Source detection in the SKA era
lessons from LOFAR

aka

How useful is PyBDSM

Niruj Mohan Ramanujam
with Huub Rottgering, David Rafferty, Oleksandr Usov

National Centre for Radio Astrophysics
(NCRA-TIFR), India

SPARCS 2016, Goa

NCRA • TIFR

- Why did LOFAR need PyBDSM ?
- How does it work ?
    - Background estimation
    - Deblending and reblending
    - Deaddiction from gaussians
    - Measuring PSF variation
    - Dealing with extended structures
    - Spectral index, RM, polarisation
- Background estimation revisited - measuring artifacts
- SKA challenge

PyBDSM - why, how, where ?

Rudimentary source extraction used to be done in AIPS, Miriad

Modern low frequency interferometry data pose challenges for source extraction

- Wide fov
  rms varying across image, speed of processing
- Non-isoplanicity within fov
  artifacts around bright sources, psf variation across image
- Many extended and diffuse sources (high brightness sens + high res)
  need non-gaussian basis to sparsify, parametrise complicated morph
- Large bandwidths
  freq dependence of fluxes, posn due to ionosphere, calibration, morphology
- Larger source density, source confusion
  affects background estimation, threshholding
- Need spectral indices, RM
  for point and extended sources, varying SNRs across source

PyBDSM - why, how, where ?

PyBDSM (Python Blob Detection and Source Measurement) written for LOFAR, at Sterrewacht Leiden (2007-10)

Used by the LOFAR survey project for catalogues and calibration

Maintained currently by Rafferty (2010-)

Being used extensively for

- SKA SA (calibration + CLEAN loop)
- GMRT (automated archive processing, catalogues)
- TGSS ADR

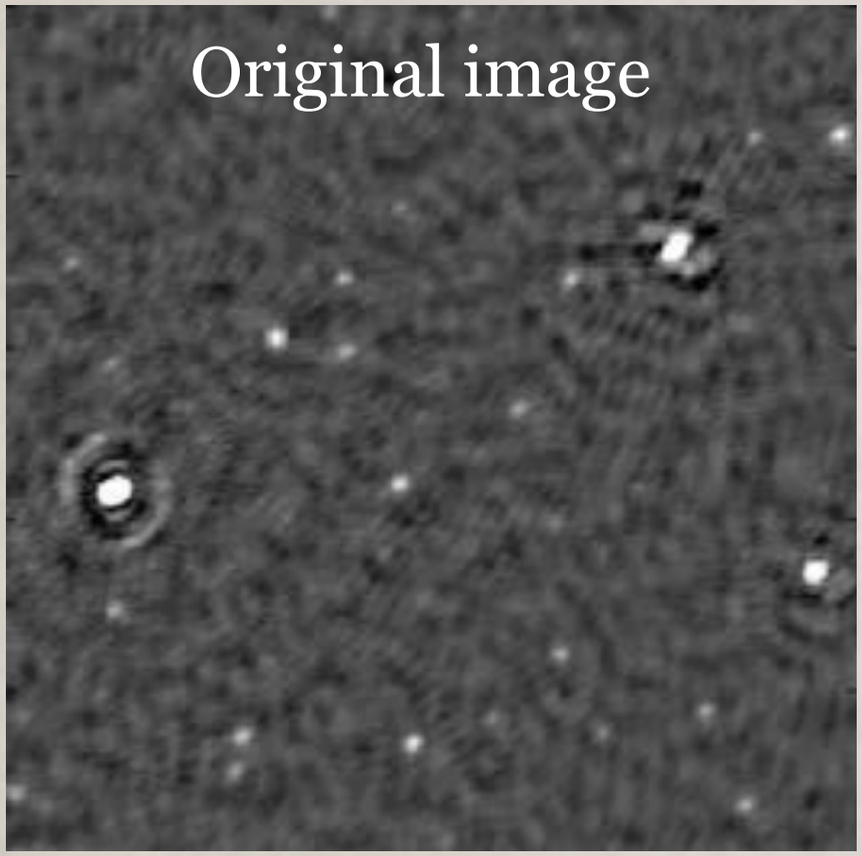
Performed well in the Source Finding Challenge (Hopkins et al, 2015) among

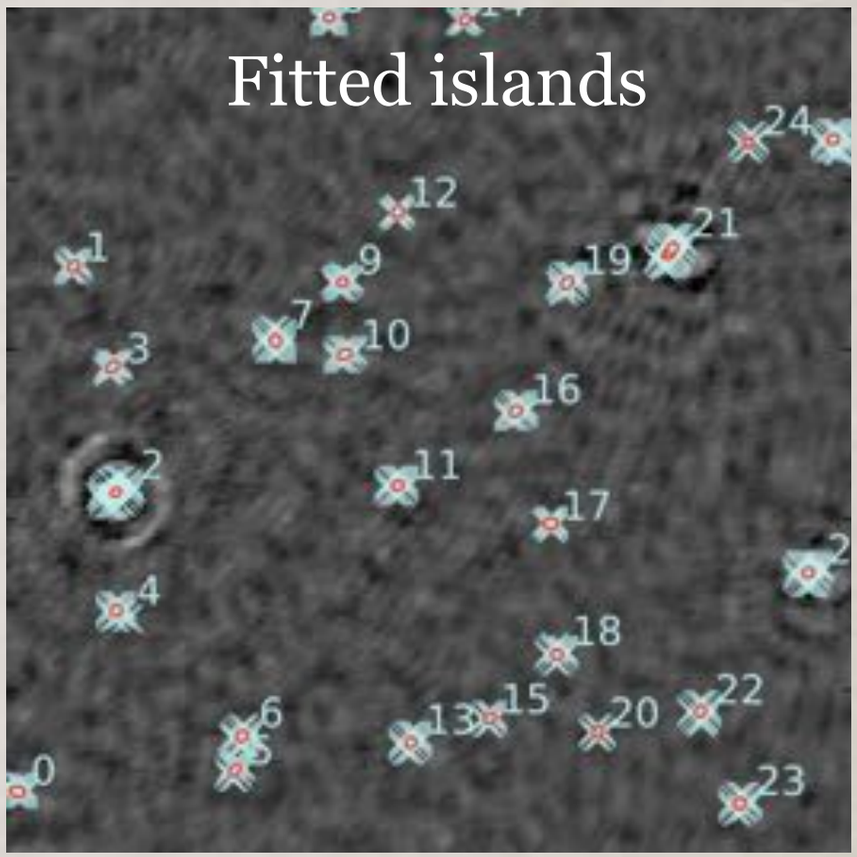
- Aegean (radio, point sources)
- APEX (Spitzer)
- Blobcat (floodfill blobs, pol)
- CuTEx (Herschel)
- IFCA (SE + filters)
- PyBDSM (LOFAR)
- PySE (LOFAR transients)
- SAD (AIPS)
- SExtractor (optical)
- SOURCE_FIND (AMI)
- Duchamp (HI)
- Selavy (ASKAP)

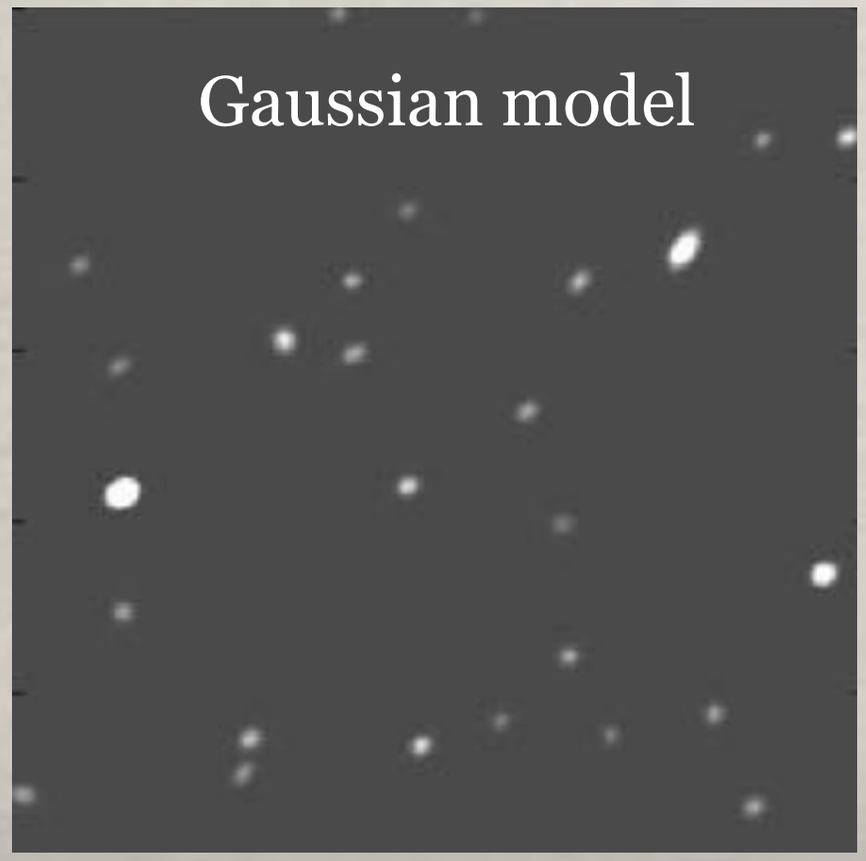
PyBDSM - the vanilla version

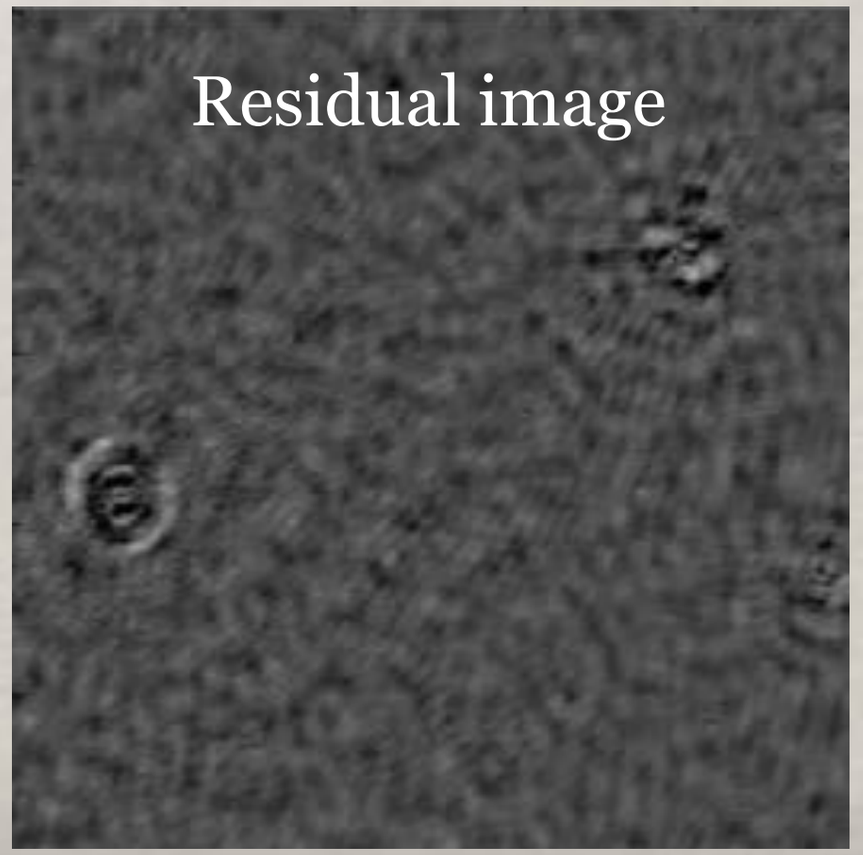
- Read in 4-d FITS of Casa-format images. Source extraction done on user-defined contm image
- Computes sensible values for parameters unless specified by user (110+ user specifiable parameter)
- Computes background rms and mean images (crucial step, more on this)
- Thresholding (hard, False Detection Rate)
- Decompose thresholded image into islands of contiguous islands
- Segmentation, deblending of islands of emission
- Each island is fit with multiple gaussians (LM+others), scale-free algorithms for initial guesses for extended structure
- Output gaussian catalogue with errors, residual images

