Detecting Sources in Chandra Data

Goal

Identify statistically significant brightness enhancements, over local background, deriving from both unresolved (point) and resolved (extended) x-ray sources. Emphasize three basic functions:

- Find sources
- Assess their credibility
- Compute their positions

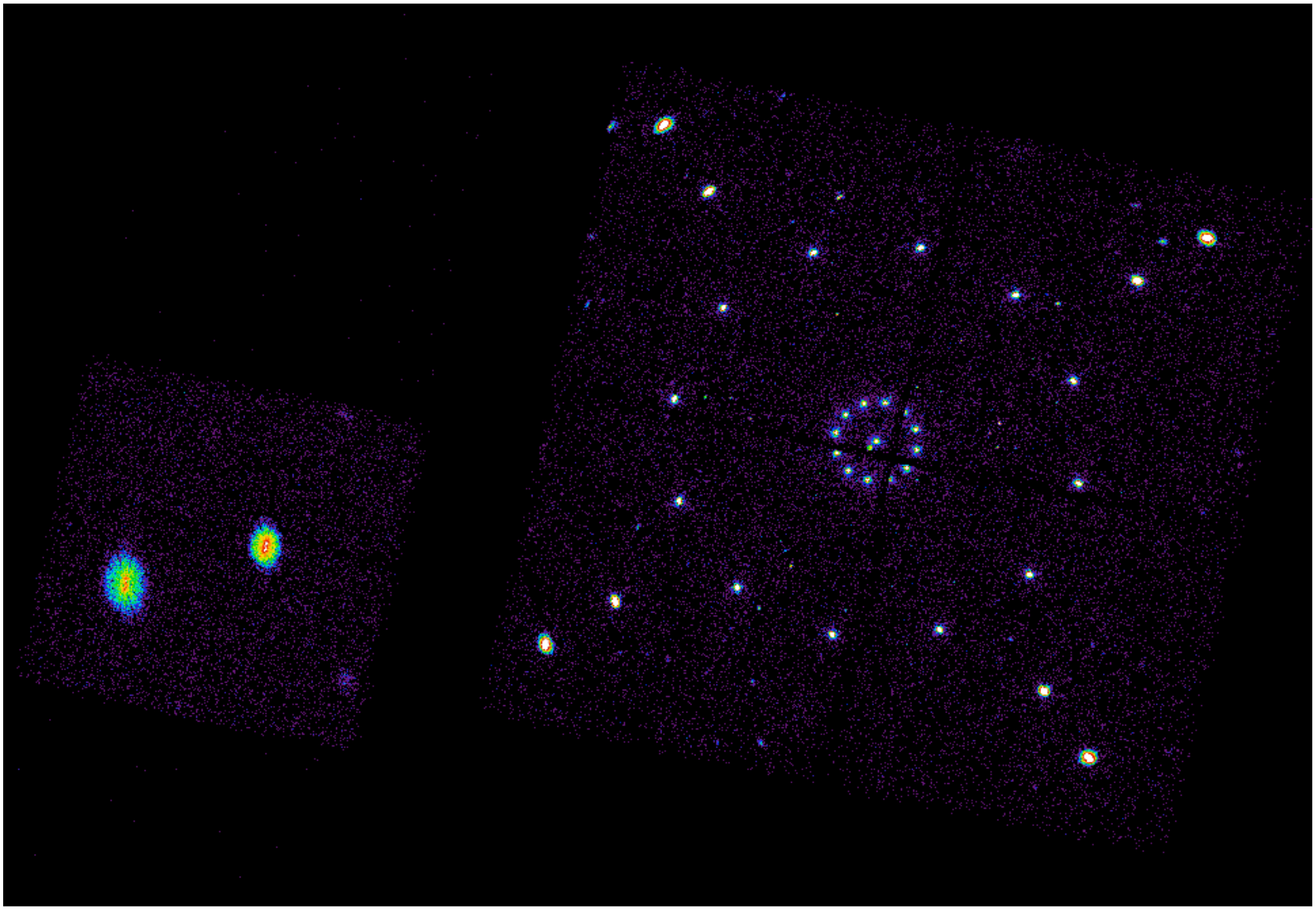
Limitations

- Deal with imaging data only
- Concerned mostly with unresolved or moderately resolved sources
- Ignore other source properties derived from DETECT tools

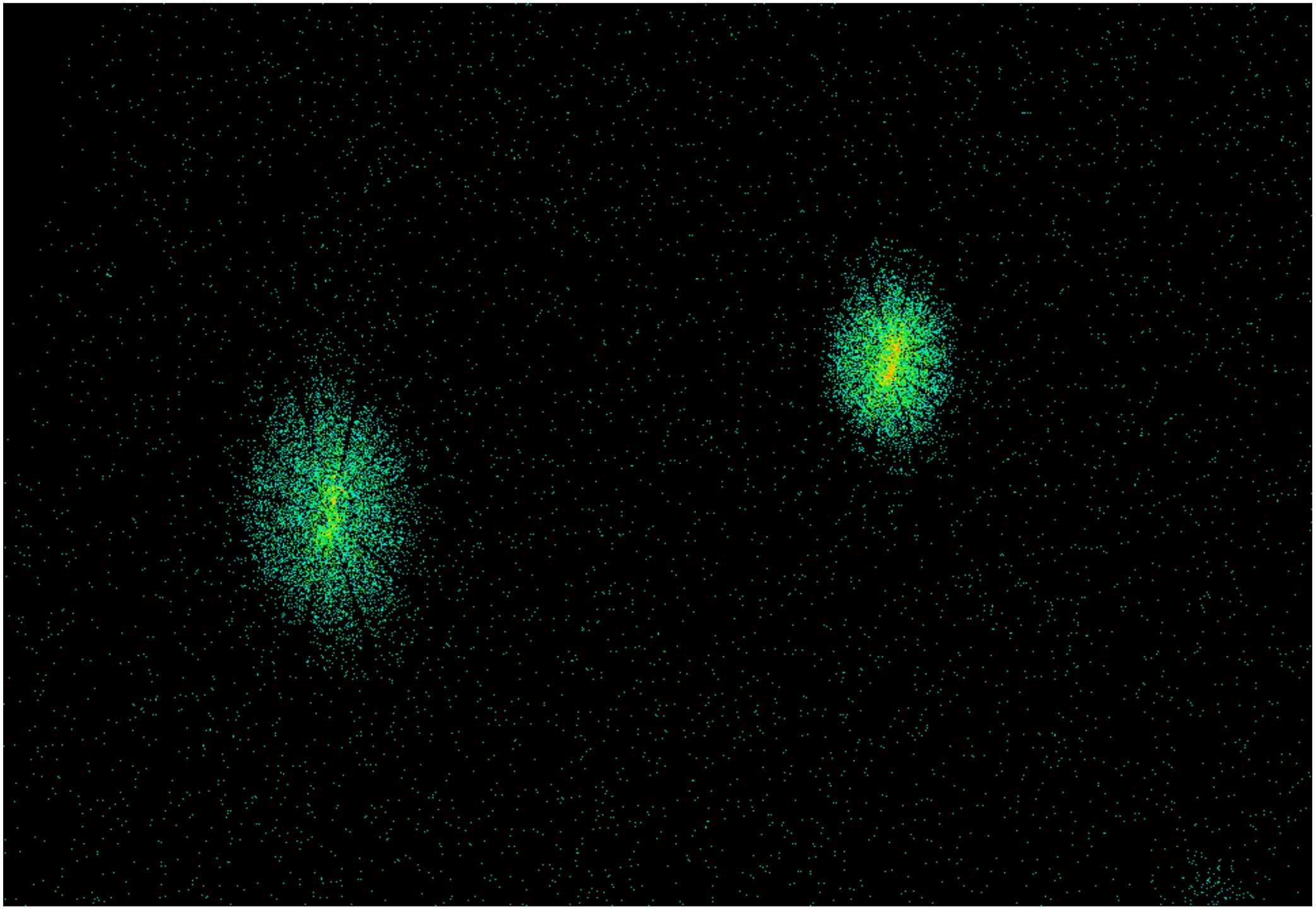
Challenges in Chandra Data

- Chandra mirrors have a ~10X higher angular resolution than other x-ray telescopes:
  - Complex source structures often seen;
  - Point sources in other telescopes are resolved into multiple, overlapping sources;
- Point Spread Function changes dramatically with position in the image;
- Images typically have only a few events per pixel.
- Chandra data have several edge effects:
  - Field-of-view boundaries;
  - Jumps in background between FI and BI chips;
  - Node boundaries inside ACIS chips.

For Example, NGC 6822, OBSID 2925 + ...



Same Data: Chip 6, Full Resolution



CIAO Source Detection Tools

- CELLDETECT - Simple “sliding-cell” tool with cell sizes dependent on off-axis angle. Can handle entire event list by automatically blocking image at multiple scales. Cheap and fast, but suffers from many false sources near edges. Not extensively calibrated.

- VTPDETECT - Analyzes clustering of events (Voronoi Tesselation and Percolation), independent of source size and shape. Good for faint, irregularly-shaped sources, but gets confused in crowded regions. Computationally prohibitive for many events or low contrast source/ background regions. Not extensively calibrated.

- WAVDETECT - Convolves image with circularly-symmetric wavelets at various scales. Works well in crowded fields and for point sources superimposed on extended emission, but careful selection of scales needed. Multiple blocked image/scale sets needed to analyze entire observation. Extensively calibrated.

How CELLDETECT Works

$C = \alpha S + B$
$T = \beta S + (b/d)^2 B$

$S = [((b^2-d^2)/d^2) C - Q]/[(b/d)^2 \alpha - \beta]$

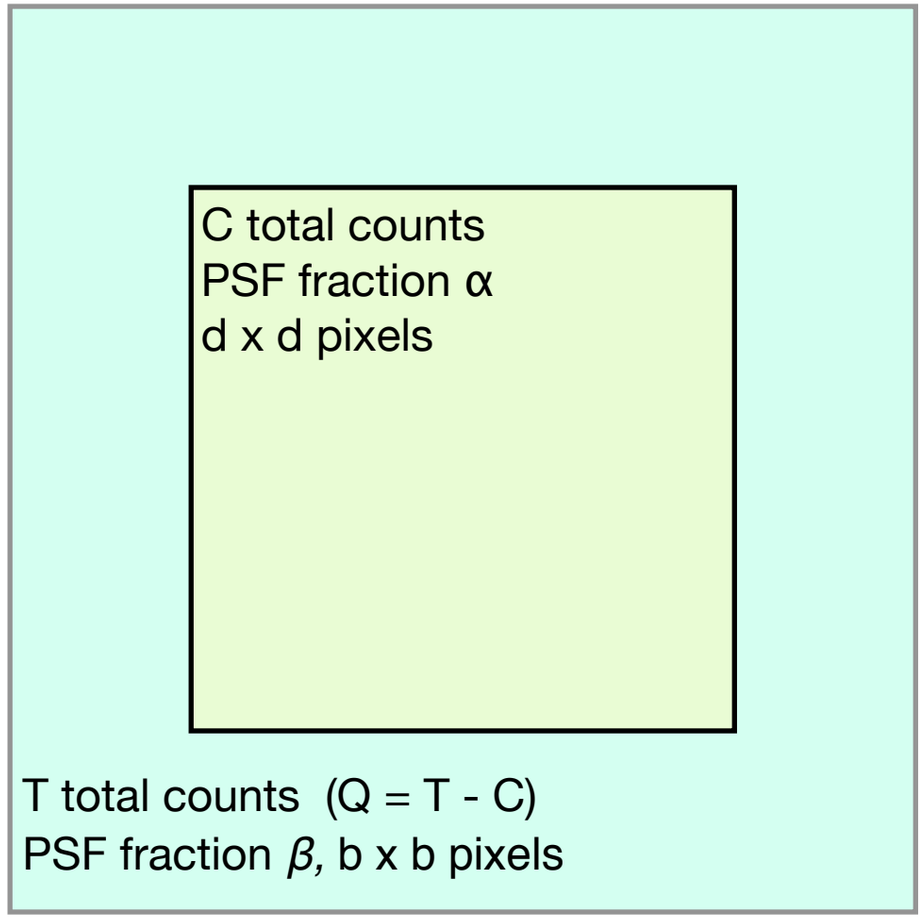
$\sigma^2_S = [((b^2-d^2)/d^2)^2 \sigma^2_C + \sigma^2_Q]/[(b/d)^2 \alpha - \beta]^2$

$SNR = S / \sigma_S$

Detect Cell “slides” by ⅓ of its size across image row, and computation repeated at each point; rows separated by ⅓ of cell size.

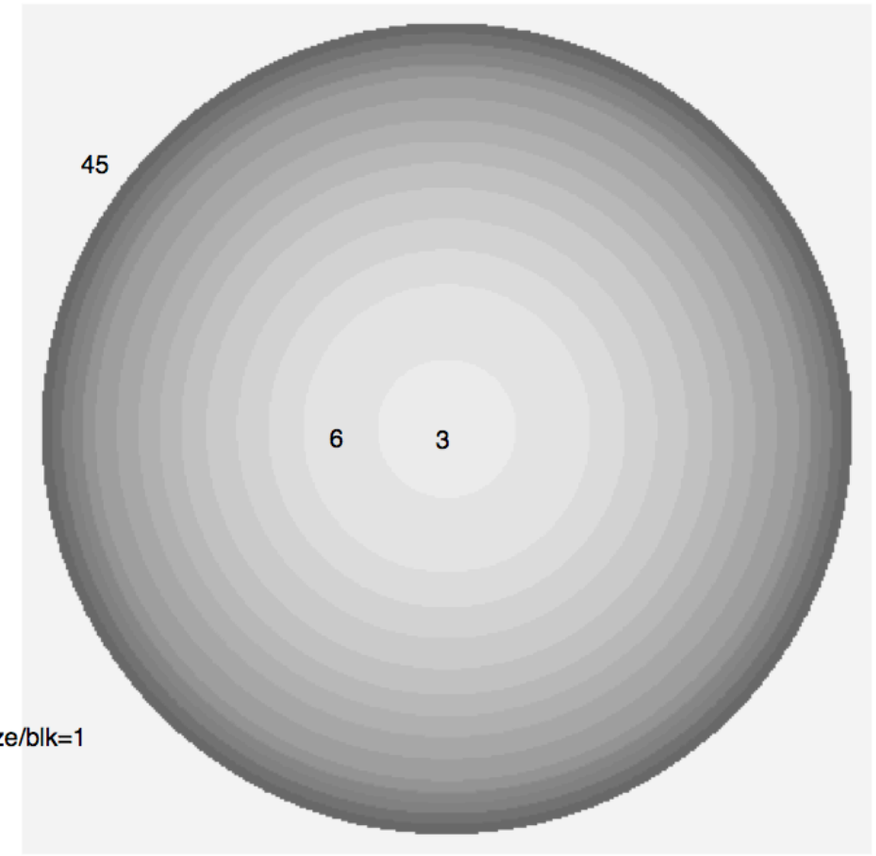
Cell locations with SNR > threshold are flagged and “grouped” into sources (local peaks).

