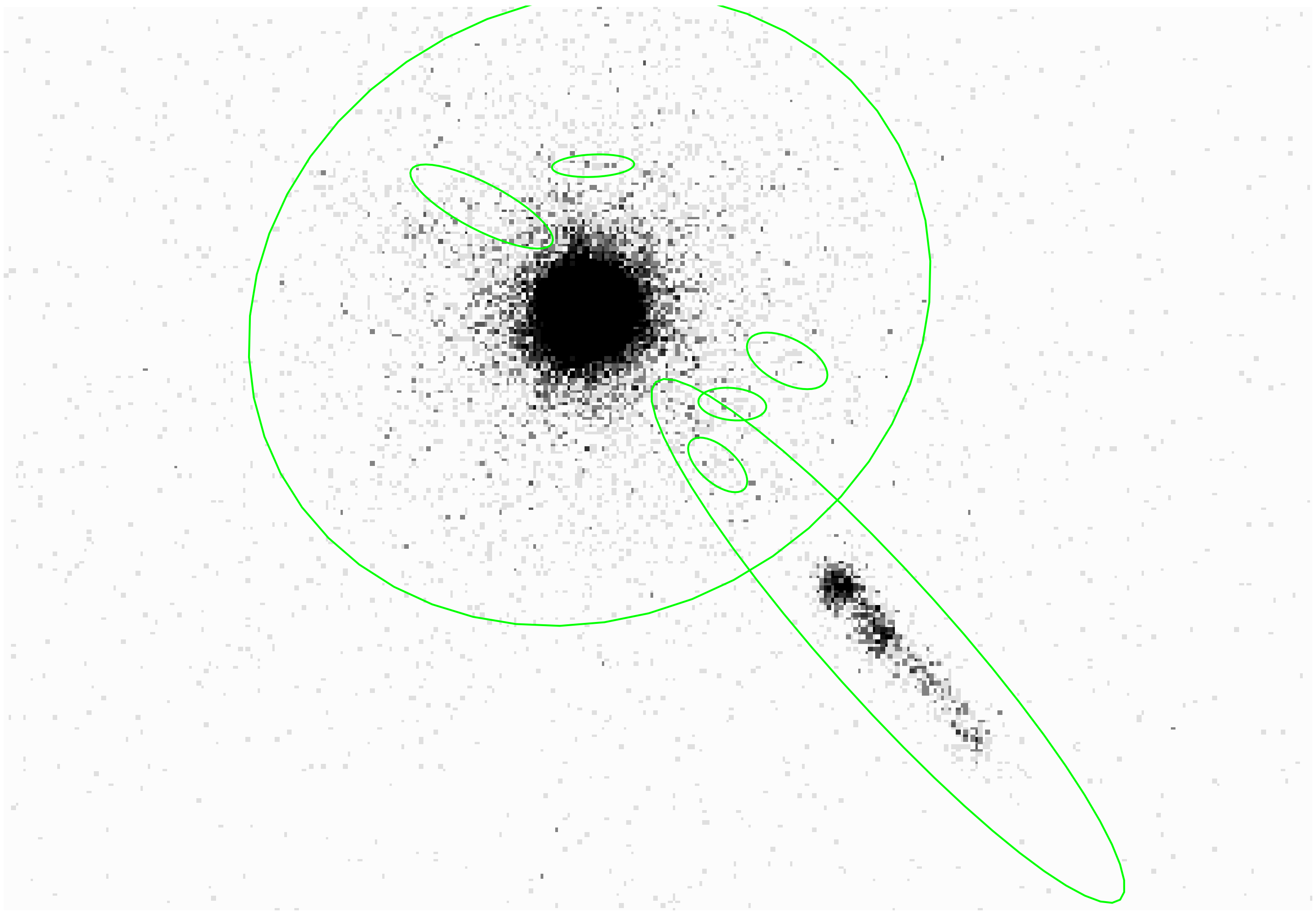

Figure 8.4: Detecting close extended sources with *vtpdetect*. The HRC-I dataset of 3C 273, with ellipses showing the *vtpdetect* results.

The results are shown in Figure 8.4.

Chapter 9

vtpdetect Theory

Reprinted with permission from Detecting Structure in Two Dimensions Combining Voronoi Tessellation and Percolation *by H. Ebeling & G. Wiedenmann, Physical Review E, 47, 704, 1993. Copyright 1993. The American Physical Society. The version herein has been edited.*

The figures from the original paper no longer exist electronically; the ones given in this chapter are reproductions.

9.1 Abstract

Conventional source-detection algorithms in high-energy astrophysics and other fields mostly use spherical or quadratic sliding windows of varying size on two-dimensionally binned representations of spatial event distributions in order to detect statistically significant event enhancements (sources) within a given field. While this is a reasonably reliable technique for nearly pointlike sources with good statistics, poor and extended sources are likely to be incorrectly assessed or even missed, as the calculations are governed by non-physical parameters like the bin size and the window geometry rather than by the actual data. The approach presented here does not introduce any artificial bias but makes full use of the unbinned two dimensional event distribution. A Voronoi tessellation on a finite plane surface yields individual densities, or fluxes, for every occupied pixel. The distribution of the densities allows us to determine the contribution from a random Poissonian background field (noise). The application of a non parametric percolation to the tessellation cells exceeding this noise level leads directly to a source list which is free of any assumptions about the source geometry. High-density fluctuations from the random background field will still be included in this tentative source list but can be eliminated in most cases by setting a lower threshold to the required number of evens per source. Since no finite-size detection windows or the like have been used, this analysis automatically yields straightforward fluxes for every source finally accepted. The main disadvantage of this approach is the considerable CPU time required for the construction of the Voronoi tessellation—it is thus applicable only to either small fields or low-event density regions.

9.2 Introduction

Many physical applications involve procedures which, in a way, can be called source-detection algorithms. Although we will use the most obvious case of a *two-dimensional spatial event distribution* to demonstrate the advantages of the technique described in the following, it is, in principle, applicable to any problem where significant density enhancements are sought in a two-dimensional parameter space (not necessarily spatial).

In high-energy astrophysics spatially resolved observations of astronomical objects often yield a few or even single photon counts per detection cell to begin with. The raw data provide the highest attainable spatial resolution which is limited by the detector hardware only; count statistics, on the contrary, can be extremely poor, especially for weak extended sources.

These poor statistics are the major handicap of conventional source detection algorithms which make use of a locally determined background in order to flag possibly significant density enhancements in the photon distribution. It is this concept of a local search which often makes the introduction of a coarser data binning inevitable in order to improve the count statistics locally at the expense of spatial accuracy. So-called sliding windows (mostly rectangular), of varying size are then moved across the binned distribution marking the positions where the count rate in the central part of the window exceeds the value expected from the background determined in the outermost regions of the window by a certain predetermined factor.

Using this technique it is clear that the decision whether a source is regarded as significant or not will be affected by several purely artificial parameters namely

- bin sizes and positions and
- window sizes and geometries.

This well-known flaw is overcome only partially by an additional commonly used detection algorithm which utilizes the results of the local detect procedure only in as much as it clips the detected sources and then computes a global background map from the remainders. A maximum likelihood (ML) routine is then used to find sources in the background subtracted distribution. However, as the ML algorithm has to assume a model profile (most applications use Gaussian distributions) to fit to the data the dependency of the decision process on an artificially fixed geometry persists.

The method we will describe in the following shows none of these shortcomings as, firstly, it does not sort the photons into artificial bins but rather works on the raw data *globally*, thus being limited only by the detector's resolution, and, secondly, it does not assume any particular source geometry for the detection process. Because of the great deal of CPU time required it should be used only on rather small datasets when searching for sources which are characterized by a low density of occupied pixels. As a sample event distribution, photon counts as detected by the xray telescope aboard the *Rosat* satellite in a $1^\circ \times 1^\circ$ field of the sky will be used.

9.3 Description of the Algorithm by Data Simulation

The actual procedure can be briefly summarized as follows.

(1) The Voronoi tessellation for the original raw photon distribution is computed.

(2) The cumulative distribution of the inverse areas of the resulting Voronoi cells is compared with that expected for a random Poisson distribution. A cutoff value for the photon density parameterizing the global background is determined.

(3) A spatial percolation algorithm is run on the individual cells, grouping cells (i.e., photons) exceeding the background density into sources.

(4) The minimal number of photons required for a true source is computed in order to discriminate against background fluctuations.

9.3.1 Voronoi Tessellation

For a given two-dimensional distribution of points (often also called atoms; in our application: photons) the Voronoi tessellation [1] is a uniquely defined set of convex cells, each of which encloses one and only one of these points. Depending on the implied boundary conditions the entire set of cells covers either the whole plane or just the area enclosed by the polygon defined by the outermost points of the distribution. In any case there are neither gaps nor overlaps between adjacent Voronoi cells. The algorithm used here for the construction of this tessellation is a two-dimensional adaption of the recipe described by Tanemura, Ogawa, and Ogita [2] for three dimensions.

We use open boundaries on a plane, finite-area surface, where the constructibility of the out most cells is guaranteed by restricting the tessellation to the central region of the area actually covered by the photon distribution. For the application presented in the following we found a confinement to the central 81% of the field area to be a good choice. Figure 9.1 illustrates the resulting cell distribution for the tessellation of a sample set of 2000 randomly positioned points.

9.3.2 Inverse Area Distribution

Let us assume for a moment that this random distribution was an actual measurement of some background radiation field: a physically interesting quantity would then not be the area assigned to each photon by the tessellation but rather the distribution of fluxes, i.e., detector counts per time interval and unit area, in order to establish a background level to compare to the fluxes of potential sources.

We are not aware of an analytical derivation of the cell area distribution function for randomly positioned points following Poissonian statistics; numerical simulations [3], however, suggest that the empirical results converge towards the differential probability distribution

$$dp(\tilde{a}) = \frac{4^4}{\Gamma(4)} \tilde{a}^3 e^{-4\tilde{a}} d\tilde{a},$$

where $\tilde{a} = a/ <a>_a$ is the cell area in units of the average cell area $<a>_a = \frac{1}{N}\sum_{i=1}^{N} a_i$. The corresponding cumulative distribution is then given by

$$P(\tilde{a}) = \int_0^{\tilde{a}} dp = 1 - e^{-4\tilde{a}} \left(\frac{32\tilde{a}^3}{3} + 8\tilde{a}^2 + 4\tilde{a} + 1 \right). \tag{9.1}$$

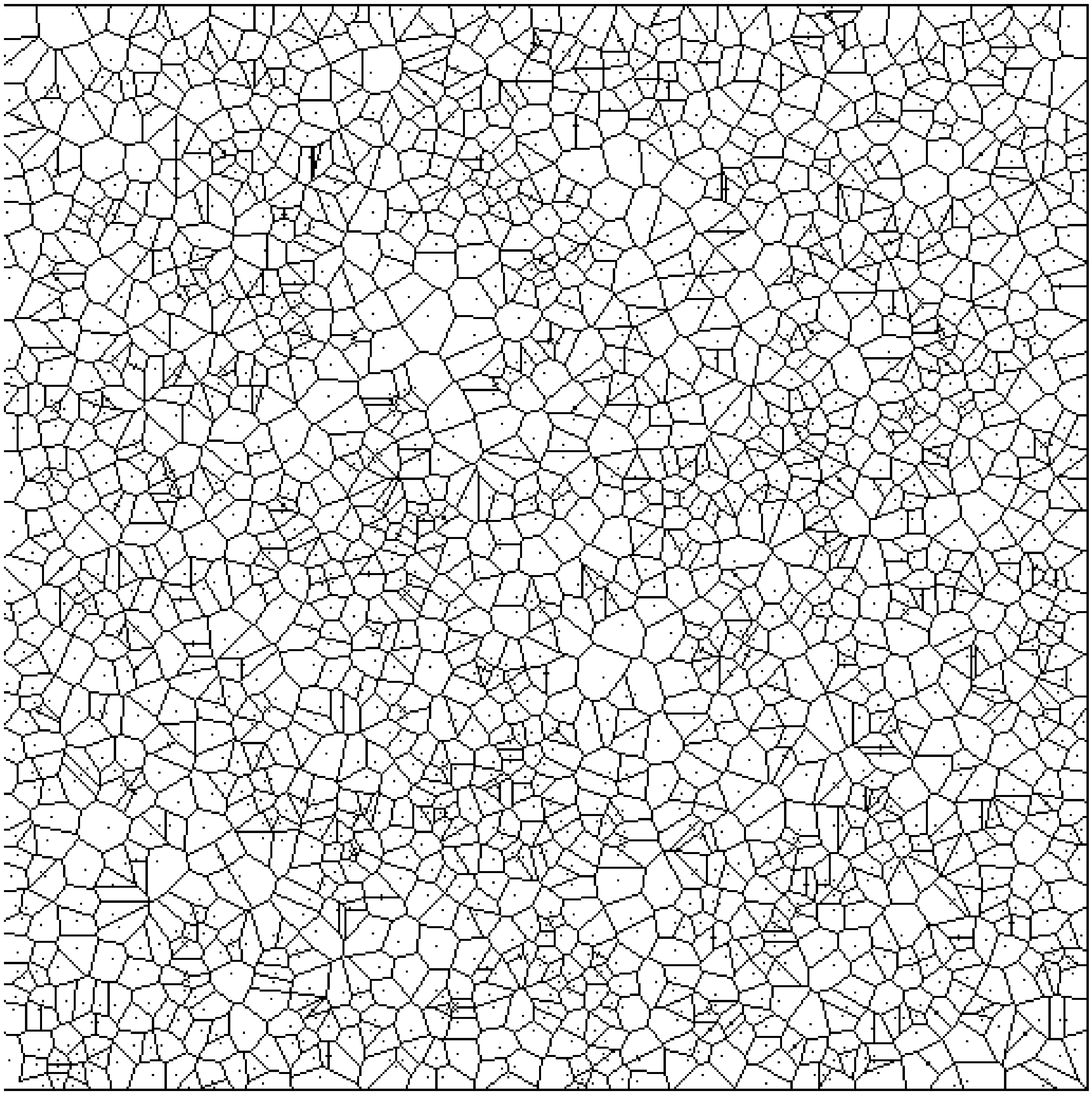

Figure 9.1: Voronoi tessellation of a sample set of 2000 randomly positioned photons in a $1^\circ \times 1^\circ$ field.

As each cell contains exactly one photon, the flux for this particular photon equals the inverse of the product of the cell area and the exposure time. Assuming a uniform exposure of unity for the time being, Eq. (9.1) yields

$$P(\tilde{f}) = e^{-4/\tilde{f}} \left(\frac{32}{3\tilde{f}^3} + \frac{8}{\tilde{f}^2} + \frac{4}{\tilde{f}} + 1 \right), \tag{9.2}$$

where $\tilde{f} = f/<f>$ is the inverse cell area $1/a$ in units of the inverse average cell area $<f>= N/\sum_{i=1}^{N} 1/f_i = 1/<a>_a$. This function is indeed in excellent agreement with the cell-area distribution of the sample shown in Fig. 9.1, as can be seen in Fig. 9.2.

Note that although the purely random photon distribution appears to reveal considerable structure in Fig. 9.1, which is what a *local* detection algorithm would look for, the cumulative flux distribution representing a *global* property of the data looks extremely smooth as statistical fluctuations tend to cancel out on a larger scale.

However, if sources are located within the field of view, a significant deviation from the Poissonian curve is observed. In simulated distributions, three sources of different extent were put on top of 2000 background photons like the ones shown in Fig. 9.1. Each of the three artificial sources consists of 200 photons in a spatial Gaussian distribution. The width of the Gaussian varies, however, from $\sigma = 1'$ (Fig. 9.4) through $\sigma = 2'$ to $\sigma = 4'$ (Fig. 9.5); no figure is given for the $\sigma = 2'$ case. Whereas the first two sources are rather compact despite their extent, the third one is extremely extended and poses a true challenge to any source-detection algorithm. The resulting flux distributions, however, are in all three cases clearly inconsistent with the random background curve which was fitted to the data in the background-dominated low-flux range where $\tilde{f} \leq 0.8$. Extensive simulations showed this range to yield the highest accuracy and reliability in the determination of the fit parameter, the total number of background photons. For all three examples this number, as determined by the fit, is within 2% of the true value.

From the residuals presented in Fig. 9.6 and Fig. 9.7, an upper flux limit for background contributions can be determined in order to allow the separation of high-flux from low-flux regions in the subsequent percolation. This cutoff is established at the flux value where the background-corrected cumulative distribution $\Delta P(\tilde{f})$ [or $N(<\tilde{f})$] reaches its minimal value, i.e., where the measured flux distribution starts to rise faster than would be expected for a purely random photon distribution. Note that for all three examples this cutoff excludes more than two-thirds of the photons from the percolation process.

Apart from the Voronoi cells in high-flux regions around the sources we are actually looking for, fluctuations in the random background will of course also be found in the upper end of the flux distribution. Most of these fluctuations consist, however, only of a couple of adjacent cells and can therefore be eliminated by means of a percolation algorithm accepting exclusive sources containing more than a certain minimum number of photons.

9.3.3 Percolation

What is usually meant by "percolation" (often called "friends-of-friends") algorithms is the search for close groups of objects in a two- or three-dimensional distribution using a maximally allowed separation $d_{\max}$ between each two of these objects as the only parameter to be set beforehand. Starting at any point, all neighboring objects closer to the starting point than the specified maximal distance are taken to be members of the agglomeration and become starting points in the next iteration themselves. Algorithms of this sort are often used as source-detection algorithms in their own right, the main difficulty being the choice of a suitable value for $d_{\max}$ which determines the scale at which sources are found.

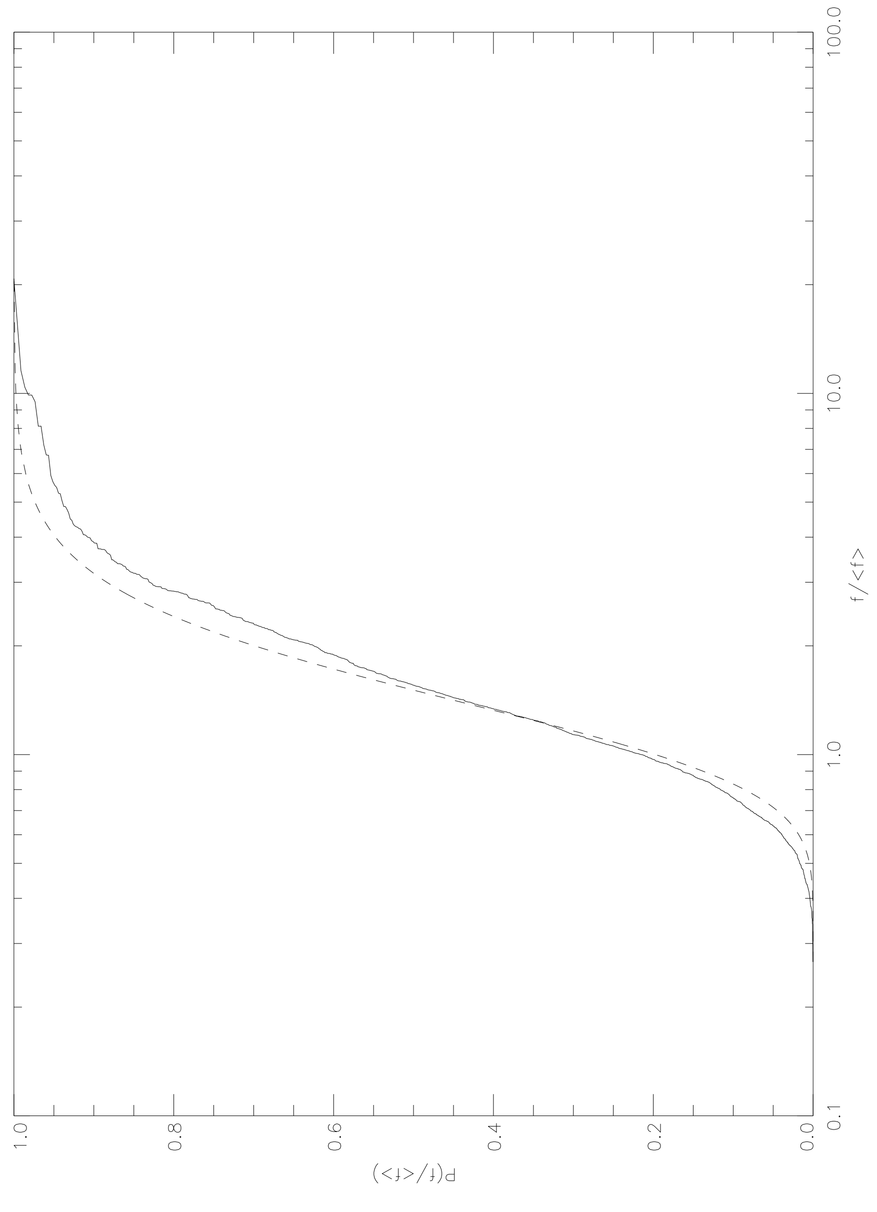

Figure 9.2: Normalized flux distribution for the randomly positioned photons of Fig. 9.1 assuming uniform exposure of the field (solid line) and theoretical model function according to Eq. (9.2) (dashed line).

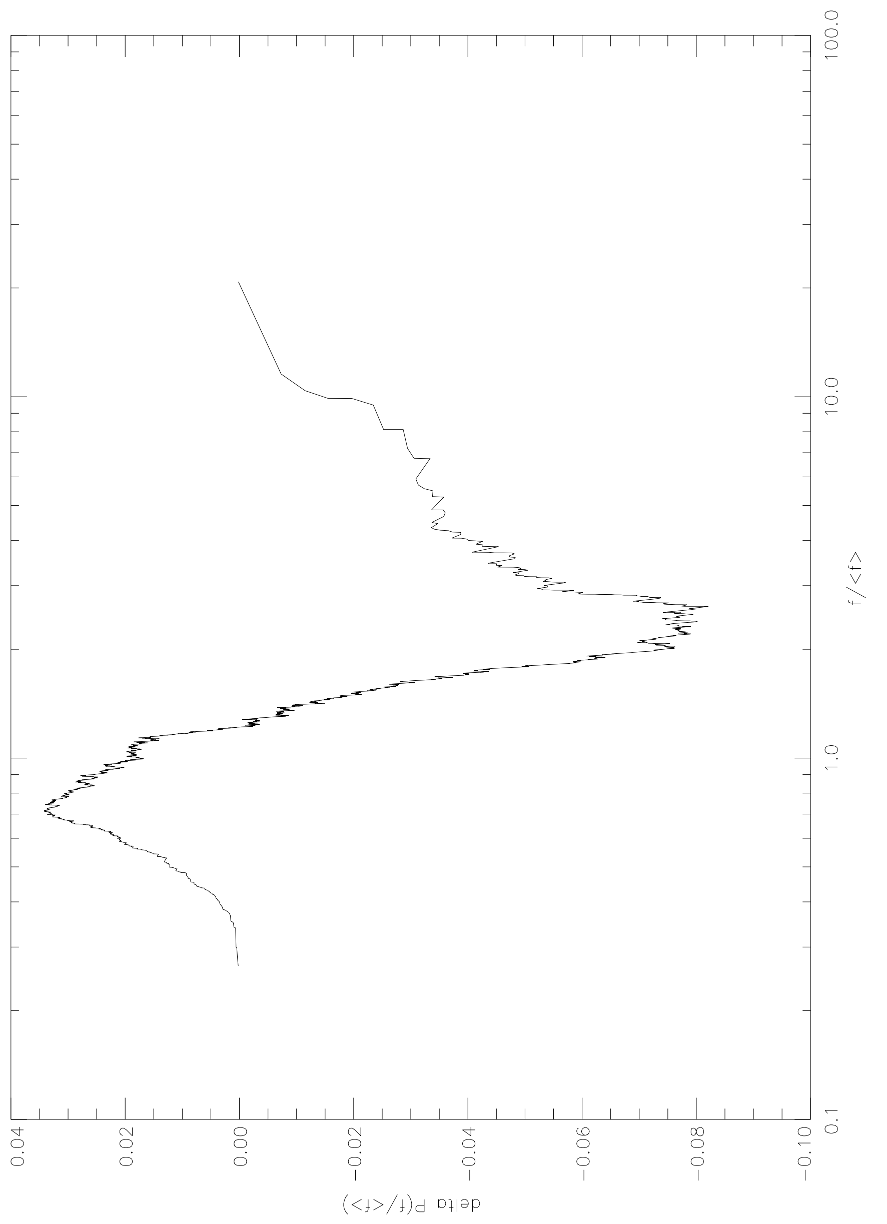

Figure 9.3: The residuals (data minus model) for Fig. 9.2.

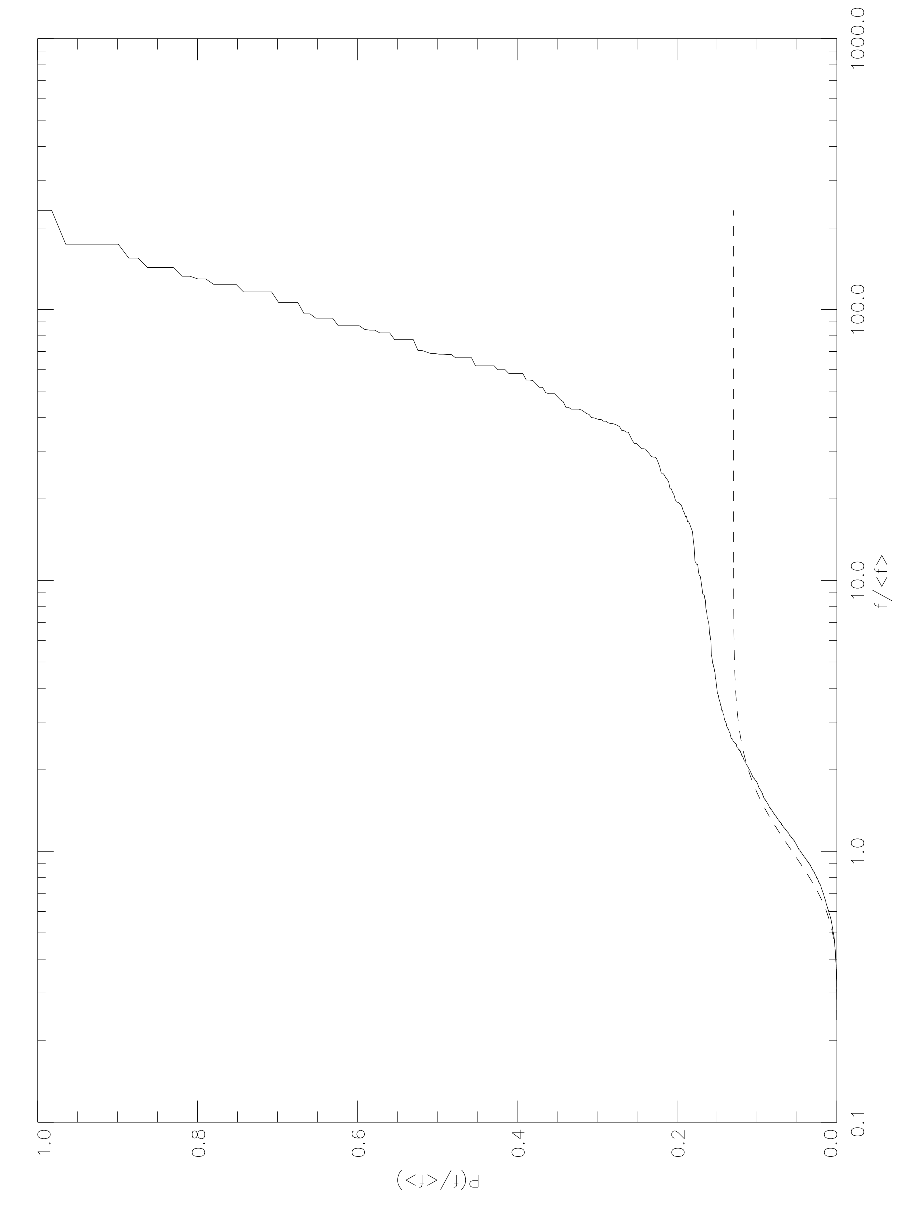

Figure 9.4: Simulated photon distributions for a field containing an extended source on Poissonian background and corresponding normalized flux distributions (solid line: data; dashed line: background fit). The 1σ width of the sources is $1'$.