PyBDSM - the vanilla version



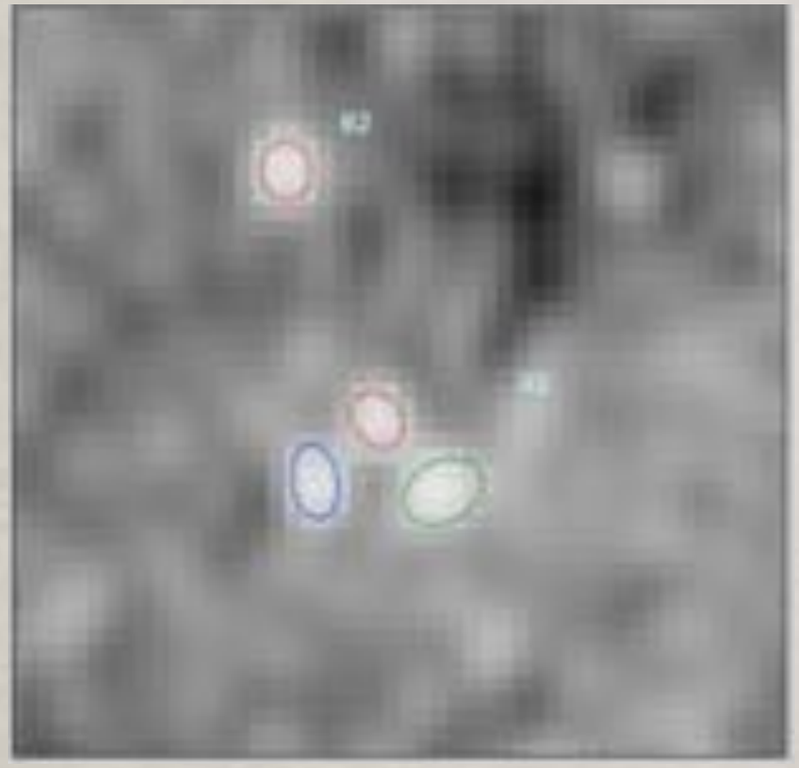


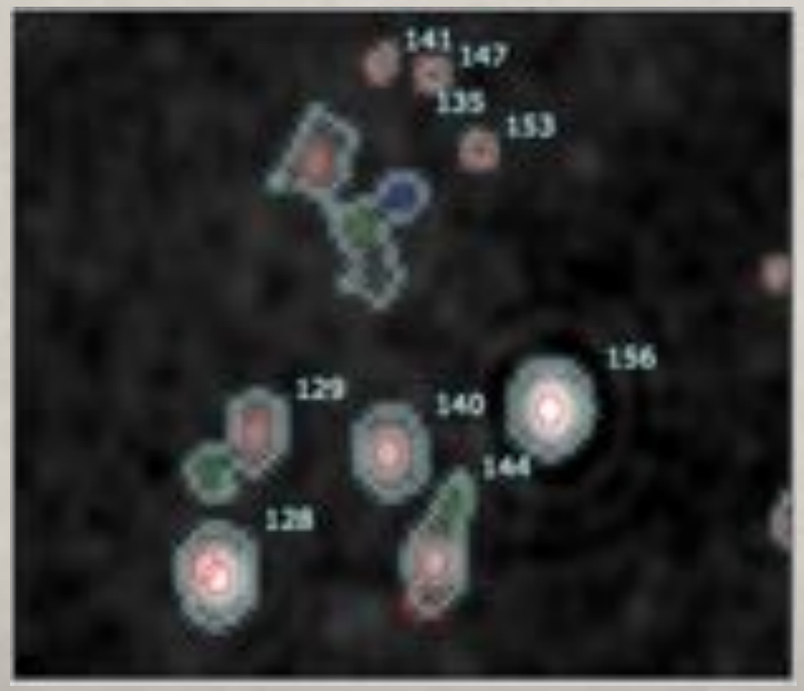


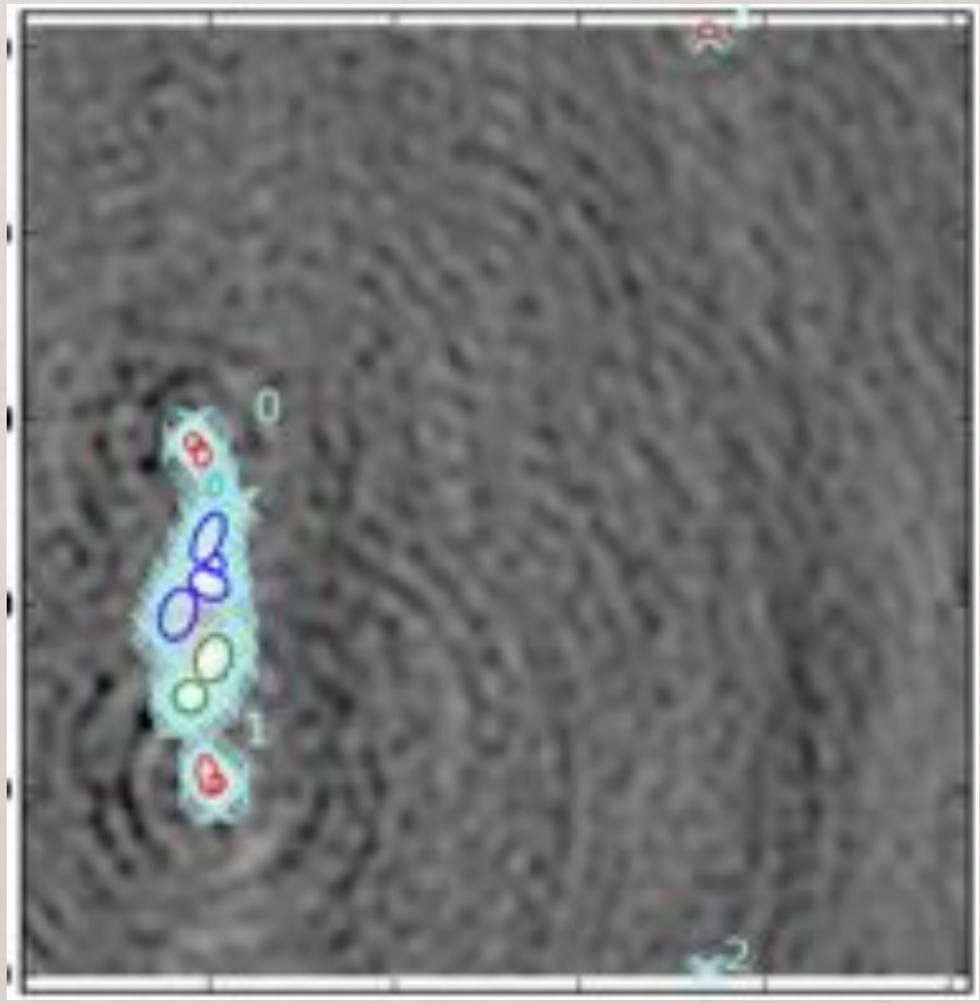


Bells and whistles : reblending

- Within each island, fitted multiple gaussians are grouped into ‘physical’ sources and classified. Corresponds well to what we would do visually







Blue crosses : outline of an island

Ellipses : fitted gaussians in the islands

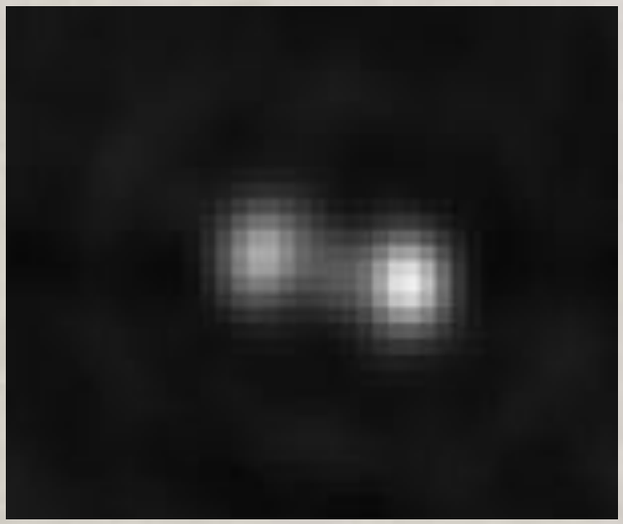
Different colours in an island are different sources

Bells and whistles : spectral index

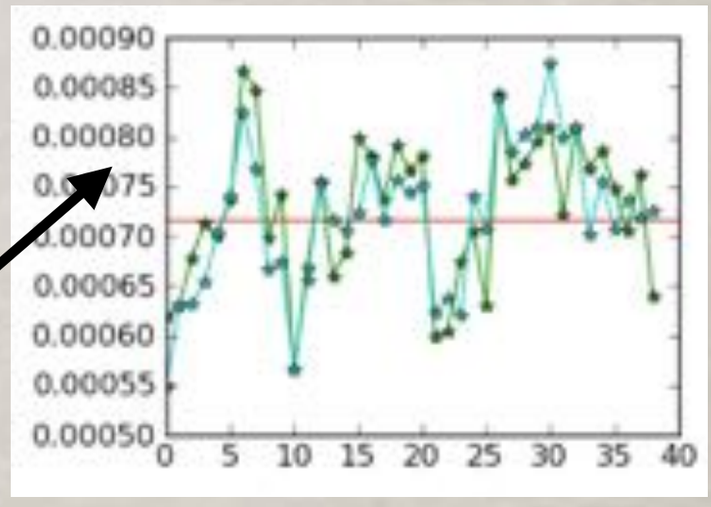
- For the detected sources, spectral indices and RMs are derived from the 4-d cubes

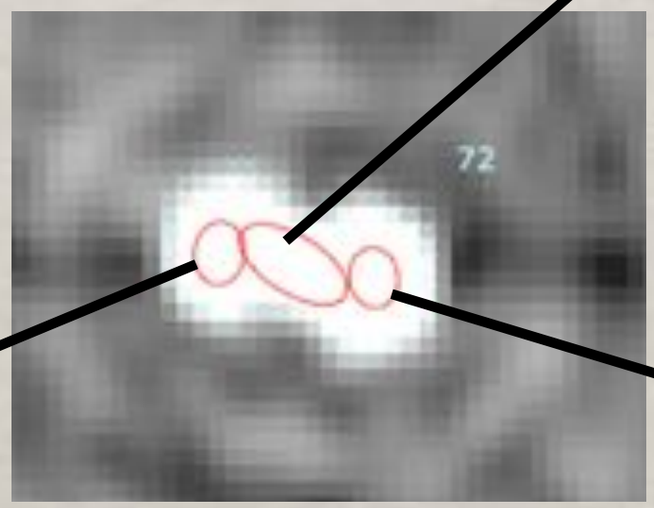
Takes into account varying scenarios of

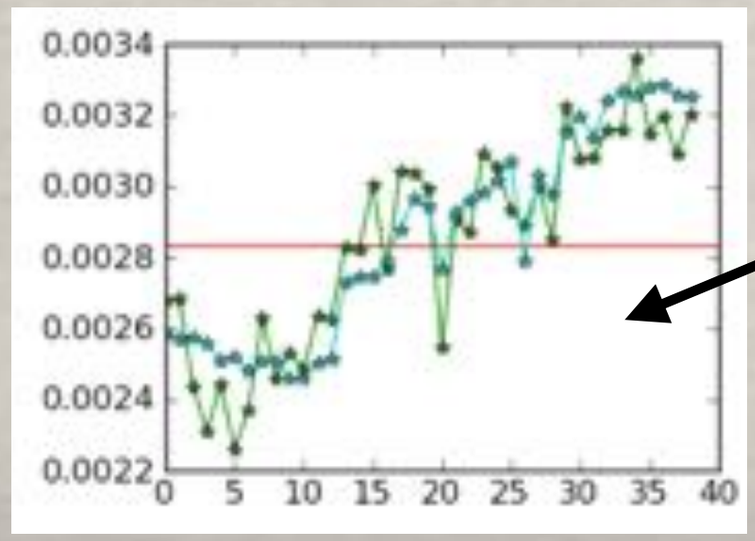
- frequency dependence of SNR of fitted components
- frequency dependence of morphology and fits of extended emission

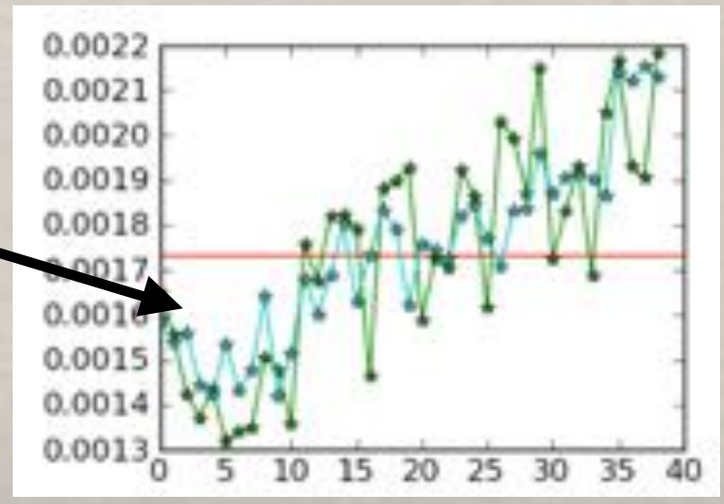


Recover flat core and steep lobes in compact AGN!