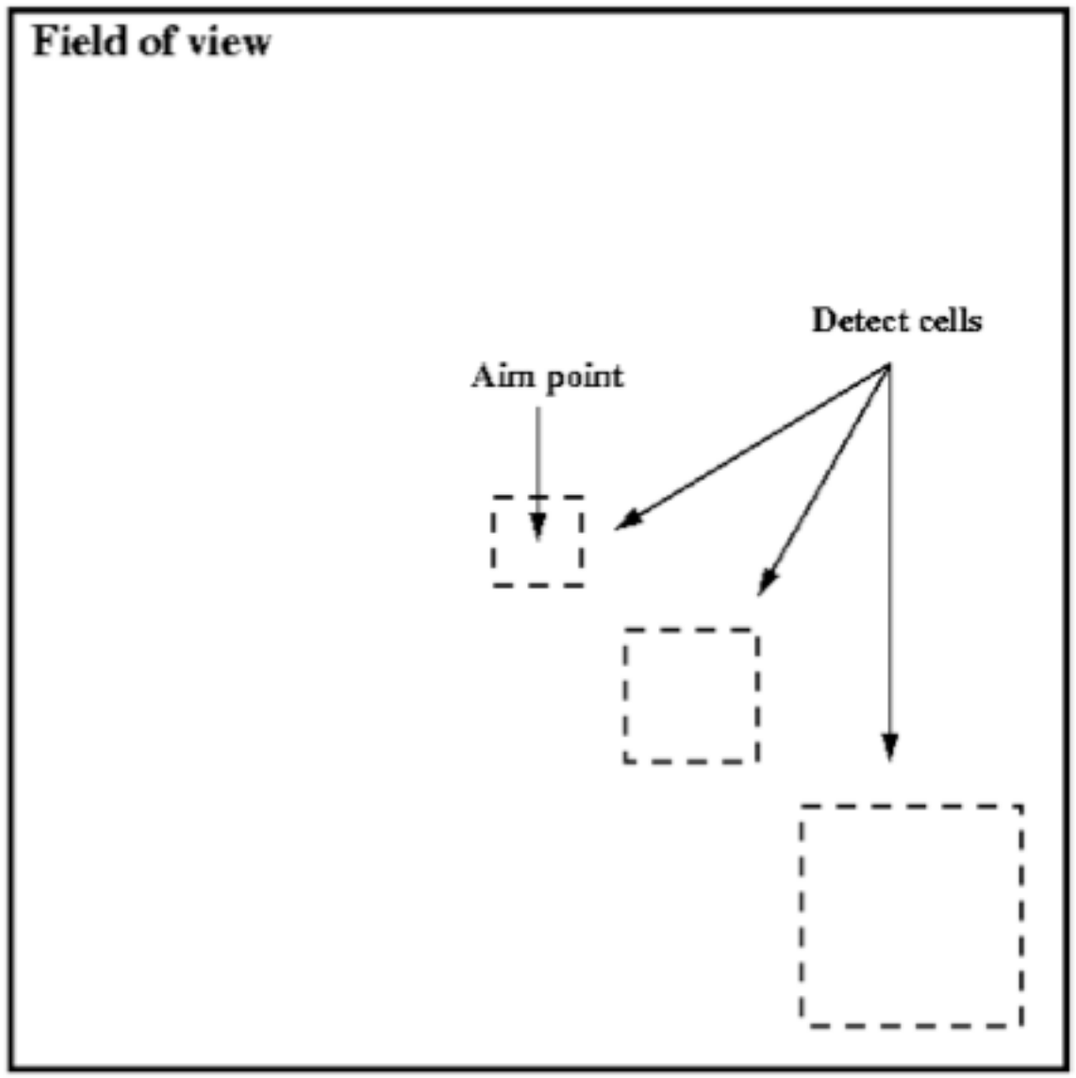
Source positions computed from centroid of events in local peak detect cell.



CELLDETECT - Variable Cell Sizes

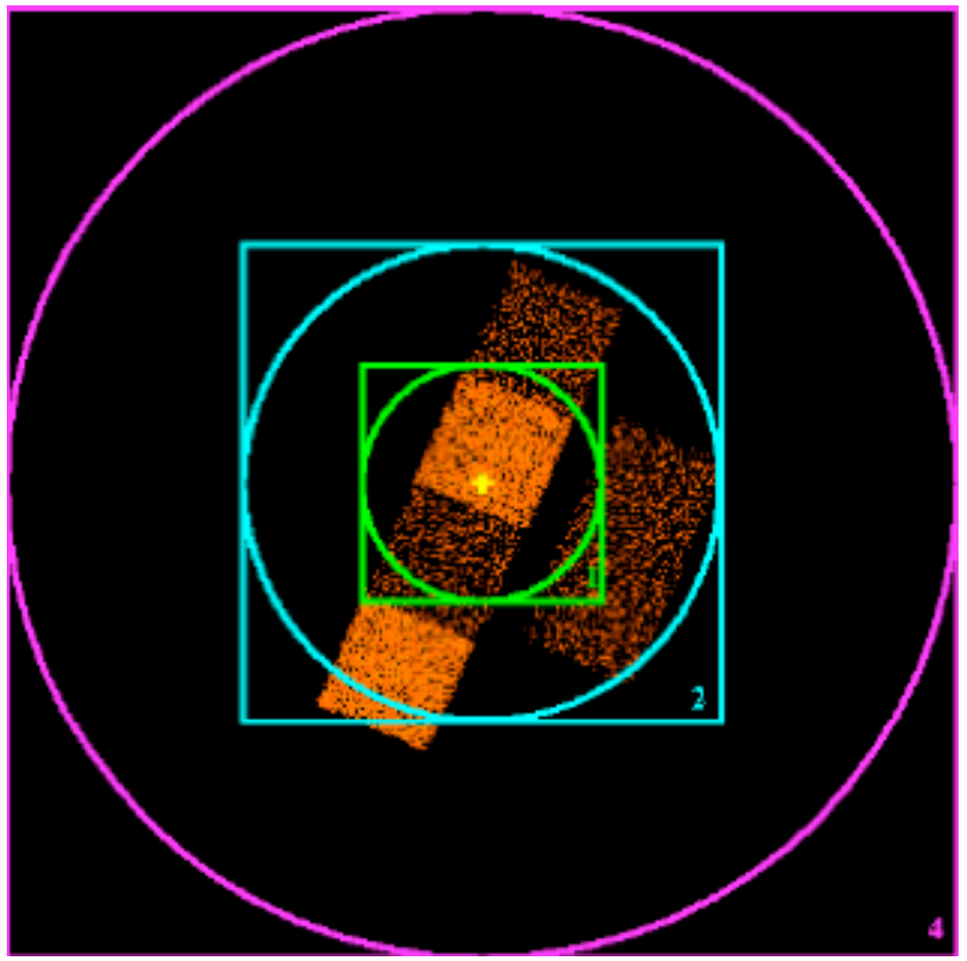
Detect cell size is based on PSF size. User can select what fraction of source counts should fall in cell (eenergy) and at what energy this is specified (eband).



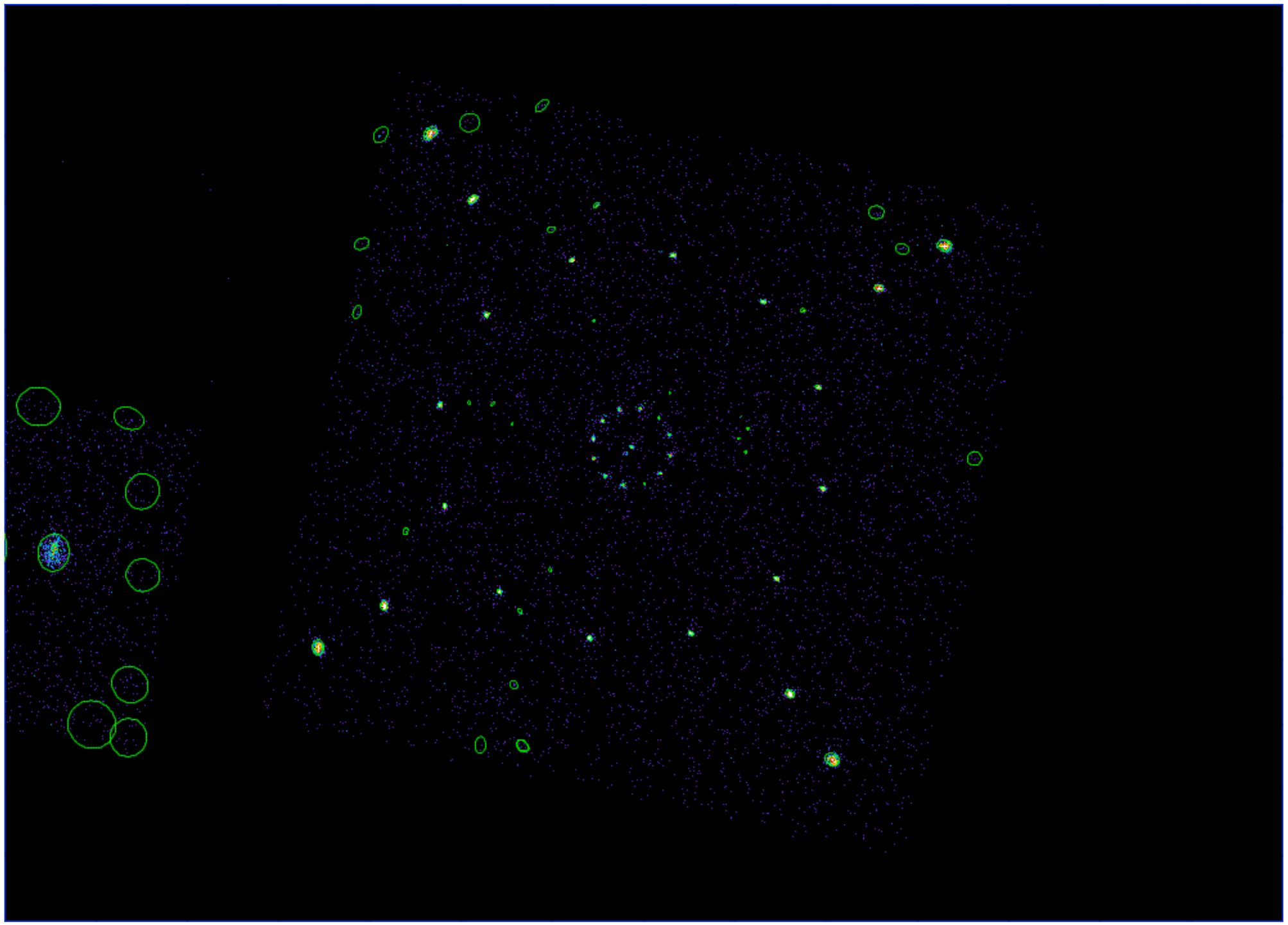


CELLDETECT - Recursive Blocking

- Center 2k x 2k scanned at full resolution, only selecting sources contained in inscribed circle;
- Center 4k x 4k scanned at block=2 resolution, excluding region included in previous step;
- Etc.

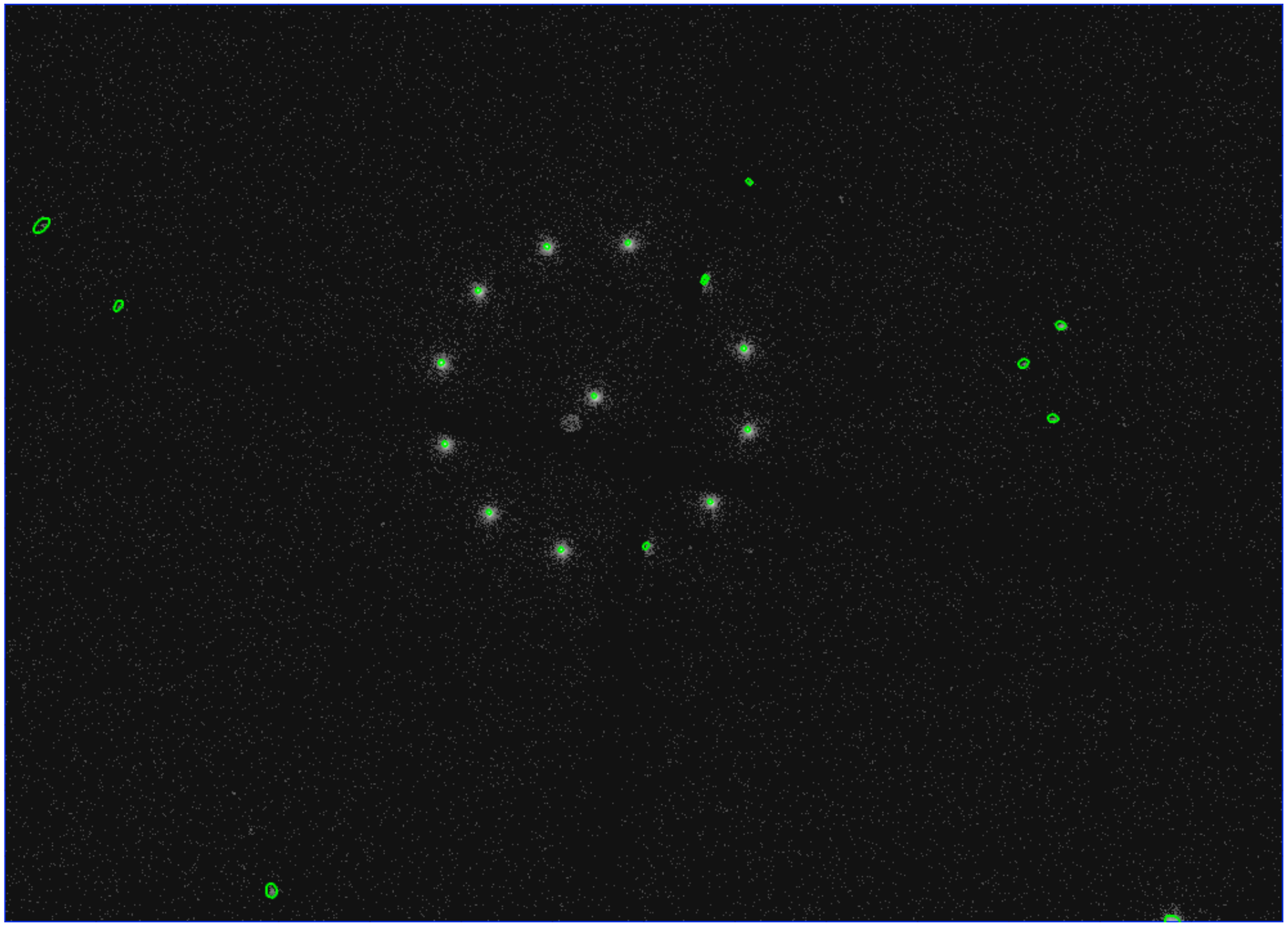


CELLDETECT on NGC 6822



celldetect infile="bl1_img.fits" expstk="" outfile="bl1_cell.fits" regfile="none" clobber="yes" thresh="3" findpeaks="yes" centroid="yes" ellsigma="3" expratio="0" fixedcell="0" xoffset="INDEF" yoffset="INDEF" eband="1.4967" eenergy="0.8" psftable="/soft/ciao-4.0.2/data/psfsize20010416.fits" cellfile="" bkgfile="" bkgvalue="0" bkgerrvalue="0" convolve="no" snrfile="" verbose="0" log="no"