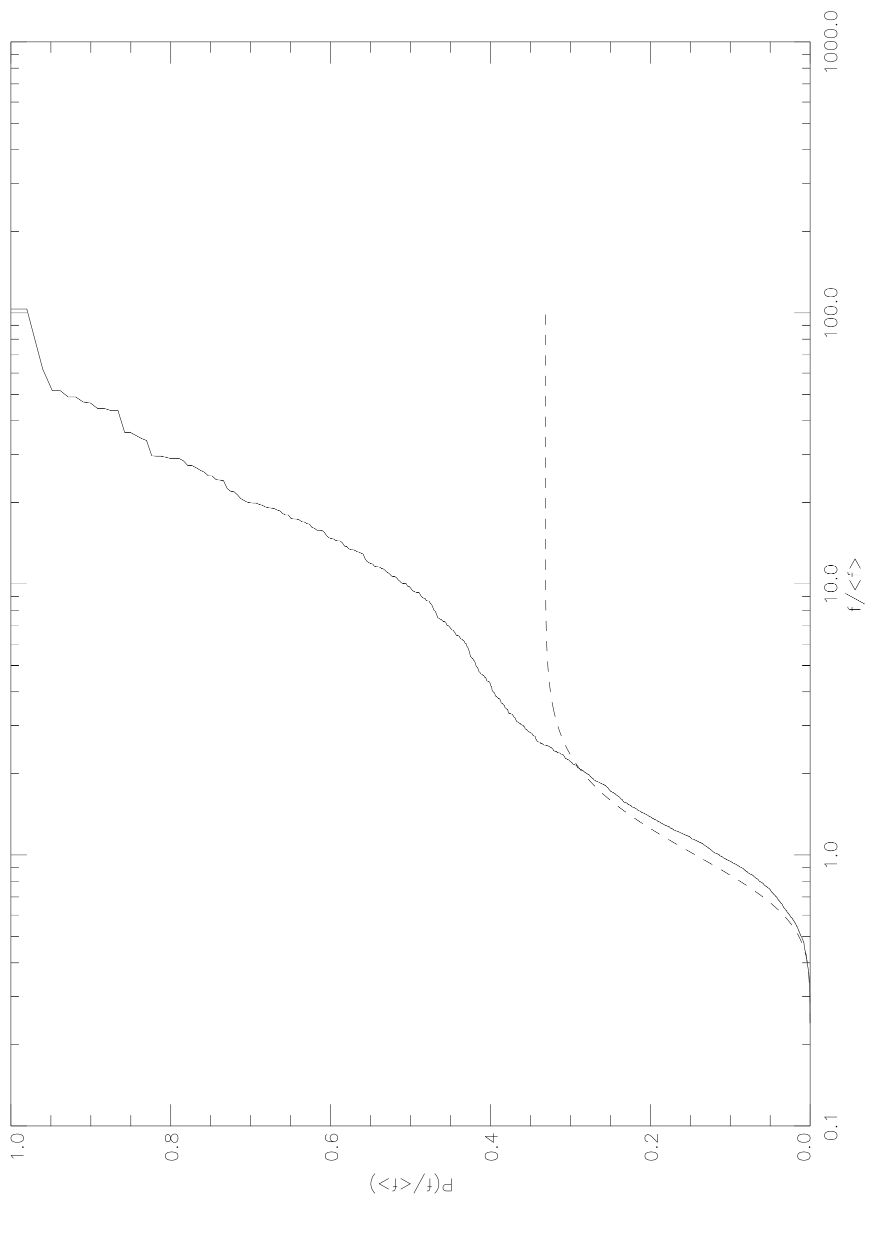

Figure 9.5: Simulated photon distributions for a field containing an extended source on Poissonian background and corresponding normalized flux distributions (solid line: data; dashed line: background fit). The 1σ width of the sources is $4'$.

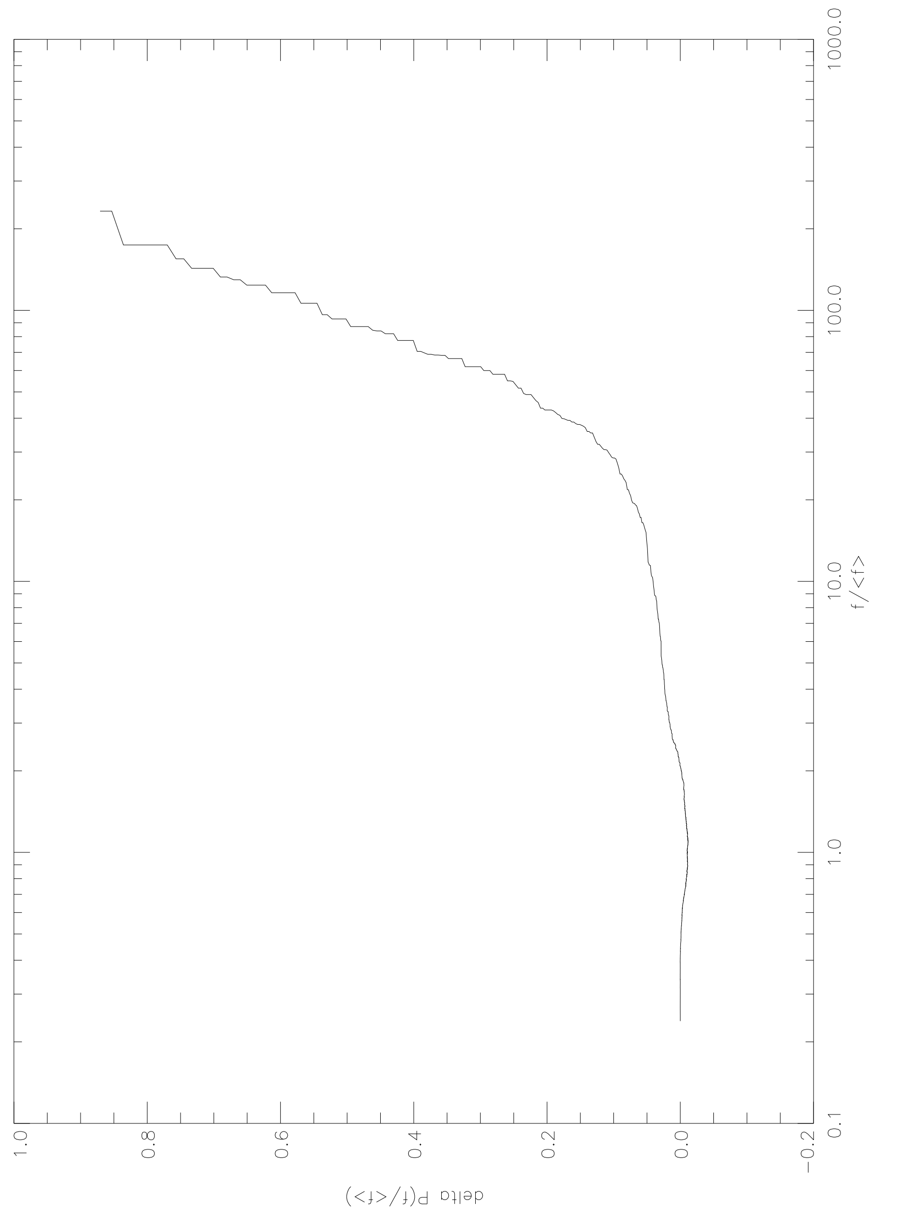

Figure 9.6: The residuals associated with Fig. 9.4.

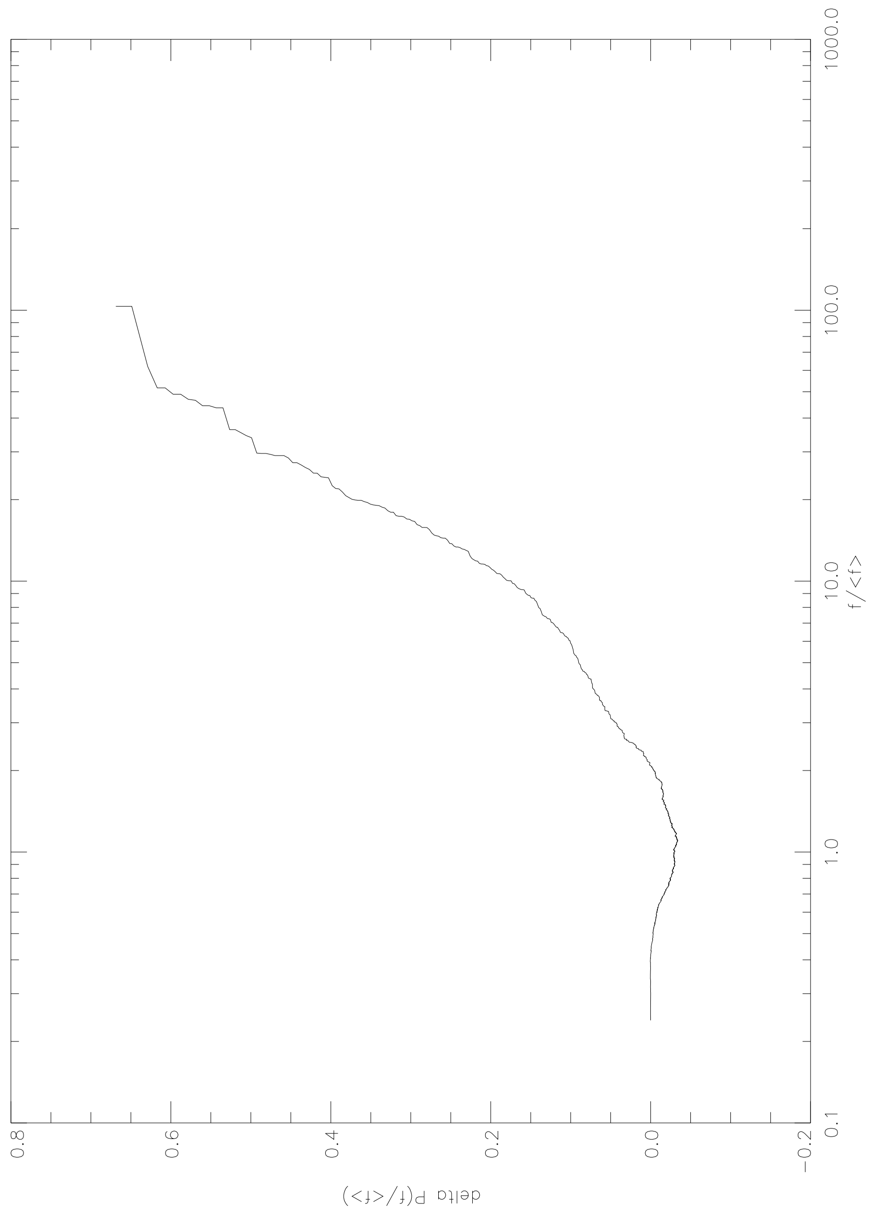

Figure 9.7: The residuals associated with Fig. 9.5.

Fortunately, the percolation algorithm used here does not need any distance parameter; $d_{\rm max}$ is, in a way, replaced by the flux cutoff determined from the comparison between the flux distributions for the data and a random background field. As this cutoff value corresponds to an area (namely that of the Voronoi cells) rather than a linear distance, it makes the percolation far more flexible and less prone to erroneously taking fluctuations for true sources

For the simulation example discussed in the previous sections, the percolation found, in fact, more sources per field (consisting of more than ten photons each) than just the one we put in. As the background radiation has entered only implicitly so far, namely in the flux threshold for the percolation, we still have to correct the source counts (i.e., the number of photons assigned to each source) for background photons. Knowing the total area covered by each source from the Voronoi tessellations, a background correction can be applied by simply subtracting the number of background photons statistically expected in the same area.

In all three cases the detections of both the simulated sources are accompanied by spurious detections in the field of background photons consisting of up to 12.3 photons (number background corrected). In general one more step is thus required in order to discriminate against random fluctuations, namely the establishment of a value for the minimal number of photons required for a real source.

9.3.4 Suppression of Fake Sources

The normalized number of sources caused by random fluctuations in the background field can be written as

$$n_{\rm src,fluct}(\tilde{f}_{\rm min}, n_{\rm ph}) = n_{\rm src,fluct}(\tilde{f}_{\rm min}, 0)\exp[-b(\tilde{f}_{\rm min})n_{\rm ph}], \tag{9.3}$$

where $\tilde{f}_{\rm min}$ is the flux cutoff value used to separate high- from low-flux events for the percolation and $n_{\rm ph}$ is the number of photons *above the background level* in the fluctuation source. $n_{\rm src,fluct}$ in Eq.(9.3) is normalized to the total number of background photons in the tessellation area, $N_{\rm bck}$ so that the actual number of random sources of size $n_{\rm ph}$ in a given field is

$$N_{\rm src,fluct}(\tilde{f}_{\rm min}, n_{\rm ph}) = N_{\rm bck} n_{\rm src,fluct}(\tilde{f}_{\rm min}, n_{\rm ph}).$$

In the range $1.2 < \tilde{f}_{\rm min} < 2.2$, which is where the flux cutoff value is found to lie for almost any event distribution, both $n_{\rm src,fluct}(\tilde{f}_{\rm min}, 0)$ and $b(\tilde{f}_{\rm min})$ are reasonably well described by linear functions of $\tilde{f}_{\rm min}$. Least squares fits yield

$$\begin{aligned} n_{\rm src,fluct}(\tilde{f}_{\rm min}, 0) &= 0.047\tilde{f}_{\rm min} - 0.04, \\ b(\tilde{f}_{\rm min}) &= 0.62\tilde{f}_{\rm min} - 0.45. \end{aligned}$$

Obviously, the statistically expected number of fluctuation sources containing at least $n_{\rm ph}$ photons above the mean background value is then given by

$$N_{\rm src,fluct}(\tilde{f}_{\rm min}, \geq n_{\rm ph}) = N_{\rm bck}\frac{n_{\rm src,fluct}(\tilde{f}_{\rm min}, 0)}{b(\tilde{f}_{\rm min})}\exp[-b(\tilde{f}_{\rm min})n_{\rm ph}]. \tag{9.4}$$

Requiring that no random source be detected in our fields at the 90% confidence level we find values for $n_{\rm ph}$ of 9.1, 13.2, and 13.2 photons, respectively, for our three simulated sources from Eq. (9.4), thus eliminating all of the additional sources and leaving only the central “true” source.

9.3.5 Comparison of Input and Output Source Characteristics

Finally, it is interesting to take a closer look at the sources' characteristics as determined by the algorithm. The algorithm is not biased towards preferential detection of spherical sources or sources of any other fixed geometry. This is the main advantage in comparison with conventional window algorithms.

The accuracy of the source positions, which are computed as the flux-weighted mean values of the individual photon coordinates, depends both on extent and brightness of the object. For our example the deviation of the detection position from the coordinates specified in the simulations amounts to $6.1''$, $18.1''$, and $39.9''$ (in order of ascending extent) which is comparable to the accuracy attained with conventional algorithms on binned data.

As for the total source flux, we find background-corrected values of 196.8, 197.8, and again 196.8 photons, which is to be compared to the actual number of 200 photons put into each source in the simulation. In all cases the source's flux was thus found to be within 2% of the true value.

9.4 Application to rosat X-ray Data

Leaving the simulated and entering the real world we applied the algorithm to a *Rosat* x-ray image which was taken during the satellite's six months all-sky survey. The $1^\circ \times 1^\circ$ field looks almost blank and in fact no source was found in the field by conventional source-detection techniques.

The reason why we chose this particular field is the fact that it is centered on the optical position of a nearby cluster of galaxies. The hot gas trapped in the potential well of a cluster being the source of extended x-ray emission one would expect to see an enhancement in the photon flux on a scale of about half a degree corresponding to the angular size of the cluster core

The algorithm run on these data is different from the one described above only in as much as the exposure time which is slightly varying over the field is taken into account by weighting the Voronoi cell areas with the local exposures, thus converting inverse areas into true fluxes.

The expected emission is indeed detectable with the presented algorithm. Three sources are found: the upper one contains 34.5 photons more than would be expected from the background field in the same area and is thus clearly a real detection. (The probability for finding a fluctuation exceeding the background by more than 19.8 photons is less than 10% for this field.) The central extremely extended source is actually split into two, the smaller of which forms the northern bulge and consists of 21.9 photons. All the rest of the central emission is interconnected to build one large source with a total of 179.6 "true" photons and coincides perfectly with the position of the cluster core as it is determined from the galaxy distribution.

9.5 Conclusions

The source-detection technique described above is widely applicable to all sorts of two-dimensional event distributions. Although we have restricted ourselves to the specific problem of detecting sources in a homogeneous background field, the method can be used quite generally to find any kind of structure embedded in a Poissonian background field. The advantages of our alternative approach which bears importance for a great variety of applications related to pattern recognition and image processing are the following:

- It does not introduce any binning of the data nor does it assume anything about the shape of the structure one is looking for.
- It allows a global and quantitative assessment of a given distribution as "background-like" or "non background-like".
- Contrary to image-processing techniques like, e.g., maximum entropy it yields a measure for the significance of the found structure. Different structures can be compared and classified according to their detection probability.

So far we have only considered a background following Poissonian statistics. As our method only uses the deviation of the cumulative probability distribution from the expected noise distribution it should, however, also be possible to discern structure from a correlated background provided that the probability distribution of the Voronoi cells for this background is known.

The algorithm's main advantage is its ability to detect sources of almost arbitrary extent, but care should be taken to provide a large enough field in order to allow the background fit to be as accurate as possible. If fields of very high source densities are studied, problems may arise by the tendency of the percolation algorithm to leave filamentary bridges between nearby adjacent sources, leading to errors in the computation of both the position and the flux of the source. In cases where potential sources are known to be rather compact and bright, one should, for CPU time's sake, rely on conventional detection algorithms unless besides the sources detection the determination of its flux is an equally important issue.

9.6 Acknowledgments

The authors would like to thank Valentin Demmel who provided much of the software this algorithm was built upon; Hans Böhringer and Gregor Morfill are thanked for stimulating discussions. Last, but not least, we are grateful to the *Rosat* SASS team at MPE for supplying the x-ray data used to illustrate a typical application.

9.7 References

[1] Voronoi, G. and Angew, J. Reine. 1908, Math. 134, 198

[2] Tanemura, M., Ogawa, T., and Ogita, N. 1983, J. Comput. Phys. 51, 191

[3] Kiang, T. 1966, Z. Astrophys. 64, 433

Additional References:

Ebeling et al. 1996, MNRAS, 281, 799

Scharf et al. 1997, ApJ, 477, 79

Jones et al. 1998, ApJ, 495, 100

Chapter 10

vtpdetect Parameters & Data Products Reference

This chapter lists all the *vtpdetect* parameters in alphabetical order for quick reference. In addition, a guide to the data products output by *vtpdetect* is provided.

10.1 Default Parameter File

```
unix% plist vtpdetect

Parameters for /soft/ciao/param/vtpdetect.par

#
# parameters for vtpdetect
#
#
# inputs -- can either be an image or table
#
        infile =                  Input file name
       expfile = none             Exposure map file name
#
# output
#
       outfile =                  Source list output file name
#
# processing parameters
#
         scale = 1                Threshold scale factor
         limit = 1e-06            Max. probability of being a false source
        coarse = 10               Minimum number of events per source
       maxiter = 10               Maximum number of iterations to allow
```

```
#
# SAOImage regions
#
      (regfile = none)            name for ASCII output region files
     (ellsigma = 3)               Size of output source ellipses (in sigmas)
         (edge = 2)               How close to edge of field to reject events
      (superdo = no)              Perform Super Voronoi Cell procedure
#
# probably use defaults for these...
#
   (maxbkgflux = 0.8)             Maximum normalized background flux to fit
   (mintotflux = 0.8)             Minimum total flux fit range
   (maxtotflux = 2.6)             Maximum total flux fit range
    (mincutoff = 1.2)             Minimum total flux cutoff value
    (maxcutoff = 3)               Maximum total flux cutoff value
       (fittol = 1e-06)           Tolerance on Possion fit
     (fitstart = 1.5)             Initial background fit starting scale factor
#
# user setable parameters
#
      (clobber = no)              Overwrite if file exists
      (verbose = 0)               Debug level
      (logfile = stderr)          Debug file name
       (kernel = default)         Output format
#
# mode
#
         (mode = ql)
```

10.2 Parameter Descriptions

In the following parameter descriptions, `"required=yes"` indicates that the user must provide a value for the indicated parameter (*e.g.* `infile`) before the program will run. Also, the empty string (`" "`) is equivalent to the string `none` for filenames.

- `clobber`
  Overwrite if file exists
  type=boolean
  def=no

  Replace output file if it already exists?

  vtpdetect does not not overwrite existing outputs unless `clobber` is set to `yes`.

- `coarse`
  Minimum number of events per source
  type=integer
  def=10
  min=1

The minimum number of events required in order for a detection to be considered a real source.

If the field is heavily populated, there will be more occurrences of statistical fluctuations for which a given number of events are close to each other. It is advisable to increase the value of `coarse` to suppress false detections for such fields. Conversely, for extremely sparse (*i.e.* low background) applications, `coarse` can be decreased.

- `edge`
  How close to edge of field to reject events
  type=integer
  def=2

  Specifies how deep into the field to ignore events.

  The Voronoi tessellation at the edges is unbounded. This parameter controls how deep into the field to ignore events; the default is 2 tessellation cells.

- `ellsigma`
  Size of output source ellipses (in sigmas)
  type=real
  def=3.0

  The size, in sigmas, to make the elliptical source detection regions.

  The size of the elliptical source regions in both of the source region files is controlled via the `ellsigma` parameter. This parameter is a multiplicative factor applied to sigma, the standard deviation of the distribution, to scale the major and minor axes of the ellipses for each source detection. This feature is included so that the graphics overlay may be made more visible; it is does not affect the detections themselves, only the display of the regions.

  The distribution is defined as the distribution of counts within the source cell (*i.e.* the observed counts), and is assumed to be Gaussian. Thus, if `ellsigma` is 1, then the elliptical source region will contain 39.3% of the source counts (*i.e.* the 1-sigma point for a 2-D Gaussian).

  This feature is included so that the graphics overlay may be made more visible; it is does not affect the detections themselves, only the display of the regions.

- `expfile`
  Exposure map file name
  type=string
  filetype=input
  def=none

  The name of an input exposure map file.

  vtpdetect works in fluxes, and so needs to know about significant exposure deviations in order to work properly. The provided exposure map must be a 2-D image with WCS that matches the input file (`infile`).

- `fitstart`
  Initial background fit starting scale factor
  type=real
  def=1.5
  min=0.9
  max=2.0

  The initial guess at Poisson fit.

  This value is used as the first guess at where to separate background from source events. It is the normalized inverse area; values larger than this "flux cutoff" are classified as source events and smaller values are considered to belong to the background distribution.

The default value is adequate for most analyses.

- `fittol`
  Tolerance on Possion fit
  type=real
  def=1.0E-6

  The fit tolerance on Poisson fit.

  The default value is adequate for most analyses.

- `infile`
  Input file name
  type=string
  filetype=input
  required=yes

  The name of the input file.

  The input FITS file can an event list or an image. *CIAO* Data Manipulation (DM) syntax may be used in the input file specification, as explained in Chapter 1, Section 1.3.3.

  If the input file is an image, a pseudo event list is created; events are added with a multiplicity equal to the pixel amplitude. The FITS primary array may not be a floating point array; *vtpdetect* only knows how to convert an image that is an integer type: byte (BITPIX=8), short (BITPIX=16), or long (BITPIX=32).

- `kernel`
  Output format
  type=string
  def=default

  The format for the output file.

  The `kernel` parameter controls the output format (`default`, `fits`, or `iraf`); the default is for the output file to have the same format as the input file.

- `limit`
  Max. probability of being a false source
  type=real
  def=1.0E-6
  min=0.0
  max=1.0

  Maximum probability of a false source per number of background events.

  This parameter indicates the number of false detections per event. The user may wish to change the `limit`, depending on the size and population of the field. The default value is 10^{-6}; if there are significantly less than a million events, `limit` may be increased accordingly.

- `logfile`
  Debug file name
  type=string
  filetype=output
  def=STDOUT

  The name of an output log file.

  The verbosity of log information is sent to `STDERR` (usually the screen). However, if `log` is set to `yes`, then the log information will be written to the specified filename.

- `maxbkgflux`
  Maximum normalized background flux to fit
  type=real
  def=0.8

  The maximum normalized background flux level to fit with the cumulative Poisson distribution.

  This parameter is one of several which limits the search for the inverse area (*i.e.* "flux") value for either the total background flux or the cutoff point where the cumulative distribution of inverse areas of the data departs from that expected of a Poisson distributed background.

  The default value is adequate for most analyses; in rare instances, the user might want to widen or lessen these intervals.

- `maxcutoff`
  Maximum total flux cutoff value
  type=real
  def=3

  The maximum absolute normalized flux level to find background cutoff.

  This parameter is one of several which limits the search for the inverse area (*i.e.* "flux") value for either the total background flux or the cutoff point where the cumulative distribution of inverse areas of the data departs from that expected of a Poisson distributed background.

  The default value is adequate for most analyses; in rare instances, the user might want to widen or lessen these intervals.

- `maxiter`
  Maximum number of iterations to allow
  type=integer
  def=10
  min=1

  The maximum number of iterations to be allowed in estimating the background level.

  vtpdetect may enter a state where the background estimate cycles between 2 or 3 values; setting the `maxiter` parameter to different values in this case will allow the user to control the exit level. Note that the tool may converge before reaching `maxiter`.

- `maxtotflux`
  Maximum total flux fit range
  type=real
  def=2.6

  The maximum normalized total flux level to fit with a parabola.

  This parameter is one of several which limits the search for the inverse area (*i.e.* "flux") value for either the total background flux or the cutoff point where the cumulative distribution of inverse areas of the data departs from that expected of a Poisson distributed background.

  The default value is adequate for most analyses; in rare instances, the user might want to widen or lessen these intervals.

- `mincutoff`
  Minimum total flux cutoff value
  type=real
  def=1.2

  The minimum absolute normalized flux level to find background cutoff.

This parameter is one of several which limits the search for the inverse area (*i.e.* "flux") value for either the total background flux or the cutoff point where the cumulative distribution of inverse areas of the data departs from that expected of a Poisson distributed background.

The default value is adequate for most analyses; in rare instances, the user might want to widen or lessen these intervals.

- `mintotflux`
  Minimum total flux fit range
  type=real
  def=0.8

  The minimum normalized total flux level to fit with parabola.

  This parameter is one of several which limits the search for the inverse area (*i.e.* "flux") value for either the total background flux or the cutoff point where the cumulative distribution of inverse areas of the data departs from that expected of a Poisson distributed background.

  The default value is adequate for most analyses; in rare instances, the user might want to widen or lessen these intervals.

- `mode`
  def=ql

  The mode defines how to handle querying for parameters. For example, setting the mode to "h" means that no prompting will occur; this is useful when running the tool from a script. See "ahelp parameter" for more information and a list of all the available modes.

- `outfile`
  Source list output file name
  type=string
  filetype=output
  required=yes

  The name of the output file.

  The `outfile` parameter is used to provide a name for the output file. The `kernel` parameter controls the output format; the default is for the output file to have the same format as the input file.

- `regfile`
  Name for ASCII output region files
  type=string

  The name of an ASCII source region output file.

  In addition to the output source list (`outfile`), the user may choose to have an ASCII source region file written by specifying a filename in `regfile` parameter. If autonaming is used (*i.e.* `"regfile=."`), then the source regions output file will be named `"<infile_root>_reg"`.

  This region file contains the location and size of the principal axes, for each detected source elliptical region. The elliptical region size is such that the region contains some fraction of the source counts. The parameter `ellsigma` specifies the desired source count fraction.

  This region file is useful for subsequent input into ds9, particularly to overlay region markers over each detection; see Chapter 1, Section 1.4 for details.

- `scale`
  Threshold scale factor
  type=real
  def=1.0
  min=0

The background flux cutoff scale.

`scale` is actually a scale factor that refers to the intensity, not the spatial size. The algorithm figures out the threshold to distinguish between background and source events, based on the area of the Voronoi cell, then the `scale` parameter allows the user to scale that value. If `scale` is left to its default value of 1, the threshold remains as calculated by the code. Setting the `scale` value > 1 will tend to de-blend sources while losing faint sources. Setting it < 1 will tend to blend sources while picking out fainter sources. The recommended range for this parameter is between 0.8 and 3.

- `superdo`
  Perform Super Voronoi Cell procedure
  type=boolean
  def=no

  Whether to perform the super Voronoi cell update algorithm at the end of the program.

  Since this is a time-consuming and uncalibrated routine, this is optional.

  After the final iteration, the Voronoi tessellation is computed from just the set of source pixel locations and the edge pixel locations (to constrain the tessellation). To better account for flux in the wings of a source profile, the total flux is taken to be the total flux contained in these "super" Voronoi cells around each source. Then the background is estimated from this larger source area.

  This option is currently implemented, but is not well-tested. The output is printed only to `STDOUT`, appearing between the lines that have "BEFORE" and "AFTER" comments.

- `verbose`
  Debug level
  type=integer
  def=0
  min=0
  max=5

  The verbosity of log output.

  The verbosity of log information ranges from 0 (none) to 5 (maximum output).

  Setting `verbose` to 4 or 5 can produce hundreds of pages of output; use these values only if necessary to diagnose a problem during the percolation.

10.3 Data Product Descriptions

Source List File (`outfile`)

The primary output of *vtpdetect* is the FITS file of source detections. The following properties are recorded in the output file:

Data item	**Unit**	**Description**
RA	deg	Source Right Ascension
RA_ERR	deg	Source Right Ascension error
DEC	deg	Source Declination
DEC_ERR	deg	Source Declination error
POS(X,Y)	pixel	Physical coordinates
X_ERR	pixel	Source X position error
Y_ERR	pixel	Source Y position error
SRC_AREA	pixel	Source region area
NET_COUNTS	count	Net source counts
NET_COUNTS_ERR	count	Error in net source counts
BKG_COUNTS	count	Background counts (scaled to source cell)
BKG_COUNTS_ERR	count	Error in BKG_COUNTS
NET_RATE	count/s	Source count rate
NET_RATE_ERR	count/s	Source count rate error
BKG_RATE	count/s/pixel	Background count rate
BKG_RATE_ERR	count/s/pixel	Background count rate error
EXPTIME	s	Effective exposure time
SRC_CUTOFF	count/s/pixel	Source flux cutoff
FSP		False source probability
EDGE_OF_FIELD		Edge-of-field flag
SHAPE		Shape of source region
R[2]		Radii of source region
ROTANG	deg	Rotation angle of source region
COMPONENT		Source number

- RA, DEC:
  Right ascension and declination coordinates, corresponding to X and Y. Calculated only if WCS keywords are present in the data, otherwise filled with zeros.
- RA_ERR, DEC_ERR:
  RA and DEC errors, corresponding to X_ERR and Y_ERR.
- POS(X,Y):
  Location of the source detection.
- X_ERR, Y_ERR:
  Uncertainties on source detection location.
- SRC_AREA:
  The geometric area of the sum of the Voronoi cells that comprise the source, in units of image pixels.
- NET_COUNTS:
  Net source counts.
- NET_COUNTS_ERR:
  Net source counts corresponding error.
- BKG_COUNTS:
  Background counts.
- BKG_COUNTS_ERR:
  Background counts corresponding error.
- NET_RATE:
  Source count rate: the NET_COUNTS divided by EXPTIME.