Wavelets for extended emission

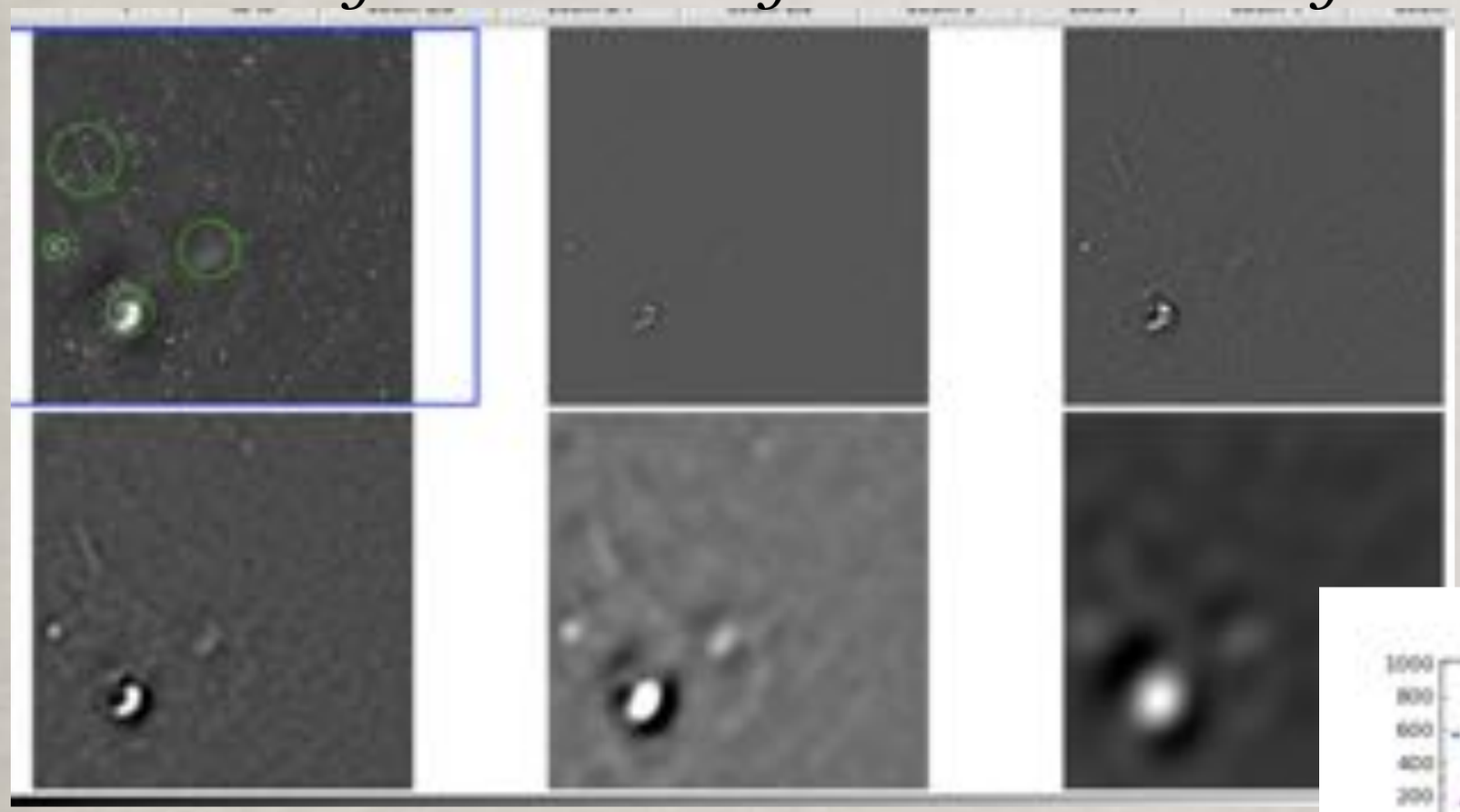
- Wavelet transforms are used to characterise extended emission

  - First run standard PyBDSM, subtract fitted gaussians
  - Run *a trous* wavelet transform on residual image
  - Now run scale-free version of PyBDSM on each wavelet transform image
  - Create catalogue of gaussians at each wavelet scale
  - Group these gaussians using the pyramidal representation to parametrise extended emission
  - Calculate wavelet model images and residual images

Wavelets for extended emission

- Wavelet transforms are used to characterise extended emission

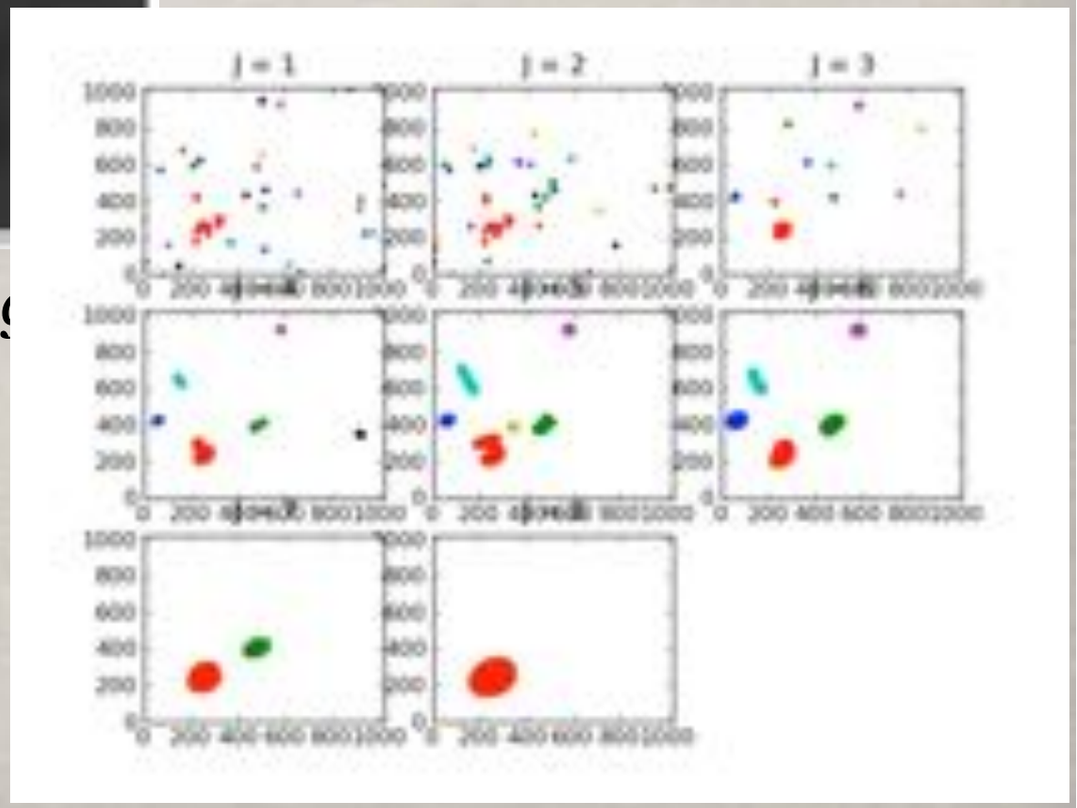
Residual image *w1 image* *w2 image*



w3 image *w4 image* *w5 imag*

WENSS image

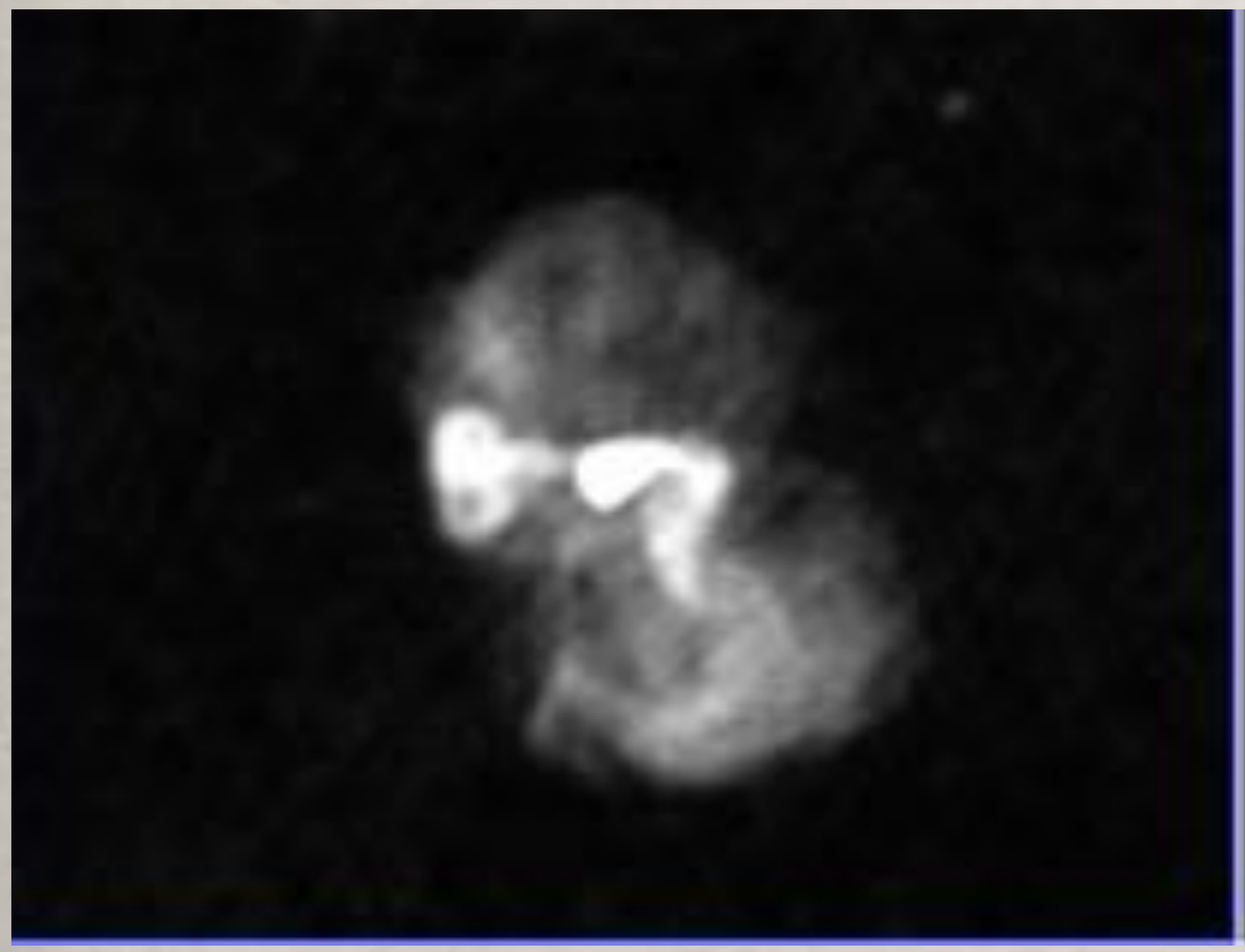
Image after subtracting gaussians, and subsequent wavelet transform images



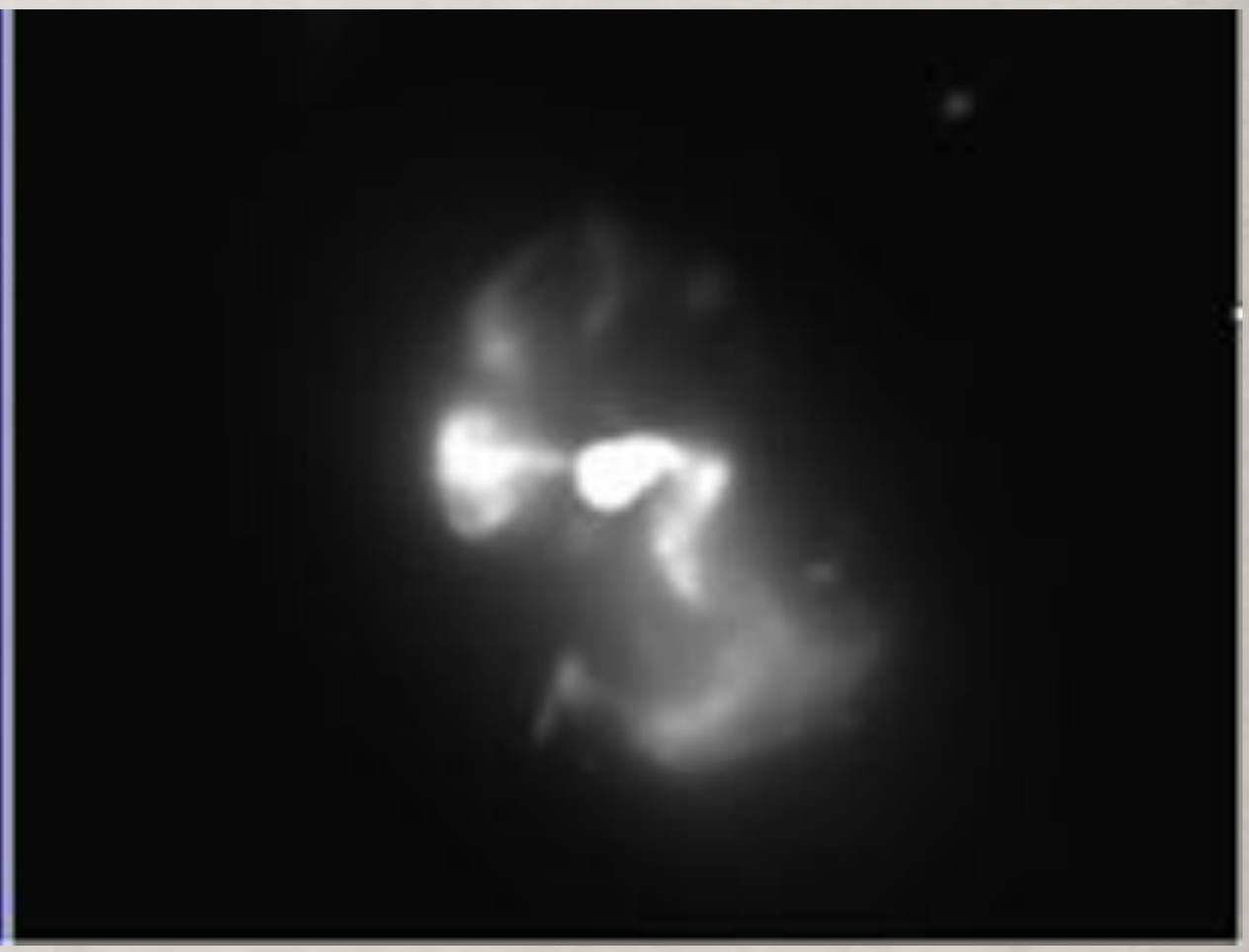
Pyramidal representation of gaussians fitted to each wavelet image