CELLDETECT on NGC 6822

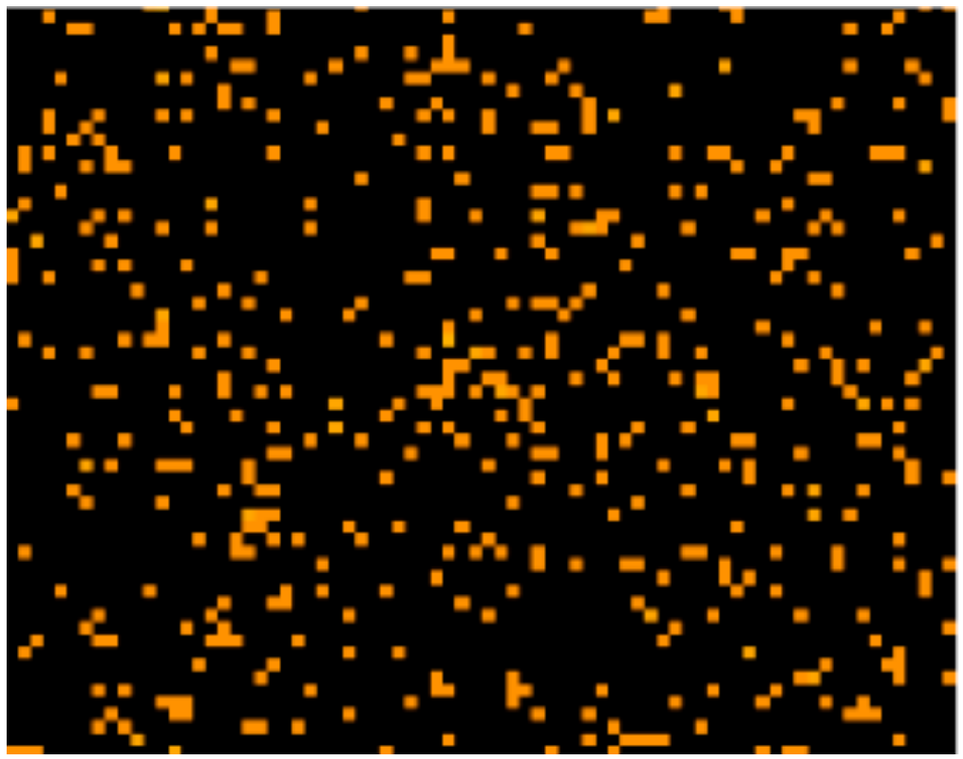


CELLDETECT: Pros and Cons

- Fast and Robust;
- Works well for point sources;
- Detailed PSF shape not needed, only approximate size;
- Can swallow entire observation in one gulp.

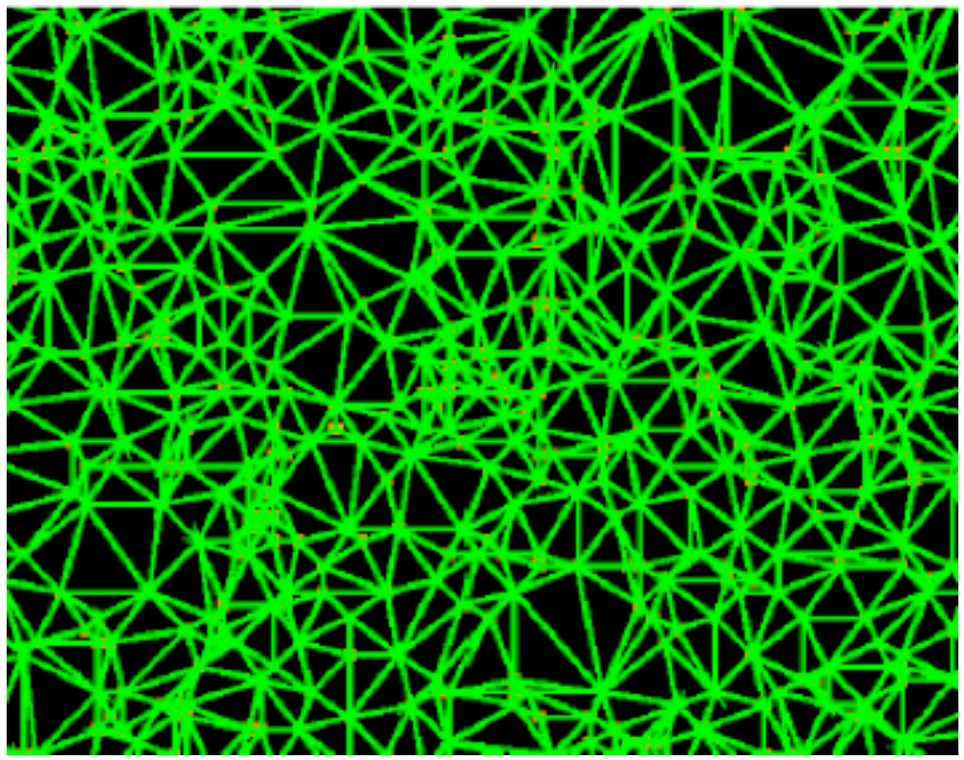
- Extended sources can be difficult, without careful selection of cell sizes;
- Can get confused in crowded fields;
- Exposure maps needed to eliminate edge sources;
- Not very sensitive, unless background maps are used, but these can be difficult to make.

VTPDETECT - How It Works



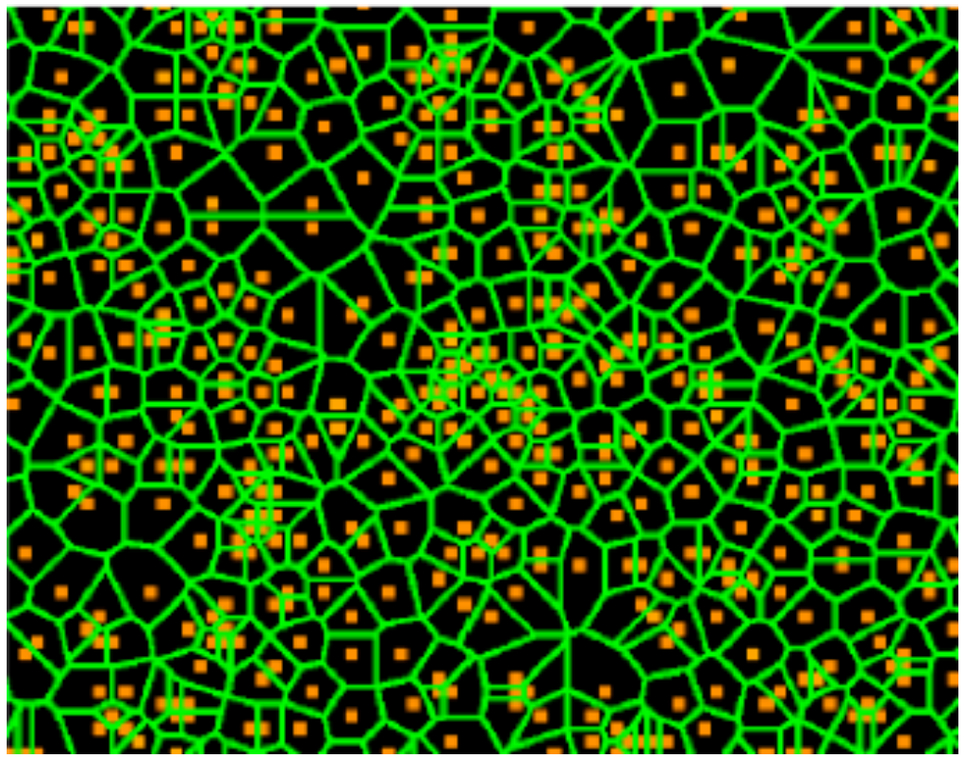
- Consider a ~75 x 50 pixel segment of an ACIS observation of 3C295

VTPDETECT - How It Works



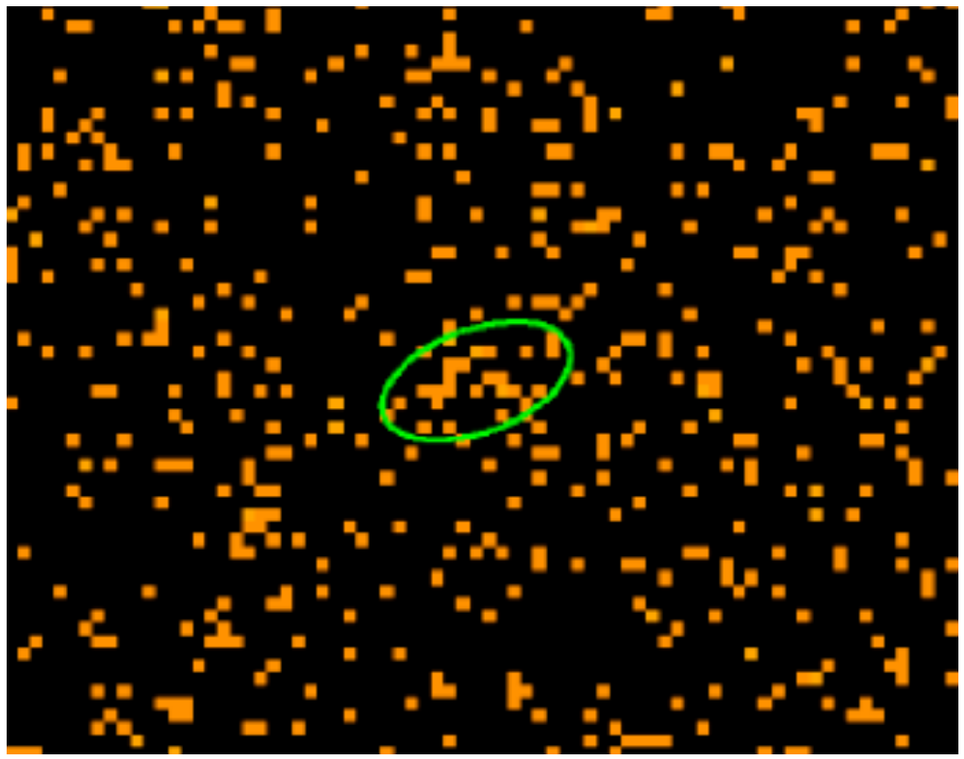
- Consider a ~75 x 50 pixel segment of an ACIS observation of 3C295
- Construct a triangulation network from all the events in the region

VTPDETECT - How It Works



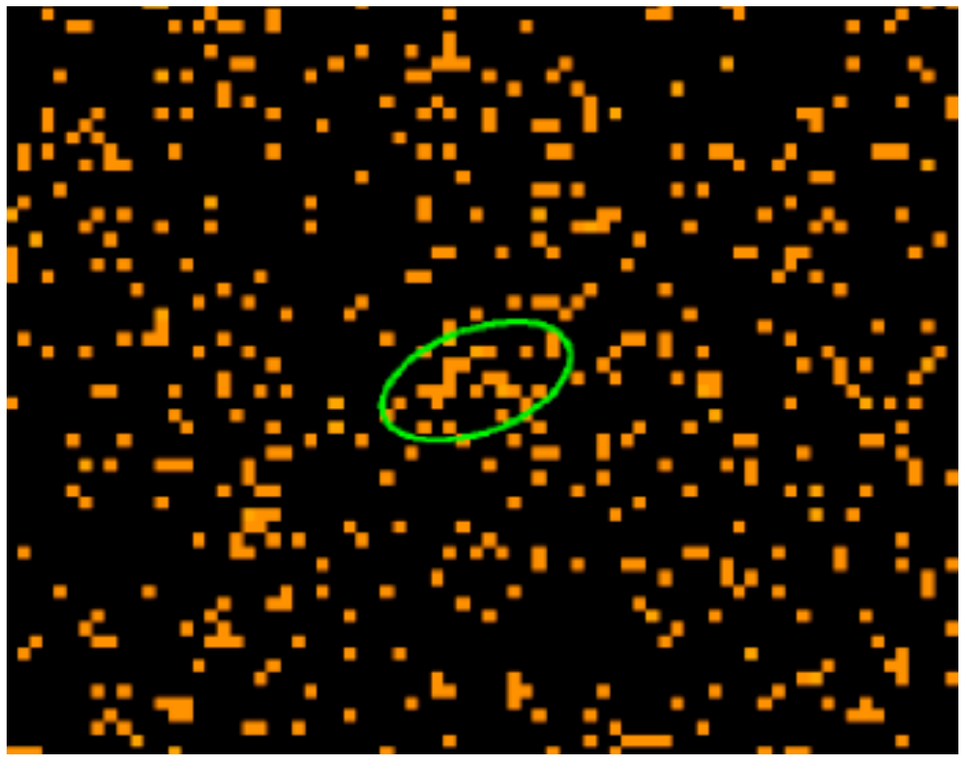
- Consider a ~75 x 50 pixel segment of an ACIS observation of 3C295
- Construct a triangulation network from all the events in the region
- Construct cells about each event by connecting centers of triangles (Voronoi Tesselation)
- Compare distribution of cell inverse areas with that derived from simulations of blank fields to set inverse area thresholds