- NET_RATE_ERR:
  Source count rate corresponding error: the NET_COUNTS_ERR divided by EXPTIME.
- BKG_RATE:
  Background count rate: the BKG_COUNTS divided by EXPTIME.
- BKG_RATE_ERR:
  Background count rate corresponding error: the BKG_COUNTS_ERR divided by EXPTIME.
- EXPTIME:
  Effective exposure time, taken from the exposure map.
- SRC_CUTOFF:
  The flux cutoff determined for the source.
- FSP:
  False source probability, *i.e.* the probability that the detection is not a real source.
- EDGE_OF_FIELD:
  Indicates whether any of the the Voronoi cells were at the edge of the field; in this case, it is possible that flux was lost.
- SHAPE:
  The shape of region output to the ASCII source region file; `ellipse` by default.
- R[2]:
  The estimated semi-major and semi-minor axis lengths (*i.e.* the x and y radius lengths in pixels, before rotation). For SHAPE = `ellipse`, this is `ellsigma` times the axis lengths.
- ROTANG:
  The counterclockwise angle between the semi-minor axis (see R) and the y-axis of the image, in degrees.
- COMPONENT:
  The number assigned to the detected source. Its primary use is for filtering of the source list by CIAO tools (*e.g.* `"component=1:10"`, `"component=1,2,5"`).

ASCII Source Region File (`regfile`)

In addition to the source list file (`outfile`), the user may also select to have a source region file (`regfile`) output.

This region file is useful for subsequent input into ds9, particularly to overlay region markers over each detection; see Chapter 1, Section 1.4 for details.

The source region file from the example in Chapter 8, Section 8.2.2:

```
unix% more example2_out.reg
physical
ellipse(370.711151,272.213654,519.869385,173.405792,0.867785)
ellipse(252.019577,452.235046,18.365868,15.496613,115.604019)
ellipse(373.667877,451.466675,11.422175,5.275846,167.176056)
.
.
.
```

Log File (`logfile`)

If `log` is set to `yes`, then the log information will be written to the file indicated (`vtpdetect.log` in this example).

The log file from the example in Chapter 8, Section 8.2.2:

```
unix% more vtpdetect.log
DEBUG: Version information -
DEBUG:     vtpdetect: 0.5.1
DEBUG:     datamodel : 1.99
DEBUG:     ascfit    : 2.2
.
.
.
```

Chapter 11

Running wavdetect

11.1 Key wavdetect Parameters

This section highlights some frequently used *wavdetect* parameters. Similar parameters and their description have been grouped together. Chapter 14 has information on each *wavdetect* parameter in alphabetical order.

1. Input file name (`infile`)

   The input file can an event list or an image. *CIAO* Data Manipulation (DM) syntax may be used in the input file specification, as explained in Chapter 1, Section 1.3.3.

   Be careful using images larger than 1024×1024, as a large amount of computer memory must be available.

2. Output source list file name (`outfile`)

   The `outfile` parameter is used to provide a name for the output file. The `kernel` parameter controls the output format; the default is for the output file to have the same format as the input file. If autonaming is used (*i.e.* `"outfile=."`), then the output file will be named `"<infile_root>_src"`.

3. Output source cell image file name (`scellfile`)

   The `scellfile` is the image showing the source cells. These source cells delimit the image pixels that are used to estimate source properties (*e.g.* count rate). If autonaming is used (*i.e.* `"scellfile=."`), then the output file will be named `"<infile_root>_scell"`.

4. Output reconstructed image file name (`imagefile`)

   The reconstructed source image filename is given by the `imagefile` parameter. If autonaming is used (*i.e.* `"imagefile=."`), then the output file will be named `"<infile_root>_image"`.

5. Output normalized background file name (`defnbkgfile`)

   The output file that will contain the default normalized (*i.e.* flat-fielded) background is specified by the `defnbkgfile` parameter. During its second stage, *wavdetect* makes one normalized background estimate from the stack of per-scale normalized backgrounds produced during the first stage. It writes the computed background to this file. If autonaming is used (*i.e.* `"defnbkgfile=."`), then the output file will be named `"<infile_root>_nbkg"`.

6. Exposure map file name (`expfile`)
   Exposure time (`exptime`)
   Minimum relative exposure needed (`expthresh`)

   To obtain true count rates with *wavdetect*, the output values must be divided by the exposure time. If an exposure map is not provided, a dummy array with all pixel values set to 1 will be used.

   The `exptime` parameter specifies the length of time in seconds over which the field was observed. If set to 0, the value will either be taken from the `expfile` FITS header or, failing that, estimated by averaging over exposure map pixel values at the center of the field.

   The `expthresh` parameter specifies the minimum relative exposure needed in a pixel in order to analyze it. This parameter should not be less than 0.1 or the accuracy of normalization will be too low. If set close to 1, only very limited regions of the field of view will be considered when source correlation maxima are listed. Typical values are 0.1-0.2.

7. Wavelet scales (`scales`)

   The `scales` parameter determines how many scaled transforms will be computed. *wtransform* produces a complete set of outputs for each scale. Small scales tend to detect small features, and larger scales find larger features. The units for `scales` are pixels. The value of `scales` is the radius of the Mexican Hat function, which crosses zero at $\sqrt{2}\times$radius.

   For the initial run, try the default value `"2.0 4.0"`. If that test is successful, try again with `scales "1.0 2.0 4.0 8.0 16.0"`. For a more extensive run, the user might wish to try the $\sqrt{2}$ series: `"1.0 1.414 2.0 2.828 4.0 5.657 8.0 11.314 16.0"`.

   The primary concern regarding this parameter is to match the first scale size to the point spread function (PSF). Once that is done, the user must decide how many scaled wavelets to use. To some extent, choices are based on the computing resources at hand. Reasonable inputs are a 512×512 array and 5 to 9 scales, where each scale is a factor of 2 or $\sqrt{2}$ larger than the preceeding one.

   In practice, one must not have too many pixels or scale sizes, otherwise the computational load becomes a problem. Data structures for a 512×512 image use up 36 MB, while a 2048×2048 image requires over 300 MB. Datasets that do not fit in physical memory will page heavily to disk and processing will progress very slowly. Scale sizes larger than 32 allocate excessive memory because it is necessary to pad the image with surrounding zeros.

   To deal with resolved sources, several convolutions are performed, each with a successively larger scale version of the wavelet. The resulting "correlation maps" are then examined for regions where the intensity is larger than some threshold, and the final source list is constructed from a comparison of the different scale runs.

8. Threshold significance for output source pixel list (`sigthresh`)

   The significance threshold for source detection. A good value to use is the inverse of the total number of pixels in the image, *e.g.* $\sim 10^{-6}$ (the default) for a 1024×1024 field. This is equivalent to stating that the expected number of false sources per field is one. If larger arrays are used without decreasing the value of `sigthresh`, there will be an increased probability of detecting false sources. Likewise, if smaller arrays are used, the user may wish to increase this parameter value. The `sigthresh` should not be smaller than roughly $\sim 10^{-9}$ to $\sim 10^{-10}$; at this value, the accuracy of the computed detection thresholds is unknown.

9. ASCII region file (`regfile`)
   Size of output source ellipses (`ellsigma`)

   In addition to the output source list (`outfile`), the user may choose to have an ASCII source region file written by specifying a filename in `regfile` parameter. If autonaming is used (*i.e.* `"regfile=."`), then the source regions output file will be named `"<infile_root>_reg"`.

This region file is useful for subsequent input into ds9, particularly to overlay region markers over each detection; see Chapter 1, Section 1.4 for details.

The size of the elliptical source regions in both of the source region files is controlled via the `ellsigma` parameter. This parameter is a multiplicative factor applied to sigma, the standard deviation of the distribution, to scale the major and minor axes of the ellipses for each source detection. This feature is included so that the graphics overlay may be made more visible; it is does not affect the detections themselves, only the display of the regions.

By default the value for `ellsigma` is 3, but a larger value is often helpful.

10. Maximum number of source-cleansing iterations (`maxiter`)
    Minimum fraction of pixels that must be cleansed to continue (`iterstop`)

    The maximum number of iterations per scale pair for cleaning sources from the data in order to estimate the background map; the default is 2. Increasing this number will increase the method's source detection sensitivity, but may not increase it enough to justify the increased computation time. More iterations are generally needed for large wavelet scales (*e.g.* 16 pixels in a *Rosat* 512×512 PSPC field). Note that the tool may converge before reaching `maxiter`.

    If the ratio of newly cleansed pixels to the overall number of pixels in the dataset is less than the parameter `iterstop` when *wavdetect* constructs a default normalized (*i.e.* flat-fielded) background (`defnbkgfile`), then *wavdetect* stops iterating and uses the current background estimate as the final background estimate. The background is calculated if there are very few new pixels being cleansed (see `bkgsigthresh`).

    The `iterstop` parameter specifies how many iterations the program should go through to calculate the background. A typical value is 0.0001 (the default): stop iterating when one of every thousand pixels has been cleansed. This parameter should not be smaller than one over the number of pixels in the dataset or larger than 1.

11. Threshold significance when estimating bkgd only (`bkgsigthresh`)

    A significance for cleaning data from the image to compute the background map. As it does not effect source detection, this parameter should be set to a more liberal value than `sigthresh` (*e.g.* 10^{-2} or 10^{-3}, the default), but not smaller than `sigthresh`). This will help reduce the effect of weak undetectable sources on the background map calculation. This value should be no larger than 0.05, the usual 5% (or 95%) statistical criterion for rejecting the null hypothesis, which is that the pixel in question has data sampled solely from the background.

12. Input background file name (`bkginput`)
    Use bkginput[2] for background error (`bkgerrinput`)
    Exposure time for input background file (`bkgtime`)

    If desired, a previously computed background image may be input instead of allowing *wavdetect* to construct a default normalized background. When using `bkginput`, enter `none` for the default normalized background output file (`defnbkgfile`).

    If `bkgerrinput` is changed to `yes`, then the background error data in the second extension of the `bkginput` file will be used (*i.e.* "bkginputfilename.fits[2]").

    The `bkgtime` parameter may specify the `LIVETIME` value for the provided background.

13. Table of PSF size data (`psftable`)
    Encircled energy of PSF (`eenergy`)
    Energy band (`eband`)
    Offset of x axis from optical axis (`xoffset`)
    Offset of y axis from optical axis (`yoffset`)

In *wavdetect*, PSF, energy, and offset information supplied via these parameters is used to select a source counts image; a source cell is computed using this image, and source properties are estimated using the data within the computed cell. Note that, unlike *celldetect*, these parameter values have no effect on the detection process. Instead, these parameter values affect reported source properties.

The PSF information used by *wavdetect* is contained in a *Chandra* calibration file distributed with *CIAO*, whose location is given by the value of the `psftable` parameter; by default this parameter is automatically set to point to the appropriate *Chandra* calibration PSF file.

The selection of a particular PSF size is governed by the energy band parameter `eband`, in keV (since the *Chandra* PSF is a function of energy). By default the energy band is set to 1.4967 keV, but the user may wish to adjust this value.

wavdetect also uses an encircled energy percentage value, given by the `eenergy` parameter, in concert with the calibration PSF file. This value is the percentage of PSF energy to be encircled by the detect cell, expressed as a fraction of 1.0. The `eenergy` is a key parameter for detecting off-axis sources, and users are urged to experiment with other values. By default, the value is set to 0.393 (the 1-sigma integrated volume of a normalized two-dimensional Gaussian); another suggested value to try is 0.5. Note that the efficiency of the algorithm is reduced if this fraction is too high or too low. If the value of `eenergy` is set too high (*e.g.* 0.9), there is the risk that the computed source cell will contain more than one source, rendering source property estimates moot. On the other hand, if the value is set too low, then the computed cell may contain only a fraction of the source counts.

By default, *wavdetect* calculates the off-axis angle using the nominal pointing of the data file as the origin. Users may change this behavior by setting offsets to any numerical values (using the `xoffset` and `yoffset` parameters); these offsets provide the location of the optical axis with respect to the center of the data file. A typical scenario in which the user may select to use offsets is when the data file is a sum of two or more observations with different pointings. The nominal pointing in such a data file is usually poorly defined and the off-axis angle could be calculated from an undesired origin, leading to suboptimal selection of the detect cell size.

14. Overwrite existing outputs (`clobber`)
    Log verbosity (`verbose`)
    Make a log file (`log`)

    wavdetect does not not overwrite existing outputs unless `clobber` is set to `yes`.

    The verbosity of log information ranges from 0 (none) to 5 (maximum output) and is sent to `STDERR` (usually the screen). However, if `log` is set to `yes`, then the log information will be written to the files `wrecon.log` and `wtransform.log`.

11.2 Examples Using Dataset A (Simulated ACIS Data)

11.2.1 Example 1: Running wavdetect with the Default Parameters

This first example illustrates running *wavdetect* using a simulated ACIS dataset as input; see Chapter 2, Section 2.1 for further information about this dataset.

Description of the Default Parameter Functionality

As required, the user specifies the following:

1. Input file name (`infile = "data/datasetA_acis_img.fits"`) The input file is an image of simulated *Chandra* data, named `data/datasetA_acis_img.fits`.
2. Output source list file name (`outfile = "example1_out.fits"`) The output file is to be named `example1_out.fits`.
3. Output source cell image file name (`scellfile = "example1_scell.fits"`) The output source cell image file is to be named `example1_scell.fits`
4. Output reconstructed image file name (`imagefile = "example1_image.fits"`) The output reconstructed image file is to be named `example1_image.fits`
5. Output normalized background file name (`defnbkgfile = "example1_defnbkg.fits"`) The output normalized background file is to be named `example1_defnbkg.fits`

None of the parameters are changed from their default values, so:

1. Wavelet scales (`scales = 2.0 4.0`)

   wavdetect will use the default scales of `"2.0 4.0"`.
2. Maximum number of source-cleansing iterations (`maxiter = 2`)

   The maximum number of iterations for cleaning sources from the data will be 2.
3. Threshold significance for output source pixel list (`sigthresh = 1e-06`)

   The significance threshold for source detection will be 10^{-6}.
4. ASCII region file (`regfile = " "`)
   Size of output source ellipses (`ellsigma = 3`)

   No ASCII region file will be produced. The FITS source list will contain a 3-sigma ellipse for each detection.
5. Minimum fraction of pixels that must be cleansed to continue (`iterstop = 0.0001`)

   wavdetect will stop iterating when one of every thousand pixels has been cleaned.

Setting Required Parameters

The `pset` tool is used to supply the required input parameter values:

```
unix% punlearn wavdetect
unix% pset wavdetect infile = data/datasetA_acis_img.fits
unix% pset wavdetect outfile = example1_out.fits
unix% pset wavdetect scellfile = example1_scell.fits
unix% pset wavdetect imagefile = example1_image.fits
unix% pset wavdetect defnbkgfile = example1_defnbkg.fits
```

Notice that a path may be include with any of the filenames. Chapter 1, Section 1.3.1 has further information on manipulating parameters.

Listing Current Parameter Settings

If desired, the user may then list the current *wavdetect* parameter settings with `plist`:

```
unix% plist wavdetect

Parameters for /home/username/cxcds_param/wavdetect.par

#
#   parameter file for wavdetect
#
#
#   input
#
        infile = data/datasetA_acis_img.fits Input file name
#
#   output
#
       outfile = example1_out.fits Output source list file name
     scellfile = example1_scell.fits Output source cell image file name
     imagefile = example1_image.fits Output reconstructed image file name
   defnbkgfile = example1_defnbkg.fits Output normalized background file name
#
#   scales
#
        scales = 2.0 4.0          wavelet scales (pixels)
      (regfile = )                ASCII regions output file
#
#   output options
#
      (clobber = no)              Overwrite existing outputs?
       (kernel = default)         Output file format (fits|iraf|default)
     (ellsigma = 3.0)             Size of output source ellipses (in sigmas)
     (interdir = .)               Directory for intermediate outputs
#
#########################################################################
#
#   wtransform parameters
#
#
#   optional input
#
     (bkginput = )                Input background file name
  (bkgerrinput = no)              Use bkginput[2] for background error
#
#   output info
#
  (outputinfix = )                Output filename infix
#
#   output content options
```

```
#
    (sigthresh = 1e-06)           Threshold significance for output source pixel list
 (bkgsigthresh = 0.001)           Threshold significance when estimating bkgd only
#
#   exposure info
#
      (exptime = 0)               Exposure time (if zero, estimate from map itself
      (expfile = )                Exposure map file name (blank=none)
    (expthresh = 0.1)             Minimum relative exposure needed in pixel to analyze it
#
#   background
#
      (bkgtime = 0)               Exposure time for input background file
#
#   iteration info
#
      (maxiter = 2)               Maximum number of source-cleansing iterations
     (iterstop = 0.0001)          Min frac of pix that must be cleansed to continue
#
#   end of wtransform parameters
#
########################################################################
########################################################################
#
#   wrecon parameters
#
#
#   PSF size parameters
#
      (xoffset = INDEF)           Offset of x axis from optical axis
      (yoffset = INDEF)           Offset of y axis from optical axis
        (eband = 1.4967)          Energy band
      (eenergy = 0.393)           Encircled energy of PSF
     (psftable = ${ASCDS_CALIB}/psfsize20010416.fits -> /soft/ciao/data/psfsize20010416.fits) Table of 
#
#   end of wrecon parameters
#
########################################################################
#
#   run log verbosity and content
#
          (log = no)              Make a log file?
      (verbose = 0)               Log verbosity
#
#   mode
#
         (mode = ql)
```

Executing *wavdetect*

To execute *wavdetect*, type the name of the tool on the command line:

```
unix% wavdetect
Input file name (data/datasetA_acis_img.fits):
Output source list file name (example1_out.fits):
Output source cell image file name (example1_scell.fits):
Output reconstructed image file name (example1_image.fits):
Output normalized background file name (example1_defnbkg.fits):
unix%
```

The parameter settings are shown in the prompt for each required parameter; these settings may be accepted by hitting the `<RETURN> key` or changed if desired.

Figure 11.1 shows the data with 3-sigma ellipses from the region file overlaid. Chapter 1, Section 1.4 has instructions on how to display the region file in ds9.

Examining Results

The *CIAO* tool `dmlist` is used to for examine the contents of the output file. To see what blocks are in the file:

```
unix% dmlist example1_out.fits opt=blocks
--------------------------------------------------------------------------------
Dataset: example1_out.fits
--------------------------------------------------------------------------------

     Block Name                          Type         Dimensions
--------------------------------------------------------------------------------
Block    1: PRIMARY                        Null
Block    2: SRCLIST                        Table        26 cols x 24       rows
```

In the `SRCLIST` block, there are 26 columns of information for each of the 24 detected sources (one row for each detected source). To list the names of these columns, along with brief column information:

```
unix% dmlist "example1_out.fits[SRCLIST]" opt=cols

--------------------------------------------------------------------------------
Columns for Table Block SRCLIST
--------------------------------------------------------------------------------

ColNo  Name                 Unit        Type             Range
   1   RA                   deg          Real8          0:       360.0
Source Right Ascension
   2   DEC                  deg          Real8          -90.0:       90.0
```

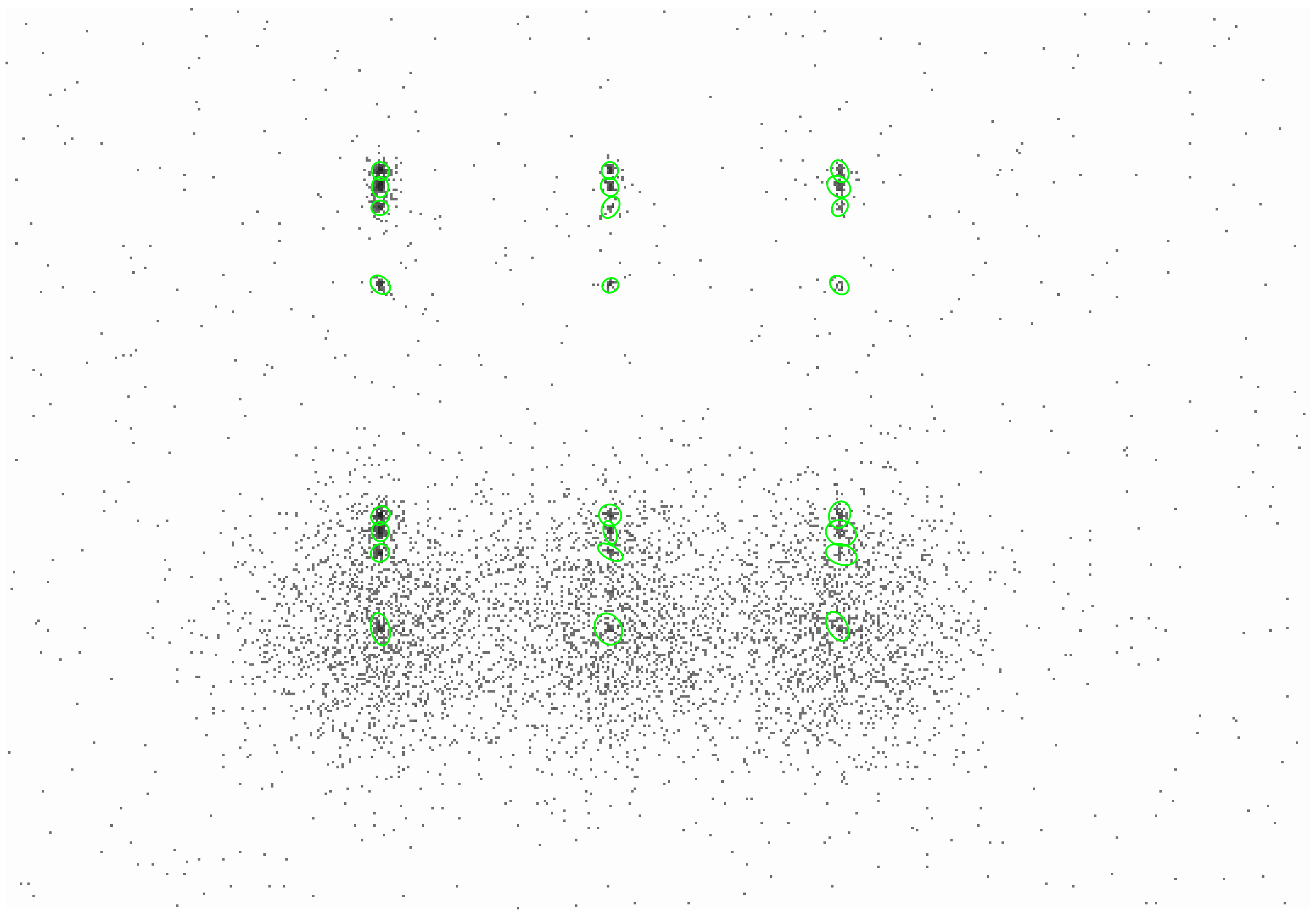

Figure 11.1: *wavdetect* with default parameters. The ACIS-S simulated field with 3-sigma ellipses showing the *wavdetect* results.

```
Source Declination
   3   RA_ERR              deg          Real8          -Inf:+Inf
Source Right Ascension Error
   4   DEC_ERR             deg          Real8          -Inf:+Inf
Source Declination Error
   5   POS(X,Y)            pixel        Real8          -Inf:+Inf
Physical coordinates
   6   X_ERR               pixel        Real8          -Inf:+Inf
Source X position error
   7   Y_ERR               pixel        Real8          -Inf:+Inf
Source Y position error
   8   NPIXSOU             pixel        Int4           -
pixels in source region
   9   NET_COUNTS          count        Real4          -Inf:+Inf
Net source counts
  10   NET_COUNTS_ERR      count        Real4          -Inf:+Inf
Error in net source counts
  11   BKG_COUNTS          count        Real4          -Inf:+Inf
Background counts (scaled to source cell)
  12   BKG_COUNTS_ERR      count        Real4          -Inf:+Inf
Error in BKG_COUNTS
  13   NET_RATE                         Real4          -Inf:+Inf
Source count rate (count/exposure map units)
  14   NET_RATE_ERR                     Real4          -Inf:+Inf
Source count rate error (count/exposure map uni
  15   BKG_RATE                         Real4          -Inf:+Inf
Background count rate (count/exposure map unit)
  16   BKG_RATE_ERR                     Real4          -Inf:+Inf
Background count rate error
  17   EXPTIME                          Real4          -Inf:+Inf
Effective exposure map value (exposure map unit
  18   EXPTIME_ERR                      Real4          -Inf:+Inf
Effective exposure map error
  19   SRC_SIGNIFICANCE                 Real4          -Inf:+Inf
Source Significance
  20   PSF_SIZE            pixel        Real4          -Inf:+Inf
Estimated size of PSF
  21   MULTI_CORREL_MAX                 Int4           -
1 if source contails 2+ correl maxima
  22   SHAPE                            String[10]
Shape of source region
  23   R[2]                             Real4(2)       -Inf:+Inf
Radii of source region source
  24   ROTANG              deg          Real4          0:      180.0
Rotation angle of source region
  25   PSFRATIO                         Real4          -Inf:+Inf
Ratio sqrt(axmax*axmin)/psf_size
  26   COMPONENT                        Int4           -
Source Number

--------------------------------------------------------------------------------
```

```
4:
--------------------------------------------------------------------------------

ColNo    Name
5:    EQSRC(RA_SRC ) = (+0)[deg] +TAN[(-0.0001361)* (POS(X)-(+256.50))]
           (DEC_SRC)   (+0)           (+0.0001361)  (    (Y) (+256.50))
```

To further examine the contents of the output file, use `dmlist` to show the position and net counts of each source detection:

```
unix% dmlist "example1_out.fits[SRCLIST][cols X,Y, NET_COUNTS]" opt=data

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

ROW    POS(X,Y)                                 NET_COUNTS

     1 (      251.9739130435,      268.6478260870)       173.5043029785
     2 (      251.8786407767,      309.1310679612)       167.4067993164
     3 (      251.9256198347,      320.4710743802)       312.7768249512
     4 (      252.0469135802,      328.9777777778)       351.6253967285
     5 (      251.9171270718,      451.4972375691)       170.5679016113
     6 (      251.7078651685,      492.3539325843)       155.3647460938
     7 (      251.8429951691,      503.3381642512)       365.8185424805
     8 (      252.0277777778,      511.8636363636)       350.7105407715
     9           (      373.06250,      268.750)       32.4447364807
    10 (      374.0681818182,      309.5227272727)        29.1281700134
    11 (      373.9565217391,      320.0289855072)        54.2208480835
    12 (      373.9577464789,      329.1690140845)        57.9016990662
    13 (      374.1111111111,      451.1944444444)        34.3589019775
    14 (      374.1818181818,      492.6363636364)        27.9628028870
    15 (      373.7014925373,      503.4776119403)        59.2189598083
    16 (      373.8285714286,      512.0857142857)        62.5913848877
    17 (      494.6417910448,      270.1791044776)        36.3386344910
    18 (      496.5862068966,      319.7356321839)        63.0490913391
    19 (      496.7272727273,      308.3409090909)        26.6807270050
    20 (      495.8470588235,      329.4470588235)        68.9102401733
    21 (      495.6571428571,      451.3714285714)        33.5286636353
    22 (      495.9583333333,      492.5833333333)        20.7489109039
    23 (      495.3918918919,      503.6351351351)        64.2184524536
    24     (      495.9583333333,      511.750)       40.6809768677
```

The graphical user interface of the *CIAO* tool *Prism*is an alternative to the command-line interface of `dmlist`:

```
unix% prism example1_out.fits &
```

11.2.2 Example 2: Running wavdetect to Create Additional Output

This next example illustrates running *wavdetect* using as input the same *Chandra* dataset from the first example (see Section 11.2.1), but with a few parameters changed from their defaults. This will cause additional output files to be produced:

- `regfile`: an ASCII region file.
- `verbose` and `log`: increased verbosity is written to the `wrecon.log` and `wtransform.log` output files.

In addition, the `ellsigma` parameter is increased to make the display of detected regions easier to see when displayed.

Setting Parameters

The input file is the same image used in the first example, and the output files are named similarly (`example2_something.fits`). We begin by setting all the parameters mentioned:

```
unix% punlearn wavdetect
unix% pset wavdetect infile = "data/datasetA_acis_img.fits[123:630,109:614]"
unix% pset wavdetect outfile = example2_out.fits
unix% pset wavdetect scellfile = example2_scell.fits
unix% pset wavdetect imagefile = example2_image.fits
unix% pset wavdetect defnbkgfile = example2_defnbkg.fits
unix% pset wavdetect regfile = example2_out.reg
unix% pset wavdetect ellsigma = 5
unix% pset wavdetect verbose = 4
unix% pset wavdetect log = yes
```

DM syntax is used to specify only a portion of the file for input (*i.e.* `[123:630,109:614]`); see Chapter 1, Section 1.3.3 for further information about DM syntax usage with

A region file named `example2_out.reg` will be produced, and the size of the output source ellipses is increased to 5-sigma. The verbosity of log information is changed to 4 and will be recorded in `wrecon.log` and `wtransform.log`.