Wavelets for extended emission

- Wavelet transforms are used to characterise extended emission



Original image

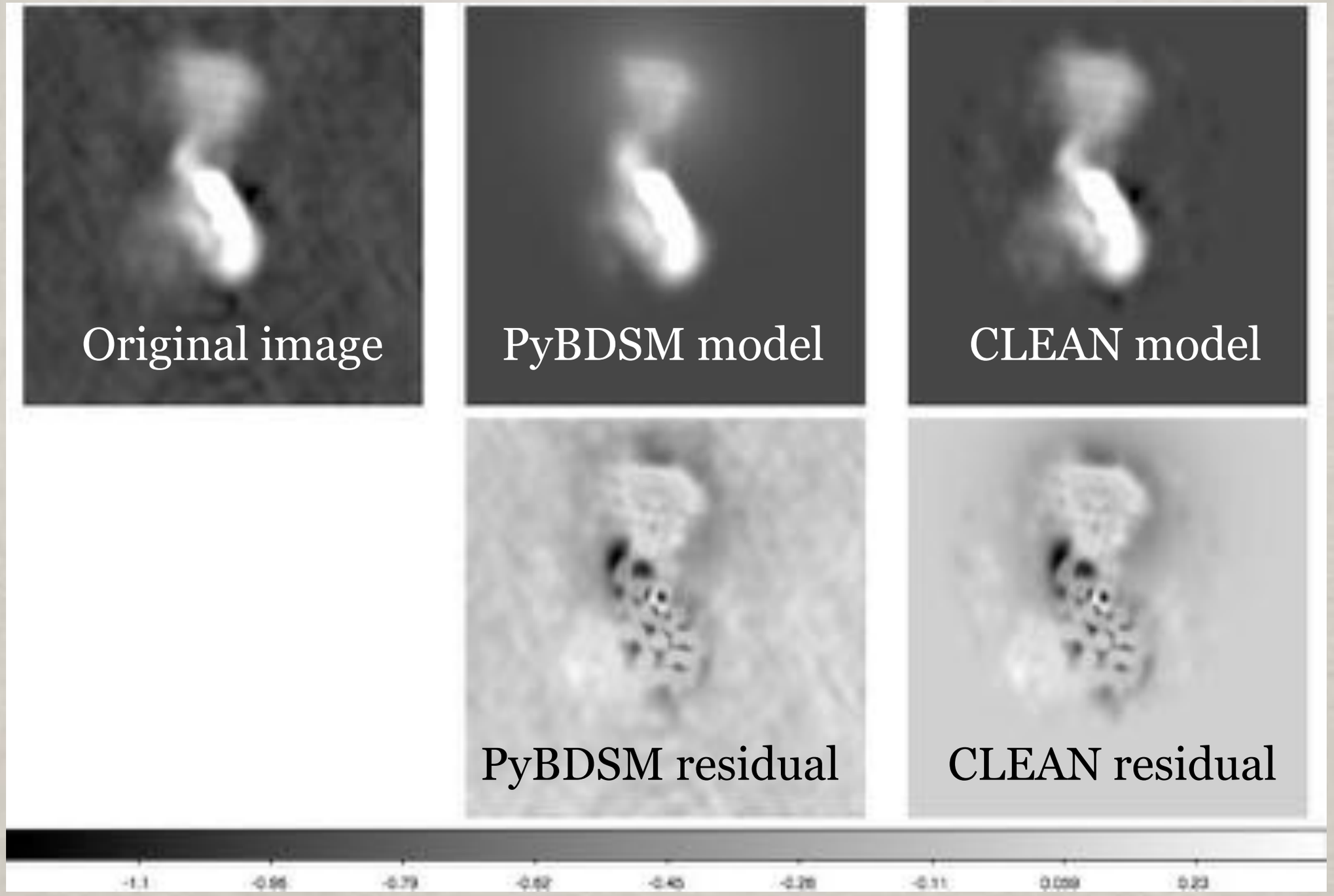


Reconstructed image from 180 gaussians from wavelet transform

180-gaussian multi-scale wavelet model of Virgo A

Wavelets for extended emission

- Wavelet transforms are used to characterise extended emission



A multi-scale wavelet model of Hydra A

Shapelet transforms as a basis

- Shapelet transforms are used to parametrise island emission in addition to gaussians

Gaussians are natural to model simple sources since the synthesized beam is a gaussian

However, gaussian decompositon is not unique and are constrained by the beam, and hence

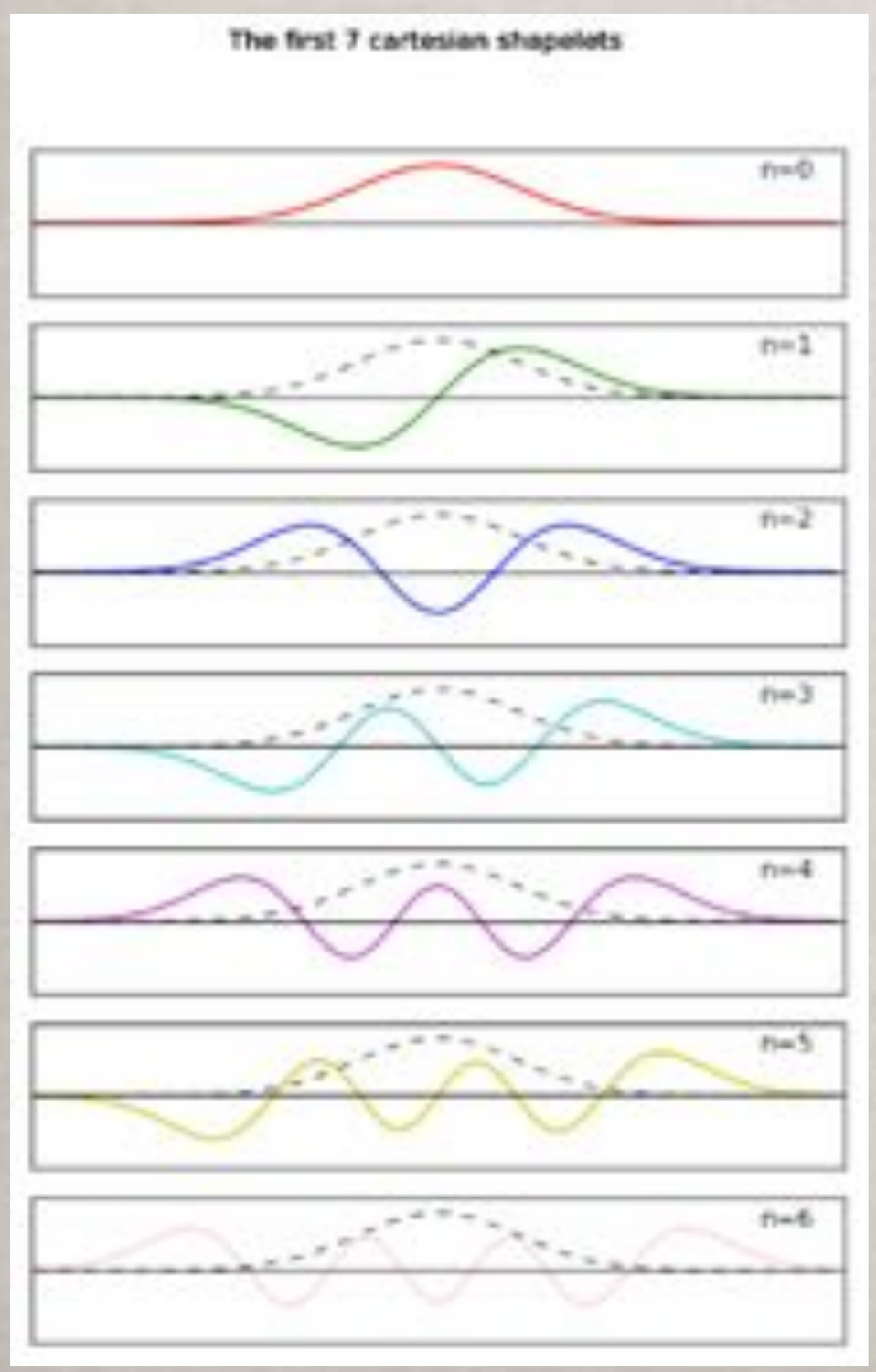
- Throwing 100s of gaussians at an extended source is not a good idea
- Cannot easily capture small departures from gaussianity in compact sources

Hence, need an orthonormal basis set that is compact, and to first order is a gaussian

The answer is the shapelet transform

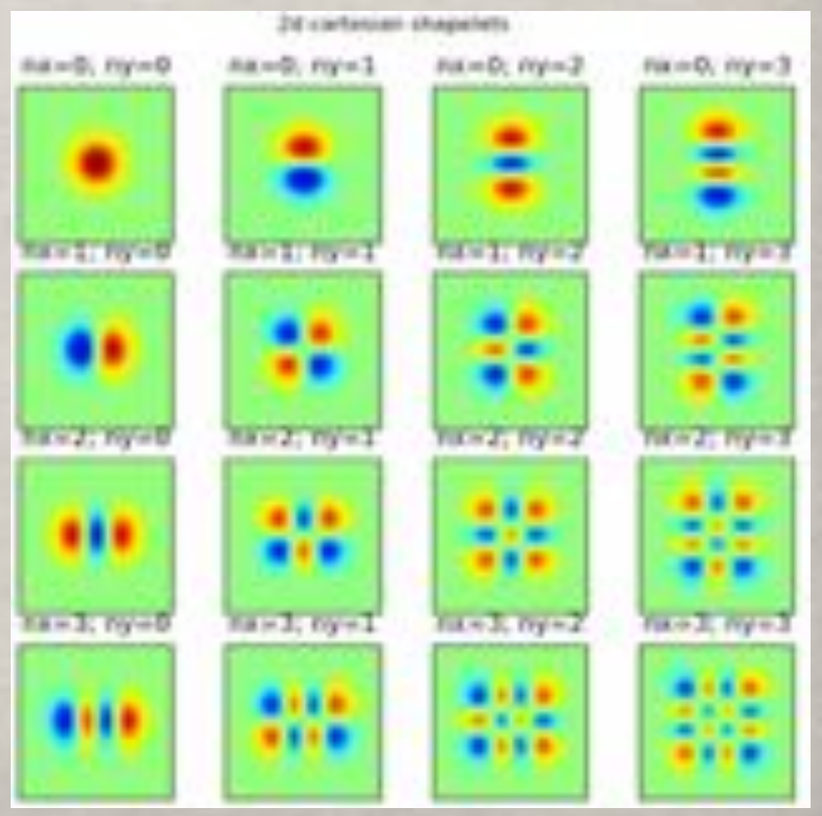
Shapelet transforms as a basis

- Shapelet transforms are used to parametrise island emission in addition to gaussians



Cartesian shapelets are Hermite polynomials multiplied by a gaussian

- First order is a gaussian
- They are highly localised
- They are orthonormal
- Sparsify emission in min number of shapelet coefficients, can reconstruct



2d cartesian shapelet basis set

Shapelet transforms as a basis

- Shapelet transforms are used to parametrise island emission in addition to gaussians