VTPDETECT - How It Works



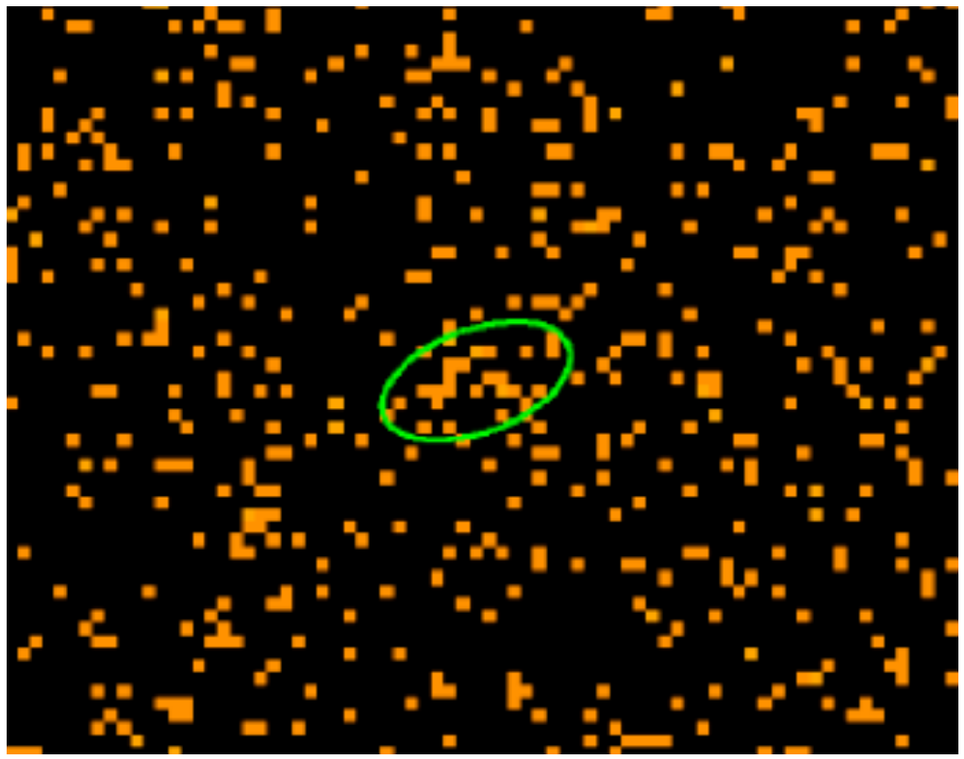
- Consider a ~75 x 50 pixel segment of an ACIS observation of 3C295
- Construct a triangulation network from all the events in the region
- Construct cells about each event by connecting centers of triangles (Voronoi Tesselation)
- Compare distribution of cell inverse areas with that derived from simulations of blank fields to set inverse area thresholds
- Group neighboring cells above threshold into source candidates (Percolation)

VTPDETECT - How It Works



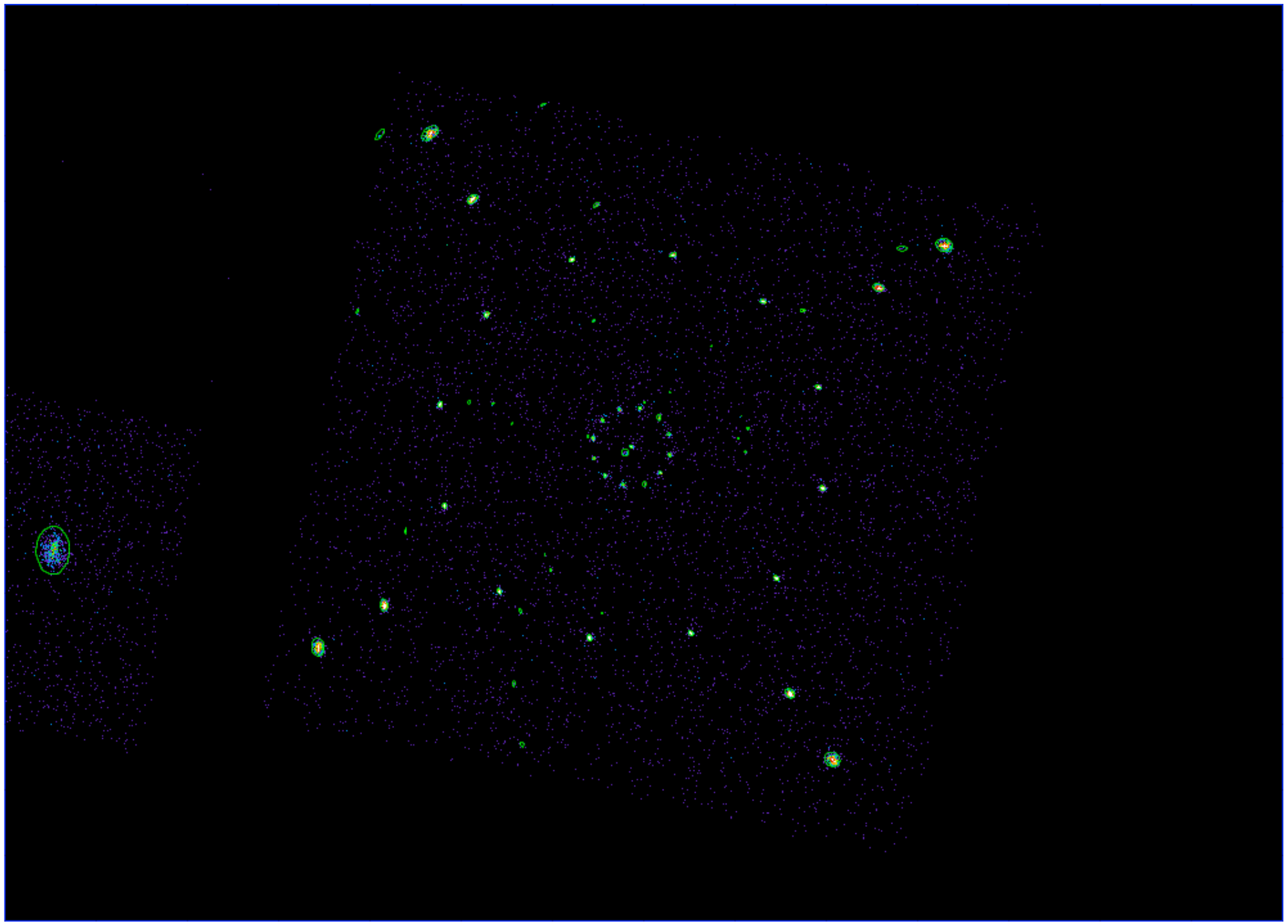
- Consider a ~75 x 50 pixel segment of an ACIS observation of 3C295
- Construct a triangulation network from all the events in the region
- Construct cells about each event by connecting centers of triangles (Voronoi Tesselation)
- Compare distribution of cell inverse areas with that derived from simulations of blank fields to set inverse area thresholds
- Group neighboring cells above threshold into source candidates (Percolation)
- Candidates with ≥ user-specified number of events are sources

VTPDETECT - How It Works



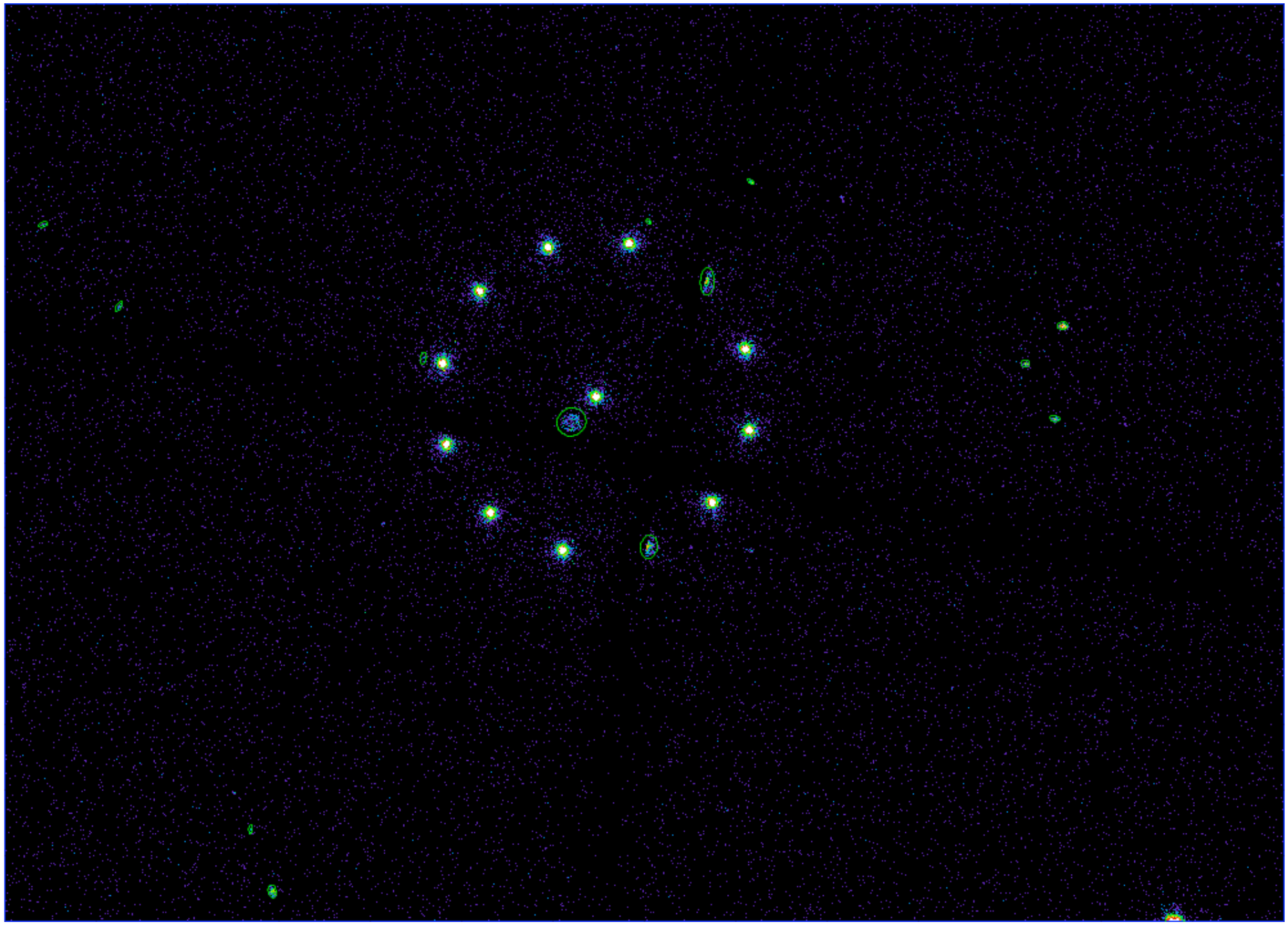
- Consider a ~75 x 50 pixel segment of an ACIS observation of 3C295
- Construct a triangulation network from all the events in the region
- Construct cells about each event by connecting centers of triangles (Voronoi Tesselation)
- Compare distribution of cell inverse areas with that derived from simulations of blank fields to set inverse area thresholds
- Group neighboring cells above threshold into source candidates (Percolation)
- Candidates with ≥ user-specified number of events are sources
- Source positions are inverse-area weighted means of event positions

VTPDETECT on NGC 6822

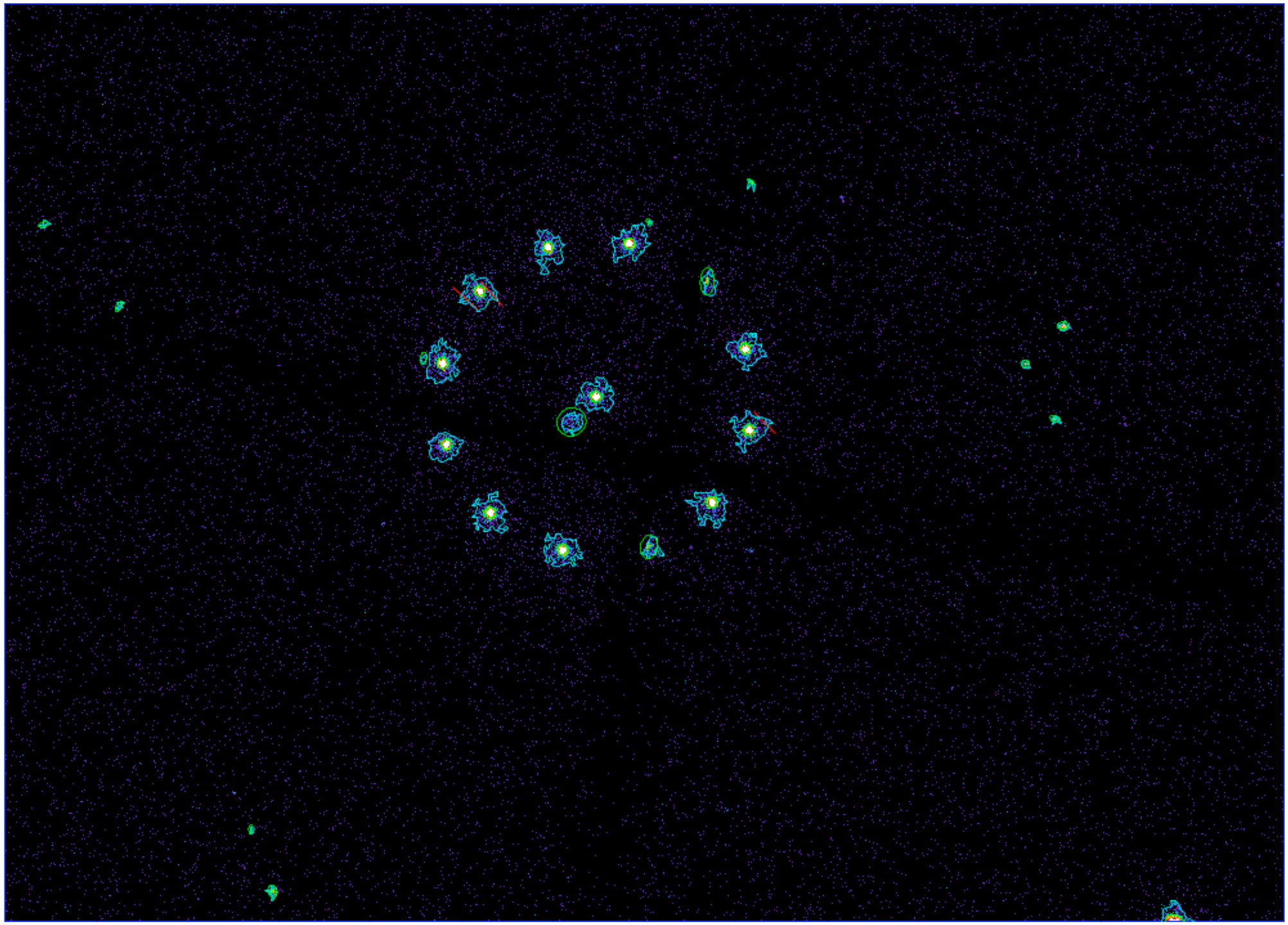


vtpdetect infile="bl1_img.fits" expfile="none" outfile="bl1_vtp.fits" regfile="none" ellsigma="3" scale="1" limit="1e-06" coarse="10" maxiter="10" edge="2" superdo="no" maxbkgflux="0.8" mintotflux="0.8" maxtotflux="2.6" mincutoff="1.2" maxcutoff="3" fittol="1e-06" fitstart="1.5" clobber="no" verbose="0" logfile="stderr"