Executing *wavdetect*

```
unix% wavdetect
Input file name (data/datasetA_acis_img.fits[123:630,109:614]):
Output source list file name (example2_out.fits):
Output source cell image file name (example2_scell.fits):
Output reconstructed image file name (example2_image.fits):
Output normalized background file name (example2_defnbkg.fits):
unix%
```

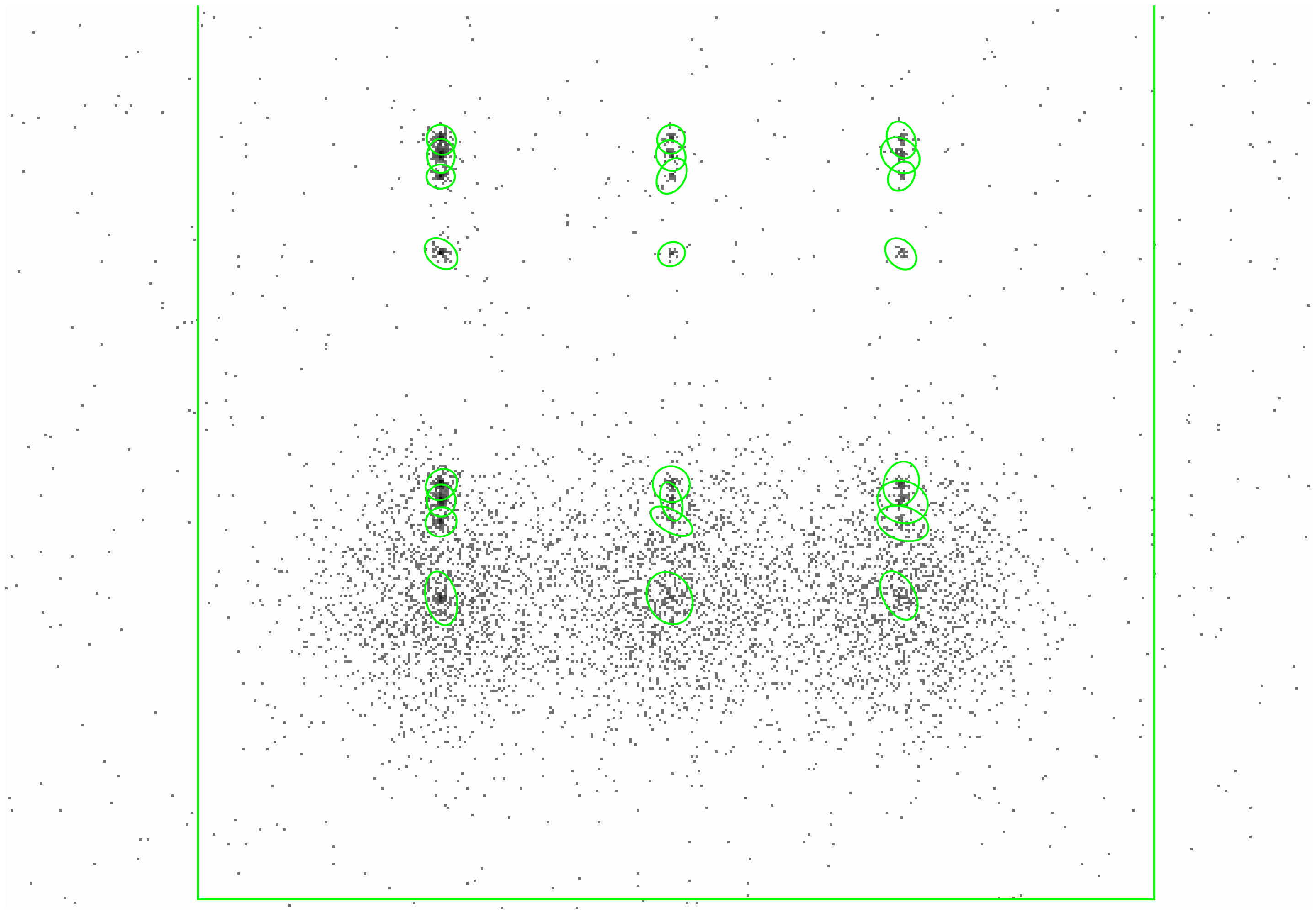

Figure 11.2: *wavdetect* results with `ellsigma=5`. The ACIS-S simulated field with ellipses showing the *wavdetect* results. The size of the ellipses has been increased from the default, which improves their visibility. The box shows the region of the chip searched by *wavdetect*.

The results are shown in Figure 11.2. Each detection has a 5-sigma ellipse overlay, since `ellsigma` was set to 5.

Examining Results

- As in the first example, this `dmlist` command shows what blocks are in the output source list file:

```
unix% dmlist example2_out.fits opt=blocks
--------------------------------------------------------------------------------
Dataset: example2_out.fits
--------------------------------------------------------------------------------
```

```
     Block Name                          Type         Dimensions
--------------------------------------------------------------------------------
Block    1: PRIMARY                        Null
Block    2: SRCLIST                        Table        26 cols x 24       rows
```

The `SRCLIST` block looks the same as Example 11.2.1: 26 columns of information for each of the 18 detected sources.

- Since the `verbose` parameter was increased and the `log` parameter was set to `yes`, output `wrecon.log` and `wtransform.log` files were created:

```
unix% more wrecon.log
reading data image
Input image file: data/datasetA_acis_img.fits[123:630,109:614]
Input correlation maxima stack: ./wd_srclist_stk
Input correlation image stack: ./wd_correl_stk
Input background stack: ./wd_nbkg_stk
Input exposure map map:
.
.

unix% more wtransform.log
input file: data/datasetA_acis_img.fits[123:630,109:614]
exposure file:
correlation maxima output stack: ./wd_srclist_stk
correlation image output stack: ./wd_correl_stk
normalized background output stack: ./wd_nbkg_stk
plain background output stack: none
threshold output stack: none
Map is multiplied by time factor 0.000000
.
.
```

11.3 Examples Using Dataset B (3C 295 on ACIS-S)

11.3.1 Example 1: Detecting Sources within Extended Emission

This first example illustrates running *wavdetect* using a *Chandra* ACIS dataset of 3C 295 as input; see Chapter 2, Section 2.2 for further information about this dataset.

All parameters are kept at defaults, except for:

- `ellsigma`: increased to make the display of detected regions easier to see.
- `regfile`: an ASCII region file.

Setting Parameters

Set the parameters:

```
unix% punlearn wavdetect
unix% pset wavdetect infile = \
"data/acisf00578N002_evt2.fits[(x,y)=rotbox(4114,4232,512,512,0)][ccd_id=7]"
unix% pset wavdetect outfile = example1_out.fits
unix% pset wavdetect scellfile = example1_scell.fits
unix% pset wavdetect imagefile = example1_image.fits
unix% pset wavdetect defnbkgfile = example1_defnbkg.fits
unix% pset wavdetect regfile = example1_out.reg
unix% pset wavdetect ellsigma = 5
```

DM syntax is used to specify only a portion of the file for input (*i.e.* `[(x,y)=rotbox(4114,4232,512,512,0)]`); see Chapter 1, Section 1.3.3 for further information about DM syntax usage with *wavdetect*.

Executing *wavdetect*

```
unix% wavdetect
Input file name (data/acisf00578N002_evt2.fits[(x,y)=rotbox(4114,4232,512,512,0)][ccd_id=7]):
Output source list file name (example1_out.fits):
Output source cell image file name (example1_scell.fits):
Output reconstructed image file name (example1_image.fits):
Output normalized background file name (example1_defnbkg.fits):
unix%
```

The results are shown in Figure 11.3.

11.3.2 Example 2: Detecting Sources within Extended Emission with an Exposure Map

This next example illustrates running *wavdetect* using as input the same *Chandra* dataset, but utilizing an exposure map. In this example, *wavdetect* will be run twice for comparison: with and without an exposure map.

Generating Input Data

Since we want to utilize an image exposure map, the data must be binned into an image. The `dmcopy` tool is used to create an image of chip S3 blocked by a factor of 2:

```
unix% punlearn dmcopy
```

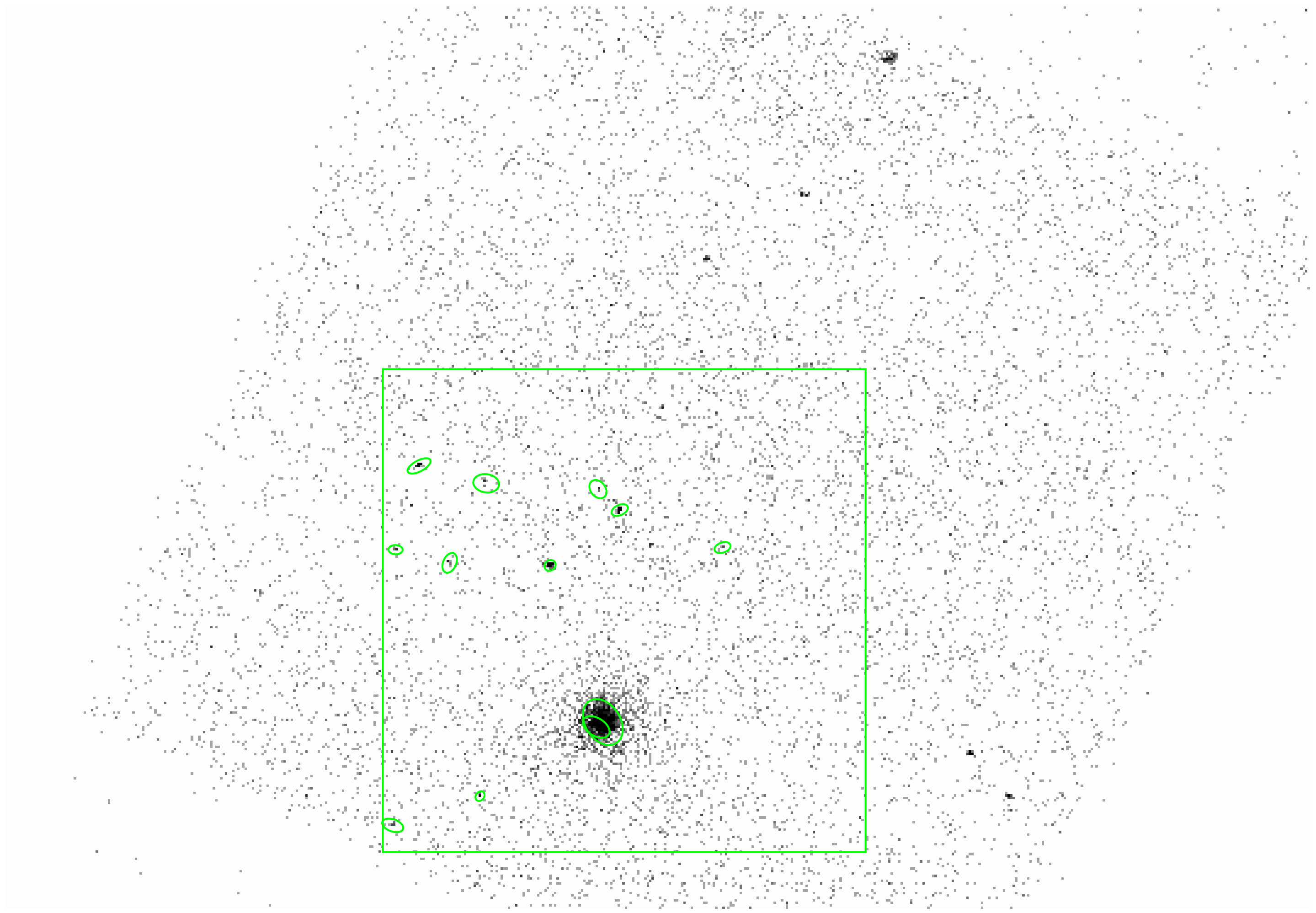

Figure 11.3: *wavdetect* on extended emission. A section of chip S3 (`ccd_id=7`) of the ACIS-S dataset of 3C 295, displayed with binning = 2; the ellipses show the *wavdetect* results. The box shows the region of the chip searched by *wavdetect*.

```
unix% dmcopy \
"data/acisf00578N002_evt2.fits[ccd_id=7][bin x=::2,y=::2][(x,y)=rotbox(4114,4232,512,512,0)]" \
acisf00578N002_evt2_ccdid7_imgbin2_reg1.fits
```

An exposure map to match this image was generated following the CIAO thread, "Compute Single Chip ACIS Exposure Map":

`http://cxc.harvard.edu/ciao/threads/expmap_acis_single/`

```
unix% punlearn asphist
unix% asphist \
      infile="pcadf052378346N002_asol1.fits[@data/acisf00578N002_evt2.fits[GTI7]]" \
      outfile="asphist_7.fits" evtfile="data/acisf00578N002_evt2.fits" mode="h"

unix% punlearn mkinstmap
unix% mkinstmap outfile="instmap_0.6kev_7.fits" monoenergy="0.6" \
      pixelgrid="1:1024:#1024,1:1024:#1024" obsfile="asphist_7.fits[asphist]" \
      detsubsys="ACIS-7" mode="h"

unix% punlearn mkexpmap
unix% mkexpmap asphistfile="asphist_7.fits" outfile="expmap_0.6kev_7_reg1.fits" \
      instmapfile="instmap_0.6kev_7.fits" normalize="no" \
      xygrid="3857.5:4371.5:#257,3975.5:4489.5:#257" mode="h"
```

The resulting file is named `expmap_0.6kev_7_reg1.fits`.

Running *wavdetect* without an Exposure Map

First, *wavdetect* is run without an exposure map:

- Set the necessary parameters:

  ```
  unix% punlearn wavdetect
  unix% pset wavdetect infile = "acisf00578N002_evt2_ccdid7_imgbin2_reg1.fits"
  unix% pset wavdetect outfile = example2a_out.fits
  unix% pset wavdetect scellfile = example2a_scell.fits
  unix% pset wavdetect imagefile = example2a_image.fits
  unix% pset wavdetect defnbkgfile = example2a_defnbkg.fits
  unix% pset wavdetect regfile = example2a_out.reg
  unix% pset wavdetect ellsigma = 5
  ```

  - `ellsigma`: increased to make the display of detected regions easier to see.
  - `regfile`: an ASCII region file.

  All other parameters are left at their default values.

- Execute *wavdetect*:

```
unix% wavdetect
Input file name (acisf00578N002_evt2_ccdid7_imgbin2_reg1.fits):
Output source list file name (example2a_out.fits):
Output source cell image file name (example2a_scell.fits):
Output reconstructed image file name (example2a_image.fits):
Output normalized background file name (example2a_defnbkg.fits):
unix%
```

Running *wavdetect* with an Exposure Map

Next, *wavdetect* is run with an exposure map:

- Set the necessary parameters:

  ```
  unix% punlearn wavdetect
  unix% pset wavdetect infile = "acisf00578N002_evt2_ccdid7_imgbin2_reg1.fits"
  unix% pset wavdetect outfile = example2b_out.fits
  unix% pset wavdetect scellfile = example2b_scell.fits
  unix% pset wavdetect imagefile = example2b_image.fits
  unix% pset wavdetect defnbkgfile = example2b_defnbkg.fits
  unix% pset wavdetect expfile = expmap_0.6kev_7_reg1.fits
  unix% pset wavdetect regfile = example2b_out.reg
  unix% pset wavdetect ellsigma = 5
  unix% pset wavdetect expfile = expmap_0.6kev_7_reg1.fits
  ```

  One additional parameter is used for the exposure map case:

  - `expfile`: the exposure map to be used in the detection.

- Execute *wavdetect*:

  ```
  unix% wavdetect
  Input file name (acisf00578N002_evt2_ccdid7_imgbin2_reg1.fits):
  Output source list file name (example2b_out.fits):
  Output source cell image file name (example2b_scell.fits):
  Output reconstructed image file name (example2b_image.fits):
  Output normalized background file name (example2b_defnbkg.fits):
  unix%
  ```

Comparing *wavdetect* Results with and without an Exposure Map

The results of both runs are shown in Figure 11.4. The use of an exposure map with *wavdetect* improves source property estimates; the detections are the same regardless of whether an exposure map was used. The detections are only affected in the case where the sources exist in an area for which the exposure map has a relative value less than `expthresh`.

The calculated source properties, however, are improved when an exposure map is supplied to *wavdetect*. For example, the `NET_RATE` column in the output will be different because the output values must be divided by the exposure time to obtain true rates with *wavdetect*.

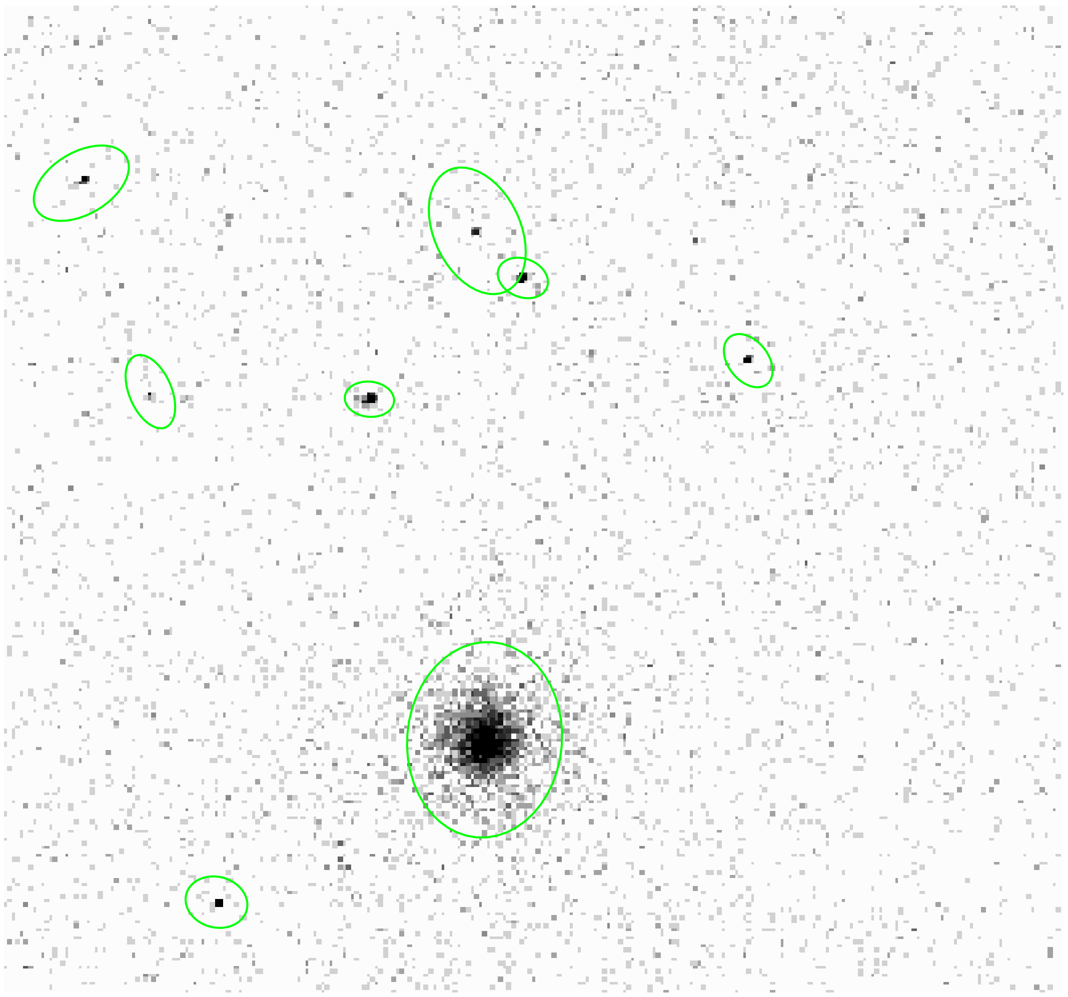

Figure 11.4: *wavdetect* with and without an exposure map. Chip S3 (`ccd_id=7`) of the ACIS-S dataset of 3C 295, binned by 2. The ellipses show the *wavdetect* results achieved with and without the use of an exposure map; there is only one set of regions as using an exposure map with this data does not change the source detection regions.

11.4 Examples Using Dataset C (3C 273 on HRC-I)

11.4.1 Example 1: Detecting Close Extended Sources

This example illustrates running *wavdetect* using a *Chandra* HRC-I dataset of 3C 273 as input; see Chapter 2, Section 2.3 for further information about this dataset.

Setting Parameters

Set the usual parameters:

```
unix% punlearn wavdetect
unix% pset wavdetect infile = \
"data/hrcf00461N003_evt2.fits[EVENTS][(x,y)=rotbox(16488.2,16285.8,512,512,0)]"
unix% pset wavdetect outfile = example1_out.fits
unix% pset wavdetect scellfile = example1_scell.fits
unix% pset wavdetect imagefile = example1_image.fits
unix% pset wavdetect defnbkgfile = example1_defnbkg.fits
unix% pset wavdetect regfile = example1_out.reg
unix% pset wavdetect ellsigma = 2
```

DM syntax is used to specify only a portion of the file for input (*i.e.* `[(x,y)=rotbox(16488.2,16285.8,512,512,0)]`); see Chapter 1, Section 1.3.3 for further information about DM syntax usage with *wavdetect*.

Executing *wavdetect*

```
unix% wavdetect
Input file name (data/hrcf00461N003_evt2.fits[EVENTS][(x,y)=rotbox(16488.2,16285.8,512,512,0)]):
Output source list file name (example1_out.fits):
Output source cell image file name (example1_scell.fits):
Output reconstructed image file name (example1_image.fits):
Output normalized background file name (example1_defnbkg.fits):
unix%
```

Note that the parameter settings made previously using `pset` are shown when the prompt for the required parameters appears, and that these settings should be accepted by hitting the `<RETURN> key`.

The results are shown in Figure 11.5.

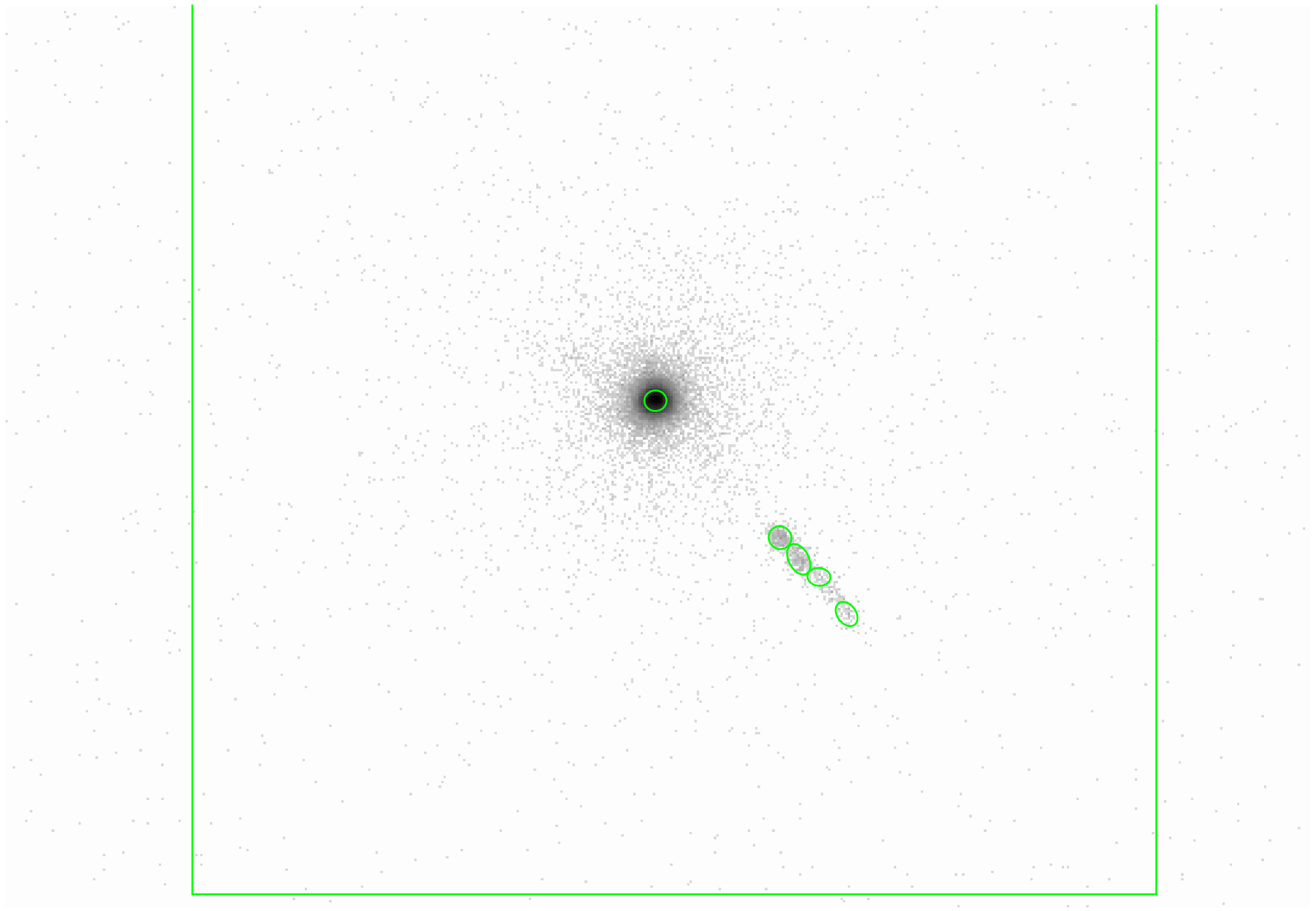

Figure 11.5: Detecting close extended sources with *wavdetect*. The HRC-I dataset of 3C 273, with ellipses showing the *wavdetect* results. The box shows the region of the chip searched by *wavdetect*.

11.4.2 Example 2: Detecting Close Extended Sources using Additional Scales

he previous example is repeated, using additional *wavdetect* scales. Increasing the number of scales means that additional transforms will be computed.

Setting Parameters

```
unix% punlearn wavdetect
unix% pset wavdetect infile = \
"data/hrcf00461N003_evt2.fits[EVENTS][(x,y)=rotbox(16488.2,16285.8,512,512,0)]"
unix% pset wavdetect outfile = example2_out.fits
unix% pset wavdetect scellfile = example2_scell.fits
unix% pset wavdetect imagefile = example2_image.fits
unix% pset wavdetect defnbkgfile = example2_defnbkg.fits
unix% pset wavdetect regfile = example2_out.reg
unix% pset wavdetect ellsigma = 2
unix% pset wavdetect scales = "1.0 2.0 4.0 8.0 16.0"
```

The `scales` parameter is changed to use five different wavelet radii.

Executing *wavdetect*

```
unix% wavdetect
Input file name (data/hrcf00461N003_evt2.fits[EVENTS][(x,y)=rotbox(16488.2,16285.8,512,512,0)]):
Output source list file name (example2_out.fits):
Output source cell image file name (example2_scell.fits):
Output reconstructed image file name (example2_image.fits):
Output normalized background file name (example2_defnbkg.fits):
unix%
```

The results are shown in Figure 11.6.

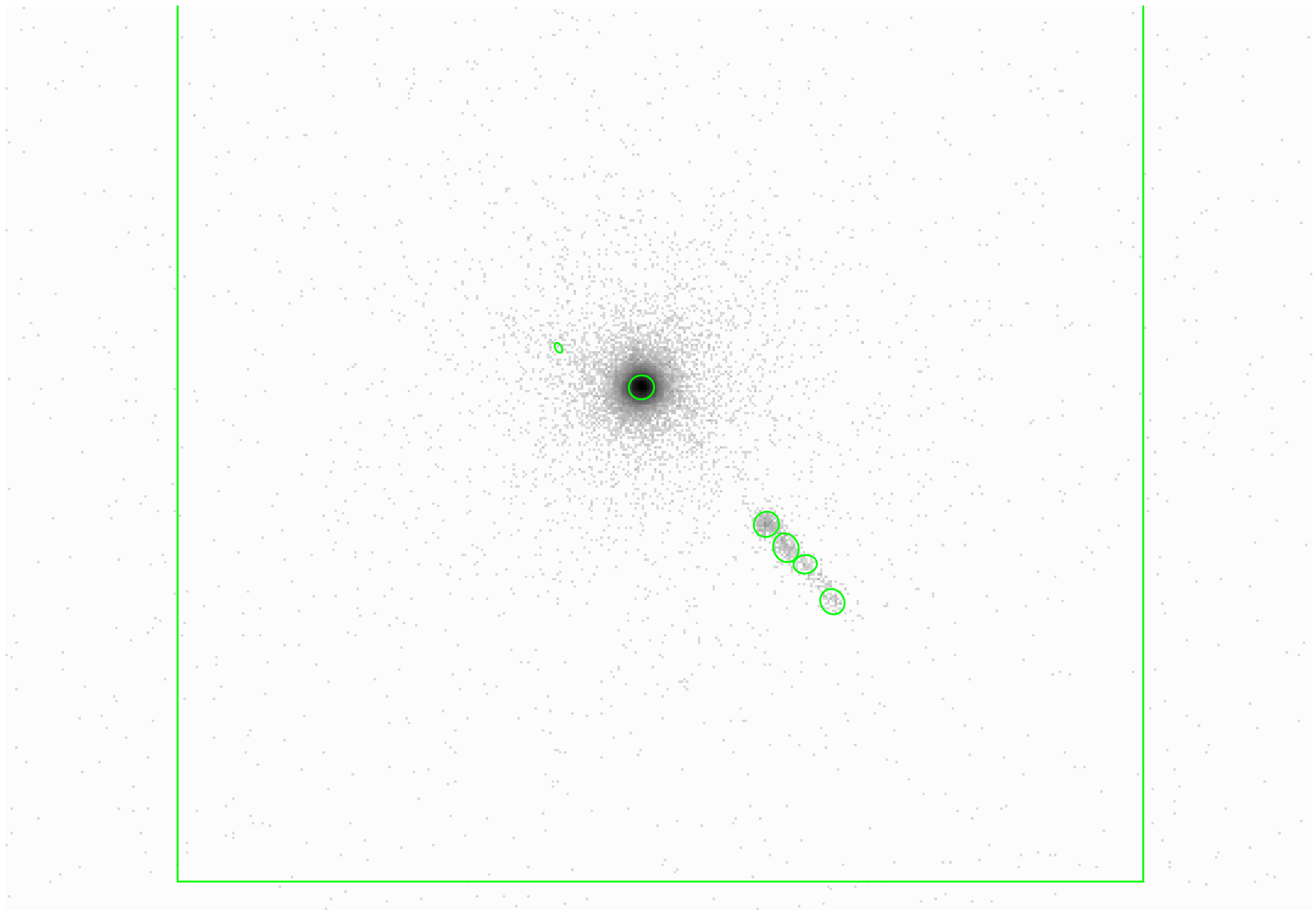

Figure 11.6: Detecting close extended sources with additional scales in *wavdetect*. The HRC-I dataset of 3C 273, with ellipses showing the *wavdetect* results. The box shows the region of the chip searched by *wavdetect*. Notice the detection to the Northwest of the core; this was "missed" with the default scales (Figure 11.5), which did not include a scale of 1.0.