Simulated image

Shapelet reconstruction

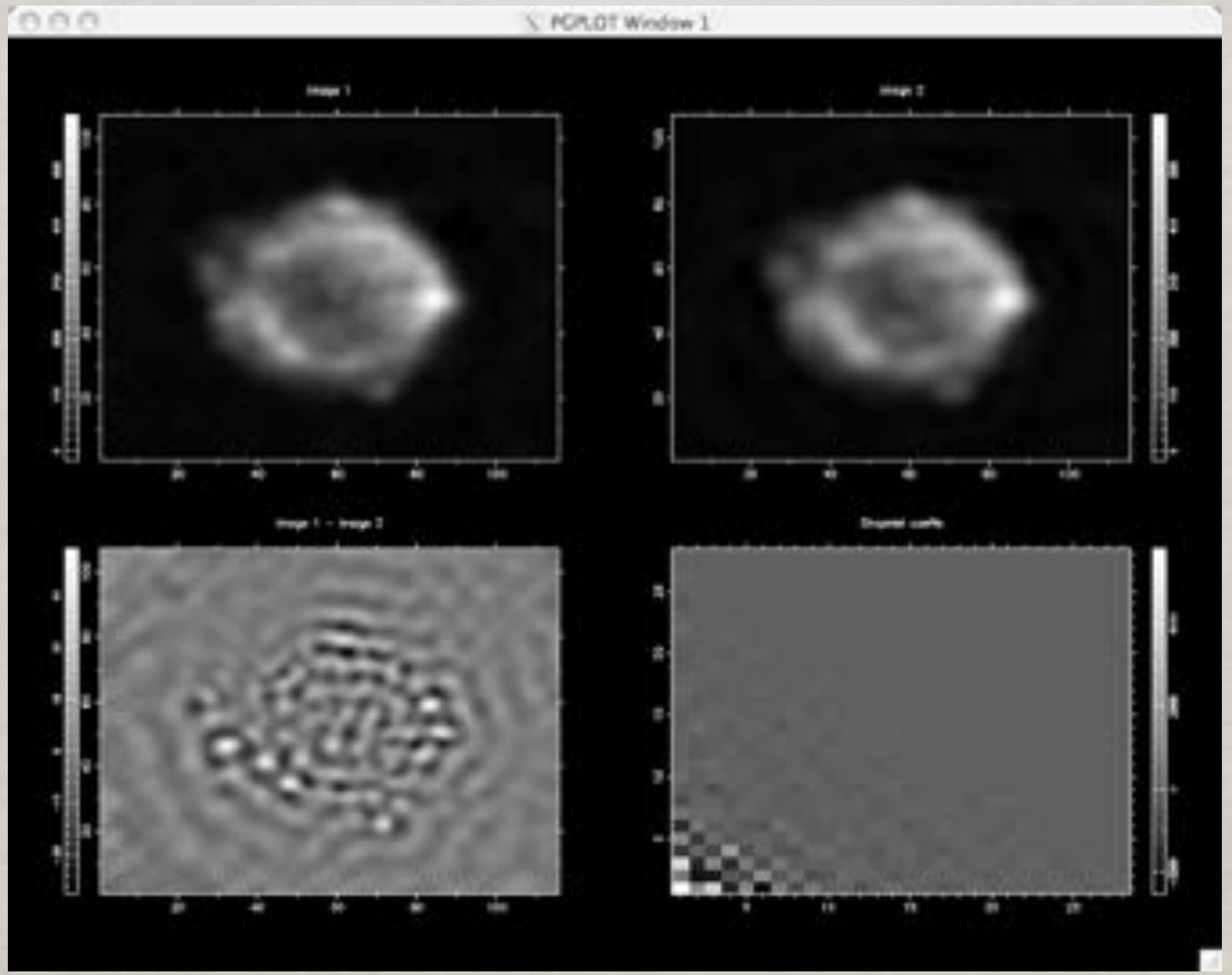
Shapelet residual

Shapelet coefficient matrix

Shapelet transforms as a basis

- Shapelet transforms are used to parametrise island emission in addition to gaussians

Cas A 74 MHz VLA image



Shapelet reconstruction

Shapelet residual

Shapelet coefficient

PSF variation across the image

- PSF variation across the image is estimated and a ‘psf map’ is created. This can be used to correct the catalogue astrometry and photometry

The effective psf, or shape of point sources, will vary across the image due to

- Field of view is a few times larger than isoplanatic size
- Calibration efficiency varies spatially for DD calibration

PyBDSM does the following

- Identify ‘true’ point sources from fitted gaussians
- Voronoi tesselate the image around bright sources
- Average psf of ‘true’ point sources in each tile
- Decompose each average psf into shapelet coefficients
- Interpolate each shapelet coefficient across the image
- Reconstruct the psf on a grid from these interpolated coefficients

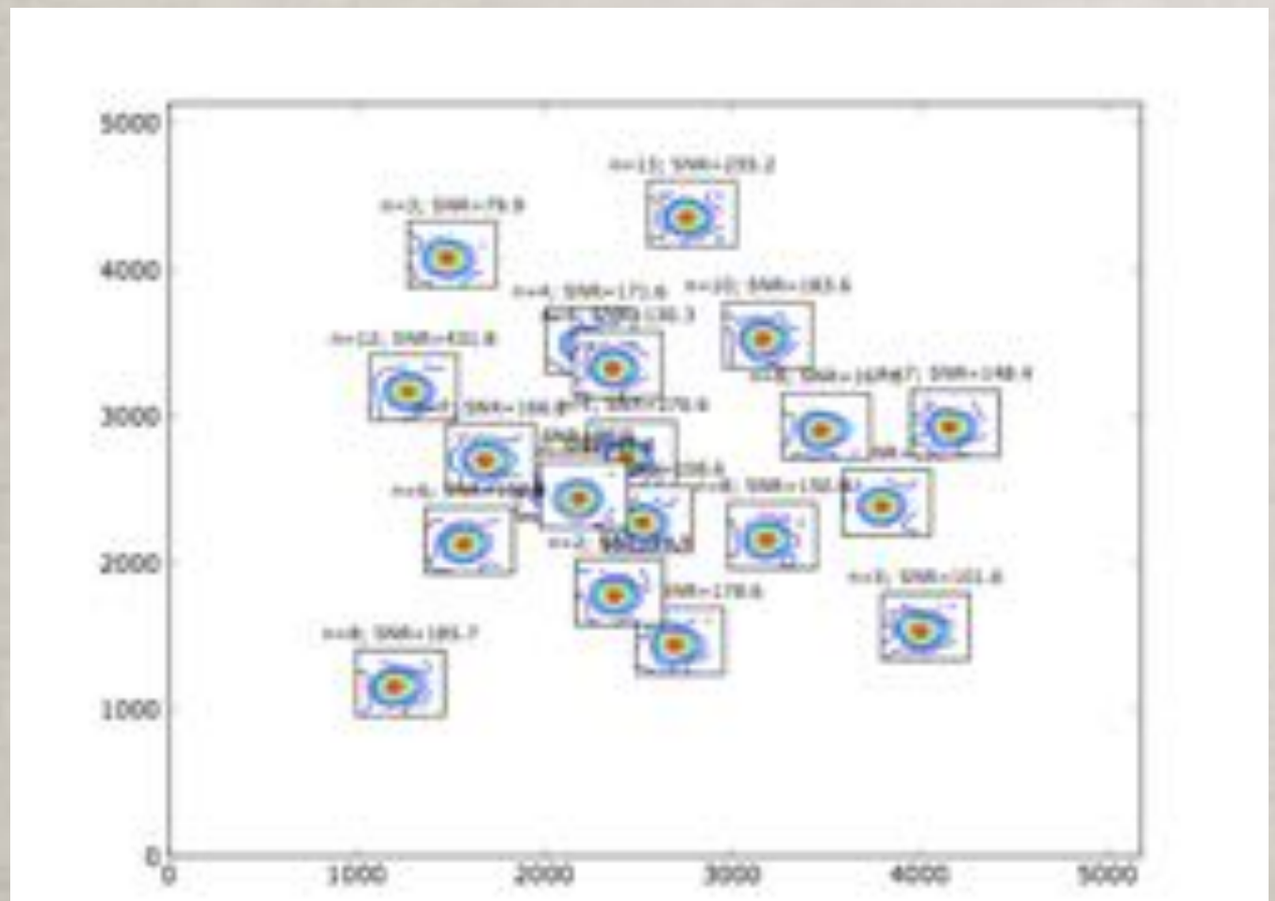
PSF variation across the image

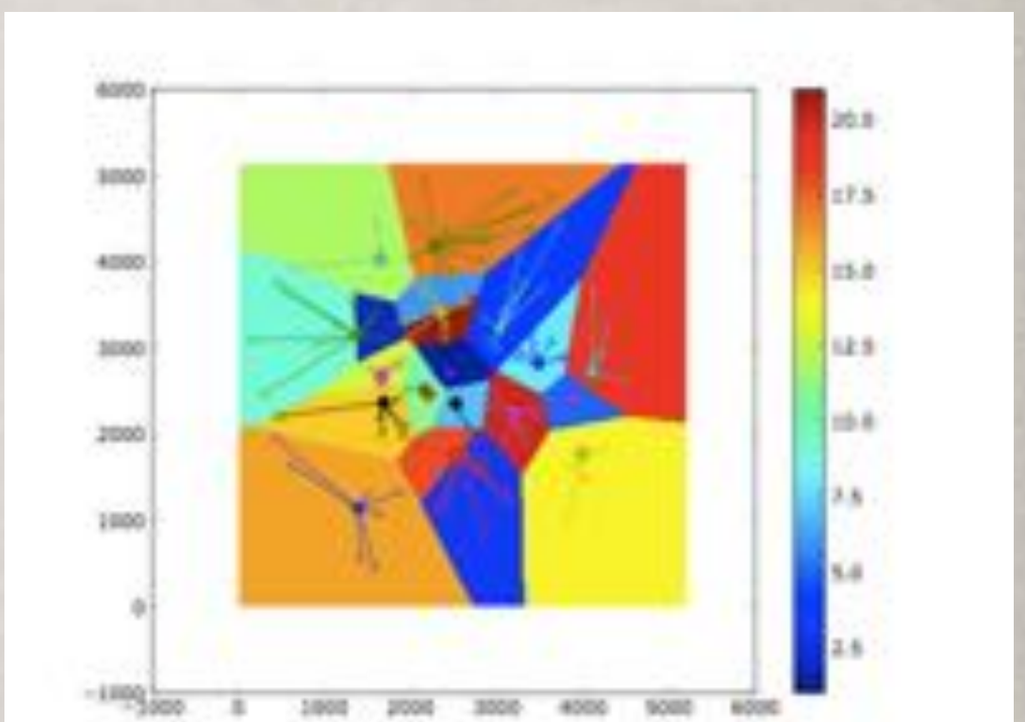
- PSF variation across the image is estimated and a ‘psf map’ is created. This can be used to correct the catalogue astrometry and photometry

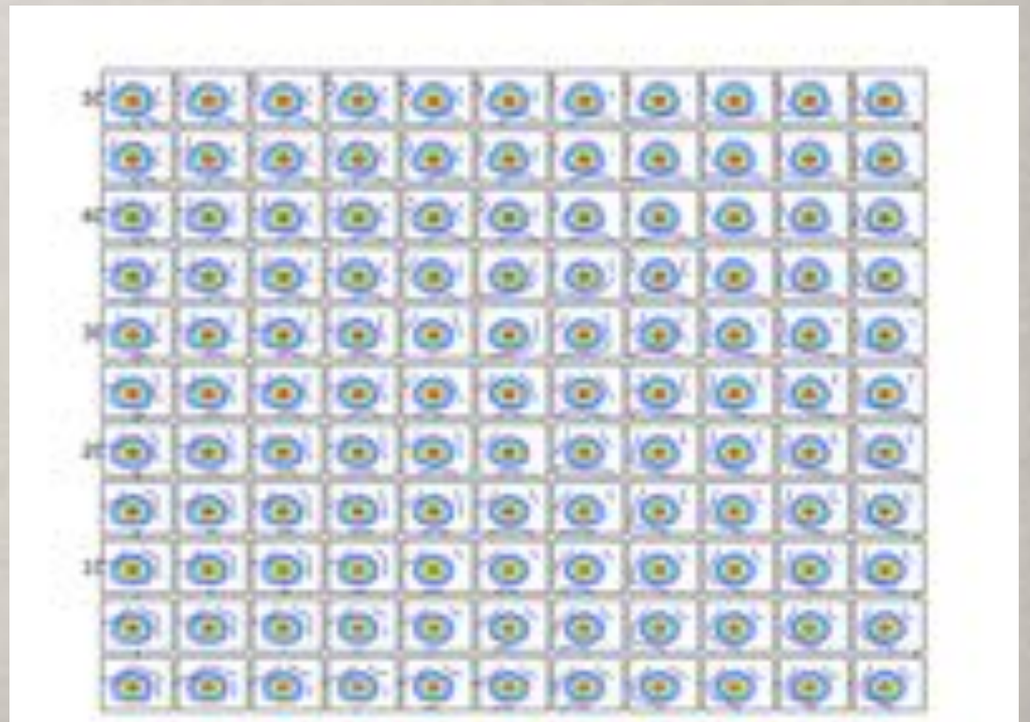
Right : Tiled image with all point sources