VTPDETECT on NGC 6822



VTPDETECT on NGC 6822



VTPDETECT: Pros and Cons

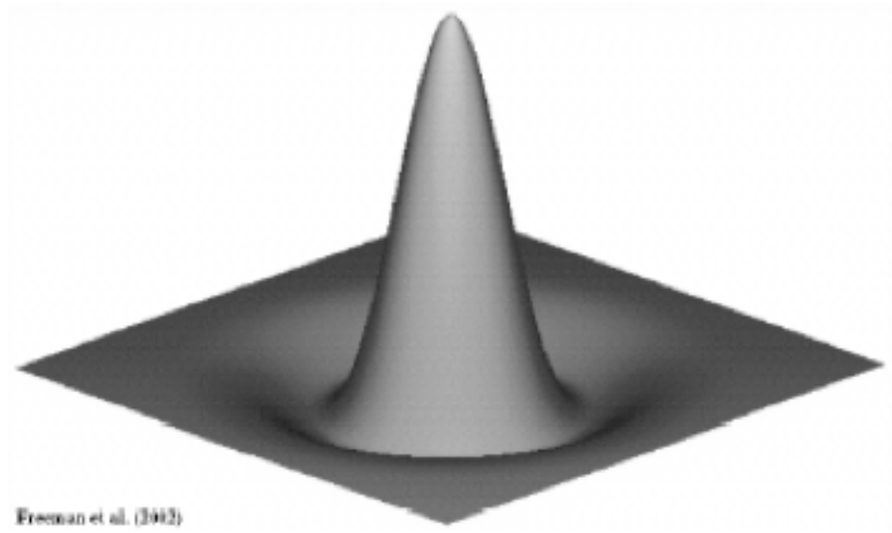
- Does not assume anything about source size/shape, and so works well for extended and/or irregularly-shaped sources.
- It's photon-based, and can work on large areas at full resolution.
- It works well for low-surface-brightness extended sources.

- Does not assume anything about source size/shape, and so can get confused in crowded fields.
- It's very slow if the number of photons is large and if the contrast between background and sources is low.

How WAVDETECT (wtransform) Works

- Wavelets are oscillatory scaleable functions which are non-zero within a limited spatial regime, and have zero normalization:

$W_{a,\sigma}(x) = \sigma^{-1} W[(x-a)/\sigma]$



- Image is convolved with wavelet at a given scale:

$$C(\sigma,x,y) = \iint dx' dy' \, W(\sigma, x-x', y-y') D(x', y')$$
$$= \langle W^{+} \otimes D \rangle + \langle W^{-} \otimes D \rangle$$

- Use $\langle W^{-} \otimes D \rangle$ to estimate background $B(x,y)$
- Compute $S(x,y)$, probability of getting $C(\sigma,x,y)$ or higher, given $B(x,y)$:

$S(x,y) = \int_{C}^{\infty} dC \, P(C|B)$

- Identify Source Pixels when $S(x,y) \leq$ user-defined threshold
- Use source pixels to get a better idea of $B(x,y)$ and iterate.

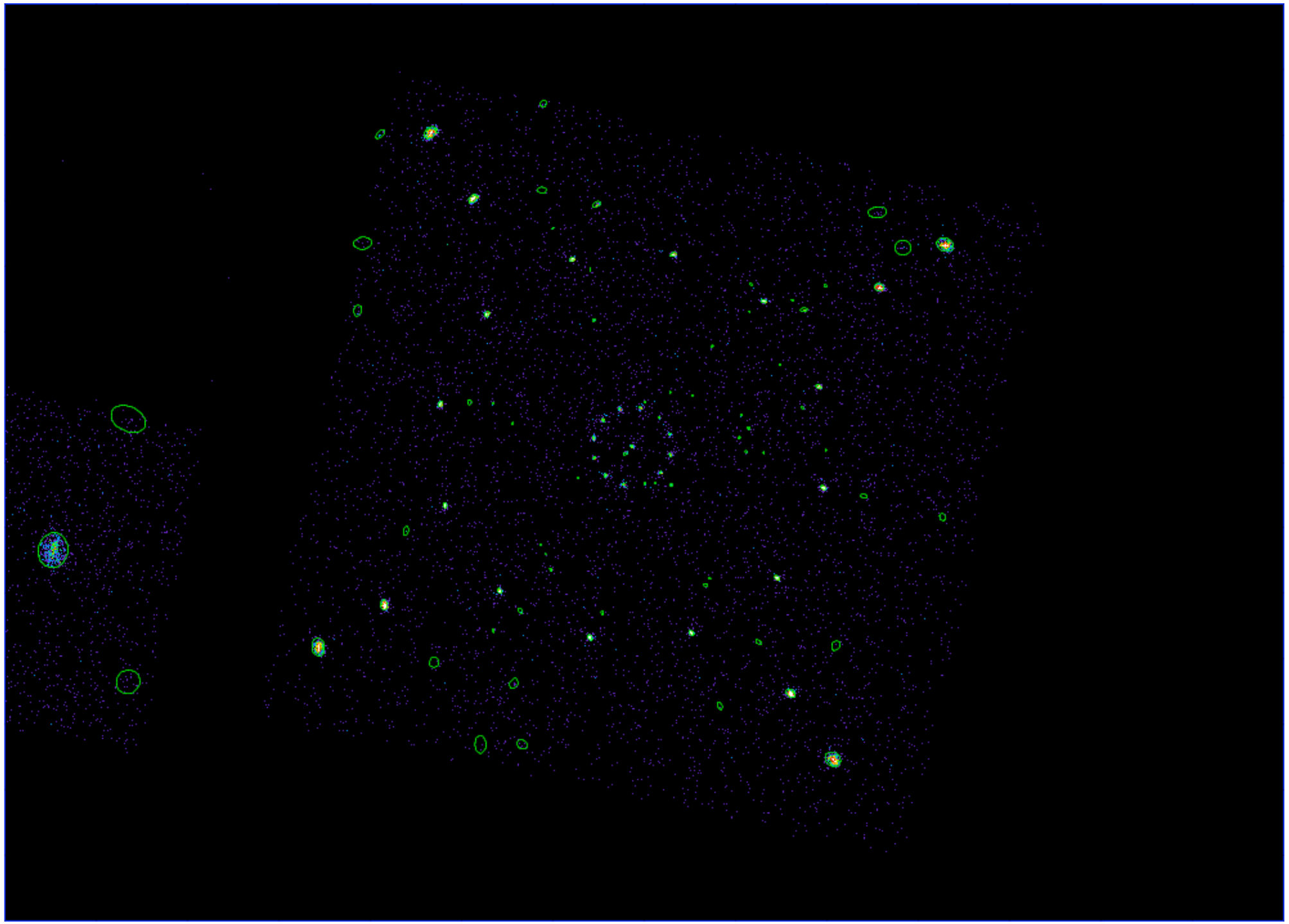
How WAVDETECT (wrecon) Works

- Construct net counts or "flux" images at scales used to compute $C(\sigma,x,y)$:

  $F(\sigma,x,y) = max(<W^{+} \otimes D> - B(x,y),0)$

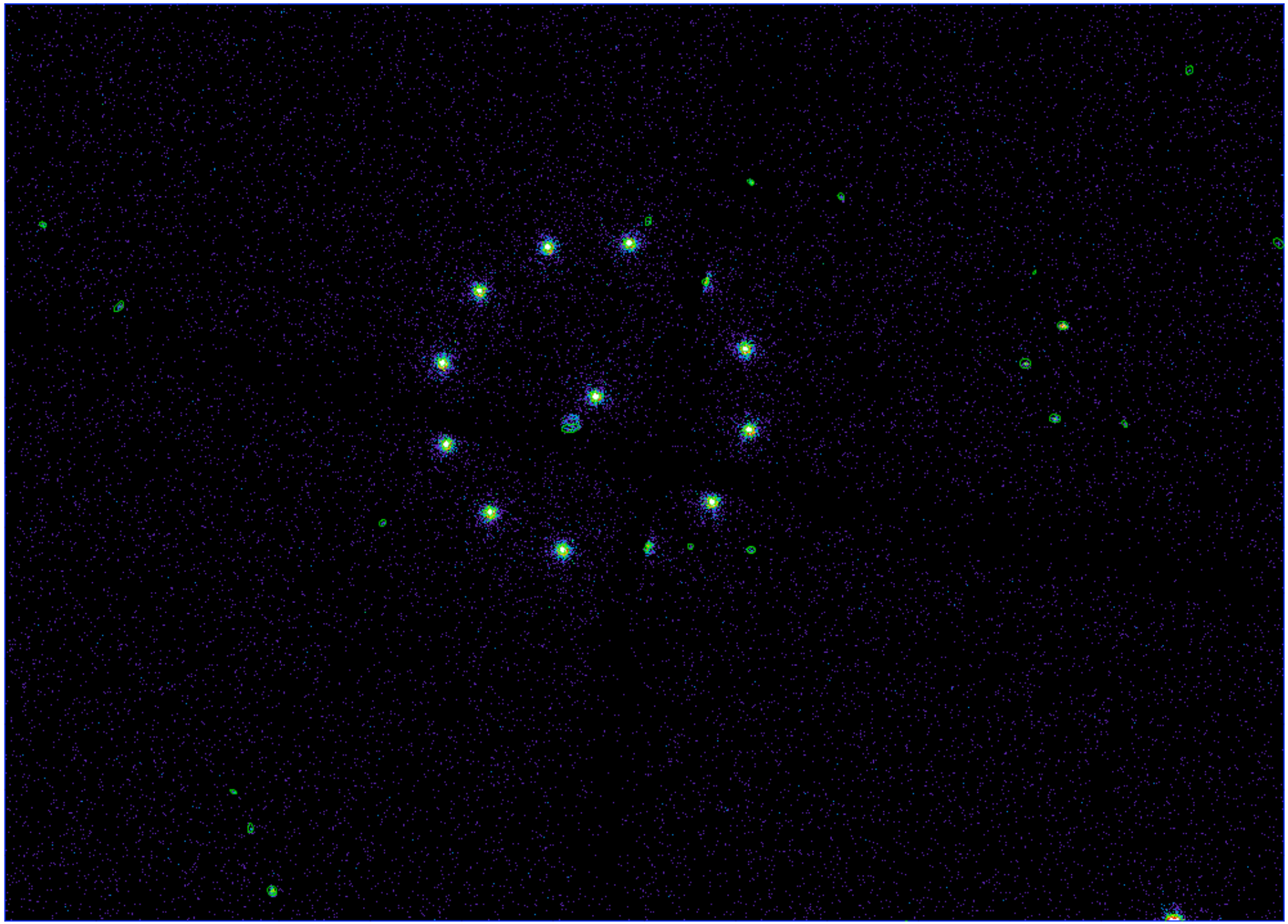
- Find local maxima in F corresponding to maxima in C
- Assign all pixels where $F > 0$ to source cell, using scale closest to PSF size, to source "cell"
- Use source cell to compute source properties; in particular, position is count-weighted average of pixel values in cell
- BE CAREFUL:
  - Candidates are identified from correlation maxima at all scales, in ascending order of scale size
  - Maxima at larger scales near smaller scale sources are associated with them