Chapter 12

Advanced wavdetect: Using wtransform & wrecon

12.1 General Discussion

wavdetect has two parts, each of which can be run as a standalone tool:

1. *wtransform* performs the convolutions between the data and the various scaled wavelet functions;
2. *wrecon* uses the *wtransform* output products to construct source lists, measure parameters for each detection, and create various maps.

There are a few advantages to running the tools separately.

- In *wavdetect*, the flux scales chosen to define the source properties are computed automatically by selecting which of the scales supplied by the user best matches the size of the point spread function (PSF) at that location on the detector. With *wrecon* however, the user has control over which of the *wtransform* wavelet scales should be included. The ability to select flux scales means that you are not tied to optimizing for unresolved structures.
- In many instances, one may want to perform the convolution for a set of wavelet scales and then use those results to construct source lists based on different selection criteria. Instead of running *wavdetect* multiple times, one can run *wtransform* once, and then perform repeated runs of *wrecon* with altered conditions.
- The Mexican Hat functions in *wavdetect* are circularly symetric, so there is a single `scales` parameter. In *wtransform*, however, the `xscales` and `yscales` parameters allows one to specify elliptical wavelet functions.

12.2 Running wtransform

This example illustrates running *wavdetect* using a *Chandra* HRC-I dataset of 3C 273 as input; see Chapter 2, Section 2.3 for further information about this dataset.

Herein we run *wtransform* with 5 scales, using autonaming for most of the output files. We then make two runs with *wrecon* to demonstrate how the output source properties change.

12.2.1 wtransform Input

The input parameters are set as follows:

```
unix% punlearn wtransform
unix% pset wtransform infile = \
      "data/hrcf00461N003_evt2.fits[EVENTS][sky=circle(16400,16400,1000)]"
unix% pset wtransform srclist = "."
unix% pset wtransform correl = "."
unix% pset wtransform nbkg = "."
unix% pset wtransform log = yes
unix% pset wtransform verbose = 2
unix% pset wtransform thresh = "."
unix% pset wtransform xscales = "1.0 2.0 4.0 8.0 16.0"
unix% pset wtransform yscales = "1.0 2.0 4.0 8.0 16.0"
unix% pset wtransform clobber = yes
```

A dot (".") indicates that autonaming will be used, as explained in Section 1.3.2.

The *wtransform* tool is then run:

```
unix% wtransform
input file (data/hrcf00461N003_evt2.fits[EVENTS][sky=circle(16400,16400,1000)]):
Source list output stack (.):
Correlation image output stack (.):
Normalized background image output stack (.):
Warning: Keyword TLMVER is not in the header.
Warning: Keyword HDUSPEC is not in the header.
Warning: Keyword HDUDOC is not in the header.
Warning: Keyword HDUVERS is not in the header.
Warning: Keyword BTIMNULL is not in the header.
Warning: Keyword BTIMRATE is not in the header.
Warning: Keyword BTIMDRFT is not in the header.
Warning: Keyword BTIMCORR is not in the header.
Warning: Keyword FLSHTIME is not in the header.
Warning: Keyword OBI_NUM is not in the header.
Warning: Keyword READMODE is not in the header.
Warning: Keyword RA_PNT is not in the header.
Warning: Keyword DEC_PNT is not in the header.
```

```
Warning: Keyword ROLL_PNT is not in the header.
```

12.2.2 wtransform Output

A number of output files are created:

```
unix% ls -1
hrcf00461N003_correl_stk.fits[EVENTS][sky=circle(16400,16400,1000)]
hrcf00461N003_evt2_010_010.fits
hrcf00461N003_evt2_010_010_correl.fits
hrcf00461N003_evt2_010_010_nbkg.fits
hrcf00461N003_evt2_010_010_thresh.fits
hrcf00461N003_evt2_020_020.fits
hrcf00461N003_evt2_020_020_correl.fits
hrcf00461N003_evt2_020_020_nbkg.fits
hrcf00461N003_evt2_020_020_thresh.fits
hrcf00461N003_evt2_040_040.fits
hrcf00461N003_evt2_040_040_correl.fits
hrcf00461N003_evt2_040_040_nbkg.fits
hrcf00461N003_evt2_040_040_thresh.fits
hrcf00461N003_evt2_080_080.fits
hrcf00461N003_evt2_080_080_correl.fits
hrcf00461N003_evt2_080_080_nbkg.fits
hrcf00461N003_evt2_080_080_thresh.fits
hrcf00461N003_evt2_160_160.fits
hrcf00461N003_evt2_160_160_correl.fits
hrcf00461N003_evt2_160_160_nbkg.fits
hrcf00461N003_evt2_160_160_thresh.fits
hrcf00461N003_nbkg_stk.fits[EVENTS][sky=circle(16400,16400,1000)]
hrcf00461N003_src_stk.fits[EVENTS][sky=circle(16400,16400,1000)]
hrcf00461N003_thresh_stk.fits[EVENTS][sky=circle(16400,16400,1000)]
wtransform.log
```

As can be seen, for each scale size there are a number of output files:

- `hrcf00461N003_evt2_xscale#_yscale#.fits`: the locations of correlation maxima.
- `hrcf00461N003_evt2_xscale#_yscale#_correl.fits`: maps of the convolution of the Mexican Hat wavelet with the data; see Figure 12.1.
- `hrcf00461N003_evt2_xscale#_yscale#_nbkg.fits`: the normalized background maps.
- `hrcf00461N003_evt2_xscale#_yscale#_thresh.fits`: optional output maps which give the S/N available at each location.

The four output "stack" files (ASCII lists of the relevent files) have adopted a filename which includes the DM filter (*e.g.* `hrcf00461N003_correl_stk.fits[EVENTS][sky=circle(16400,16400,1000)]`). This is caused by the use of autonaming and is generally undesirable. It can be avoided by not using autonaming, or rename the files after the fact:

```
unix% mv 'hrcf00461N003_correl_stk.fits[EVENTS][sky=circle(16400,16400,1000)]' \
      hrcf00461N003_correl_stk.fits
unix% mv 'hrcf00461N003_nbkg_stk.fits[EVENTS][sky=circle(16400,16400,1000)]' \
      hrcf00461N003_nbkg_stk.fits
unix% mv 'hrcf00461N003_src_stk.fits[EVENTS][sky=circle(16400,16400,1000)]' \
      hrcf00461N003_src_stk.fits
unix% mv 'hrcf00461N003_thresh_stk.fits[EVENTS][sky=circle(16400,16400,1000)]' \
      hrcf00461N003_thresh_stk.fits

unix% more hrcf00461N003_correl_stk.fits
hrcf00461N003_evt2_010_010_correl.fits
hrcf00461N003_evt2_020_020_correl.fits
hrcf00461N003_evt2_040_040_correl.fits
hrcf00461N003_evt2_080_080_correl.fits
hrcf00461N003_evt2_160_160_correl.fits
```

12.3 Running wrecon

12.3.1 wrecon Input

wrecon is set up to use some of the *wtransform* output files:

```
unix% punlearn wrecon
unix% pset wrecon infile = \
      "data/hrcf00461N003_evt2.fits[EVENTS][sky=circle(16400,16400,1000)]"
unix% pset wrecon sourcefile = src1.fits
unix% pset wrecon scellfile = scell1.fits
unix% pset wrecon imagefile = image1.fits
unix% pset wrecon regfile = flux1.reg
unix% pset wrecon clobber = yes
unix% pset wrecon srclist = hrcf00461N003_src_stk.fits
unix% pset wrecon correl = hrcf00461N003_correl_stk.fits
unix% pset wrecon nbkg = hrcf00461N003_nbkg_stk.fits
unix% pset wrecon defnbkgfile = defnbkgfile1.fits
unix% pset wrecon fluxfile = flux1.stk
unix% pset wrecon xscales = "1.0 2 4 8 16"
unix% pset wrecon yscales = "1.0 2 4 8 16"
unix% pset wrecon fluxscales = 1
unix% pset wrecon log = yes
unix% pset wrecon verbose = 2
```

The *wrecon* tool is then run:

```
unix% wrecon
Input file name (data/hrcf00461N003_evt2.fits[EVENTS][sky=circle(16400,16400,1000)]):
```

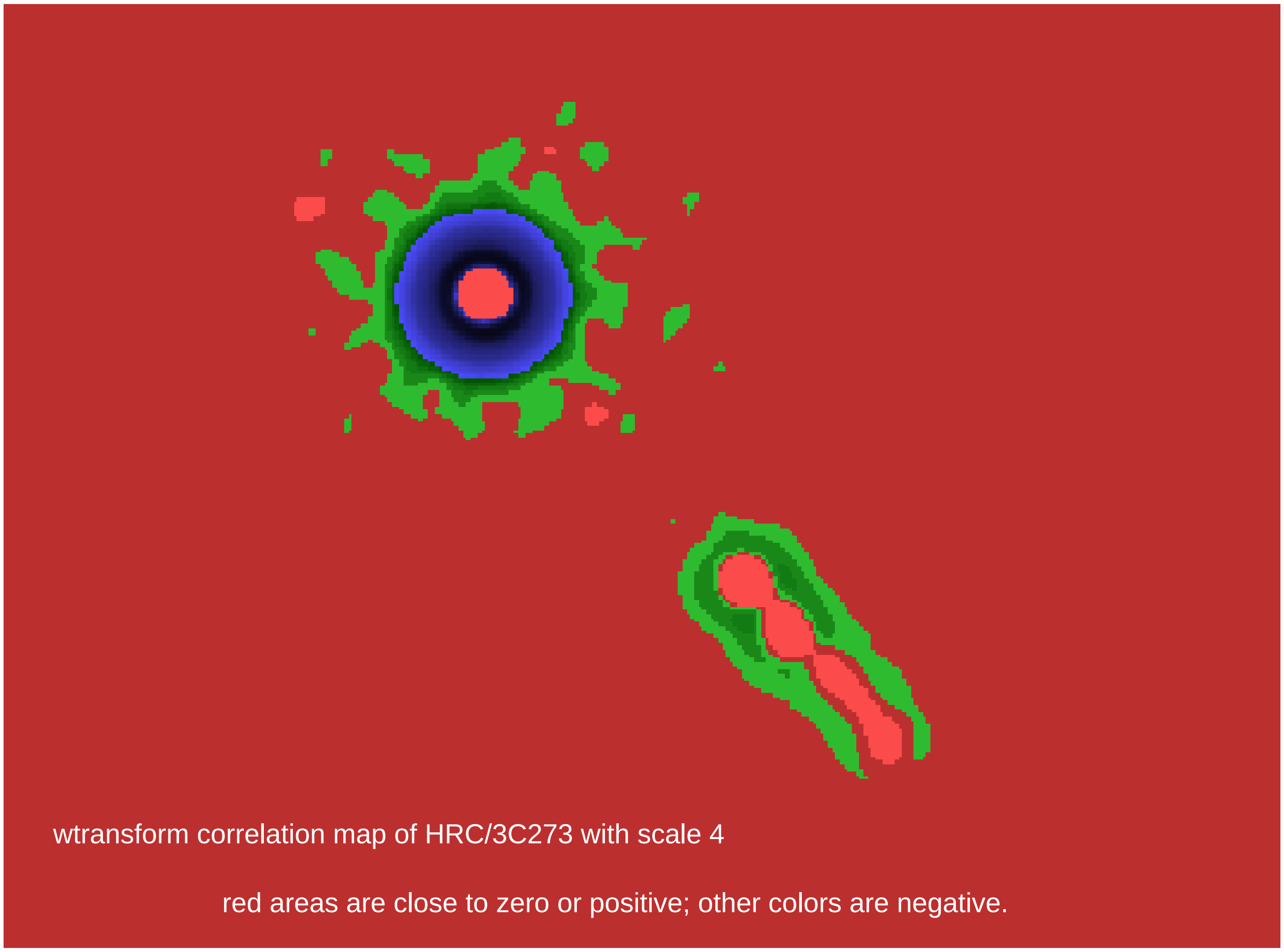

Figure 12.1: A correlation map produced by *wtransform*for `scale = 4` pixels. The 3C 273 jet stands out. Red areas are positive (or just slightly negative), while the other colors correspond to significantly negative values. The map peaks around 200,000 and has negative regions as low as -25,000.

```
Output source list file name (src1.fits):
Warning: Keyword TLMVER is not in the header.
Warning: Keyword HDUSPEC is not in the header.
Warning: Keyword HDUDOC is not in the header.
Warning: Keyword HDUVERS is not in the header.
Warning: Keyword BTIMNULL is not in the header.
Warning: Keyword BTIMRATE is not in the header.
Warning: Keyword BTIMDRFT is not in the header.
Warning: Keyword BTIMCORR is not in the header.
Warning: Keyword FLSHTIME is not in the header.
Warning: Keyword OBI_NUM is not in the header.
Warning: Keyword READMODE is not in the header.
Warning: Keyword RA_PNT is not in the header.
Warning: Keyword DEC_PNT is not in the header.
Warning: Keyword ROLL_PNT is not in the header.
Output source cell image file name (scell1.fits):
Output image file name (image1.fits):
unix%
```

Next, we repeat the *wrecon* run, changing only the output file names and the `fluxscale` parameter (from `1` to `"3 4"`). The logfile from the first run (`wrecon.log`) is also renamed so that it does not get overwritten.

```
unix% mv wrecon.log wrecon1.log
unix% pset wrecon sourcefile = src2.fits
unix% pset wrecon scellfile = scell2.fits
unix% pset wrecon imagefile = image2.fits
unix% pset wrecon regfile = flux2.reg
unix% pset wrecon defnbkgfile = "defnbkgfile2.fits"
unix% pset wrecon fluxfile = flux2.stk
unix% pset wrecon fluxscales = "3 4"
unix%
unix% wrecon
...
unix% mv wrecon.log wrecon2.log
unix%
```

12.3.2 wrecon Output

The first run of *wrecon* produced:

```
defnbkgfile1.fits
flux1.reg
flux1.stk
image1.fits
scell1.fits
src1.fits
wrecon1.log
```

and the second run produced:

```
defnbkgfile2.fits
flux2.reg
flux2.stk
image2.fits
scell2.fits
src2.fits
wrecon2.log
```

Descriptions of the output files of *wrecon* are given in the *wavdetect* discussions (Chapters 11 and 14).

12.3.3 Comparing Two Runs of wrecon

Both runs made the same detections since they were based on the results from the same run of *wtransform*. However, the source properties, including location and counts, depend on the size and location of the source cell, so the actual values between the two *wrecon* runs differ somewhat. We have selected a few columns of interest from the output source lists, and show them here to demonstrate the sort of differences that occur:

- `POS(X,Y)`: the pixel coordinates of the centroid of the distribution within the cell.
- `NPIXSOU`: number of pixels in the cell.
- `NET_COUNTS`: net counts within the cell.
- `R[2]`: the major and minor major axes of the count distribution within the cell, multiplied by the `ellsigma` parameter value.

1. Source Properties from `fluxscale = 1`

```
unix% dmlist "src1.fits[cols POS,NPIXSOU,NET_COUNTS,SRC_SIGNIFICANCE,R]" opt=data
--------------------------------------------------------------------------------
Data for Table Block SRCLIST
--------------------------------------------------------------------------------
ROW  POS(X,Y)           NPIXSOU  NET_COUNTS  SRC_SIGNIFICANCE  R[2]

1   (16478.13,16291.77) 806      213089.18   2627.78           [10.09 9.60]
2   (16544.38,16219.29) 170         395.94     72.64           [ 8.87 8.08]
3   (16556.39,16206.0)   67         146.50     39.68           [ 7.34 5.01]
4   (15914.40,16323.60)  16           4.93      2.59           [ 3.10 1.37]
5   (16565.68,16196.76)  37          35.42     12.54           [ 5.04 3.76]
6   (16580.92,16177.14)  33          26.67     10.92           [ 5.38 2.95]
7   (16417.0 ,16691.0)    5           0.97      0.51           [ 0    0]
```

2. Source Properties from `fluxscale = "3 4"`

```
unix% dmlist "src2.fits[cols POS,NPIXSOU,NET_COUNTS,SRC_SIGNIFICANCE,R]" opt=data
--------------------------------------------------------------------------------
```

```
Data for Table Block SRCLIST
--------------------------------------------------------------------------------
ROW  POS(X,Y)            NPIXSOU  NET_COUNTS  SRC_SIGNIFICANCE  R[2]

1   (16478.13,16291.76) 1244     213218.03   2405.44           [10.51 10.03]
2   (16544.58,16219.00)  497        498.73     60.22           [12.05 11.10]
3   (16554.74,16206.77)  385        352.63     50.30           [12.63 10.06]
4   (15913.19,16323.80)  342         19.85      8.36           [10.05  8.08]
5   (16567.28,16194.35)  385        150.90     25.21           [14.30 12.05]
6   (16580.52,16178.00)  410        100.61     20.05           [12.16 11.50]
7   (16417.40,16690.40)  103          4.40      2.03           [ 6.31  4.29]
```

The regions are shown in Figure 12.2.

The cell maps are shown in Figures 12.3 and 12.4. Note how the cell sizes are larger for the jet detections when using larger flux scales.

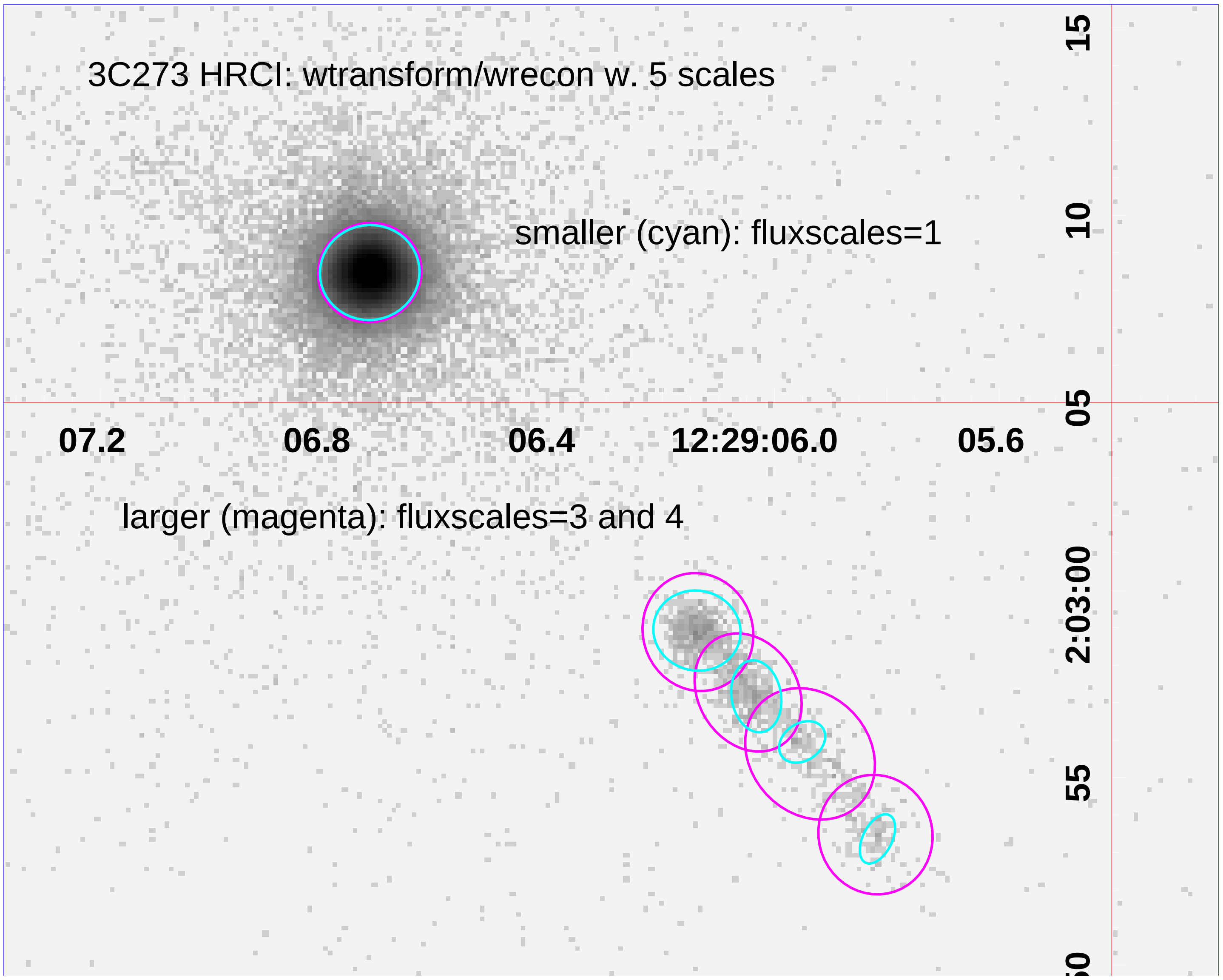

Figure 12.2: The region files produced by two runs of *wrecon*. For the bright core of 3C 273, the regions are essentially the same size, but they differ for the jet detections. The smaller regions (cyan/blue) are for `fluxscales = 1`, and the larger regions (magenta) are for `fluxscales = "3 4"`.

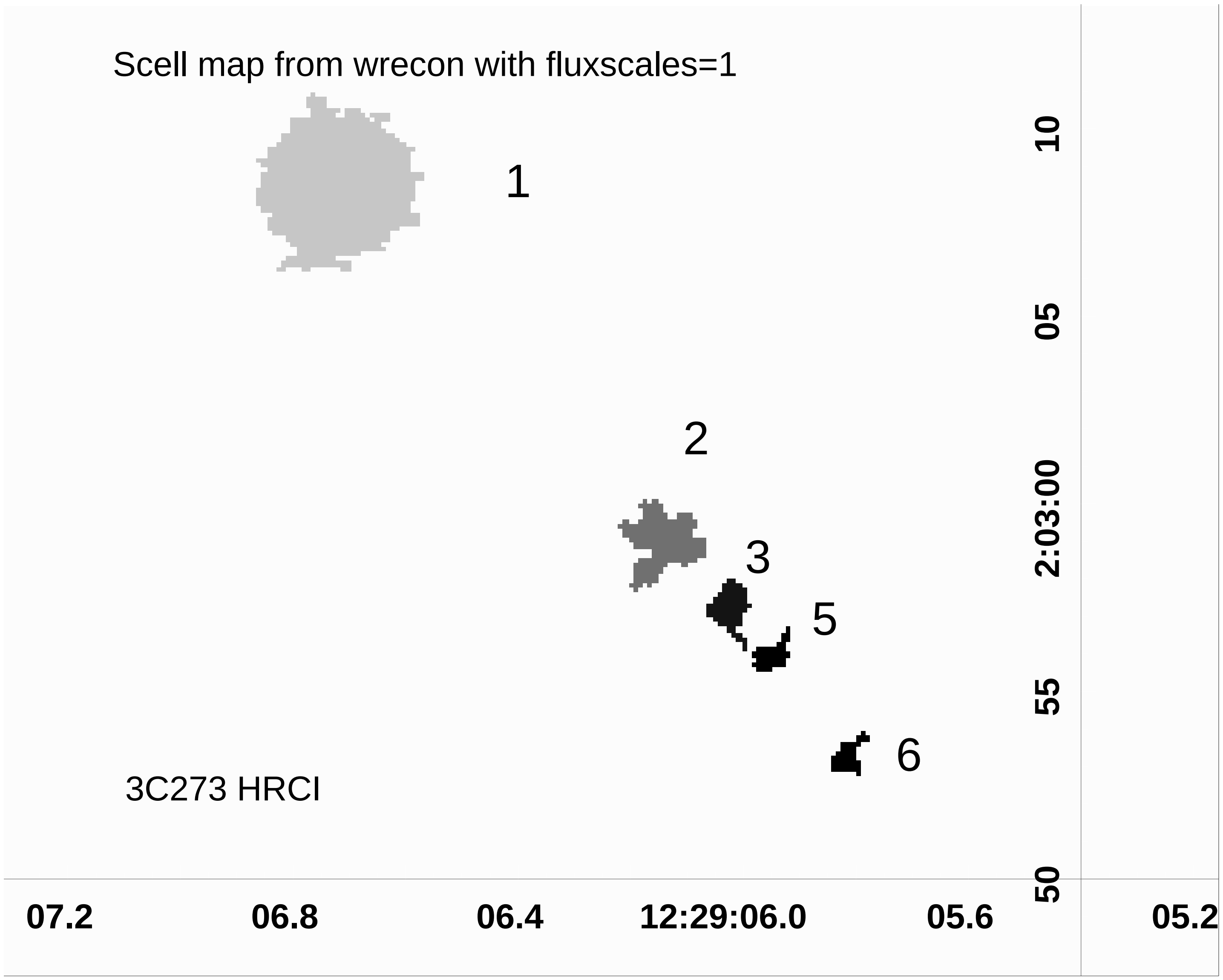

Figure 12.3: The source cells for 3C 273 with `fluxscales = 1`. The map values denote the detection number.

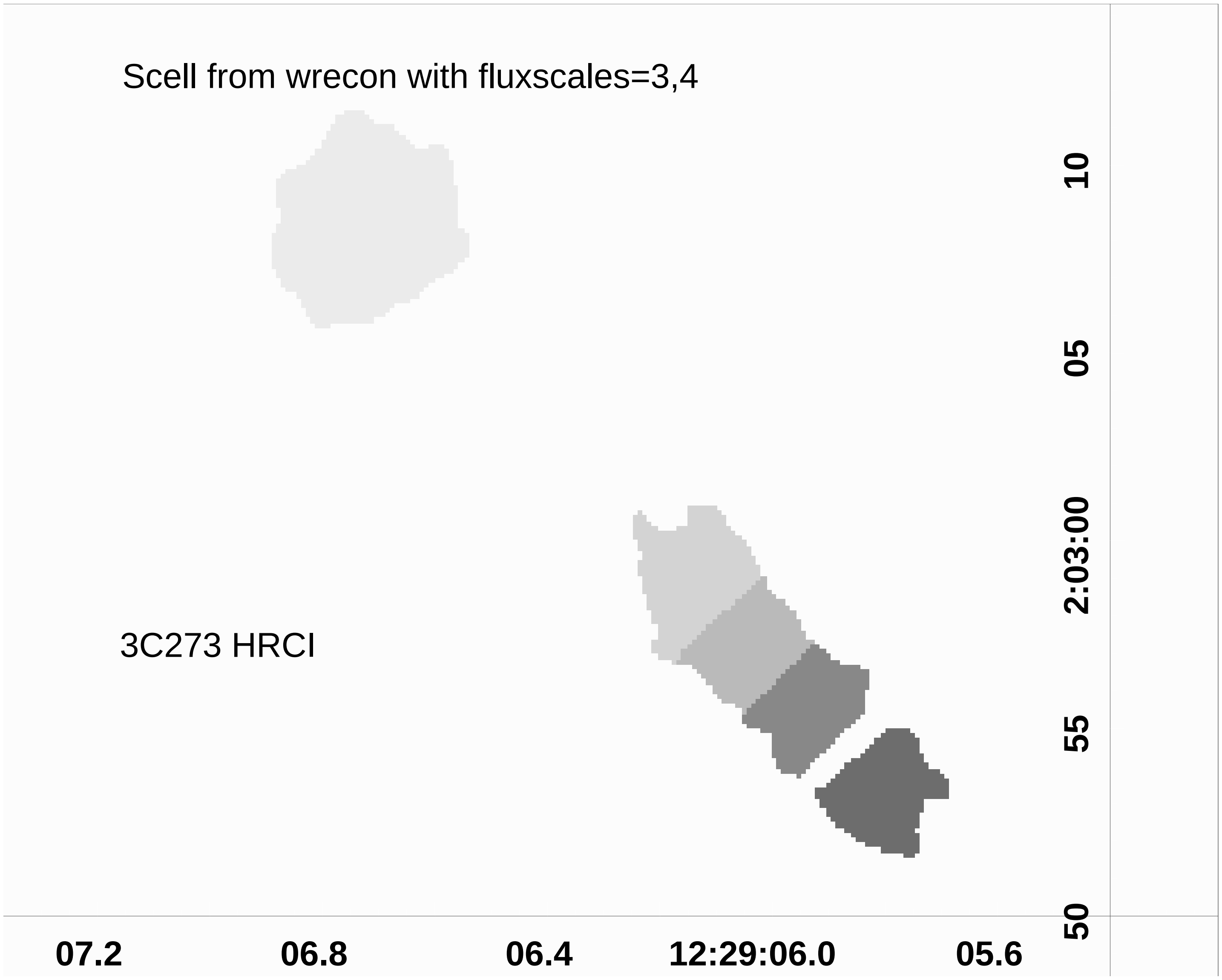

Figure 12.4: The source cells for 3C 273 with `fluxscales = "3 4"`. Compare the cell sizes to those in Figure 12.3.