Left bottom : Co-added psf in each tile

Right bottom: gridded interpolated PSFs



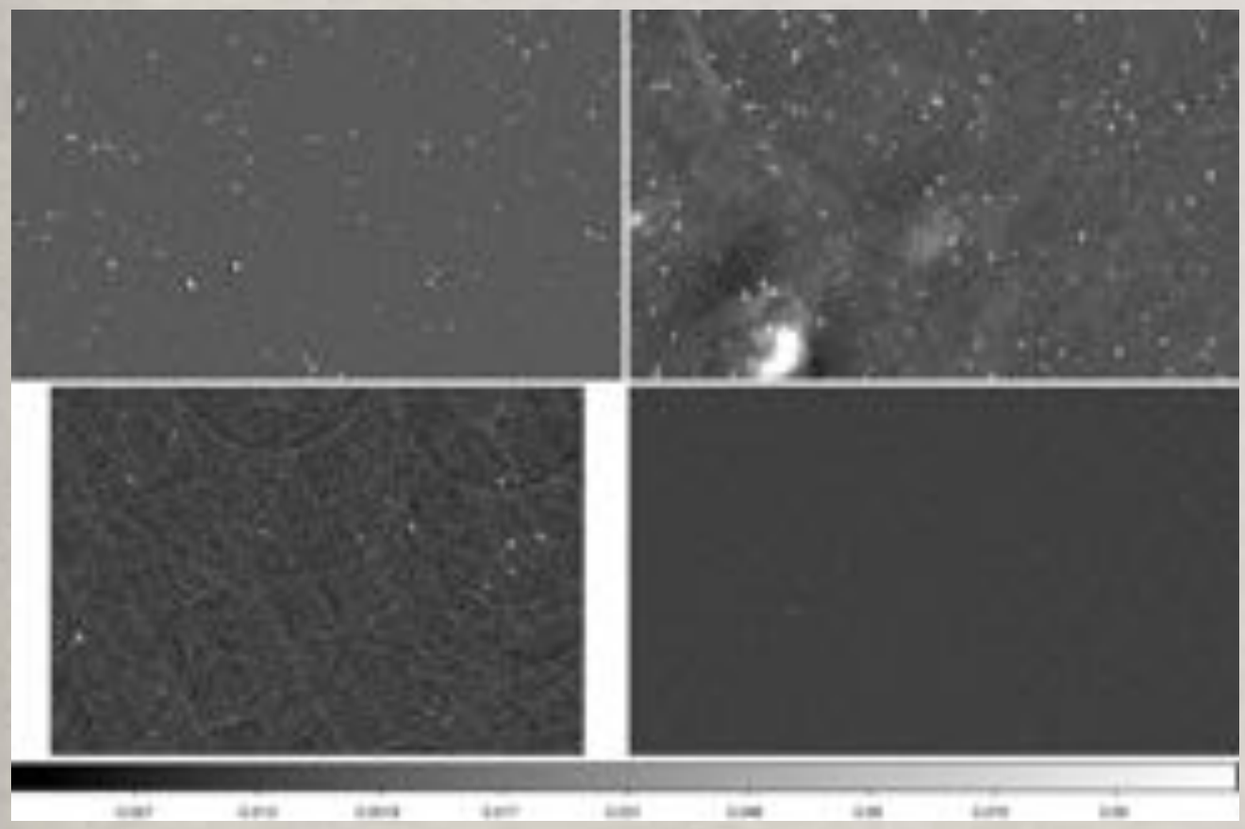




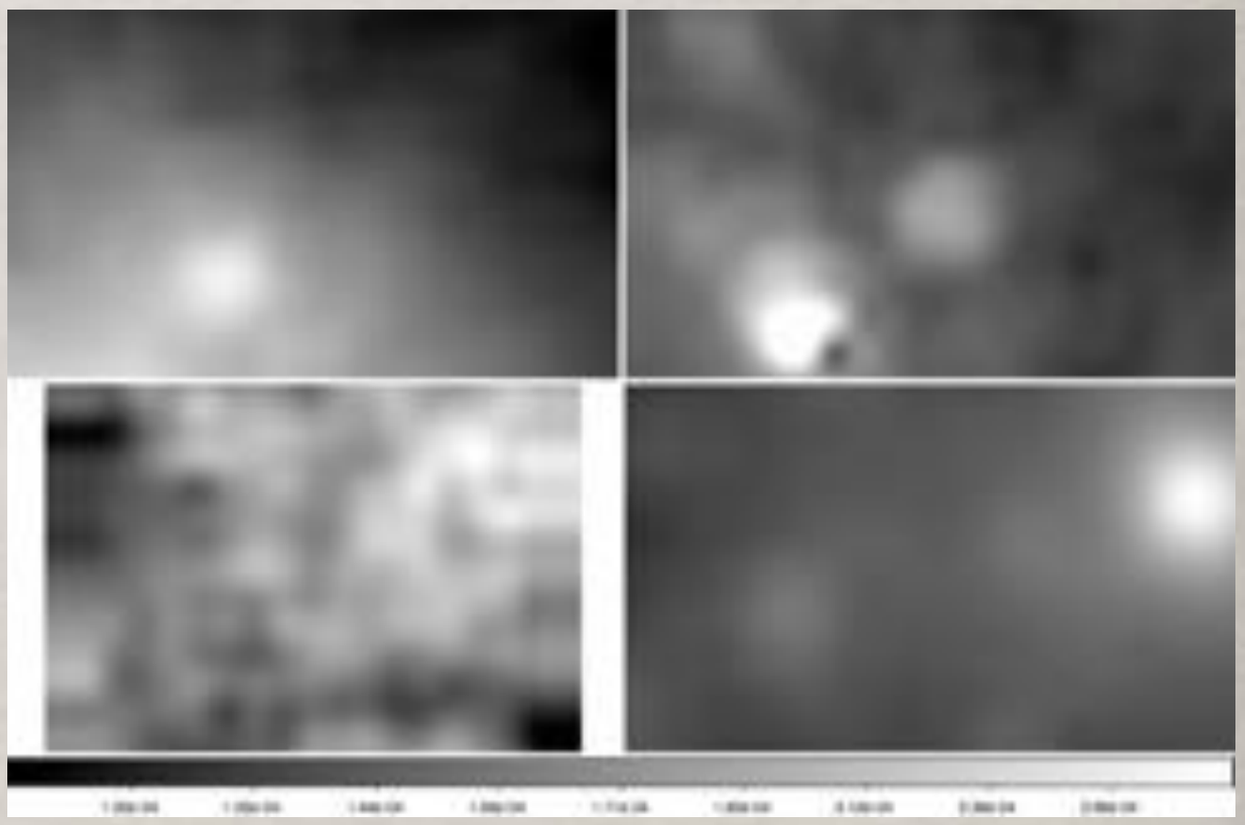
Background rms image

The most crucial step in any source extraction software is the estimation of background rms variation across the image

The rms is estimated within boxes of a certain size and then interpolated across the image. This box size has to be carefully chosen



4 different images



Corresponding rms images

The choice of the box size can create false sources and destroy real ones. All source extraction softwares leave this as a user defined parameter

No one size to fit them all

The optimal box size depends on

- Intrinsic variation scale of rms (primary beam)
- Source density
- Distributon function of source sizes
- *Sizes of artifacts due to calibration errors*

Choice of box size prevents full automation of source extraction

Value chosen determines your source catalogue

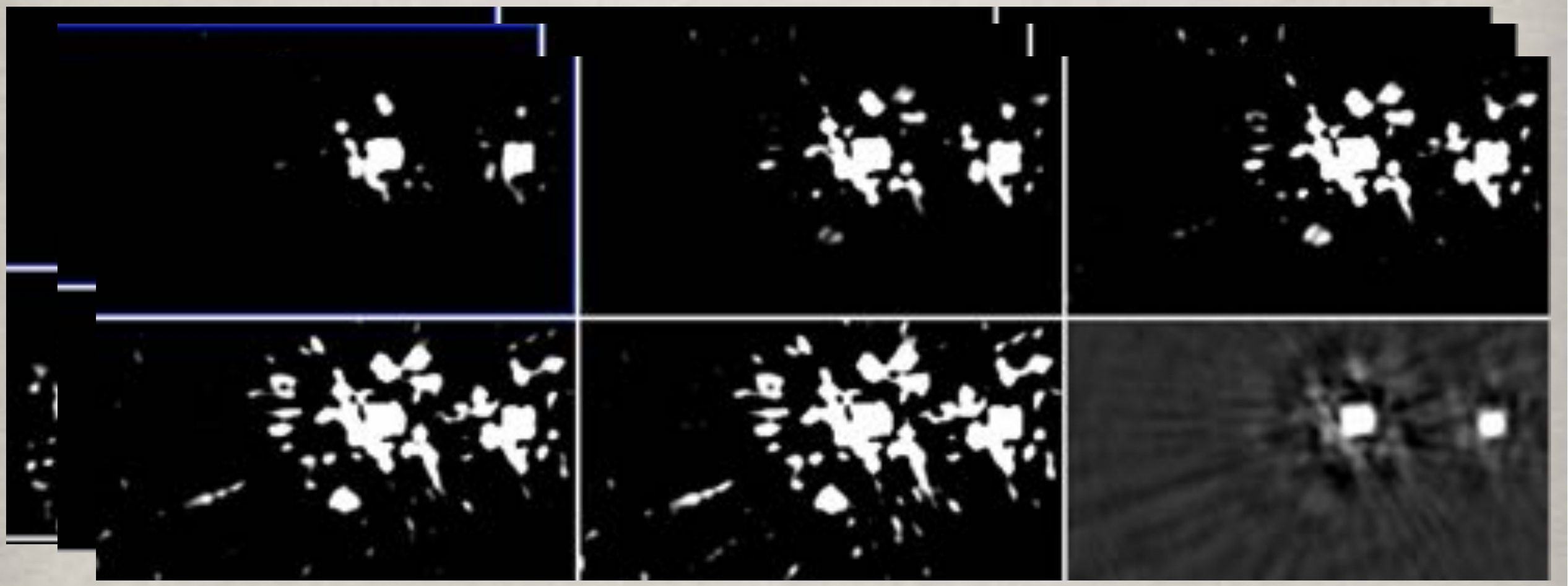
The first three factors are constant for an image and can be automated

The last factor is flux dependent, and is hence visually estimated

No one size to fit them all

If the box size is larger than the artifact size, the rms is underestimated and artifacts are detected as real sources