WAVDETECT on NGC 6822

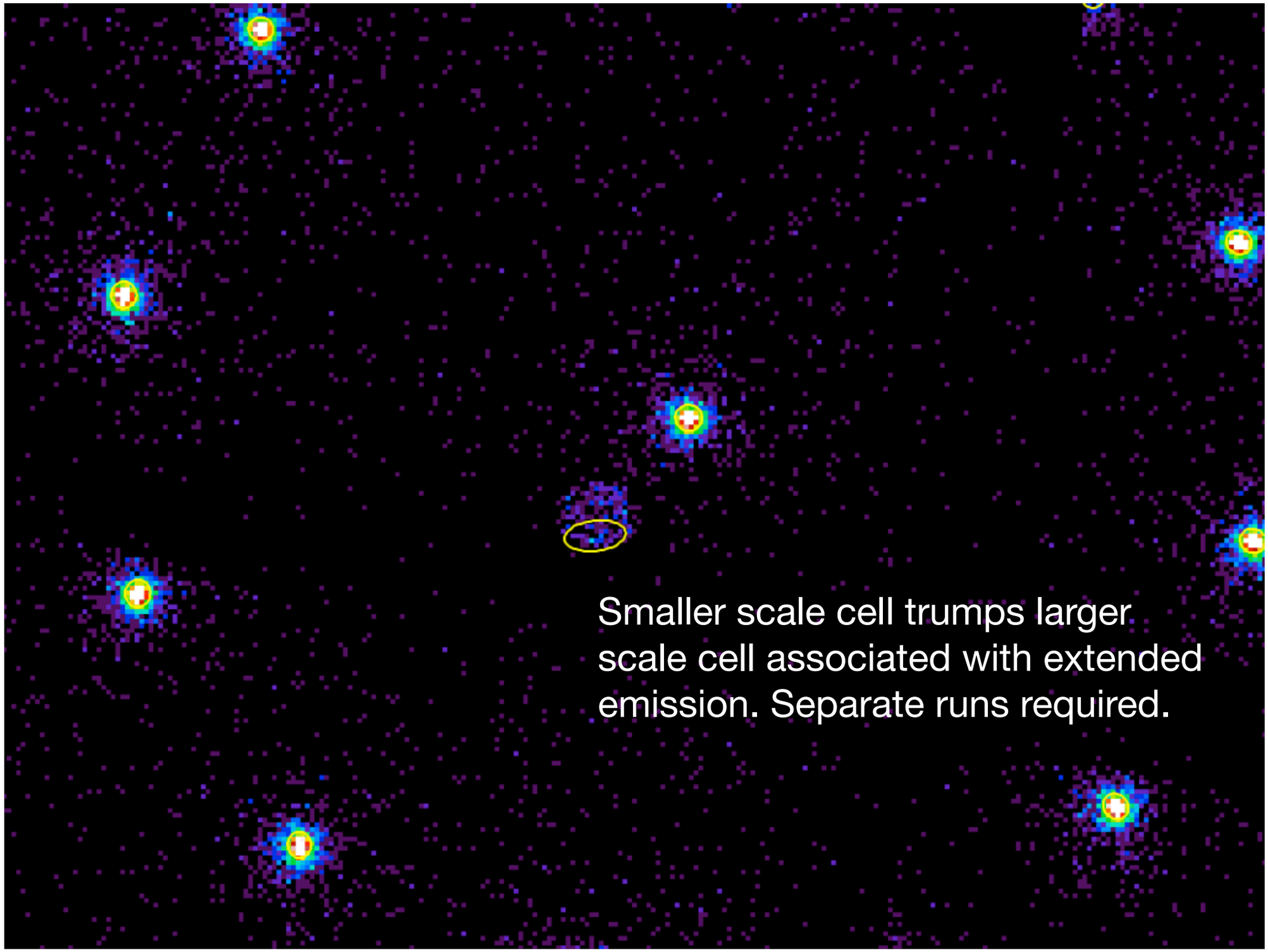


wavdetect infile="bl1_img.fits" outfile="bl1_wav.fits" scellfile = "/tmp/cell.fits" imagefile = "/tmp/img.fits" defnbkgfile = "/tmp/bkg.fits" scales = "1 2 4 8 16" regfile="none" sigthresh=5.96e-8

WAVDETECT on NGC 6822



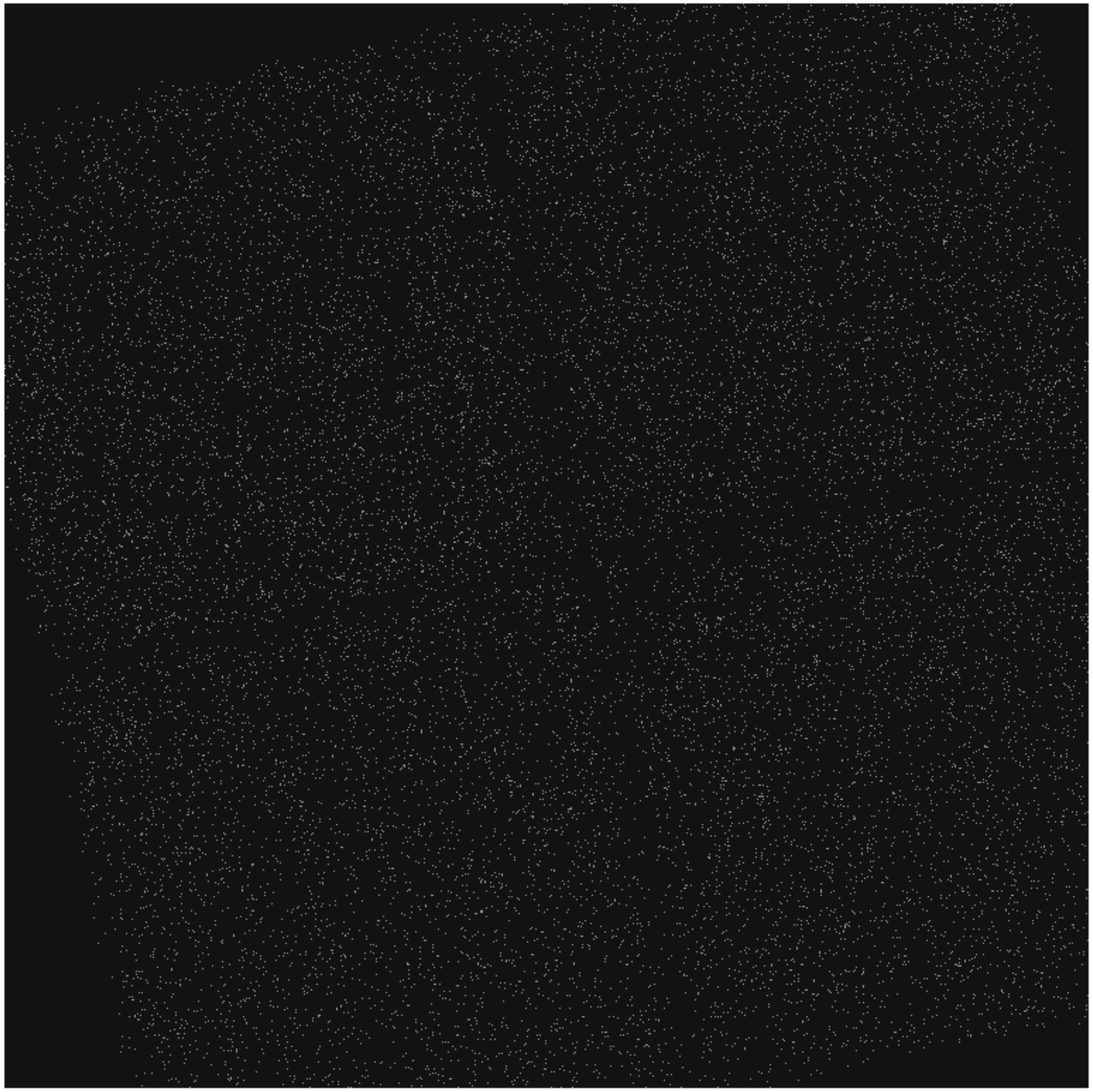
WAVDETECT on NGC 6822



WAVDETECT Calibration

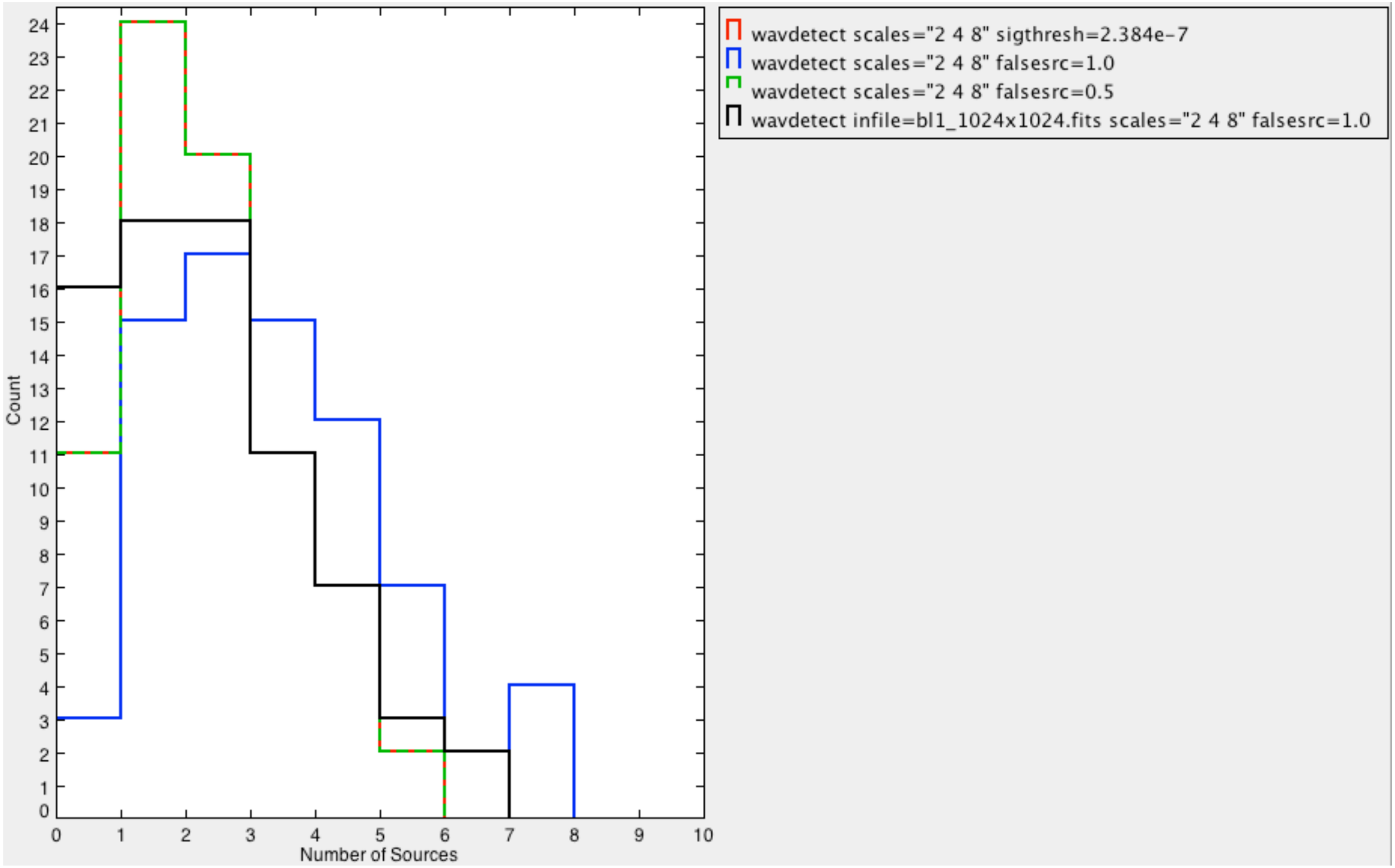
- Type I Errors: false source detections, determined from simulations of blank fields, by wavdetect authors (to set detect threshold parameters) or by users
- Type II errors: missing source detections, determined from simulations of point sources with different counts, positions, backgrounds
- Position Uncertainties
  - Statistical - simulated point sources with typical flux distribution, distributed throughout fields
  - Systematic - very bright sources at specific values of θ, ϕ

WAVDETECT Calibration: Type I Errors



- ACIS-I 125 ksec. obs.
- 2048 x 2048 image
- block = 1

WAVDETECT Calibration: Type I Errors



WAVDETECT Calibration: Type I Errors

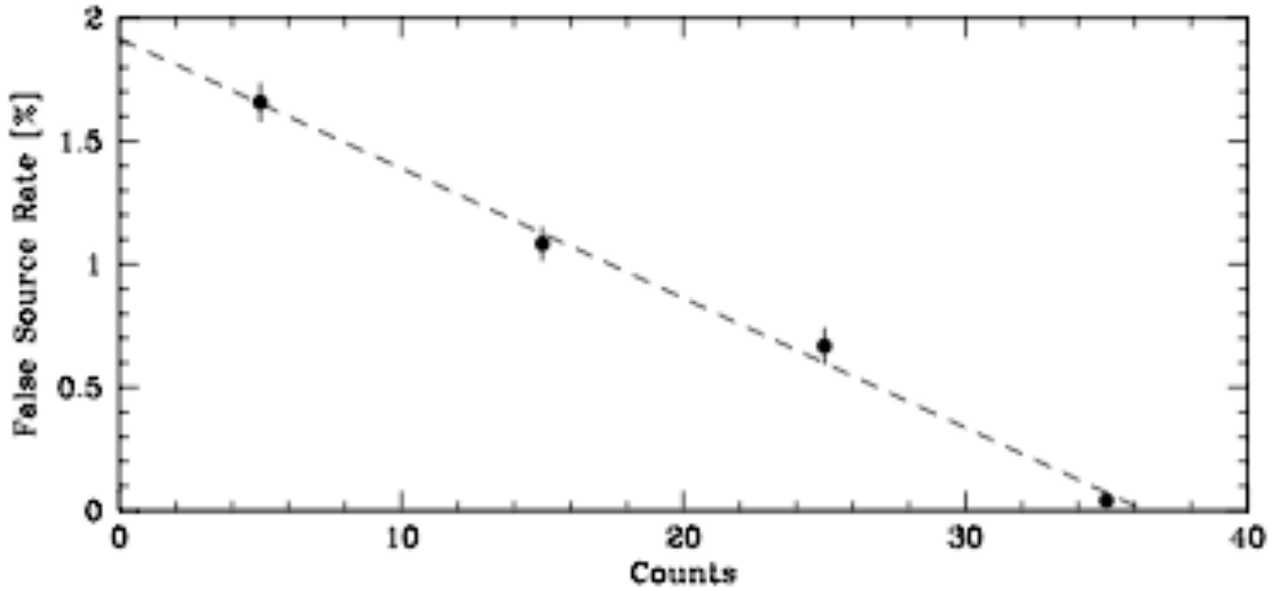


FIG. 13.—False source detection rate as a function of source counts in the B-band extracted by `wavdetect` (*top*) and off-axis angle (*bottom*). ~1% of the total detected sources are spurious sources, and 80% of spurious sources have counts less than ~30. The false source detection rate increases as the source counts decrease and as the off-axis angle increases. The dashed lines indicate the best linear least-squares fit results (see § 4.3.2).