Chapter 13

wavdetect Theory

This chapter is a draft copy of the paper "A Wavelet-Based Algorithm for the Spatial Analysis of Poisson Data" by P.E. Freeman, V. Kashyap, R. Rosner, D.Q. Lamb, *Ap. J. Suppl. Series*, Vol. 138, p. 185, 2002. The published paper is available online via `astro-ph`: `http://arXiv.org/abs/astro-ph/0108429` .

13.1 Source Detection Using Wavelet Functions

A wavelet is a member of a family of oscillatory scaleable functions which deviates from zero within a limited spatial regime, and has zero normalization. Perhaps the simplest example of such a function is a positively valued "top hat," with amplitude A and width w, flanked by two negatively valued troughs with total integrated area $-Aw$. (This in fact describes the Haar wavelet family.) Any function with zero normalization satisfying the property

$$W_{a,\sigma}(x) \equiv \frac{1}{\sigma} W\left(\frac{x-a}{\sigma}\right) \tag{13.1}$$

may be used as a wavelet function. σ is the *scaling* or *dilation* parameter and a is the *translation* parameter.

The importance of wavelet functions only began to be recognized in early 1980s, when they were used in the analysis of seismic data (Goupillaud *et al.* 1984). Wavelets are now used as tools in atomic physics, the study of fractal structures, time series, sound, and image analysis, *etc.* Slezak, Bijaoui, & Mars (1990) were the first to apply wavelets in astronomy, applying them to galaxy catalogue data to search for statistically significant clusters and voids. The interested reader may find more information on wavelets in Daubechies (1992) or Holschneider (1995).

Wavelet transforms, convolutions of an arbitrary analyzable function with a set of suitably scaled wavelet functions, provides a superior means to detect and characterize sources in an image, compared with classical "sliding-box"[1] and Fourier methods. Because a wavelet is a function with limited spatial extent, it can be used to determine source locations, like the sliding-box. Unlike the sliding-box, however, it can also be used to identify the dominant frequencies in the analyzed function, *i.e.* to characterize source shape and extent. Such characterization could also be done with Fourier analysis, but unlike Fourier methods, wavelets

[1]The method used in the *Chandra* source detection routine `celldetect`.

determine *localized* frequency information, as a consequence of their localization in both the spatial and Fourier domains.

The program *wtransform* applies wavelets to the problem of detecting sources in X-ray data, and the associated program *wrecon* analyzes the detected sources and constructs the source list. While others have developed codes with similar analysis goals (*e.g.* Vikhlinin, Forman, & Jones 1994; Rosati *et al.* 1995; Grebenev, Forman, & Jones 1995), *wtransform* and *wrecon* are considerably more complex, *e.g.* correcting for exposure variation across the field of view (FOV), and making error estimates, while not requiring that the point spread function (PSF) have a particular functional form (which is particularly important for analysis of *Chandra* images). Another program of similar complexity but using different methods is the `palermo wdetect` program of Damiani *et al.* (1997).

In §13.2 we describe the basic recipe one would follow to detect sources in data, without reference to any particular wavelet function. We do this is to underscore the fact that the basic details of the method are not dependent on the choice of wavelet function, so long as that function is simple, with one centralized positive amplitude mode. The fact that the PSF of X-ray detectors often has Gaussian-like shape motivates our use of the Marr wavelet, or "Mexican Hat" (MH) function, which we describe in Appendix 13.5.

The program *wtransform* uses the correlation of the MH function with a given image D to identify putative source pixels. In §13.3 we describe in detail the steps that *wtransform* follows, in particular discussing how the program estimates the local background at each pixel, how it corrects for the systematic shift that a pixel's correlation value will take if the exposure varies within the limited spatial extent of the wavelet function, and how it performs error analysis. In appendices which relate to §13.3, we present derivations of analytic quantities related to the MH function (Appendix 13.6), as well as describe the simulations used to estimate the threshold correlation value for source detection in each pixel (Appendix 13.7). In §13.4 we discuss in detail how the source list is constructed by *wrecon*, and in an appendix we describe how *wrecon* computes a default background image for use in determining source properties (Appendix 13.8).

13.2 Basic Algorithm

The algorithm by which one uses a simple unimodal wavelet function $W(\sigma_x, \sigma_y, x, y)$ to detect sources in a two-dimensional image D is conceptually simple.[2] This function is convolved with D to produce a "correlation image" C:

$$C(\sigma_x, \sigma_y, x, y) \;=\; \int\!\!\int dx'\; dy'\; W(\sigma_x, \sigma_y, x - x', y - y')\; D(x', y') \equiv\; < W{*}D > . \qquad (13.2)$$

(Note that the data are assumed to have units counts, and not counts per second.) The expectation value of $C(\sigma_x, \sigma_y, x, y)$ is zero, if there are no sources within the limited spatial extent of the wavelet function, and the background count rate is locally constant, because the normalization of $W(\sigma_x, \sigma_y, x, y)$ is zero.

A clump of counts will manifest itself as a local maximum in a correlation image if the scale sizes (σ_x, σ_y) are approximately the same as, or larger than, the size of the clump. One can see why this is so more easily if we write $C \;=\; < PW{*}D > + < NW{*}D >$, where PW and NW denote the positive and negative amplitude portions of the wavelet function, respectively. If the clump is contained completely within PW, then the contribution of the positive term to C outweighs that of the negative term, producing a maximum.[3] If the

[2]The current version of *wtransform* works only with binned images, though it may be updated in the future to also work with event lists.

[3]By writing C in this way, we also note that the term $< NW{*}D >$, which is ≤ 0 by definition, acts to subtract the contribution of the local background around the clump from the value of C. We return to this point in §13.3.

scale sizes are smaller, then the wavelet function will extend over a smaller region within the clump, within which the inferred counts amplitude will tend more and more to a constant. Hence $C \to 0$. For larger scale sizes, the correlation value tends asymptotically to a maximum, $C_{\rm max}$.

Source detection boils down to the question: how large must the value $C(\sigma_x, \sigma_y, x, y)$ become before we may safely accept that a source has contributed counts to the observed clump? To answer this, we compute the Type I error, or the probability that we would erroneously accept a source when the null hypothesis, that there is no source, is correct. We compute this quantity, also called the significance, in each image pixel (i, j):

$$S_{i,j} \ = \ \int_{C_{i,j}}^{\infty} dC p(C|n_{B,i,j}). \tag{13.3}$$

$n_{B,i,j}$ is the inferred number of background counts within the limited spatial extent of the wavelet function, and $p(C|n_{B,i,j})$ is the probability sampling distribution for C given $n_{B,i,j}$. By definition, $0 \leq S_{i,j} \leq 1$; if a source is present, $S_{i,j} \to 0$. If $S_{i,j} \leq S_o$, a user-defined threshold (*e.g.* 10^{-6}), pixel (i, j) is identified as a source pixel. (Note that this identification may be made for any correlation image pixel, not just those which are local maxima in correlation space.)

If $n_{B,i,j}$ is estimated from the raw data themselves, this estimate will be biased if source counts are present, so that $S_{i,j} \gtrsim S_{i,j,{\rm true}}$. Thus an iterative procedure is used to remove source counts from the image:

- estimate $B_{i,j}$, the number of background counts in pixel (i, j), and compute $C_{i,j} =< W{*}D >_{i,j}$;
- compute $S_{i,j}$;
- determine which pixels are associated with (the strongest) sources;
- replace $D_{i,j}$ with $B_{i,j}$ in those pixels, and call the new data image D';
- estimate $B'_{i,j}$ using D', and compute $C'_{i,j} =< W{*}D' >_{i,j}$;
- compute $S'_{i,j}$;
- determine which pixels are associated with (weaker) sources;
- *etc.*

One continues to iterate until $B_{i,j}^{\rm final}$ is determined. (The usual number of iterations depends upon many factors, but is usually $\approx$ 3-4.) With this final background estimate, one computes a final significance $S_{i,j}^{\rm final}$ for each value $C_{i,j} =< W{*}D >_{i,j}$ (the correlation of the wavelet function with the *raw* image data) so that a final listing of source pixels may be made.

After this algorithm is used to determine lists of source pixels for many wavelet scale sizes, cross-identification of pixels across scales is performed to create the final source list. This cross-identification allows the weeding out of spurious sources which may be detected even if a high significance threshold is used.

13.3 Source Pixel Identifications

13.3.1 Background Estimation

In §13.2, we noted how the condition $< NW{*}D > \le 0$ can be interpreted to mean that the NW performs a "background subtraction." It is thus natural to use a function of $< NW{*}D >$ to estimate B in pixel (i, j), if it is *a priori* unknown:

$$B_{i,j} \;=\; \frac{1}{N_{i,j}} < NW{*}D >_{i,j} \; . \tag{13.4}$$

$N_{i,j}$ is the normalization. Variations in exposure within the limited spatial extent of the wavelet function will affect the estimate of $N_{i,j}$, which we estimate as $N_{i,j} = \frac{<NW*E>_{i,j}}{E_{i,j}}$. (If exposure variations may be safely ignored, then $N_{i,j}$ can be computed analytically; see Appendix 13.6.) The normalized (or flat-field) background $B_{i,j,\mathrm{norm}}$ is directly related to $B_{i,j}$:

$$B_{i,j} \;=\; E_{i,j}\frac{< NW{*}D >_{i,j}}{< NW{*}E >_{i,j}} \;=\; E_{i,j}B_{i,j,\mathrm{norm}} \tag{13.5}$$

$B_{i,j,\mathrm{norm}}$ is an estimate of the number of counts that would be detected if there were no exposure variations; it differs from the number of photons seen from a similarly sized region of sky by a factor related to the efficiencies of the mirror assembly and detector. Note that $B_{i,j,\mathrm{norm}}$ is set to zero in pixels with exposure less than a user-specified threshold.

We stress that one should not interpret $B_{i,j}$ blindly as the actual background amplitude $B_{i,j,\mathrm{true}}$; it should only be viewed as a computation made in order to select source pixels. For instance, if a small wavelet scale is applied in a region with a large PSF, then $B_{i,j} \gg B_{i,j,\mathrm{true}}$ around any source, because the estimate of $B_{i,j}$ is unavoidably biased by the source counts which lie in the region spanned by NW. In *wrecon*, we provide a default method of combining information across scales to estimate $B_{i,j,\mathrm{true}}$ (which we discuss in Appendix 13.8); it is this quantity which is used to help determine source properties.

13.3.2 Correlation Image

Analytic Correlation. We detect sources in the image D by correlating these data with the wavelet function:

$$\begin{aligned} C_{i,j} &= \; < W{*}D >_{i,j} \\ &= \sum_{i'}\sum_{j'}\left[\int_{\Delta x'}\int_{\Delta y'} dx'dy'W(\sigma_x,\sigma_y,x',y')\right]D_{i',j'} \qquad (13.6) \\ &= \sum_{i'}\sum_{j'} W_{i-i',j-j'}D_{i',j'} \qquad (13.7) \end{aligned}$$

The wavelet is centered in pixel (i, j) (*e.g.* (x, y) = [0.5,0.5] for pixel (i, j) = [1,1]), and $\Delta x'$ and $\Delta y'$ represent the limits of integration over pixel (i', j'). The integral in Eq. (13.6) can be solved analytically if the MH function (Eq. [13.37]) is used; we present this solution in Appendix 13.6. For the MH function, summation over i' and j' within an ellipse with semi-major and minor axes of length $\approx 5\sigma$ is sufficient to determine $C_{i,j}$ to high accuracy.

Calculation using the Fast Fourier Transform. Analytic integration may be too computationally intensive if the image or scale size is large. In these situations, we invoke the Correlation Theorem, which allows us to use Fast Fourier Transforms (FFTs) to compute C:

$$C \;\approx\; \mathrm{FFT}^{-1}[N \times \mathrm{FFT}(W) \times \mathrm{FFT}^{*}(D)]. \tag{13.8}$$

The right side of this equation may be read as follows: "the inverse FFT of the product of a normalization factor N, the FFT of W, and the complex conjugate of the FFT of D." To avoid problems associated with FFT aliasing, we pad the image D with zeros; an axis-length in the padded image is, in general, the next power of two larger than the length of that axis in the unpadded image (*e.g.* for an image of size 1024×1024, the padded image has size 2048×2048, with the contents of the unpadded image in the center). We compute the FFT using `realft` and associated routines of Press *et al.* (1992). For the particular case of the MH function, we may compute the Fourier Transform analytically; we present the derivation in Appendix 13.6.

Correcting for Exposure Variation

Variations in exposure within the limited spatial extent of the MH function, *e.g.* caused by the shadows of mirror supports and ribs, or by the edge of the FOV itself, will reduce the accuracy of the estimate of $C_{i,j}$. This quantity may be overestimated or underestimated, depending upon the location of pixel (i,j) relative to regions of reduced exposure, but an overestimate is more damaging, as it can lead to the detection of a spurious source. An overestimate will occur if the exposure reduction primarily occurs within NW. To see this, let us assume the counts in regions without exposure variation to be a constant, and that the exposure reduction causes a reduction in counts ΔD. Then:

$$\begin{aligned} C_{i,j} &= \;< PW{*}D >_{i,j} \;+\; < NW{*}(D - \Delta D) >_{i,j} \\ &= \;- < NW{*}\Delta D >_{i,j} \\ &> \;0 \end{aligned}$$

In this example, $C_{i,j}^{\mathrm{novar}} = 0$. Thus exposure variations can lead to the detection of spurious sources near shadows (but not within them) and near the FOV edge. Below, we present two methods for correcting the estimate $C_{i,j}$.

"Full" Exposure Correction. We represent the data as the sum of source counts S and background counts B:

$$C_{i,j} \;=\; < W{*}D >_{i,j} \;=\; < W{*}S >_{i,j} \;+\; < W{*}B >_{i,j} \tag{13.9}$$

We rewrite the second term in terms of the exposure E and normalized background B_{norm}:

$$< W{*}B >_{i,j} = < W{*}(EB_{\mathrm{norm}}) >_{i,j} . \tag{13.10}$$

We estimate what the correlation value would be if $E = E_{i,j}$ throughout the region spanned by the wavelet function:

$$C_{i,j,\mathrm{cor}} \;=\; C_{i,j} \;-\; < W{*}(EB_{\mathrm{norm}}) >_{i,j} \;+\; E_{i,j} < W{*}B_{\mathrm{norm}} >_{i,j} . \tag{13.11}$$

As noted above, $C_{i,j,\mathrm{cor}} \lesssim C_{i,j}$ if exposure reductions primarily occur in NW. As a default, E is set to one in each pixel, so that the edge-of-field correction may be made even if the exposure map is not read; this default array is then overwritten if the exposure map is entered. We note that the calculation of $< W{*}B_{\mathrm{norm}} >_{i,j}$ is prone to systematic error unless B_{norm} is extrapolated over regions of low exposure (where B_{norm} is set to zero) and beyond the edge of the FOV.[4]

[4]We "tile over" regions where the normalized background has been set to zero. We subdivide the image into smaller and smaller boxes: if in any one box the percentage of non-zero pixels is less than 20%, *wtransform* determines the average value in those pixels and assigns that value to the other pixels of the box. "Extrapolation" beyond the FOV is currently accomplished by simple mirror-image reflection.

The source counts, S, may also be altered by variations in exposure if the source is extended; however, correcting S requires knowledge of $S_{\rm norm}$, which is *a priori* unknown. Also, our prime motivation for correcting $C_{i,j}$ is to reduce the number of spurious sources, and correcting S does not help us in this regard.

"Fast" Exposure Correction. The full exposure correction requires *wtransform* to perform calculations during each iteration, since the value of $B_{i,j,\rm norm}$ will change from iteration to iteration until sources are cleansed from the image. However, we can make a "faster" exposure correction if we assume that $B_{\rm norm}$ is locally constant (which is often a safe assumption, especially if σ_x and σ_y are small). If this is the case, then $< W{*}B_{\rm norm} >_{i,j} = 0$, and

$$C_{i,j,\rm cor} \ = \ C_{i,j} \ - \ B_{i,j,\rm norm} < W{*}E >_{i,j} \ . \qquad (13.12)$$

This correction is deemed "fast" because *wtransform* calculates $< W{*}E >_{i,j}$ once, as opposed to once per iteration. To ensure accuracy, however, the full exposure correction is performed once the estimate of $B_{i,j,\rm norm}$ has stabilized.

Calculation of Significance

In §13.2, we show how the significance is computed in each pixel. However, for reasons discussed fully in Appendix 13.7, *wtransform* does not compute significances directly, but rather compares each correlation value $C_{i,j}$ with a threshold correlation value $C_{i,j,o}$. This threshold is found by estimating the lower bound on the integral in Eq. (13.3), given S_o, a user-specified significance threshold (*e.g.* 10^{-6}), and $q_{i,j} = 2\pi\sigma_x\sigma_y B_{i,j}$. If $C_{i,j} > C_{i,j,o}$, pixel (i,j) is identified as a possible source pixel.

In *wtransform*, the user actually specifies two threshold significance values: $S_{o,sc}$, for source cleansing, and S_o. During iterations, $S_{i,j}$ is compared to $S_{o,sc}$, which may be set to be much larger than S_o (*e.g.* 10^{-2}) to help reduce the bias caused by source counts in estimates of $B_{i,j}$.

13.3.3 Error Estimation

The analytic calculations of $C_{i,j}$ and $B_{i,j}$ involve the pixel-by-pixel summation of the independent Poisson-distributed random variable $D_{i,j}$. To compute the variances of these quantities, we use the general formula

$$V[Y] \ = \ V[\sum_m a_m X_m] \ = \ \sum_m a_m^2 V[X_m] + 2\sum_m \sum_{n>m} a_m a_n {\rm cov}[X_m, X_n]. \qquad (13.13)$$

where $Y = \sum_m a_m X_m$, and X_m are the random variables (see, *e.g.*, Eadie *et al.* 1971). If Y cannot be written simply as a linear function of random variables, we use the approximation formula

$$V[Y] \ \approx \ \sum_m \sum_n \frac{\partial Y}{\partial X_m}\frac{\partial Y}{\partial X_n}{\rm cov}[X_m, X_n] \qquad (13.14)$$

If Y depends directly on $D_{i,j}$, the covariance terms are zero, and the approximation formula simplies to the familiar formula $V[Y] \ \approx \ \sum_m \left(\frac{\partial Y}{\partial X_m}\right)^2 V[X_m]$.

Background. We rewrite Eq. (13.5) in terms of summations:

$$B_{i,j} \ = \ \frac{E_{i,j}}{< NW{*}E >_{i,j}} \sum_{i'} \sum_{j'} NW_{i-i',j-j'} D_{i',j'} \qquad (13.15)$$

The random variables $D_{i',j'}$ have variance $V[D_{i',j'}] = D_{i',j'}$.

$$\begin{aligned} V[B_{i,j}] &= \left(\frac{E_{i,j}}{< NW{*}E >_{i,j}}\right)^2 \sum_{i'}\sum_{j'} NW^2_{i-i',j-j'} D_{i',j'} & (13.16)\\ &= \left(\frac{E_{i,j}}{< NW{*}E >_{i,j}}\right)^2 < NW^2{*}D >_{i,j} & (13.17)\\ &= E^2_{i,j} V[B_{i,j,\mathrm{norm}}] \end{aligned}$$

This quantity is calculated only after the data are cleansed of sources. It may be calculated analytically (Eq. [13.16]) or using FFTs (Eq. [13.17]).

Correlation: No Exposure Correction.

$$\begin{aligned} V[C_{i,j}] &= V[\sum_{i'}\sum_{j'} W_{i-i',j-j'} D_{i',j'}]\\ &= \sum_{i'}\sum_{j'} W^2_{i-i',j-j'} D_{i',j'} & (13.18)\\ &= < W^2{*}D >_{i,j} \end{aligned}$$

This variance is computed once, after the first iteration, either analytically (Eq. [13.18]) or through the use of FFTs (Eq. [13.19]). To speed computation when using Fourier Transforms, we use the analytically-derived quantity $FT(W^2)$. Details on its derivation may be found in Appendix 13.6.

Correlation: Full Exposure Correction. Rewriting Eq. (13.11) in terms of summations and rearranging terms, we write the full exposure correction as:

$$\begin{aligned} C_{\mathrm{cor},i,j} &= \sum_{i'}\sum_{j'} \left[W_{i-i',j-j'} - \left(\sum_{i''}\sum_{j''} \frac{W_{i-i'',j-j''} NW_{i'-i'',j'-j''}(E_{i,j} - E_{i'',j''})}{< NW{*}E >_{i'',j''}} \right) \right] D_{i',j'}\\ &= \sum_{i'}\sum_{j'} a_{i,i',i'',j,j',j''} D_{i',j'} \end{aligned}$$

The variance is then

$$V[C_{\mathrm{cor},i,j}] = \sum_{i'}\sum_{j'} a^2_{i,i',i'',j,j',j''} D_{i',j'} \quad (13.19)$$

The presence of one summation over image pixels within another makes the computation time for any one given pixel ~ 1 CPU-minute. Thus in *wtransform* the covariance terms are currently ignored and the variance is computed as

$$V[C_{\mathrm{cor},i,j}] = V[C_{i,j}] + < W^2{*}(E^2 V[B_{\mathrm{norm}}]) >_{i,j} + E^2_{i,j} < W^2{*}V[B_{\mathrm{norm}}] >_{i,j} . \quad (13.20)$$

Initial tests indicate that this formula is accurate away from sources, but greatly *overestimates* the variance around sources. These tests indicate that the true variance around sources does not differ greatly from the true variance away from sources ($\lesssim 50\%$), so that this equation can provide the analyst a good estimate of the magnitude of the variances.

13.4 Constructing the Source List

The program *wrecon* is used to construct the final source list after *wtransform* has been run at all desired scales. To construct this list, it uses:

- lists of correlation space maxima whose amplitudes are greater than the threshold specified in *wtransform*, created at each wavelet scale;
- correlation images created by *wtransform* at each wavelet scale;
- and "flux images" computed within *wrecon* itself.

Next we discuss how the flux images are computed, then show how they are used to define the cells within the FOV that delineate the extent of a putative source and which allow us to estimate its properties.

13.4.1 Flux Image

The flux image is created by smoothing the raw data, then subtracting the normalized background from the smoothed image[5] on a pixel-by-pixel basis:

$$F_{i,j} \; = \; \max\left(\frac{< PW{*}D >_{i,j}}{< PW{*}E >_{i,j}} \; - \; B_{i,j,\mathrm{norm}}, 0\right). \tag{13.21}$$

We use PW as the smoothing function because it has the desirable properties of being localized, and, for the particular case of the MH function, of mimicking the shape of a canonical Gaussian PSF.

The scale pairs (σ_x, σ_y) at which to compute the flux images are user-specified; the number of pairs can be less than the number of wavelet scale pairs examined overall. Using fewer pairs reduces computation time, but may lead to a less-precise determination of source properties. Hereafter, we assume that $\sigma = \sigma_x = \sigma_y$, forcing symmetric wavelets.

The Source Cell

To determine source properties, it is necessary to define a "cell" on the detector, in which some portion of a source's counts are detected. If we choose the size and shape of the cell, then we need detailed knowledge of the PSF to estimate the fraction of a source's counts that are recorded in the cell. However, *wrecon* and the other *Chandra* detection algorithms assume no details about a detector's PSF except for its characteristic size $r_{\rm PSF}$, which is computed for *Chandra* data using the routine `psf_calc_size`; its functional form is unknown. But even if we did have detailed knowledge of the PSF, its use would provide a good estimate of the total source counts only if the source were point-like.

wrecon uses the flux images to define the source cell. In regions where there are no sources, the mean value of F is zero; only in the vicinity of sources does F deviate markedly from zero. Hence *wrecon* defines the source cell by

[5]The normalized background image is input by the user, or a default image is constructed from images made at each scale by *wtransform* (see Appendix 13.8).

- using the flux image computed at the scale σ_F closest to the PSF size, by minimizing the function $|\log_2 r_{\rm PSF} - \log_2 \sigma_{F,i}|$;
- determining the local maximum (i_F, j_F) in F-space corresponding to (i_C, j_C), the estimated location of the putative source;
- and determining those pixels for which (i_F, j_F) is the local maximum in F-space. Pixels for which $F_{i,j} = 0$ are not considered.

The use of $< PW{*}E >$ in Eq. (13.21) makes F a normalized quantity, which is beneficial since exposure variations can cause spurious flux-image maxima, which would reduce the cell size and adversely affect the determination of source properties. We stress, however, that the flux image is not used in the actual determination of source properties.

A cell defined in this manner has advantageous properties:

- there are no *a priori* assumptions about source shape and extent;
- saddle points in F-space allow the definition of boundaries between nearby sources;
- and the creation of the flux image involves smearing, or de-focusing, data, so that cell boundaries will lie outside the region where the source is evident to the eye. Hence, at least for isolated sources, nearly all source counts should have been detected within cell boundaries.

This approach is not always optimal when analyzing extended sources: such sources may have more than one local maximum in counts space, leading to multiple "sources" being listed. One way to counteract this is to set $\sigma_{F,\rm min} \gg r_{\rm PSF}$, as this will increase the probability of the extended object being associated with one maximum in F-space (which will ensure a sufficiently large cell). However, it also increases the probability that point sources in the vicinity of the extended source will be "swallowed up" by the larger source cell, so care must be exercised when interpreting derived source properties such as count rate. This approach is also not optimal if PSF is bimodal, as it is far off-axis for *Chandra*; in this case, *wtransform* will output two correlation maxima for a well-sampled source, and *wrecon* will characterize each maximum independently of the other. Only with the *a posteriori* application of a detailed PSF can the two "sources" be properly combined.

It is also possible for a source cell to contain two or more underlying sources. This is indicated when there are multiple correlation maxima in the source cell *at the PSF scale.* The code indicates such a condition with a flag in the source list; it does not currently attempt to "break up" the cell to refine source property estimates.

Source Selection Algorithm

To create the source list, *wrecon* examines the lists of correlation maxima in ascending order of wavelet size. If the location of a maximum (i_C, j_C) is not already associated with a source cell (see below), it will be provisionally associated with a real source if

- $F_{i_C,j_C} \neq 0$;

- and the ellipse defined by

$$\frac{(x-i_C)^2}{2\sigma_{\rm cur}^2} + \frac{(y-j_C)^2}{2\sigma_{\rm cur}^2} = 1$$

  does not contain one or more sources first detected at scales smaller than the currently considered scale $\sigma_{\rm cur}$. This is an important check when $\sigma_{\rm cur} \gtrsim r_{\rm PSF}$, as sources in a crowded field merge, possibly creating "new" sources at new locations in the FOV.

After determining for the putative source its cell and properties, *wrecon* first scans through the lists of all correlation maxima at all scales, and marks as "examined" those which lie within the current source cell, and then verifies that the putative source appears at one or more scales $\gtrsim r_{\rm PSF}$. It does this by determining the scale size closest to the PSF size, $\sigma_{\rm PSF}$ (by minimizing $|\log_2 r_{\rm PSF} - \log_2 \sigma_i|$), then counting the number of correlation maxima that lie within the source cell for $\sigma_i \geq \sigma_{\rm PSF}$. If the number is zero, *wrecon* rejects the putative source.

13.4.2 Source Properties

Location. *wrecon* calculates the expectation values $x_{\rm source}$ and $y_{\rm source}$ by summing pixel numbers within the source cell, using the data $D_{i,j}$ as a weighting function:

$$x_{\rm source} = \frac{1}{D_o}\sum_{i\in sc}\sum_{j\in sc} D_{i,j}\, i \qquad (13.22)$$

$$y_{\rm source} = \frac{1}{D_o}\sum_{i\in sc}\sum_{j\in sc} D_{i,j}\, j \qquad (13.23)$$

where $D_o = \sum_{i\in sc}\sum_{j\in sc} D_{i,j}$, and sc represents the source cell. We note that the estimate of location may be biased if the source cell is truncated because of the presence of a nearby source.

A preferred weighting function would be the source flux $D_{i,j} - E_{i,j}B_{i,j,{\rm norm}}$, but its use greatly complicates the computation of variances $V[x_{\rm source}]$ and $V[y_{\rm source}]$. (For similar reasons, $F_{i,j}$ is not used as a weighting function.) $D_{i,j}$ and $D_{i,j} - E_{i,j}B_{i,j,{\rm norm}}$ will give similar estimates if the source is strong or if background is small.