rmsbox=100 rmsbox=200 rmsbox=300



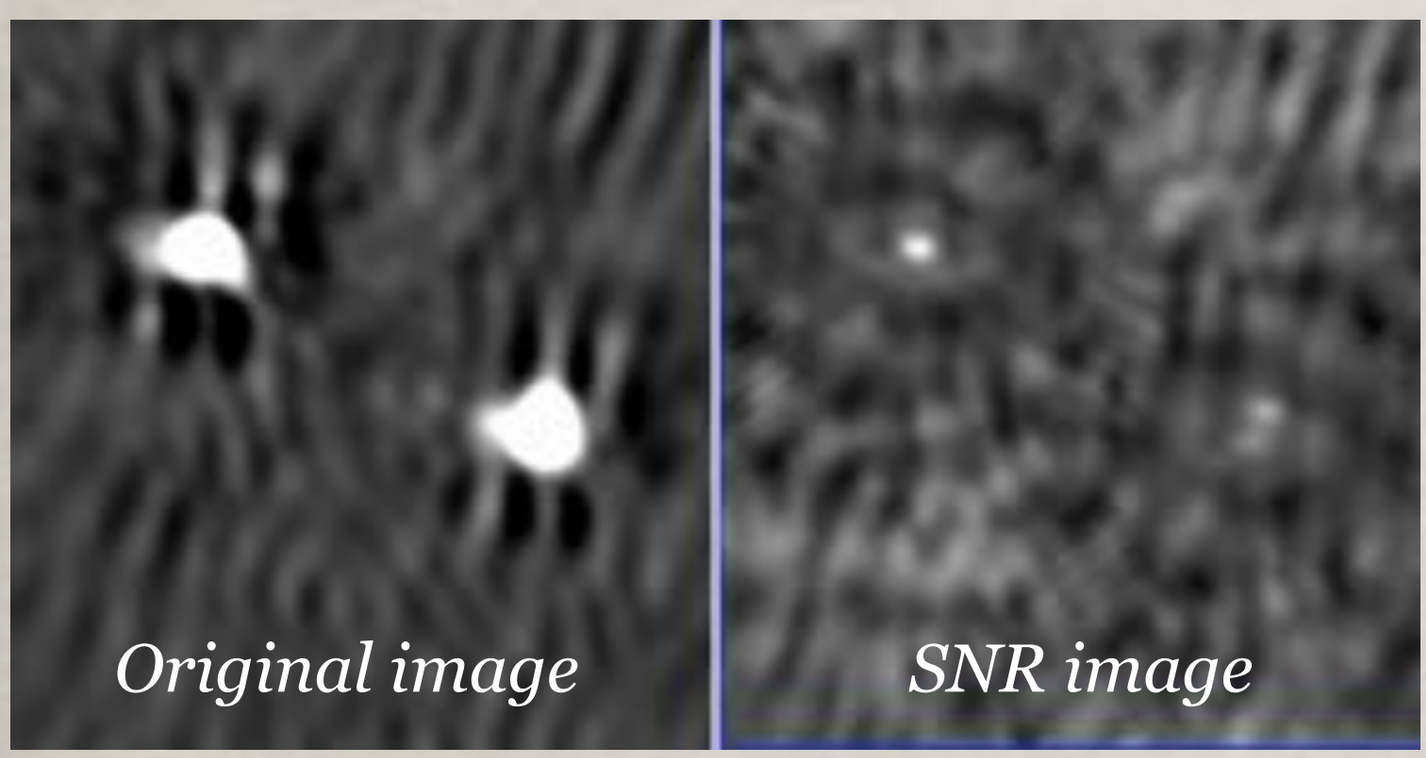
rmsbox=500 rmsbox=700 Original image

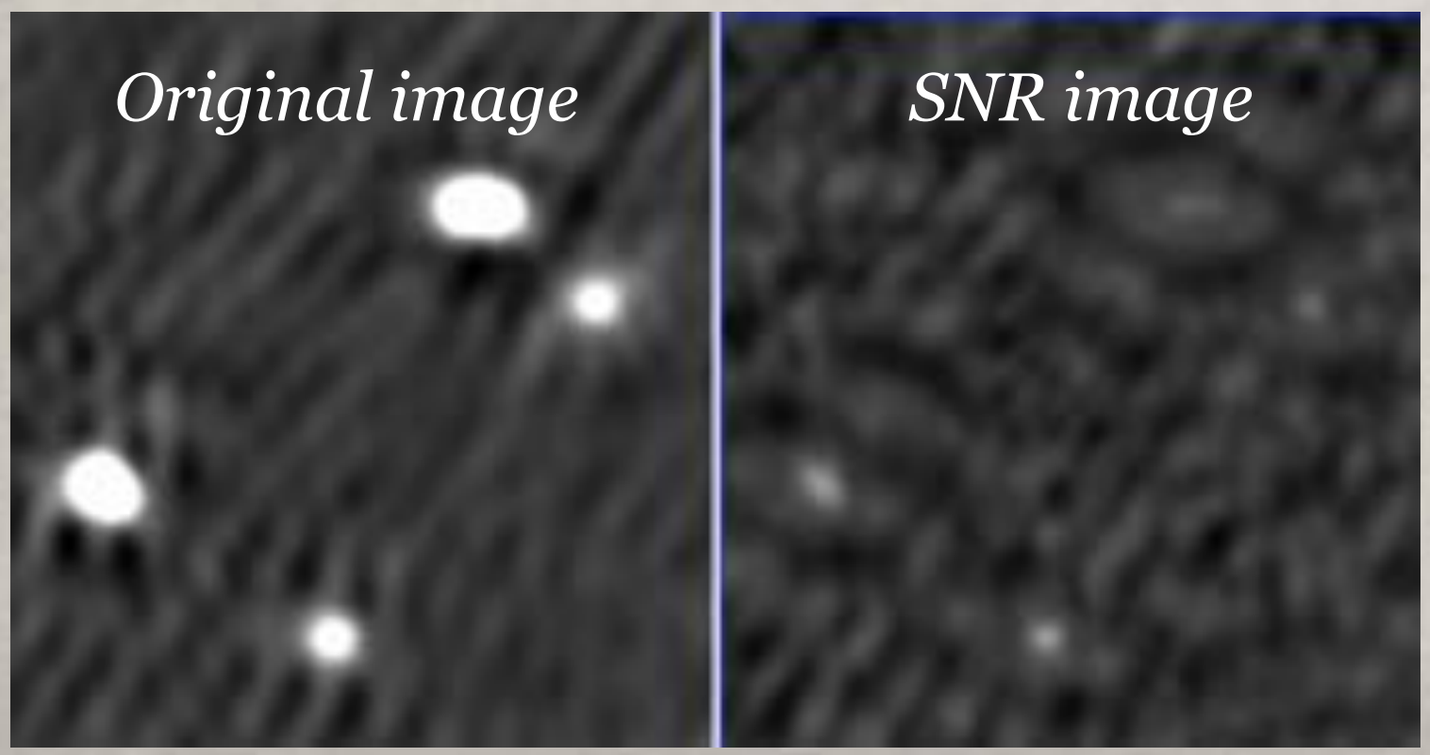
Parts of P37 image from LOFAR HBA survey (image courtesy Shimwell)

Greyscale for the rmsbox images are saturated at 3 sigma

No one size to fit them all

If the box size is smaller than the size of extended sources, then the rms is overestimated and the large parts of the emission goes undetected





Bright sources dissapear when a 2nd box of 30 pixels was used in the LOFAR PyBDSM pipeline

Parts of P2 image from LOFAR HBA survey (image courtesy Shimwell)

No one size to fit them all

PyBDSM circa 2008 :

Internally calculates box size based on expected source counts, sizes and primary beam variation

User visually estimates maximum artifact size in image and inputs it instead

PyBDSM circa 2012 (Rafferty) :

User specifies one box size for entire image and a smaller box size around N brightest sources

PyBDSM circa 2017 :

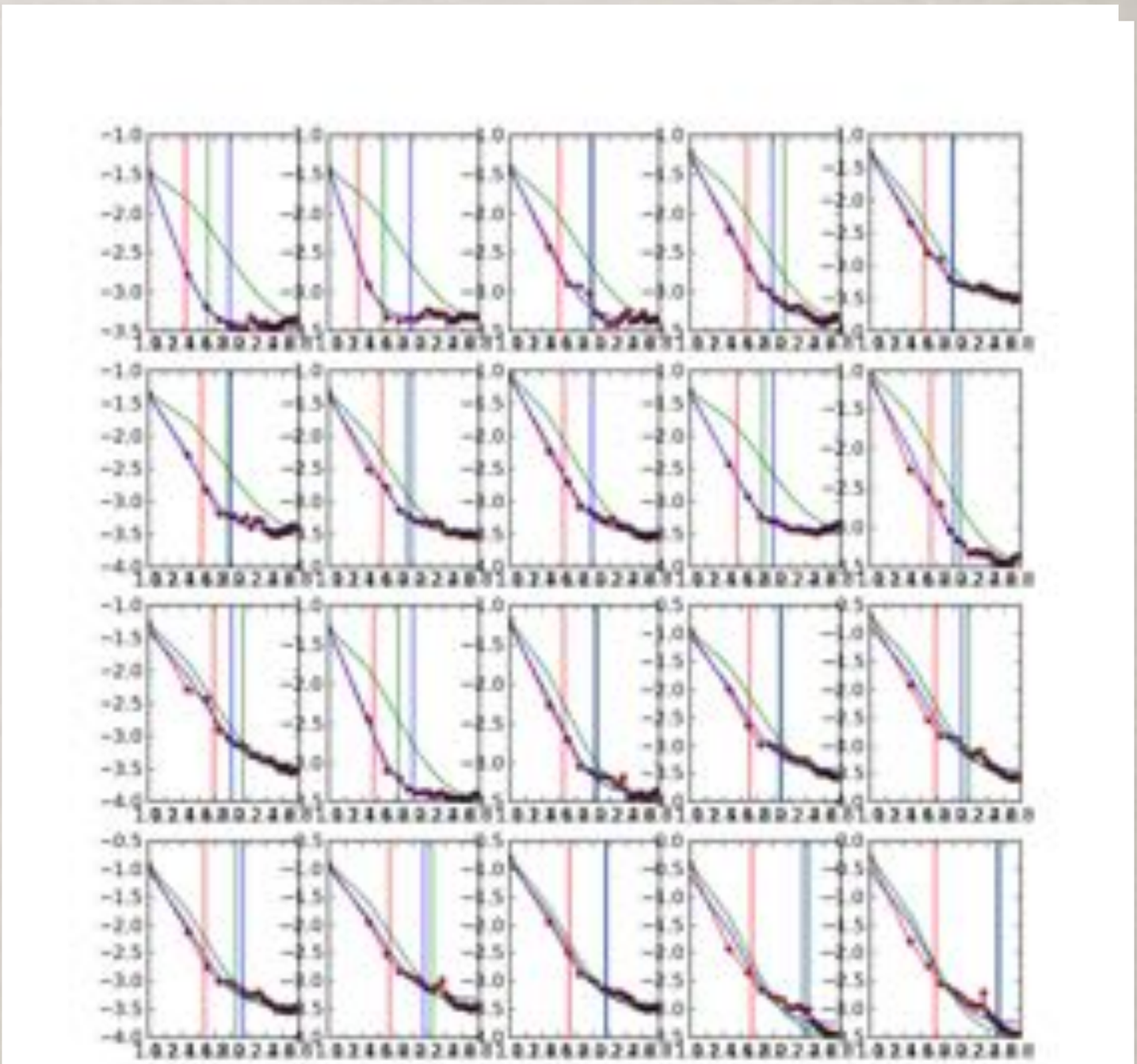
Calculate the artifact size as a function of source flux from the image itself

Calculate rms in boxes with flux-dependent sizes and interpolate across the image

No one size to fit them all

Identify N brightest sources in an image

For each source, calculate rms in concentric squares, fit a function



P18 image from the LOFAR HBA survey (image courtesy Shimwell)