WAVDETECT Calibration: Type II Errors

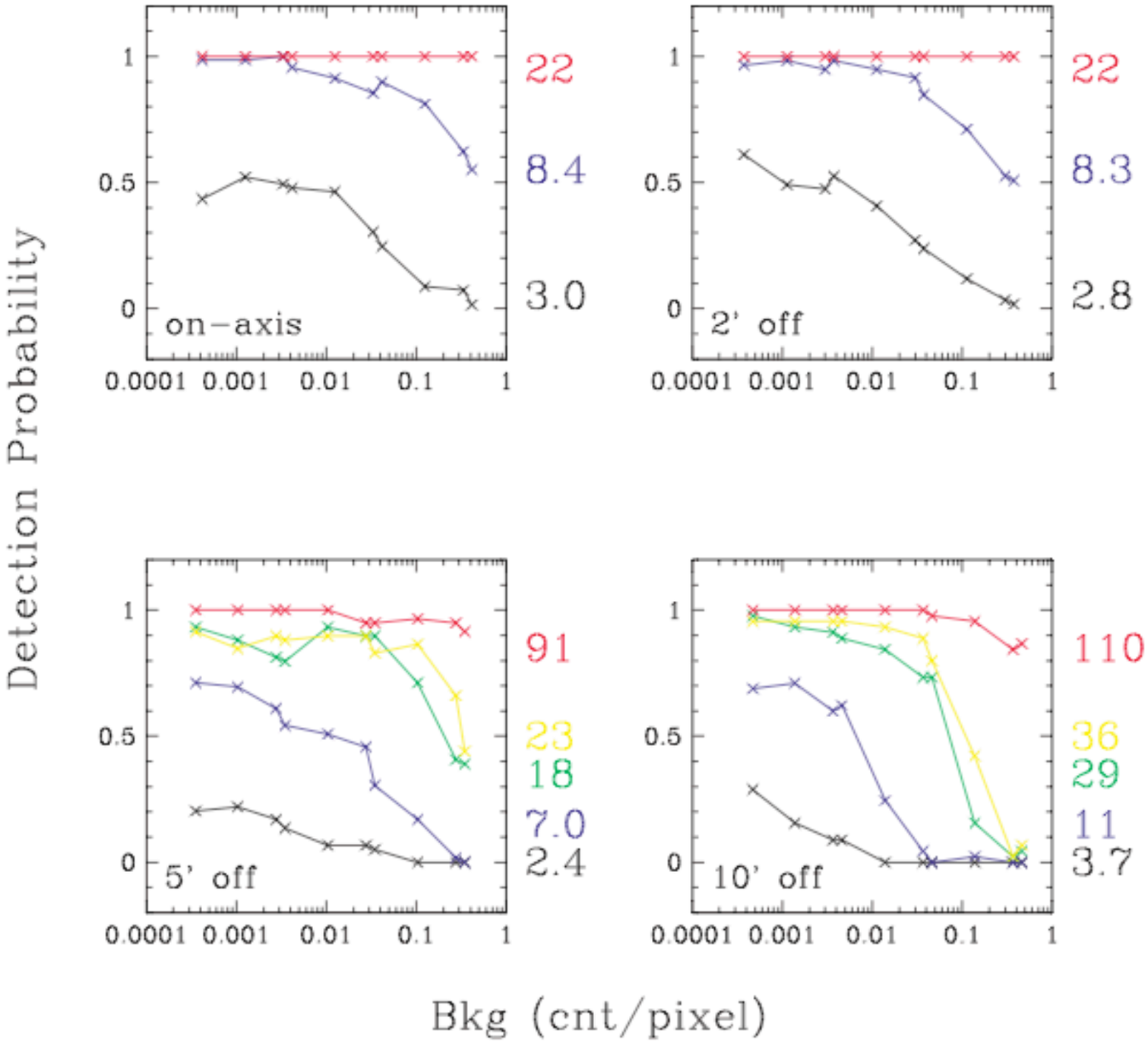
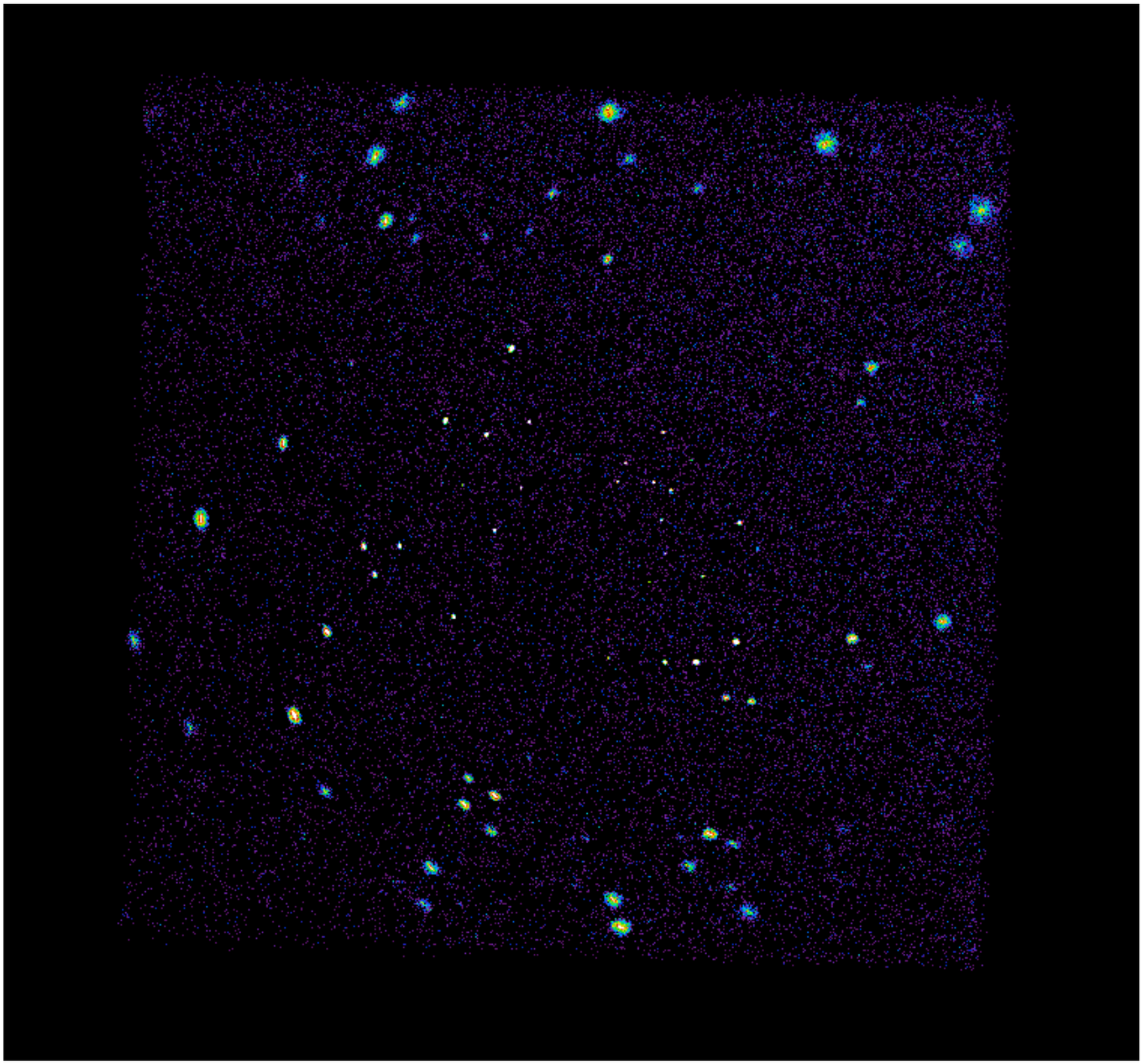


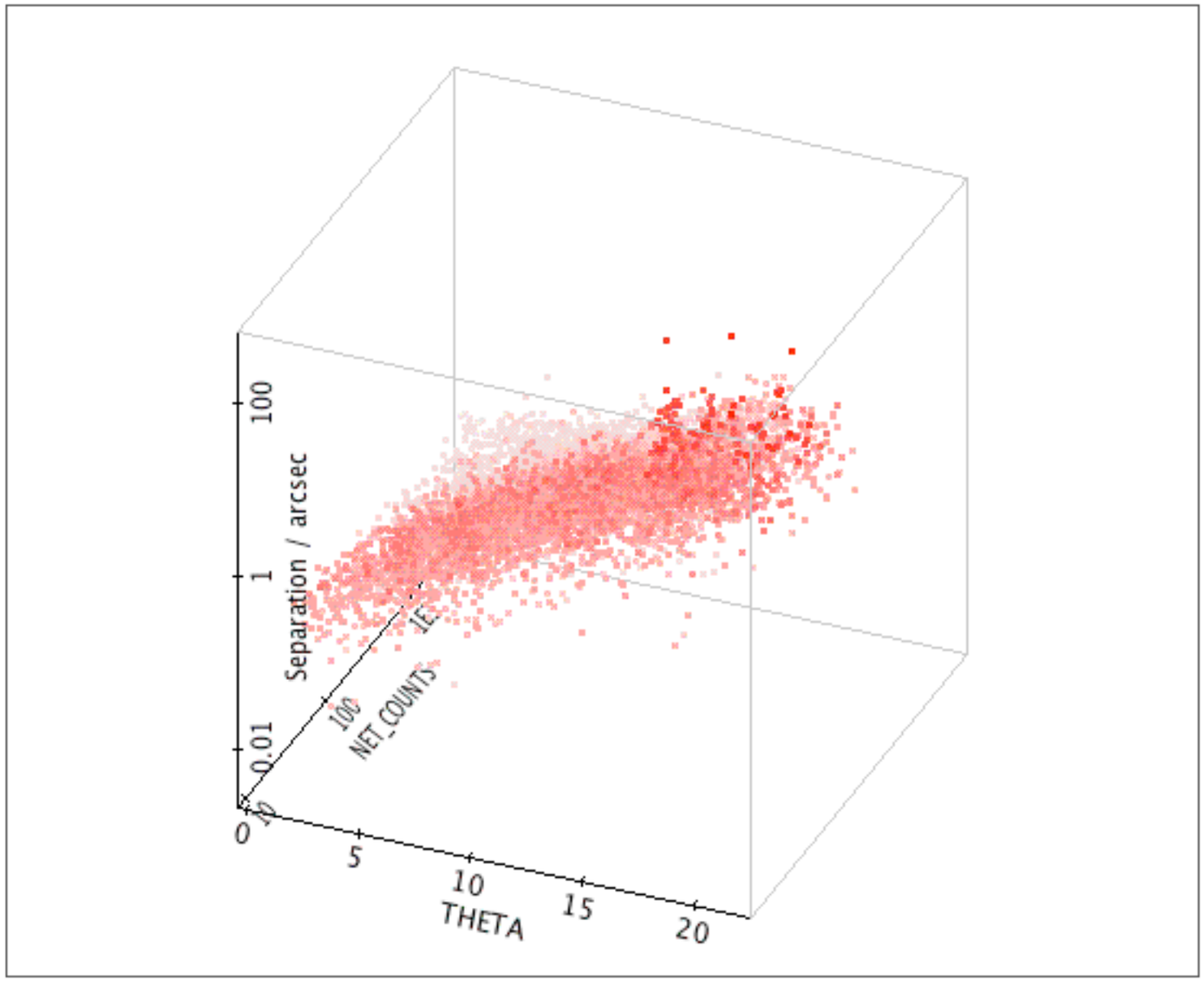
FIG. 9.—Detection probability as a function of background counts with various source counts (from a few to ~100, indicated at the right side of figures) and off-axis distances: (*a*) on-axis, (*b*) 2′ off-axis, (*c*) 5′ off-axis, and (*d*) 10′ off-axis.

WAVDETECT: Statistical Position Uncertainties

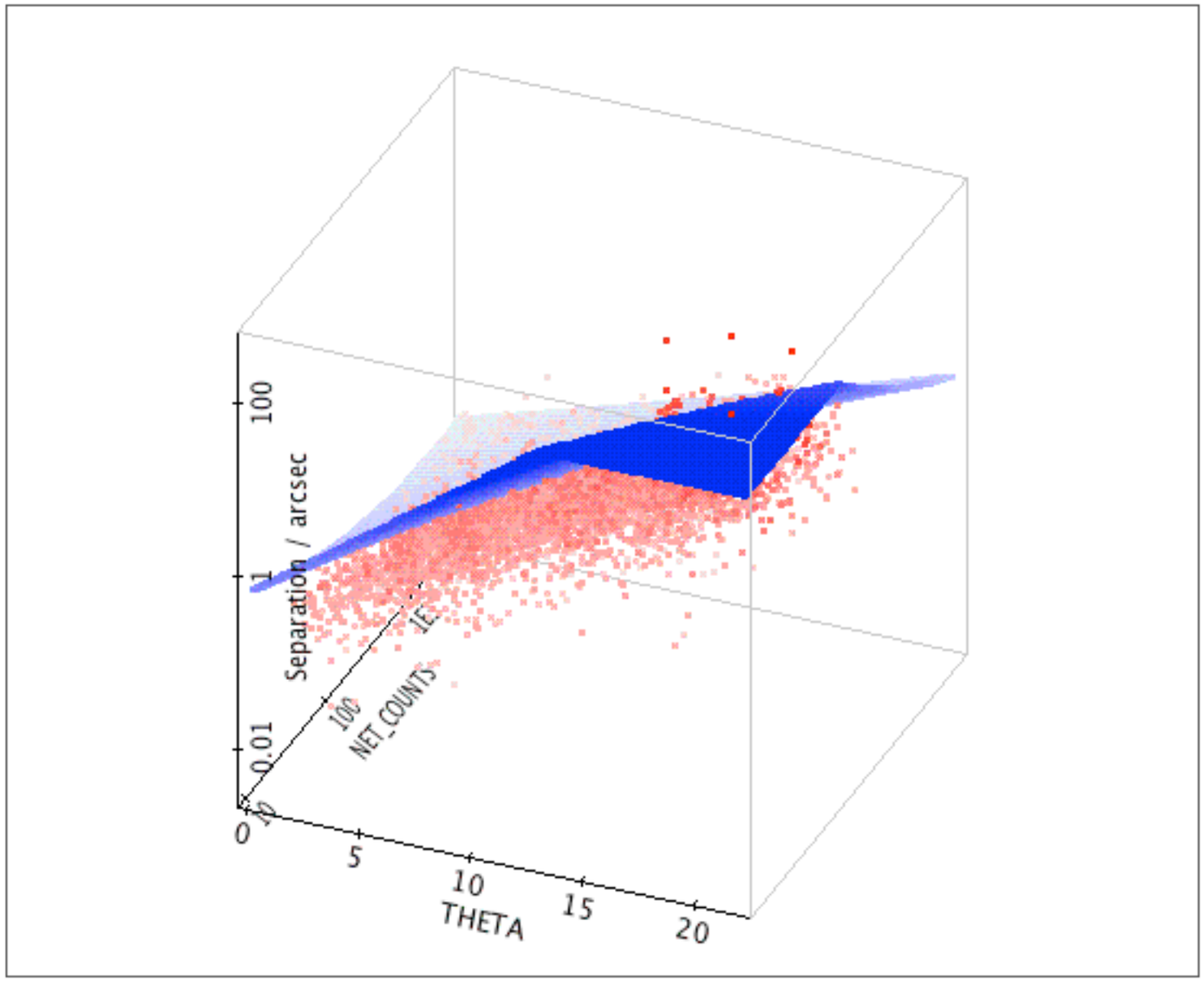


- 50 Simulations of HRC-I
- OBSID 1912 meta-data (~50 ksec).
- Source Counts range from ~10 - ~4000
- ~6600 sources simulated