The variance of the location estimate, $V[x_{\rm source}]$, is estimated using Eq. (13.14):

$$\begin{aligned} V[x_{\rm source}] &= \sum_{i\in sc}\sum_{j\in sc}\left(\frac{\partial x_{\rm source}}{\partial D_{i,j}}\right)^2 V[D_{i,j}] \\ &= \sum_{i\in sc}\sum_{j\in sc}\left(\frac{i - x_{\rm source}}{D_o}\right)^2 D_{i,j} \qquad (13.24) \end{aligned}$$

$V[y_{\rm source}]$ is computed similarly.

Counts. The source counts are estimated by summing count flux within the source cell:

$$C_{\rm source} = \sum_{i\in sc}\sum_{j\in sc}(D_{i,j} - E_{i,j}B_{{\rm norm},i,j}) \qquad (13.25)$$

It is possible to rewrite this equation in the form $C = \sum_i \sum_j A_{i,j} D_{i,j}$, which in theory makes the variance directly computable. However, this calculation is computationally intensive and is not performed by the current version of the program (see the discussion around Eq. [13.65] for details). The variance is computed using the formula

$$V[C_{\rm source}] \approx \sum_{i\in sc}\sum_{j\in sc} D_{i,j} + E_{i,j}^2 V[B_{{\rm norm},i,j}] \tag{13.26}$$

The missing covariance terms are positive, so Eq. (13.26) underestimates the true variance.

Source Exposure Time. The effective exposure time of the source is the expectation value of t_oE, where t_o is the total detector livetime, $E_{i,j}$ is the fractional exposure in each bin within the source cell ($0 \le E_{i,j} \le 1$), and $D_{i,j}$ is the weighting function:

$$t_{\rm source} = \frac{t_o}{D_o}\sum_{i\in sc}\sum_{j\in sc} D_{i,j}E_{i,j} \tag{13.27}$$

$V[t_{\rm source}]$ is computed in an analogous manner to $V[x_{\rm source}]$, by replacing i with $t_oE_{i,j}$:

$$\begin{aligned} V[t_{\rm source}] &= \sum_{i\in sc}\sum_{j\in sc}\left(\frac{\partial t_{\rm source}}{\partial D_{i,j}}\right)^2 V[D_{i,j}] \\ &= \sum_{i\in sc}\sum_{j\in sc}\left(\frac{t_oE_{i,j} - t_{\rm source}}{D_o}\right)^2 D_{i,j} \end{aligned} \tag{13.28}$$

Source Count Rate. The source count rate is

$$R_{\rm source,C} = \frac{C_{\rm source}}{t_{\rm source}}. \tag{13.29}$$

The variance on the count rate estimate is

$$\begin{aligned} V[R_{\rm source,C}] &\approx \left(\frac{\partial R_{\rm source,C}}{\partial C_{\rm source}}\right)^2 V[C_{\rm source}] + \left(\frac{\partial R_{\rm source,C}}{\partial t_{\rm source}}\right)^2 V[t_{\rm source}] \\ &= \frac{1}{t_{\rm source}^2}\left(V[C_{\rm source}] + R_{\rm source,C}^2 V[t_{\rm source}]\right) \end{aligned} \tag{13.30}$$

where $V[C_{\rm source}]$ and $V[t_{\rm source}]$ are given in eqs. (13.26) and (13.28) respectively. Equation (13.30) underestimates the true variance.

Background Counts in Source Cell. The number of background counts in the source cell is

$$B_{\rm source} = \sum_{i\in sc}\sum_{j\in sc} E_{i,j}B_{i,j,{\rm norm}}. \tag{13.31}$$

The (underestimated) variance is given by

$$V[B_{\rm source}] \approx \sum_{i\in sc}\sum_{j\in sc} E_{i,j}^2 B_{i,j,{\rm norm}}. \tag{13.32}$$

Background Count Rate in Source Cell. The background count rate in the source cell is

$$R_{\mathrm{source,B}} \;=\; \frac{B_{\mathrm{source}}}{t_{\mathrm{source}}}, \tag{13.33}$$

which has (underestimated) variance

$$V[R_{\mathrm{source,B}}] \;\approx\; \frac{1}{t^2_{\mathrm{source}}}\left(V[B_{\mathrm{source}}] + R^2_{\mathrm{source,B}} V[t_{\mathrm{source}}]\right). \tag{13.34}$$

Source Image. *wrecon* constructs a noise-free image of detected sources using the equation

$$S_{i,j} \;=\; \sum_{k=1}^{N} \frac{C_{i,j,k}}{\sigma_{x,k}\sigma_{y,k}} \nu_{i,j,k}, \tag{13.35}$$

where N is the number of input scale pairs, and $\nu_{i,j,k} = 1$ if the following conditions hold:

- $C_{i,j,k} > 0$;
- the associated local correlation maximum is identified as a source pixel by *wtransform*;
- and the associated local correlation maximum is contained within a source cell.

Otherwise, $\nu_{i,j,k} = 0$. The second condition ensures that random maxima which are not associated with a source but which happen to lie within a source cell are not included in the source image; the last condition ensures that putative sources rejected by *wrecon* are also not included in the image.

13.5 The Mexican Hat (MH) Function

The source detection algorithm we present in this paper should not depend on the details of function itself, at least for simple wavelets which have one central positive mode. In this Appendix, we describe the Marr wavelet function, or "Mexican Hat function," (MH) which is used by *wtransform* and *wrecon*.

A wavelet function may be derived from a real-valued, non-negative, infinitely differentiable function $\phi(x, y)$ that satisfies the condition $\int \phi(x, y) = 1$ (see, *e.g.*, Holschneider 1995). One function that satisfies these requirements is the Gaussian:

$$\phi(x, y) \;=\; \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2}\right), \tag{13.36}$$

The relationship between ϕ and the wavelet function W in two dimensions is

$$\begin{aligned} W(\sigma_x, \sigma_y, x, y) \;&=\; [(x\frac{\partial}{\partial x} + 1) + (y\frac{\partial}{\partial y} + 1)]\phi(x, y) \\ &=\; [((x, y)\cdot\nabla) + 2]\phi(x, y) \\ &=\; \frac{1}{2\pi\sigma_x\sigma_y}\left[2 - \frac{x^2}{\sigma_x^2} - \frac{y^2}{\sigma_y^2}\right] e^{-\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2}} \end{aligned} \tag{13.37}$$

The integral of W over all space is, by definition, zero. This allows us to disregard the factor $\frac{1}{2\pi\sigma_x\sigma_y}$ in Eq. (13.37) when performing correlations.

The MH has a positive kernel surrounded by a negative annulus; the boundary between these two regions is an ellipse with axis-lengths $\sqrt{2}\sigma_x$ and $\sqrt{2}\sigma_y$:

$$\frac{x^2}{2\sigma_x^2} + \frac{y^2}{2\sigma_y^2} = 1 \qquad (13.38)$$

The area of this ellipse, used to compute correlation thresholds, is $A(\sigma_x, \sigma_y) = 2\pi\sigma_x\sigma_y$. The amplitude of the MH function is 2 at its center, regardless of scale; its minimum amplitude is $\approx$ -0.27 on the ellipse defined by axis-lengths $2\sigma_x$ and $2\sigma_y$.

The MH function has several advantageous properties which motivate its use.

- When rotationally symmetric, its Gaussian-like positive kernel has similar shape to a canonical PSF.
- The kernel's limited extent allows only those sources with intrinsically or instrumentally broadened size $\sim (\sigma_x, \sigma_y)$ to be strongly detected.
- It has an analytically-derivable Fourier Transform, speeding its correlation with data via the Correlation Theorem (Appendix 13.6).
- It has a limited extent in Fourier space, such that limited, discrete sampling in σ_x and σ_y (*e.g.* at values separated by factors of two) is sufficient to sample the entire frequency domain.
- Its limited extent in both spatial and Fourier domains helps to minimize effects of aliasing.

13.6 Solutions to Integrals Involving the MH Function

13.6.1 Analytic Integration of the MH Function

Pixel-by-Pixel Integration

If the width of the wavelet function is of order the size of the image pixel, then using FFTs to compute $< W{*}D >$ and other related quantities becomes impractible. To determine $C_{i,j}$, for instance, we integrate $W(x, y)$ on a pixel-by-pixel basis, as shown in Eq. (13.7).

We substitute the form of $W(x, y)$ in Eq. (13.37) into Eq. (13.7):

$$W_{\rm p} = \int_{x_1}^{x_2} dx \int_{y_1}^{y_2} dy \left[2 - \frac{x^2}{\sigma_x^2} - \frac{y^2}{\sigma_y^2}\right] \exp\left(-\frac{x^2}{2\sigma_x^2}\right) \exp\left(-\frac{y^2}{2\sigma_y^2}\right). \qquad (13.39)$$

Here, (x, y) represent coordinates relative to the center of pixel (i, j), and the integration is carried out over the pixel (i', j'):

$$\begin{aligned} x1 &= i' - i - \frac{1}{2} \qquad y1 = j' - j - \frac{1}{2} \\ x2 &= i' - i + \frac{1}{2} \qquad y2 = j' - j + \frac{1}{2} \end{aligned}$$

$W(x,y) \approx 0$ if (x,y) are outside an ellipse with axis-lengths $5\sigma_x$ and $5\sigma_y$, limiting the range of integration.

If we expand Eq. (13.39), we find that there are two basic integrals which must be performed. The first is:

$$\begin{aligned}
\int_{x_1}^{x_2} dx \exp\left(-\frac{x^2}{2\sigma_x^2}\right) &= \sqrt{2}\sigma_x \int_{t_1}^{t_2} dt \exp(-t^2) \\
&= \int_0^{t_2} dt \exp(-t^2) - \int_0^{t_1} dt \exp(-t^2) \\
&= \sqrt{\frac{\pi}{2}}\sigma_x \left[\mathrm{erf}\left(\frac{x_2}{\sqrt{2}\sigma_x}\right) - \mathrm{erf}\left(\frac{x_1}{\sqrt{2}\sigma_x}\right)\right] \\
&= \sqrt{\frac{\pi}{2}}\sigma_x x_{\mathrm{erf}}
\end{aligned}$$

where the substitution $t = \frac{x}{\sqrt{2}\sigma_x}$ is made, and $\mathrm{erf}(x)$ is the error function (see, *e.g.*, Press *et al.* 1992). The second integral is:

$$\int_{x_1}^{x_2} dx \frac{x^2}{\sigma_x^2} \exp\left(-\frac{x^2}{2\sigma_x^2}\right) = 2\sqrt{2}\sigma_x \int_{t_1}^{t_2} dt t^2 \exp(-t^2).$$

This integral can be solved by parts, by taking the derivative of t and the integral of $t\exp(-t^2)$, leaving an integral of $\exp(-t^2)$ (solved above). Leaving out intermediate steps, we present the solution:

$$x_1 \exp\left(-\frac{x_1^2}{2\sigma_x^2}\right) - x_2 \exp\left(-\frac{x_2^2}{2\sigma_x^2}\right) - \sqrt{\frac{\pi}{2}}\sigma_x x_{\mathrm{erf}} = x_{\mathrm{diff}} - \sqrt{\frac{\pi}{2}}\sigma_x x_{\mathrm{erf}}.$$

Taking the products of these solutions, and exchanging y for x when needed, we find:

$$W_{\mathrm{p}} = \pi\sigma_x\sigma_y x_{\mathrm{erf}} y_{\mathrm{erf}} - \sqrt{\frac{\pi}{2}}\sigma_y y_{\mathrm{erf}}\left(x_{\mathrm{diff}} + \sqrt{\frac{\pi}{2}}\sigma_x x_{\mathrm{erf}}\right) - \sqrt{\frac{\pi}{2}}\sigma_x x_{\mathrm{erf}}\left(y_{\mathrm{diff}} + \sqrt{\frac{\pi}{2}}\sigma_y y_{\mathrm{erf}}\right). \qquad (13.40)$$

Integration of the Negative Annulus

If exposure variations and the FOV edge may be ignored in the computation of the background, then we may replace $< NW{*}E >_{i,j}$ in Eq. (13.5) with a background normalization factor, which we derive here.

The overall normalization of the MH function is zero. Hence the background normalization factor may be derived by integrating over its positive core:

$$N_{\mathrm{back}} = \int_{-\sqrt{2}\sigma_x}^{\sqrt{2}\sigma_x} dx \int_{-\sigma_y\sqrt{2-\frac{x^2}{\sigma_x^2}}}^{\sigma_y\sqrt{2-\frac{x^2}{\sigma_x^2}}} dy \left[2 - \frac{x^2}{\sigma_x^2} - \frac{y^2}{\sigma_y^2}\right] \exp\left(-\frac{x^2}{2\sigma_x^2}\right) \exp\left(-\frac{y^2}{2\sigma_y^2}\right). \qquad (13.41)$$

The limits of integration reflect that the core extends over an ellipse with axes $\sqrt{2}\sigma_x$ and $\sqrt{2}\sigma_y$. We reparametrize the integral using polar coordinates (r,θ), after first mapping the boundary ellipse to a boundary circle using the transformation $y' = \frac{\sigma_x}{\sigma_y}y$:

$$\begin{aligned}
N_{\rm back} &= \frac{\sigma_y}{\sigma_x}\int_{-\sqrt{2}\sigma_x}^{\sqrt{2}\sigma_x} dx \int_{-\sigma_x\sqrt{2-\frac{x^2}{\sigma_x^2}}}^{\sigma_x\sqrt{2-\frac{x^2}{\sigma_x^2}}} dy' \left[2-\frac{x^2+y'^2}{\sigma_x^2}\right]\exp\left(-\frac{x^2+y'^2}{2\sigma_x^2}\right) &(13.42)\\
&= \frac{\sigma_y}{\sigma_x}\int_0^{2\pi} d\theta \int_0^{\sqrt{2}\sigma_x} drr\left[2-\frac{r^2}{\sigma_x^2}\right]\exp\left(-\frac{r^2}{2\sigma_x^2}\right) &(13.43)\\
&= 2\pi\frac{\sigma_y}{\sigma_x}\left[\int_0^{\sqrt{2}\sigma_x} dr2r\exp\left(-\frac{r^2}{2\sigma_x^2}\right) - \int_0^{\sqrt{2}\sigma_x} dr\frac{r^3}{\sigma_x^2}\exp\left(-\frac{r^2}{2\sigma_x^2}\right)\right]. &(13.44)
\end{aligned}$$

The determinant of the Jacobian of the transformation from $(x, y') \to (r, \theta)$ is r.

We evaluate the second integral using integration by parts:

$$\int_0^{\sqrt{2}\sigma_x} dr\frac{r^3}{\sigma_x^2}\exp\left(-\frac{r^2}{2\sigma_x^2}\right) = -\left[r^2\exp\left(-\frac{r^2}{2\sigma_x}\right)|_0^{\sqrt{2}\sigma_x}\right] + \int_0^{\sqrt{2}\sigma_x} dr2r\exp\left(-\frac{r^2}{2\sigma_x^2}\right). \qquad (13.45)$$

The second integral in Eq. (13.45) cancels with the first integral in Eq. (13.44), leaving:

$$\begin{aligned}
N_{\rm back} &= 2\pi\frac{\sigma_y}{\sigma_x}\left[r^2\exp\left(-\frac{r^2}{2\sigma_x}\right)|_0^{\sqrt{2}\sigma_x}\right]\\
&= 2\pi\frac{\sigma_y}{\sigma_x}\frac{2\sigma_x^2}{\exp(1)}\\
&= \frac{4\pi\sigma_x\sigma_y}{\exp(1)}. &(13.46)
\end{aligned}$$

13.6.2 Analytic Fourier Transform of the MH Function

The Fourier Transform of $W(x, y)$ is

$$\begin{aligned}
FT(W) &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} dxdyW(x,y)e^{2\pi i(k_x x+k_y y)}\\
&= \int_{-\infty}^{+\infty} dye^{2\pi i k_y y}\int_{-\infty}^{+\infty} dxW(x,y)\cos(2\pi k_x x)\\
&+\ i\int_{-\infty}^{+\infty} dye^{2\pi i k_y y}\int_{-\infty}^{+\infty} dxW(x,y)\sin(2\pi k_x x) &(13.47)\\
&= \int_{-\infty}^{+\infty} dye^{2\pi i k_y y}\int_{-\infty}^{+\infty} dxW(x,y)\cos(2\pi k_x x)\\
&= \left[\int_{-\infty}^{+\infty} dy\cos(2\pi k_y y) + i\int_{-\infty}^{+\infty} dy\sin(2\pi k_y y)\right]\int_{-\infty}^{+\infty} dxW(x,y)\cos(2\pi k_x x)\\
&= \int_{-\infty}^{+\infty} dy\cos(2\pi k_y y)\int_{-\infty}^{+\infty} dxW(x,y)\cos(2\pi k_x x)\\
&= \int_{-\infty}^{+\infty} dy\cos(2\pi k_y y)\times C_{\rm W} &(13.48)
\end{aligned}$$

The wave-number k equals $\frac{2(i_k-1)k_{\rm Nyquist}}{l}$, where i_k is the pixel number in Fourier space, $k_{\rm Nyquist} = \frac{1}{2}$ is the Nyquist wave-number, and l is half the length of the relevant axis in the padded image. The fourth integral in Eq. (13.47) and the second integral in Eq. (13.48) are zero, as the integrands are products of even and odd functions.

$$\begin{aligned} C_{\rm W} &= \exp\left(-\frac{y^2}{2\sigma_y^2}\right)\int_{-\infty}^{+\infty} dx \left[2-\frac{x^2}{\sigma_x^2}-\frac{y^2}{\sigma_y^2}\right]\exp\left(-\frac{x^2}{2\sigma_x^2}\right)\cos(2\pi k_x x) \\ &= \exp\left(-\frac{y^2}{2\sigma_y^2}\right)\left[\left(2-\frac{y^2}{\sigma_y^2}\right)\Psi_{0,{\rm c}}(2\pi k_x x, \frac{1}{\sqrt{2}\sigma_x}, 0) - \frac{1}{\sigma_x^2}\Psi_{2,{\rm c}}(2\pi, \frac{1}{\sqrt{2}\sigma_x})\right] \end{aligned} \quad (13.49)$$

$\Psi_{0,{\rm c}}(p,q,\lambda)$ and $\Psi_{2,{\rm c}}(a,p)$ represent two integral solutions which one may find, e.g., in Gradshteyn & Ryzhik (1980; formulae 3.896-2 and 3.952-4 respectively):

$$\Psi_{0,{\rm c}}(p,q,\lambda) = \frac{1}{q}\sqrt{\pi}\exp\left(-\frac{p^2}{4q^2}\right)\cos(p\lambda),$$

and

$$\Psi_{2,{\rm c}}(a,p) = \sqrt{\pi}\frac{2p^2-a^2}{4p^5}\exp\left(-\frac{a^2}{4p^2}\right).$$

Substituting these quantities into Eq. (13.49), one finds that

$$C_{\rm W} = \exp\left(-\frac{y^2}{2\sigma_y^2} - 2\pi^2 k_x^2\sigma_x^2\right)\sigma_x\sqrt{2\pi}\left[4\pi^2k_x^2\sigma_x^2+1-\frac{y^2}{\sigma_y^2}\right] \quad (13.50)$$

Substituting Eq. (13.50) into Eq. (13.48) and solving, we find:

$$\begin{aligned} & FT(W) \\ = \; & \exp(-2\pi^2k_x^2\sigma_x^2)\sigma_x\sqrt{2\pi}\int_{-\infty}^{+\infty} dy\cos(2\pi k_y y)\exp\left(-\frac{y^2}{2\sigma_y^2}\right)\left[4\pi^2k_x^2\sigma_x^2+1-\frac{y^2}{\sigma_y^2}\right] \\ = \; & \exp(-2\pi^2k_x^2\sigma_x^2)\sigma_x\sqrt{2\pi}\left[(4\pi^2k_x^2\sigma_x^2+1)\Psi_{0,{\rm c}}(2\pi k_y, \frac{1}{\sqrt{2}\sigma_y}, 0) - \frac{1}{\sigma_y^2}\Psi_{2,{\rm c}}(2\pi k_y, \frac{1}{\sqrt{2}\sigma_y})\right] \\ = \; & (2\pi)^3\sigma_x\sigma_y(k_x^2\sigma_x^2+k_y^2\sigma_y^2)\exp[-2\pi^2(k_x^2\sigma_x^2+k_y^2\sigma_y^2)]. \end{aligned} \quad (13.51)$$

To compute the error of the correlation values in each pixel, we calculate $< W^2 * D >_{i,j}$ using an analytic Fourier Transform $FT(W^2)$. The derivation of this function is similar to the derivation of $FT(W)$. A new integral which appears is:

$$\int_{-\infty}^{+\infty} dx\frac{x^4}{\sigma_x^4}\exp\left(-\frac{x^2}{\sigma_x^2}\right)\cos(2\pi k_x x) \quad (13.52)$$

One may solve this integral by parts, differentiating the term $\frac{x^3}{\sigma_x^2}\cos(2\pi k_x x)$ and integrating the term $\frac{x}{\sigma_x^2}\exp\left(-\frac{x^2}{\sigma_x^2}\right)$. The integral

$$\int_{-\infty}^{+\infty} dx x^3\sin(2\pi k_x x)\exp\left(-\frac{x^2}{\sigma_x^2}\right) \quad (13.53)$$

has solution (see, e.g., Gradshteyn & Ryzhik 1980, formula 3.952-5):

$$\int_{-\infty}^{+\infty} dx x^3\sin(2\pi k_x x)\exp\left(-\frac{x^2}{\sigma_x^2}\right) = \sqrt{\pi}\sigma_x^7\frac{3\pi\frac{k_x}{\sigma_x^2}-2\pi^3k_x^3}{2}\exp(-\pi^2\sigma_x^2k_x^2). \quad (13.54)$$

The final solution is

$$FT(W^2) = \left[2\pi\sigma_x\sigma_y + \pi^5\sigma_x\sigma_y(k_x^2\sigma_x^2+k_y^2\sigma_y^2)^2\right]\exp[-\pi^2(k_x^2\sigma_x^2+k_y^2\sigma_y^2)]. \quad (13.55)$$

13.7 Computation of Threshold Correlation Values

An image pixel (i,j) is associated with an astronomical source if the significance $S_{i,j}$ of its correlation value $C_{i,j}$ is greater than a user-defined threshold significance S_o:

$$S_{i,j} \;=\; \int_{C_{i,j}}^{\infty} p(C|n_{B,i,j})) > S_o.$$

$n_{B,i,j}$ is the inferred number of background counts within the limited spatial extent of the wavelet function, and $p(C|n_{B,i,j})$ is the probability sampling distribution for C given $n_{B,i,j}$. This sampling distribution does not depend upon the scale size of the wavelet function. We assume that the background count rate is locally constant, so that instead of $n_{B,i,j}$, we use the expected number of background counts in the positive kernel of the wavelet, a quantity which we denote $q_{i,j}$. For the MH function, $q_{i,j} = 2\pi\sigma_x\sigma_y B_{i,j}$.

We find that $p(C|q_{i,j})$ does not have analytic form for those values of $q_{i,j}$ most likely to be seen in analyses of *Chandra* images, so we must estimate this distribution using simulations. The computation time needed for such simulations means that we cannot accurately assign significances to pixels if $S_{i,j} \lesssim 10^{-7}$. So instead of computing the significance, *wtransform* compares the value of $C_{i,j}$ in each pixel with a *threshold* correlation value $C_{i,j,o}(S_o, q_{i,j})$, defined by the equation

$$S_o \;=\; \int_{C_{i,j,o}}^{\infty} dCp(C|q_{i,j}). \qquad (13.56)$$

We determine $C_{i,j,o}$ by:

- randomly selecting values logq_o from the range -10 $\leq$ log$q_o \leq$ 3.25;
- creating 1024$\times$1024 images, sampling data in each pixel from the Poisson distribution, with the expected number of counts in each pixel being $B_o = \frac{q_o}{2\pi\sigma^2}$, and with $\sigma = 4$ pixels;
- computing $C_{i,j}$ and $q_{i,j}$ in each image pixel;
- and recording $C_{i,j}$ in bins of size Δlog$q_{i,j} = 0.2$, for -6.9 $\leq$ log$q_{i,j} \leq$ 3.1, with one bin being used for all values of log$q_{i,j}$ < -6.9.[6] From these distributions $p(C|q_{i,j})$, we can determine $C_{i,j,o}$.

If a simulated dataset has no counts, we do not analyze it. We chose log$q_{i,j} = 3.25$ as a upper limit because of the report by Damiani *et al.* (1997) that for log$q_{i,j} \gtrsim 3$, the probability sampling distribution is analytically representable as a Gaussian with width $\sigma = \sqrt{q_{i,j}}$. (Note that the value of $q_{i,j}$ that we use in this work is larger than that used by Damiani *et al.* by a factor of 2π.) We return to this point below.

By analyzing over 50,000 simulated images, we are able to determine 25 values of $C_o(S_o)$ in each log$q_{i,j}$ bin, for values of $S_o \gtrsim 10^{-7}$, using the central 68% (17) of the values to estimate variances. We fit simple functions to the estimated threshold values, using the χ^2 method and the estimated variances. We present our results below. These functions describe the observed threshold values well, except in the regime log$q_{i,j} \lesssim 0$ and log$S_o \gtrsim -4$, where a binned look-up table is used instead. We note that these fits allow us in principle to compute threshold correlation values for significance values below our computational lower limit $S_o \sim 10^{-7}$ (such as for $S_o \sim 10^{-9}$, the significance corresponding to one false source pixel in an *Chandra* HRC image).

[6]Small values of $q_{i,j}$ occur even when there are no counts in the negative annulus of the wavelet function because the FFT is inexact.

In the regime $\log q_{i,j} \lesssim 0$ and $\log S_o \lesssim -4$, $\log C_o(q_{i,j})$ is well-described by the parabola:

$$\log C_o(q_{i,j}) = A_{\rm lo}(\log q_{i,j})^2 + B_{\rm lo}\log q_{i,j} + C_{\rm lo}, \qquad (13.57)$$

with

$$\begin{aligned} A_{\rm lo} &= 0.00462 \\ B_{\rm lo} &= 0.0661 \\ C_{\rm lo} &= -0.0154(\log S_o)^2 - 0.252(\log S_o) - 0.031 \end{aligned}$$

(Note that errors have not been estimated for these parameters.)

In the regime $\log q_{i,j} \gtrsim 0$, $C_o(q_{i,j})$ is described by the function

$$C_o(q_{i,j}) = A_{\rm hi}\sqrt{q_{i,j}} + B_{\rm hi}, \qquad (13.58)$$

a function used by Damiani *et al.*, with

$$A_{\rm hi} = \begin{cases} -0.509 \log S_o + 1.897 - .00172\ (\log S_o + 7)^{3.606} & \log S_o \geq -7 \\ -0.509 \log S_o + 1.897 & \log S_o < -7 \end{cases}$$

$$B_{\rm hi} = -1.115 \log S_o - 1.038$$

For values $\log q_{i,j} \sim 0$, we find that we must sometimes use the linear equation

$$\log C_o(q_{i,j}) = A_{\rm mid}\log q_{i,j} + B_{\rm mid} \qquad (13.59)$$

to represent the threshold, with

$$A_{\rm mid} = 0.00182(\log S_o)^3 + 0.0279(\log S_o)^2 + 0.158\log S_o + 0.607 \qquad (13.60)$$

and

$$B_{\rm mid} = \begin{cases} -0.064\log S_o + 0.612 - 0.00015(\log S_o + 7)^{4.75} & -2 < \log S_o \leq -1 \\ -0.064\log S_o + 0.612 - 0.00085(\log S_o + 7)^{3.5} & -7 \leq \log S_o \leq -2 \\ -0.064\log S_o + 0.612 & \log S_o < -7 \end{cases}$$

If the probability sampling distribution is Gaussian with width $\sigma = \sqrt{q_{i,j}}$, then the significance is given by:

$$\begin{aligned} S_{i,j} &= \frac{1}{\sqrt{2\pi q_{i,j}}}\int_{C_{i,j}}^{\infty} dC \exp\left(-\frac{C_{i,j}^2}{2q_{i,j}}\right) \\ &= 1 - \frac{1}{2} - \frac{1}{\sqrt{2\pi q_{i,j}}}\int_0^{C_{i,j}} dC \exp\left(-\frac{C_{i,j}^2}{2q_{i,j}}\right) \\ &= \frac{1}{2}\left[1 - {\rm erf}\left(\frac{C_{i,j}}{\sqrt{2q_{i,j}}}\right)\right] \qquad (13.61) \end{aligned}$$

We find that if we use Eq. (13.58) in the regime $\log q_{i,j} \gtrsim 3$, then the derived values of $C_{i,j,o}$ are larger than those predicted by Eq. (13.61) above. Because it is more conservative, we use Eq. (13.58) for all values $\log q_{i,j} \gtrsim 0$.

13.8 Normalized Background Map

If the user does not provide a normalized background image, *wrecon* will create such an image by combining the normalized background images that *wtransform* creates at each wavelet scale:

$$B^{\text{total}}_{\text{norm},i,j} = \sum_{k=1}^{N} \epsilon_{i,j,k}\sigma_k^2 B_{\text{norm},i,j,k} \tag{13.62}$$

$$w^{\text{total}}_{i,j} = \sum_{k=1}^{N} \epsilon_{i,j,k}\sigma_k^2 \tag{13.63}$$

where

$$\epsilon_{i,j,k} = \begin{cases} 1 & \sigma_k \gtrsim r_{\text{PSF},i,j} \\ 0 & \text{otherwise} \end{cases}$$

and N is the number of scales used. "$\sigma_k \gtrsim r_{\text{PSF}}$" denotes all $\sigma > r_{\text{PSF}}$, along with the largest $\sigma < r_{\text{PSF}}$. For instance, if $r_{\text{PSF}} = 9$ pixels, $\epsilon = 1$ for the standard progression values $\sigma = [8,8\sqrt{2},16,...]$, and $\epsilon = 0$ for $\sigma = [...4,4\sqrt{2}]$. We exclude the smallest scales because for $\sigma \ll r_{\text{PSF}}$, the background is greatly overestimated in the vicinity of sources. The weighting reflects that the area of the FOV used to estimate the background goes as σ^2. The default background for *wrecon* is then:

$$B^{\text{default}}_{\text{norm},i,j} = \frac{B^{\text{total}}_{\text{norm},i,j}}{w^{\text{total}}_{i,j}}. \tag{13.64}$$

We may algebraically manipulate the combination of eqs. (13.5), (13.62), and (13.63), to express the default normalized background as

$$\begin{aligned} B^{\text{default}}_{\text{norm},i,j} &= \sum_{i'}\sum_{j'}\left[\sum_{k=1}^{N}\frac{\epsilon_{i,j,k}\sigma_k^2}{< NW{*}E >_{i,j,k}} NW_{i-i',j-j',k}\right] D_{i',j'} \\ &= \sum_{i'}\sum_{j'}\left[\sum_{k=1}^{N} a_{i,j,k} NW_{i-i',j-j',k}\right] D_{i',j'} \end{aligned} \tag{13.65}$$

The presence of the term $a_{i,j,k}$ makes it impossible to compute the variance of the default background estimate using FFTs. Hence the current version of the program computes the variance of the background estimate by summing variance estimates at each scale:

$$V[B^{\text{default}}_{\text{norm},i,j}] = \sum_{k=1}^{N}\left(\frac{\epsilon_{i,j,k}\sigma_k^2}{w^{\text{total}}_{i,j}}\right)^2 V[B_{\text{norm},i,j,k}] \tag{13.66}$$

The missing covariance terms are positive, so Eq. (13.66) provides an *underestimate* of the true variance. If the user wishes to reduce the contribution of covariance terms, he or she may apply a sparser set of scales, since the magnitude of the covariance is proportional to the amount of overlap between negative annuli. For instance, if only scales separated by a factor of $2\sqrt{2}$ are used, the covariance terms will be ≈ 0, since the overlap of annuli is minimal.[7]

[7] A new method of determining the variance, based upon a suggestion by Kolaczyk (private communication), is being tested and may be used in future versions of *wrecon*.

13.9 References

Damiani, F., Maggio, A., Micela, G., & Sciortino, S. 1997, ApJ 483, 350

Daubechies, I. 1992, Ten Lectures on Wavelets (Philadelphia: SIAM)

Eadie, W. T., Drijard, D., James, F. E., Roos, M., & Sadoulet, B. 1971, Statistical Methods in Experimental Physics (Amsterdam: North-Holland)

Goupillaud, P., Grossmann, A., & Morlet, J. 1984, Geoexploration 23, 85

Gradshteyn, I., & Ryzhik, I. 1980, Table of Integrals, Series, and Products (San Diego: Academic Press)

Grebenev, S. A., Forman, W., & Jones, C. 1995, ApJ 445, 607

Holschneider, M. 1995, Wavelets: An Analysis Tool (Oxford: Oxford University Press)

Micela, G., Sciortino, S., Kashyap, V., Harnden, F. R., Jr., & Rosner, R. 1996, ApJS 102, 75

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P.1992, Numerical Recipes (Cambridge: Cambridge Univ. Press)

Rosati, P., Della Cecai, R., Burg, R., Norman, C., & Giacconi, R. 1995, ApJL 445, 11

Slezak, E., Bijaoui, A., & Mars, G. 1990, A&A 227, 301

Vikhlinin, A., Forman, W., & Jones, C. 1994, ApJ 435, 162

Chapter 14

wavdetect Parameters & Data Products Reference

This chapter lists all the *wavdetect* parameters in alphabetical order for quick reference. In addition, a guide to the data products output by *wavdetect* is provided.

14.1 Default Parameter File

```
unix% plist wavdetect

Parameters for /soft/ciao/param/wavdetect.par

#
#   parameter file for wavdetect
#
#
#   input
#
        infile =                  Input file name
#
#   output
#
       outfile =                  Output source list file name
     scellfile =                  Output source cell image file name
     imagefile =                  Output reconstructed image file name
   defnbkgfile =                  Output normalized background file name
#
#   scales
#
        scales = 2.0 4.0          wavelet scales (pixels)
      (regfile = )                ASCII regions output file
```

```
#
#   output options
#
      (clobber = no)              Overwrite existing outputs?
       (kernel = default)         Output file format (fits|iraf|default)
     (ellsigma = 3.0)             Size of output source ellipses (in sigmas)
     (interdir = .)               Directory for intermediate outputs
#
#########################################################################
#
#   wtransform parameters
#
#
#   optional input
#
     (bkginput = )                Input background file name
  (bkgerrinput = no)              Use bkginput[2] for background error
#
#   output info
#
  (outputinfix = )                Output filename infix
#
#   output content options
#
    (sigthresh = 1e-06)           Threshold significance for output source pixel list
 (bkgsigthresh = 0.001)           Threshold significance when estimating bkgd only
#
#   exposure info
#
      (exptime = 0)               Exposure time (if zero, estimate from map itself
      (expfile = )                Exposure map file name (blank=none)
    (expthresh = 0.1)             Minimum relative exposure needed in pixel to analyze it
#
#   background
#
      (bkgtime = 0)               Exposure time for input background file
#
#   iteration info
#
      (maxiter = 2)               Maximum number of source-cleansing iterations
     (iterstop = 0.0001)          Min frac of pix that must be cleansed to continue
#
#   end of wtransform parameters
#
########################################################################
########################################################################
#
#   wrecon parameters
#
#
#   PSF size parameters
```

```
#
      (xoffset = INDEF)           Offset of x axis from optical axis
      (yoffset = INDEF)           Offset of y axis from optical axis
        (eband = 1.4967)          Energy band
      (eenergy = 0.393)           Encircled energy of PSF
     (psftable = ${ASCDS_CALIB}/psfsize20010416.fits -> /soft/ciao/data/psfsize20010416.fits) Table of 
#
#   end of wrecon parameters
#
########################################################################
#
#   run log verbosity and content
#
          (log = no)              Make a log file?
      (verbose = 0)               Log verbosity
#
#   mode
#
         (mode = ql)
```

14.2 Parameter Descriptions

In the following parameter descriptions, `"required=yes"` indicates that the user must provide a value for the indicated parameter (*e.g.* `infile`) before the program will run. Also, the empty string (`" "`) is equivalent to the string `none` for filenames.

- `bkgerrinput`
  Use bkginput[2] for background error
  type=boolean
  def=no
  required=no

  Whether the background error data in the `bkginput` file will be used.

  If `bkgerrinput` is changed to `yes`, then the background error data in the second extension of the `bkginput` file will be used (*i.e.* "bkginputfilename.fits[2]").

- `bkginput`
  Input background file name
  type=string
  filetype=input
  required=no

  The name of a background image.

  If desired, a previously computed background image may be input instead of allowing *wavdetect* to construct a default normalized background. When using `bkginput`, enter `none` for the default normalized background output file (`defnbkgfile`).

- `bkgsigthresh`
  Threshold significance when estimating bkgd only

type=real
def=0.001
required=no

The significance threshold for cleansing data from the image to compute the background map.

As it does not effect source detection, this parameter should be set to a more liberal value than `sigthresh` (*e.g.* 10^{-2} or 10^{-3}, the default), but not smaller than `sigthresh`). This will help reduce the effect of weak undetectable sources on the background map calculation. This value should be no larger than 0.05, the usual 5% (or 95%) statistical criterion for rejecting the null hypothesis, which is that the pixel in question has data sampled solely from the background.

- `bkgtime`
  Exposure time for input background file
  type=real
  def=0
  required=no

  Exposure time for the background file.

  The `bkgtime` parameter may specify the `LIVETIME` value for the provided background (`bkginput`). If `bkgtime` is non-zero and no background map is supplied, the source characteristics in the output file will be wrong.

- `clobber`
  Overwrite existing outputs?
  type=boolean
  def=no
  required=no

  If "yes", overwrite existing outputs.

  wavdetect does not not overwrite existing outputs unless `clobber` is set to `yes`.

- `defnbkgfile`
  Output normalized background file name
  type=string
  filetype=output
  required=yes
  autoname=yes

  Output default normalized background output file.

  The output file that will contain the default normalized (*i.e.* flat-fielded) background is specified by the `defnbkgfile` parameter. During its second stage, *wavdetect* makes one normalized background estimate from the stack of per-scale normalized backgrounds produced during the first stage. It writes the computed background to this file. If autonaming is used (*i.e.* `"defnbkgfile=."`), then the output file will be named `"<infile_root>_nbkg"`.

- `eband`
  Energy band
  type=real
  def=1.4967
  units=keV

  The photon energy (keV) at which the point spread function (PSF) is chosen.

  The selection of a particular PSF size is governed by the energy band `eband` parameter [keV], since the *Chandra* PSF is a function of energy. By default the energy band is set to 1.4967 keV.

- `eenergy`
  Encircled energy of PSF
  type=real
  def=0.393
  required=no

  The percentage of PSF energy to be encircled, expressed as a fraction of 1.0.

  wavdetect also uses an encircled energy percentage value, given by the `eenergy` parameter, in concert with the calibration PSF file. This value is the percentage of PSF energy to be encircled by the detect cell, expressed as a fraction of 1.0. The `eenergy` is a key parameter for detecting off-axis sources, and users are urged to experiment with other values. By default, the value is set to 0.393 (the 1-sigma integrated volume of a normalized two-dimensional Gaussian); another suggested value to try is 0.5. Note that the efficiency of the algorithm is reduced if this fraction is too high or too low. If the value of `eenergy` is set too high (*e.g.* 0.9), there is the risk that the computed source cell will contain more than one source, rendering source property estimates moot. On the other hand, if the value is set too low, then the computed cell may contain only a fraction of the source counts.

- `ellsigma`
  Size of output source ellipses (in sigmas)
  type=real
  def=3.0

  The size, in sigmas, to make the elliptical source detection regions.

  The size of the elliptical source regions in both of the source region files (`regfile` and `outfile`) is controlled via the `ellsigma` parameter. This parameter is a multiplicative factor applied to sigma, the standard deviation of the distribution, to scale the major and minor axes of the ellipses for each source detection.

  The distribution is defined as the distribution of counts within the source cell (*i.e.* the observed counts), and is assumed to be Gaussian. Thus, if `ellsigma` is 1, then the elliptical source region will contain 39.3% of the source counts (*i.e.* the 1-sigma point for a 2-D Gaussian).

  This feature is included so that the graphics overlay may be made more visible; it is does not affect the detections themselves, only the display of the regions.

- `expfile`
  Exposure map file name (blank=none)
  type=string
  filetype=input

  The name of an exposure map file.

  This is an image of the data set exposure time. Although the expectation is that `expfile` will have units of time, exposure maps generated by the CIAO tool `mkexpmap`, which have units of cm^2, work as well.

- `expthresh`
  Minimum relative exposure needed in pixel to analyze it
  type=real
  def=0.1
  required=no

  The minimum relative exposure needed in a pixel in order to analyze it.

  For each pixel, a relative exposure (pixel exposure value over maximum value of exposure in map) is calculated. If this relative exposure is less than the parameter `expthresh`, then the pixel in question is not analyzed; all associated values (background, correlation, *etc.*) are set to zero.

This parameter should not be less than 0.1 or the accuracy of normalization will be too low. If set close to 1, only very limited regions of the field of view will be considered when source correlation maxima are listed. Typical values are 0.1-0.2.

- `exptime`
  Exposure time (if zero, estimate from map itself
  type=real
  def=0
  units=seconds

  The length of time over which the field was observed, in seconds.

  If set to 0, the value will either be taken from the `expfile` FITS header or, failing that, estimated by averaging over exposure map pixel values at the center of the field.

- `imagefile`
  Output reconstructed image file name
  type=string
  filetype=output
  required=yes
  autoname=yes

  The name of a reconstructed source image output file.

  The reconstructed source image filename is given by the `imagefile` parameter. If autonaming is used (*i.e.* `"imagefile=."`), then the output file will be named `"<infile_root>_image"`.

- `infile`
  Input file name
  type=string
  filetype=input
  required=yes

  The name of the input file.

  The input file can an event list or an image. *CIAO* Data Manipulation (DM) syntax may be used in the input file specification, as explained in Chapter 1, Section 1.3.3.

  Be careful using images larger than 1024×1024, as a large amount of computer memory must be available.

  Pixel values should be reasonable numbers, as /wavdetect/ probability distributions have been derived assuming Poisson statistics. /wavdetect/ simply will not work with fluxed images, as the calculated correlation values are so small that no detections occur.

- `interdir`
  Directory for intermediate outputs
  type=string
  filetype=output
  def=.
  required=no

  A directory for intermediate results.

- `iterstop`
  Min frac of pix that must be cleansed to continue
  type=real
  def=0.0001
  required=no

The minimum fraction of pixels that must be cleansed to continue constructing the background.

If the ratio of newly cleansed pixels to the overall number of pixels in the dataset is less than the parameter `iterstop` when *wavdetect* constructs a default normalized (*i.e.* flat-fielded) background (`defnbkgfile`), then *wavdetect* stops iterating and uses the current background estimate as the final background estimate. The background is calculated if there are very few new pixels being cleansed (see `bkgsigthresh`).

The `iterstop` parameter specifies how many iterations the program should go through to calculate the background. A typical value is 0.0001 (the default): stop iterating when one of every thousand pixels has been cleansed. This parameter should not be smaller than one over the number of pixels in the dataset or larger than 1.

- `kernel`
  Output file format (fits—iraf—default)
  type=string
  def=default
  required=no

  The format for the output file.

  The `kernel` parameter controls the output format (`default`, `fits`, or `iraf`); the default is for the output file to have the same format as the input file.

- `log`
  type=boolean
  def=no
  required=no

  Whether log information will go to a file or STDOUT.

  The verbosity of log information is sent to `STDOUT` (usually the screen). However, if `log` is set to `yes`, then the files `wrecon.log` and `wtransform.log` will be created.

- `maxiter`
  Maximum number of source-cleansing iterations
  type=integer
  def=2
  required=no

  The maximum number of iterations per scale pair for cleaning sources from the data in order to estimate the background map.

  Increasing this number will increase the method's source detection sensitivity, but may not increase it enough to justify the increased computation time. More iterations are generally needed for large wavelet scales (*e.g.* 16 pixels in a *Rosat* 512×512 PSPC field). Note that the tool may converge before reaching `maxiter`.

- `mode`
  def=ql

  The mode defines how to handle querying for parameters. For example, setting the mode to "h" means that no prompting will occur; this is useful when running the tool from a script. See "ahelp parameter" for more information and a list of all the available modes.

- `outfile`
  Output source list file name
  type=string
  filetype=output

required=yes
autoname=yes

The name of the output file.

The `outfile` parameter is used to provide a name for the output file. The `kernel` parameter controls the output format; the default is for the output file to have the same format as the input file. If autonaming is used (*i.e.* `"outfile=."`), then the output file will be named `"<infile_root>_src"`.

- `outputinfix`
  Output filename infix
  type=string
  required=no

  A unique string for use in output filenames.

  This may be used to avoid file naming collisions with other wavdetect users.

- `psftable`
  type=string
  filetype=ARD
  def=psfsize20010416.fits

  The path and name of the file containing PSF information.

  In *wavdetect*, PSF, energy, and offset information supplied via these parameters is used to select a source counts image; a source cell is computed using this image, and source properties are estimated using the data within the computed cell. Note that, unlike *celldetect*, these parameter values have no effect on the detection process. Instead, these parameter values affect reported source properties.

  The PSF information used by *wavdetect* is contained in a *Chandra* calibration file distributed with *CIAO*, whose location is given by the value of the `psftable` parameter; by default this parameter is automatically set to point to the appropriate *Chandra* calibration PSF file.

- `regfile`
  ASCII regions output file
  type=string
  autoname=yes

  The name of an ASCII source region output file.

  In addition to the output source list (`outfile`), the user may choose to have an ASCII source region file written by specifying a filename in `regfile` parameter. If autonaming is used (*i.e.* `"regfile=."`), then the source regions output file will be named `"<infile_root>_reg"`.

  This region file contains the location and size of the principal axes, for each detected source elliptical region. The elliptical region size is such that the region contains some fraction of the source counts. The parameter `ellsigma` specifies the desired source count fraction.

- `scales`
  Wavelet scales (pixels)
  type=string
  def=2.0 4.0

  The wavelet radii, in pixels.

  The `scales` parameter determines how many scaled transforms will be computed. *wtransform* produces a complete set of outputs for each scale. Small scales tend to detect small features, and larger scales find larger features. The units for `scales` are pixels. The value of `scales` is the radius of the Mexican Hat function, which crosses zero at $\sqrt{2}\times$radius.

For the initial run, try the default value `"2.0 4.0"`. If that test is successful, try again with `scales` `"1.0 2.0 4.0 8.0 16.0"`. For a more extensive run, the user might wish to try the $\sqrt{2}$ series: `"1.0 1.414 2.0 2.828 4.0 5.657 8.0 11.314 16.0"`.

The primary concern regarding this parameter is to match the first scale size to the point spread function (PSF). Once that is done, the user must decide how many scaled wavelets to use. To some extent, choices are based on the computing resources at hand. Reasonable inputs are a 512×512 array and 5 to 9 scales, where each scale is a factor of 2 or $\sqrt{2}$ larger than the preceeding one.

In practice, one must not have too many pixels or scale sizes, otherwise the computational load becomes a problem. Data structures for a 512×512 image use up 36 MB, while a 2048×2048 image requires over 300 MB. Datasets that do not fit in physical memory will page heavily to disk and processing will progress very slowly. Scale sizes larger than 32 allocate excessive memory because it is necessary to pad the image with surrounding zeros.

To deal with resolved sources, several convolutions are performed, each with a successively larger scale version of the wavelet. The resulting "correlation maps" are then examined for regions where the intensity is larger than some threshold, and the final source list is constructed from a comparison of the different scale runs.

- `scellfile`
  Output source cell image file name
  type=string
  filetype=output
  required=yes
  autoname=yes

  The output image of the source cells.

  The `scellfile` is the image showing the source cells. These source cells delimit the image pixels that are used to estimate source properties (*e.g.* count rate). If autonaming is used (*i.e.* `"scellfile=."`), then the output file will be named `"<infile_root>_scell"`.

- `sigthresh`
  Threshold significance for output source pixel list
  type=real
  def=1e-06
  required=no

  The significance threshold for identifying a pixel as belonging to a source.

  A good value to use is the inverse of the total number of pixels in the image, *e.g.* $\sim 10^{-6}$ (the default) for a 1024×1024 field. This is equivalent to stating that the expected number of false sources per field is one. If larger arrays are used without decreasing the value of `sigthresh`, there will be an increased probability of detecting false sources. Likewise, if smaller arrays are used, the user may wish to increase this parameter value. The `sigthresh` should not be smaller than roughly $\sim 10^{-9}$ to $\sim 10^{-10}$; at this value, the accuracy of the computed detection thresholds is unknown.

- `verbose`
  Log verbosity
  type=integer
  def=0
  min=0
  max=5
  required=no

  The verbosity of log output.

The verbosity of log information ranges from 0 (none) to 5 (maximum output) and is sent to `STDOUT` (usually the screen).

- `xoffset`
  Offset of x axis from optical axis
  type=integer
  def=INDEF
  units=pixels
  required=no

  The offsets of the x-axis for the calculation of the off-axis angle.

  By default, *celldetect* calculates the off-axis angle using the nominal pointing of the data file as the origin. Users may change this behavior by setting offsets to any numerical values (using the `xoffset` and `yoffset` parameters); these offsets provide the location of the optical axis with respect to the center of the data file. A typical scenario in which the user may select to use offsets is when the data file is a sum of two or more observations with different pointings. The nominal pointing in such a data file is usually poorly defined and the off-axis angle could be calculated from an undesired origin, leading to suboptimal selection of the detect cell size.

- `yoffset`
  Offset of y axis from optical axis
  type=integer
  def=INDEF
  units=pixels
  required=no

  The offsets of the y-axis for the calculation of the off-axis angle.

  See the description for `xoffset`.

14.3 Data Product Descriptions

Source List File (`outfile`)

The primary output of *wavdetect* is the FITS file of source detections. The following properties are recorded in the output file:

Data item	**Unit**	**Description**
RA	deg	Source right ascension
DEC	deg	Source declination
RA_ERR	deg	Source right ascension error
DEC_ERR	deg	Source declination error
POS(X,Y)	pixel	Physical coordinates
X_ERR	pixel	Source X position error
Y_ERR	pixel	Source Y position error
NPIXSOU	pixel	Pixels in source region
NET_COUNTS	count	Net source counts
NET_COUNTS_ERR	count	Error in net source counts
BKG_COUNTS	count	Background counts (scaled to source cell)
BKG_COUNTS_ERR	count	Error in background counts
NET_RATE		Source count rate (count/exposure map units)
NET_RATE_ERR		Source count rate error
BKG_RATE		Background count rate (count/exposure map unit)
BKG_RATE_ERR		Background count rate error
EXPTIME		Effective exposure map value (exposure map unit)
EXPTIME_ERR		Effective exposure map error
SRC_SIGNIFICANCE		Source significance
PSF_SIZE	pixel	Estimated size of PSF
MULTI_CORREL_MAX		1 if source contains 2+ correl maxima
SHAPE		Shape of source region
R[2]		Radii of source region
ROTANG	deg	Rotation angle of source region
PSFRATIO		The ratio of the equivalent sigma of the count distribution to the PSF_SIZE
COMPONENT		Source number

Source cells are arbitrarily shaped groups of pixels within which source properties are estimated. A given source's cell is constructed by: (1) estimating the PSF size; (2) choosing the source counts image created using a smoothing scale closest to that PSF size; and (3) going to the source's location in that image, where the image amplitude is positive, and determining the closest zero-amplitude contour around that point. (If there is a saddle point in the source counts distribution closer than the zero-point in a particular direction, then the saddle point is used to define the contour.) Pixels within that contour constitute the source cell.

- RA, DEC:
  Right ascension and declination coordinates, corresponding to X and Y. Calculated only if WCS keywords are present in the data, otherwise filled with zeros.
- RA_ERR, DEC_ERR:
  RA and DEC errors, corresponding to X_ERR and Y_ERR.
- POS(X,Y):
  Estimated X and Y positions of the detected source. Calculated with a "center-of-mass" algorithm using the counts data within the source cell. If the cell is asymmetric and/or the source is relatively weak, the position may be offset from the true position apparent to the eye.
- X_ERR, Y_ERR:
  Estimated uncertainties on the X or Y position of the source.
- NPIXSOU:
  The number of image pixels contained in the source cell.

- NET_COUNTS:
  The sum of all counts in the source cell minus the sum of the estimated background counts.
- NET_COUNTS_ERR:
  Estimated uncertainty of NET_COUNTS. This estimate does not include contributions from covariance terms (which are non-zero because the background estimates in adjacent pixels are not statistically independent); these contributions are generally ignorable.
- BKG_COUNTS:
  The sum of the estimated background counts in each pixel in the source cell. The estimated background is output as the default normalized background image.
- BKG_COUNTS_ERR:
  Estimated uncertainty of BKG_COUNTS. This estimate does not include contributions from covariance terms (which are non-zero because the background estimates in adjacent pixels are not statistically independent); these contributions are generally ignorable.
- NET_RATE:
  Source count rate: the NET_COUNTS divided by EXPTIME. Note that the units of NET_RATE are not necessarily seconds: typical *Chandra* exposure maps have units cm^2, so the NET_RATE is in $\mathrm{counts/cm}^2$.
- NET_RATE_ERR:
  Estimated uncertainty of NET_RATE. This estimate does not include contributions from covariance terms (which are non-zero because the background estimates in adjacent pixels are not statistically independent); these contributions are generally ignorable.
- BKG_RATE:
  Background count rate: the BKG_COUNTS divided by EXPTIME. Note that the units of BKG_RATE are not necessarily seconds: typical *Chandra* exposure maps have units cm^2, so the BKG_RATE is in $\mathrm{counts/cm}^2$.
- BKG_RATE_ERR:
  Estimated uncertainty of BKG_RATE. This estimate does not include contributions from covariance terms (which are non-zero because the background estimates in adjacent pixels are not statistically independent); these contributions are generally ignorable.
- EXPTIME:
  The sum of the exposure within the source cell. Note that the units of EXPTIME are not necessarily seconds: typical *Chandra* exposure maps have units cm^2.
- EXPTIME_ERR:
  Estimated uncertainty of EXPTIME. This estimate takes into account the Poisson fluctuations of the data within each pixel of the source cell, but ignores any uncertainty in the pixel-by-pixel estimate of EXPTIME itself.
- SRC_SIGNIFICANCE:
  A significance estimate in units of σ, found by dividing the NET_COUNTS by the "Gehrels error" σ_G of the BKG_COUNTS ($\sigma_G = 1 + \sqrt{BKG\_COUNTS + 0.75}$). This significance *should not be taken at face value*, particularly in the low-counts limit. The best statement of a source significance, regardless of the value of SRC_SIGNIFICANCE, is that it is less than or equal to the input value `sigthresh`.
- PSF_SIZE:
  The estimate "size," in image pixels, of the PSF at the source location. This is an encircled-energy radius, not a diameter.
- MULTI_CORREL_MAX:
  If 1, then multiple correlation maxima are observed within the output source cell: an indication that a detected source *may* in actuality be a composite of more than one source.

- SHAPE:
  The shape of region output to the ASCII source region file; `ellipse` by default.
- R[2]:
  The estimated semi-major and semi-minor axis lengths (*i.e.* the x and y radius lengths in pixels, before rotation). For SHAPE = `ellipse`, this is `ellsigma` times the axis lengths.
- ROTANG:
  The counterclockwise angle between the semi-minor axis (see R) and the y-axis of the image, in degrees.
- PSFRATIO:
  The ratio of the estimated "radius" of the source cell to PSF_SIZE. The radius of the source cell is found by taking the square root of the product of the two components of R[2] and dividing this by 2*ellsigma. A ratio $\gg 1$ is an *indication* that the detected source may be extended.
- COMPONENT:
  The number assigned to the detected source. The source numbers correspond to those in the output source list. Its primary use is for filtering of the source list by CIAO tools (*e.g.* `"component=1:10"`, `"component=1,2,5"`).

ASCII Source Region File (`regfile`)

In addition to the source list file (`outfile`), the user may also select to have a source region file (`regfile`) output.

This region file is useful for subsequent input into ds9, particularly to overlay region markers over each detection; see Chapter 1, Section 1.4 for details.

The source region file from the example in Chapter 11, Section 11.2.2:

```
unix% more example2_out.reg
ellipse(251.973913,268.647826,14.517026,8.139225,102.509720)
ellipse(251.878641,309.131068,8.314850,7.646153,26.671946)
ellipse(251.925620,320.471074,8.546168,7.646372,90.888733)
.
.
.
```

Log File (`logfile`)

If `log` is set to `yes`, then the files `wrecon.log` and `wtransform.log` are created.

The log files from the example in Chapter 11, Section 11.2.2:

```
unix% more wrecon.log
reading data image
Input image file: data/datasetA_acis_img.fits[123:630,109:614]
Input correlation maxima stack: ./wd_srclist_stk
.
.
.

unix% more wtransform.log
input file: data/datasetA_acis_img.fits[123:630,109:614]
exposure file:
correlation maxima output stack: ./wd_srclist_stk
correlation image output stack: ./wd_correl_stk
```

.
.
.

Source Cell Image File (`scellfile`)

The source cell image shows which pixels are used in the computation of properties for each detected source. The value for each pixel is either 0 (not a source pixel) or an integer (corresponding to a particular numbered source in the source list, *i.e.* a pixel value "2" means that that pixel is associated with source #2). The extent of each source cell is determined by using the particular source counts image closest in size to the PSF size at the source location; see Chapter 13 for more details on /wavdetect/ theory.

Reconstructed Image File (`imagefile`)

The reconstructed source image is a noise-free reconstruction of the raw data using the correlation images output from *wtransform*. Note that the overall normalization of the source image differs from that of the raw data image, thus values in particular pixels should not be read as estimates of the number of counts in those pixels. Chapter 13 contains details on how this image is constructed.

Normalized Background File (`defnbkgfile`)

During its second stage, *wavdetect* makes one normalized background estimate from the stack of per-scale normalized backgrounds produced during the first stage. It writes this computed background to this file.

The normalized (or flat-field) background image shows the number of expected background counts in each detector pixel, if the exposure time in each pixel is constant and equal to the amount of time the field was observed.

Computationally, the normalized background is:

$$B_{\mathrm{norm},i,j} \;=\; \frac{< NW{*}D' >_{i,j}}{< NW{*}E >_{i,j}}, \tag{14.1}$$

where NW represents the wavelet function with positive values reset to zero ("Negative Wavelet"), $D'_{i,j}$ are the cleansed data in each pixel (the data with putative source counts removed), and $E_{i,j}$ is the ratio of exposure time in each pixel to the overall observation time. $< NW{*}D' >_{i,j}$ and $< NW{*}E >_{i,j}$ are the correlation of the negative wavelet function with the cleansed data and exposure, respectively.

Because $B_{\mathrm{norm},i,j}$ is a function of wavelet scale size, there is a distinct normalized background image output from *wtransform* at each chosen pair of scales (σ_x, σ_y).