WAVDETECT: Statistical Position Uncertainties



WAVDETECT: Statistical Position Uncertainties

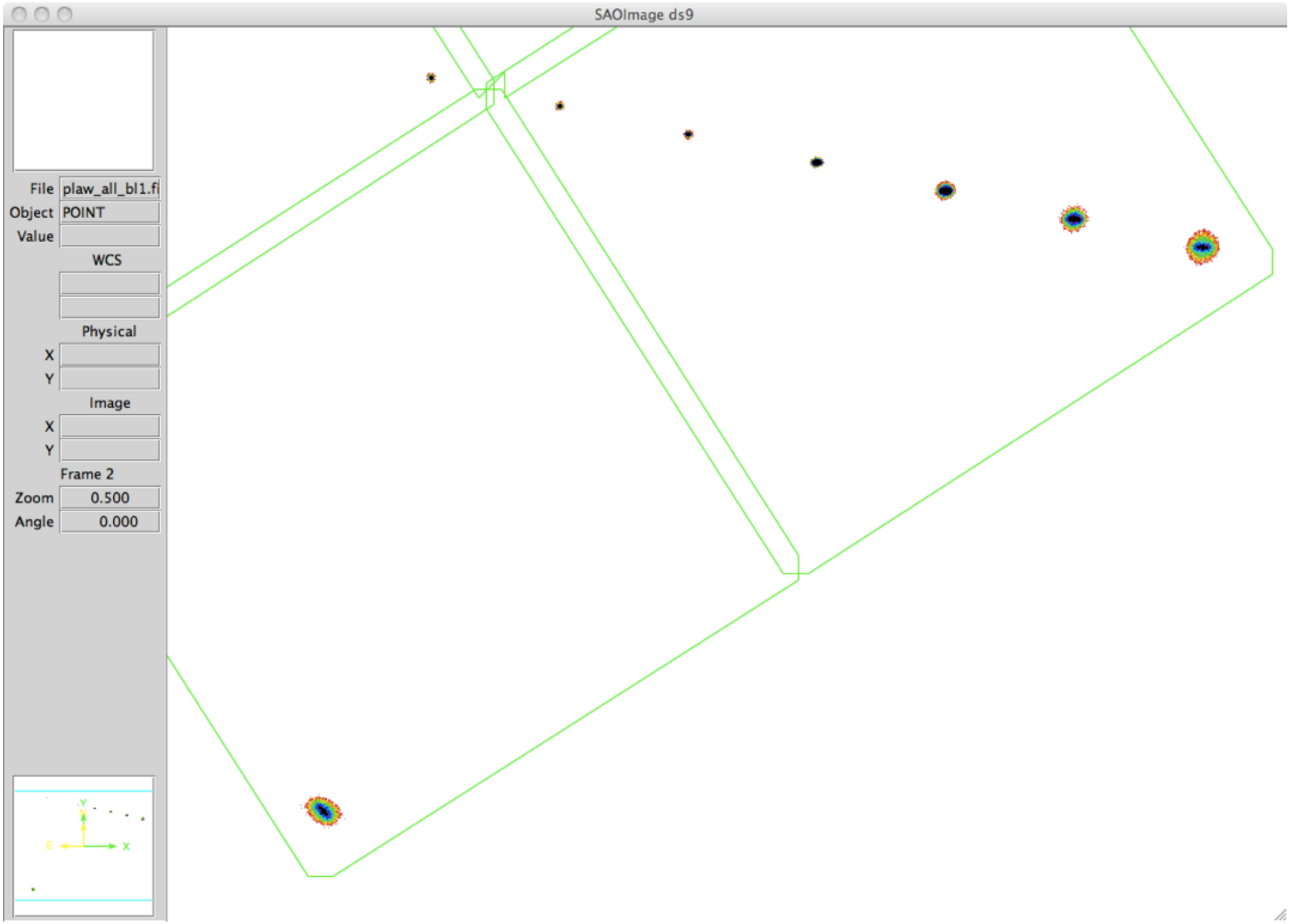


$$\log \sigma_{95} = \min(\log(134.398), 0.752569 + 0.216985 \times \theta + 0.000242 \times \theta^2 \; -1.142476 \times \log C + 0.172132 \times (\log C)^2 + -0.040549 \times \theta \times \log C)$$

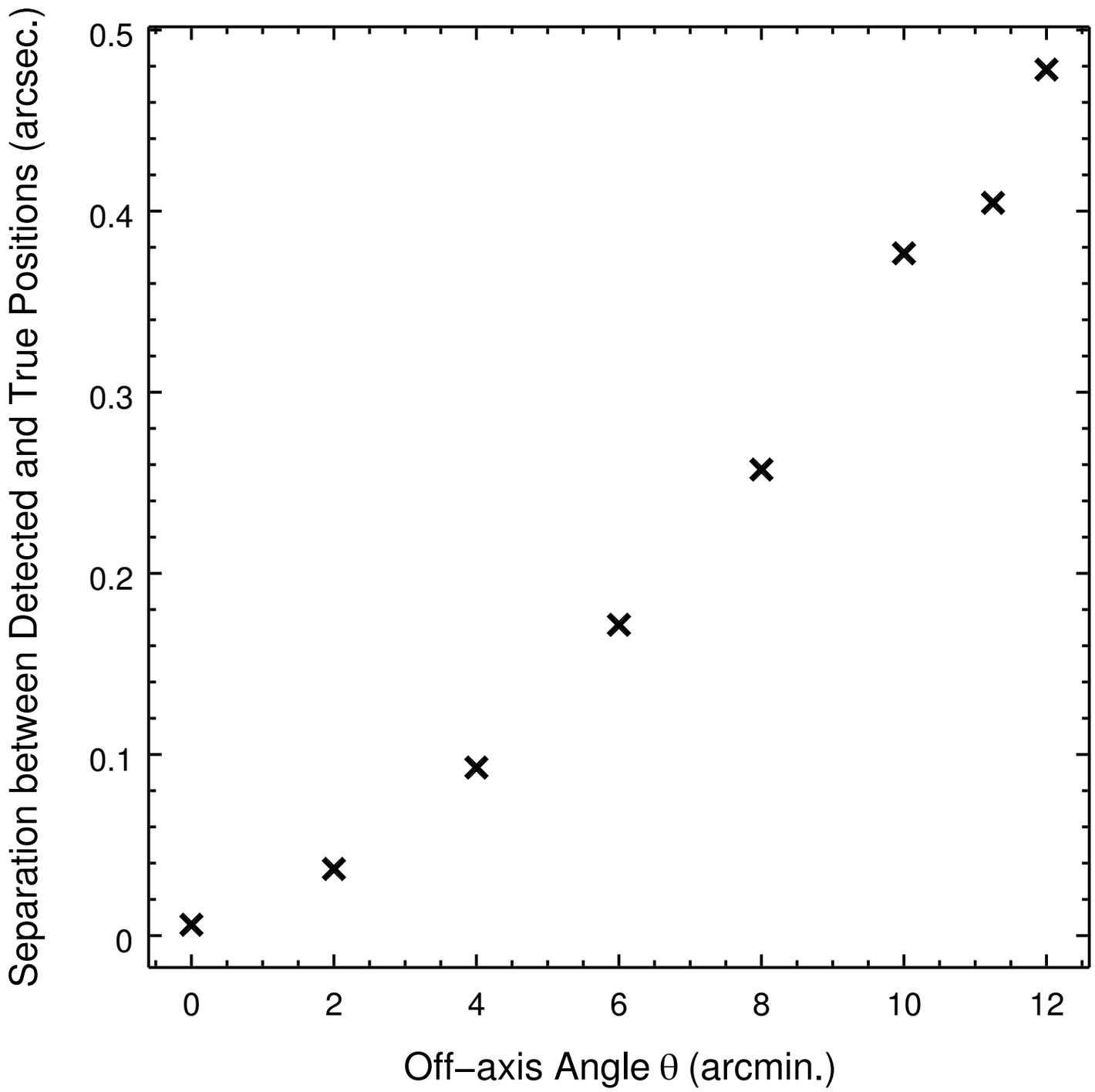
WAVDETECT: Systematic Position Uncertainties

- Due to structure of PSFs, centroids are not a good representation of true source position far off-axis
- Demonstrate via simulations with no background and > 10^5 counts per source, so statistical uncertainty is negligible

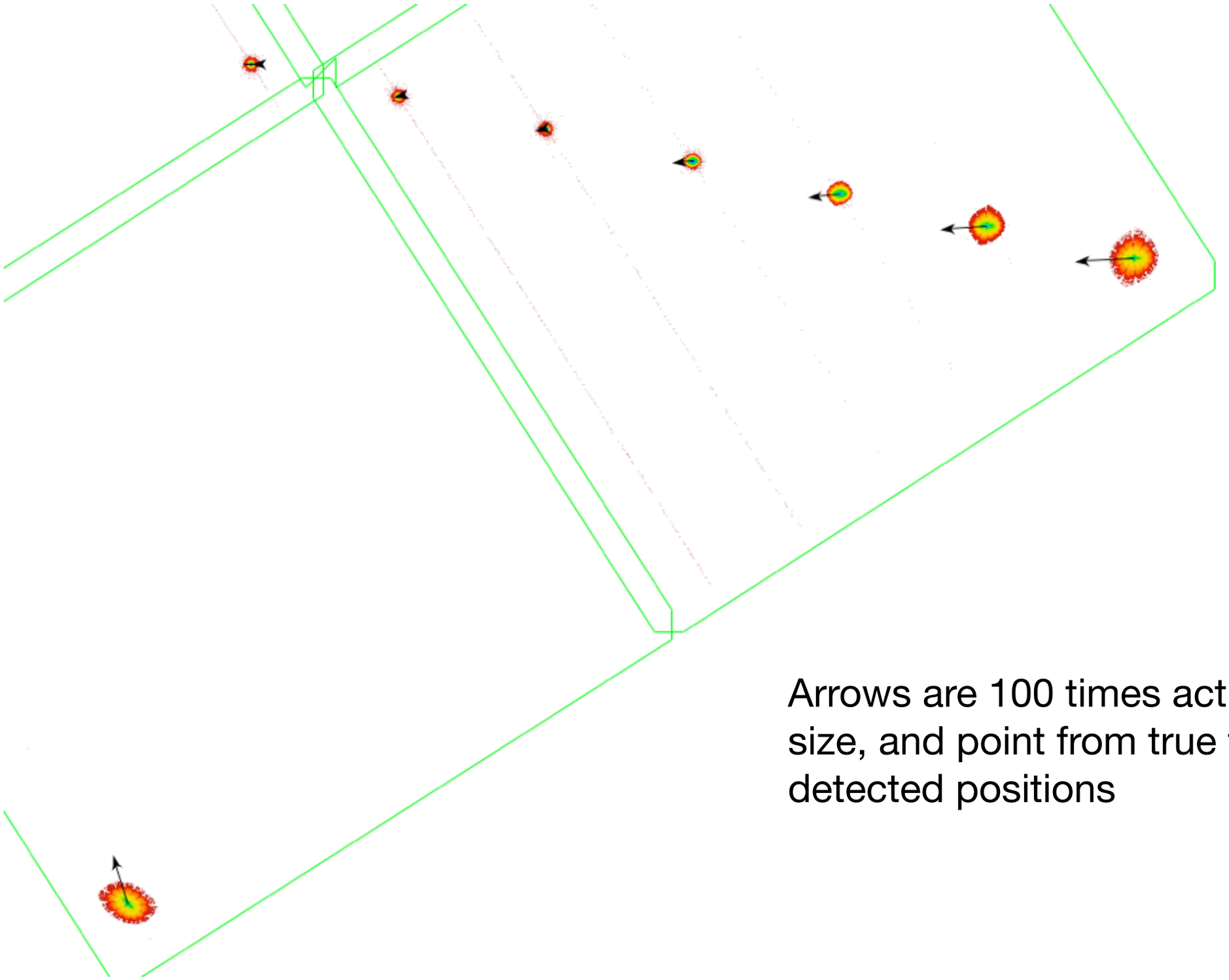
WAVDETECT: Systematic Position Uncertainties



WAVDETECT: Systematic Position Uncertainties



WAVDETECT: Systematic Position Uncertainties



Arrows are 100 times actual size, and point from true to detected positions

WAVDETECT: Pros and Cons

- Works well in crowded fields.
- Works well for point sources and extended emission, provided scales chosen carefully (may have to merge scales manually).
- Only approximate PSF shape is needed.
- Edge of field effects not a problem.
- Systematic centroiding errors can be calibrated out, but typically much smaller than statistical errors

- Slow, especially if many wavelet scales used.
- Memory-intensive, 2k x 2k OK, but larger image sizes can be a problem.
- No recursive blocking built-in, so running on entire image requires multiple, blocked images. Source lists must then be combined.