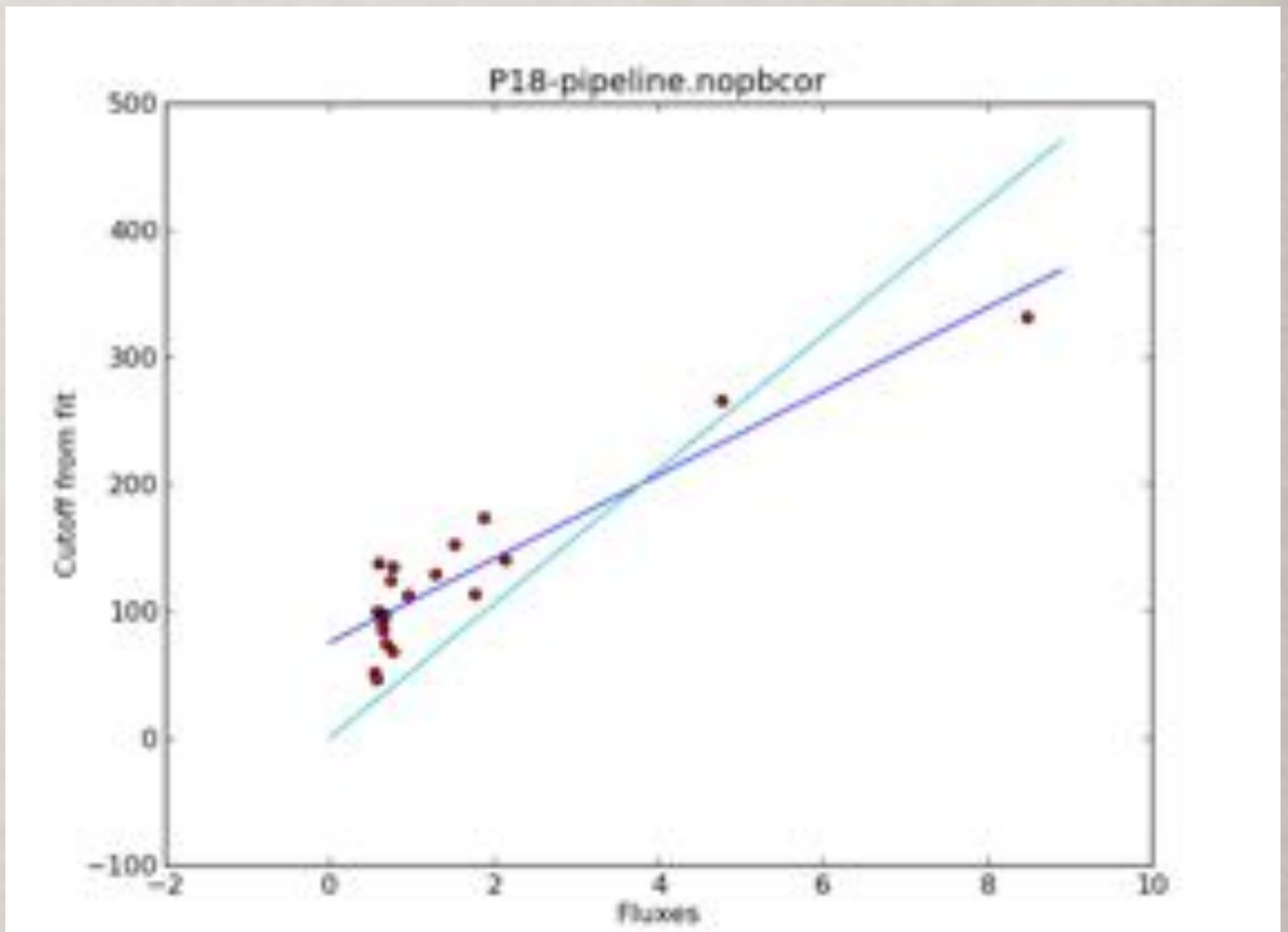
No one size to fit them all

Identify N brightest sources in an image

For each source, calculate rms in concentric squares, fit a function

Calculate radius at which rms is a factor from global rms for an outer scale

Also calculate ‘e-folding’ radius for an inner scale



Artifact ‘size’ versus source flux

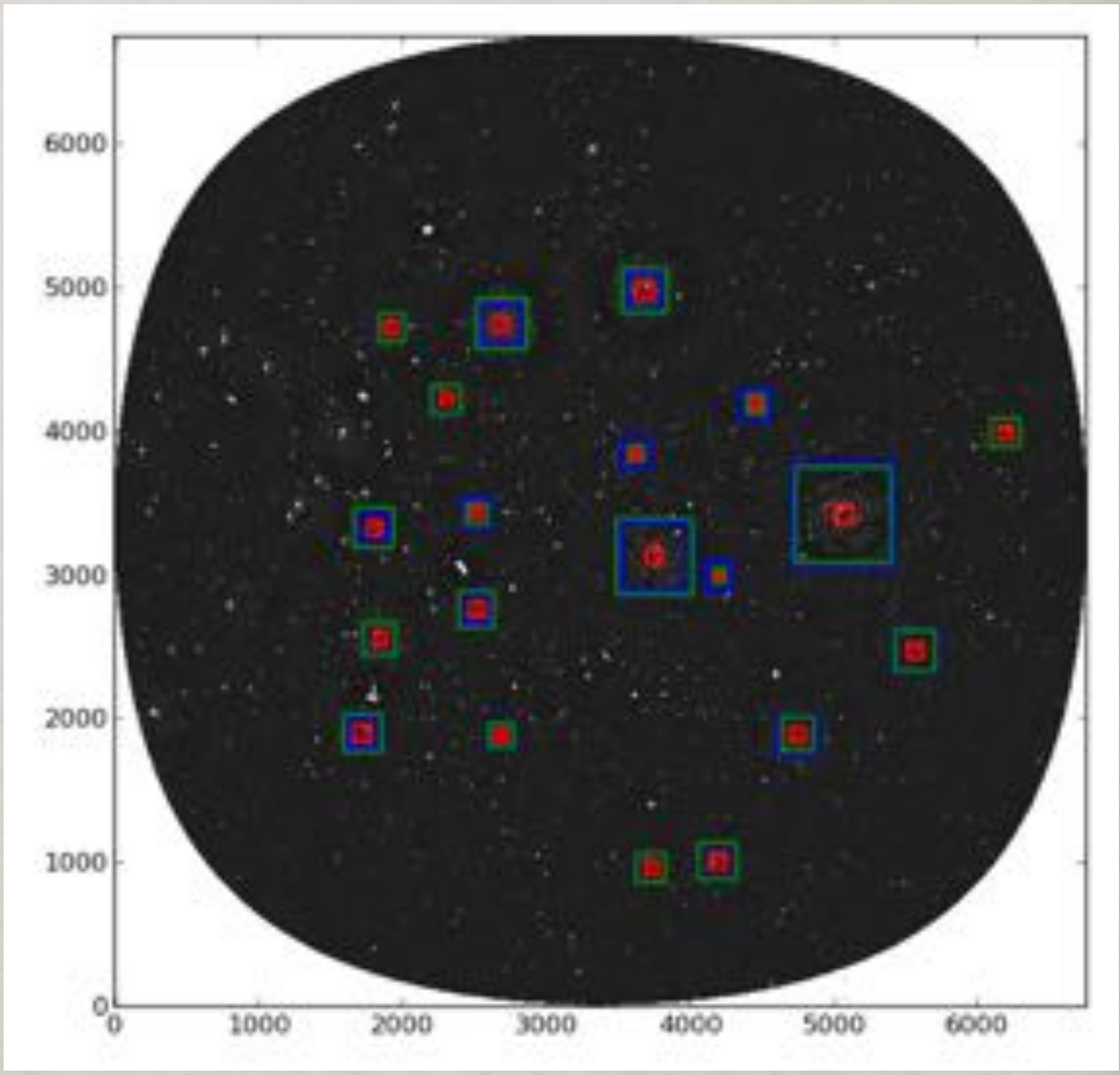
No one size to fit them all

Identify N brightest sources in an image

For each source, calculate rms in concentric squares, fit a function

Calculate radius at which rms is a factor from global rms for an outer scale

Also calculate ‘e-folding’ radius for an inner scale



Artifacts around 20 brightest sources (green/blue : outer scale; red : inner scale)

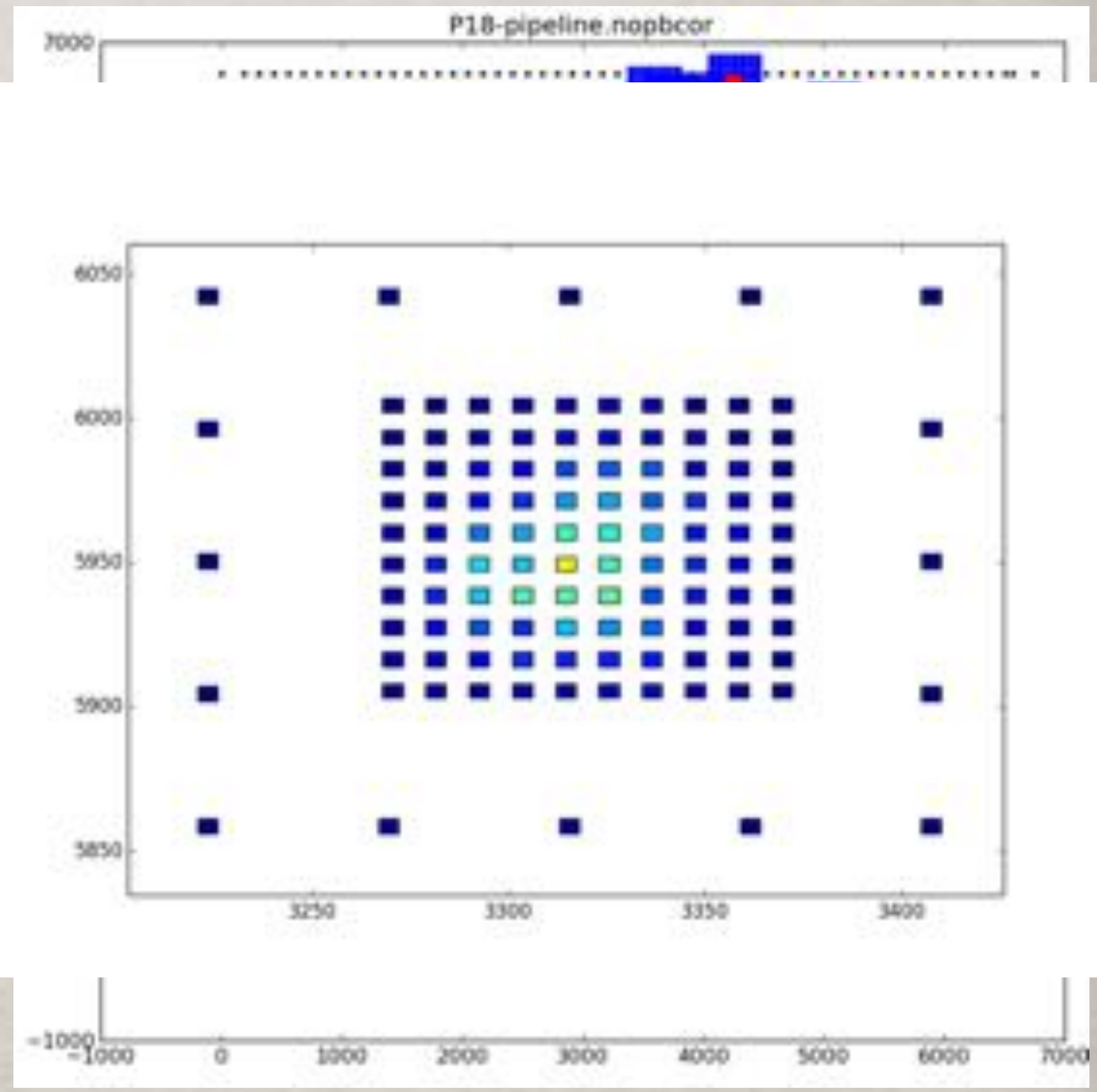
No one size to fit them all

Implement a scheme to decide on nested boxes of decreasing size around each source

These boxes depend on the flux of each source

RMS is evaluated inside each of these boxes and interpolated

To correct the rms image around bright sources

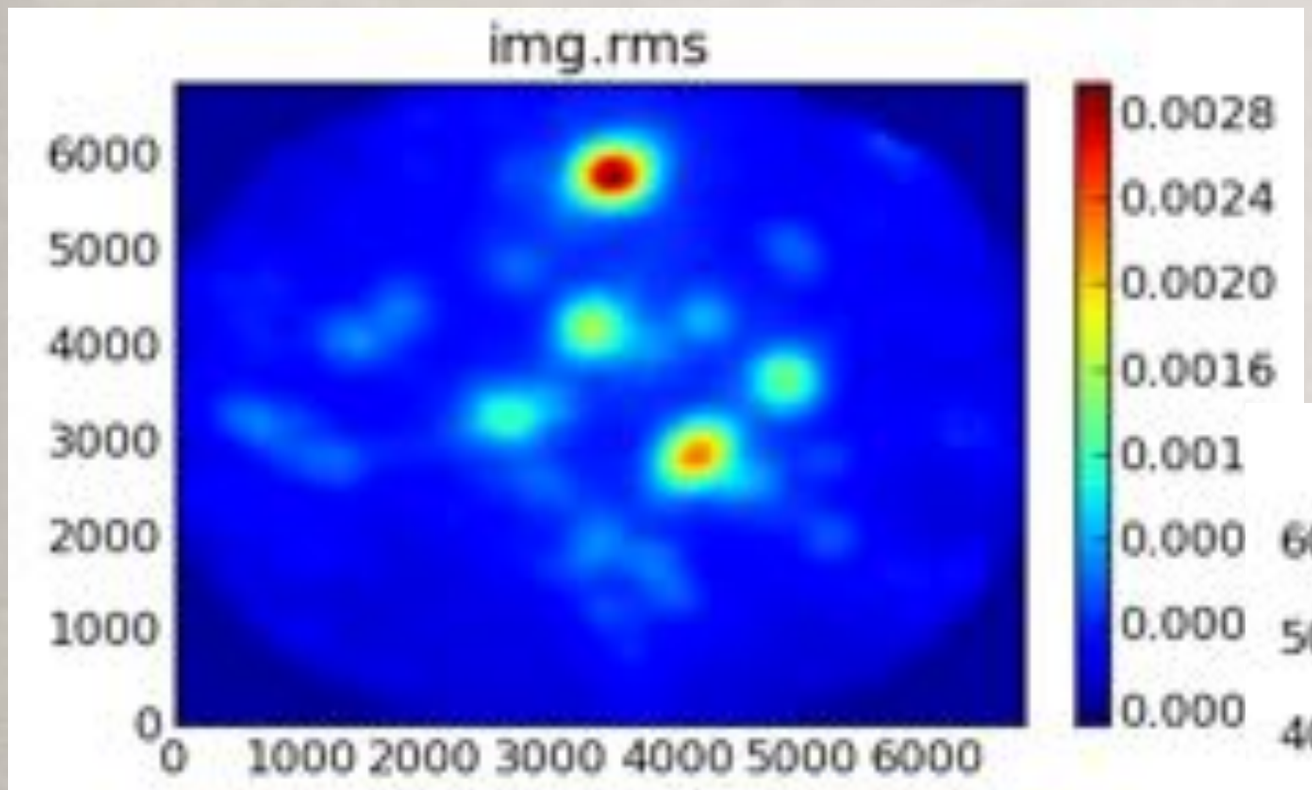


Black : original regular grid

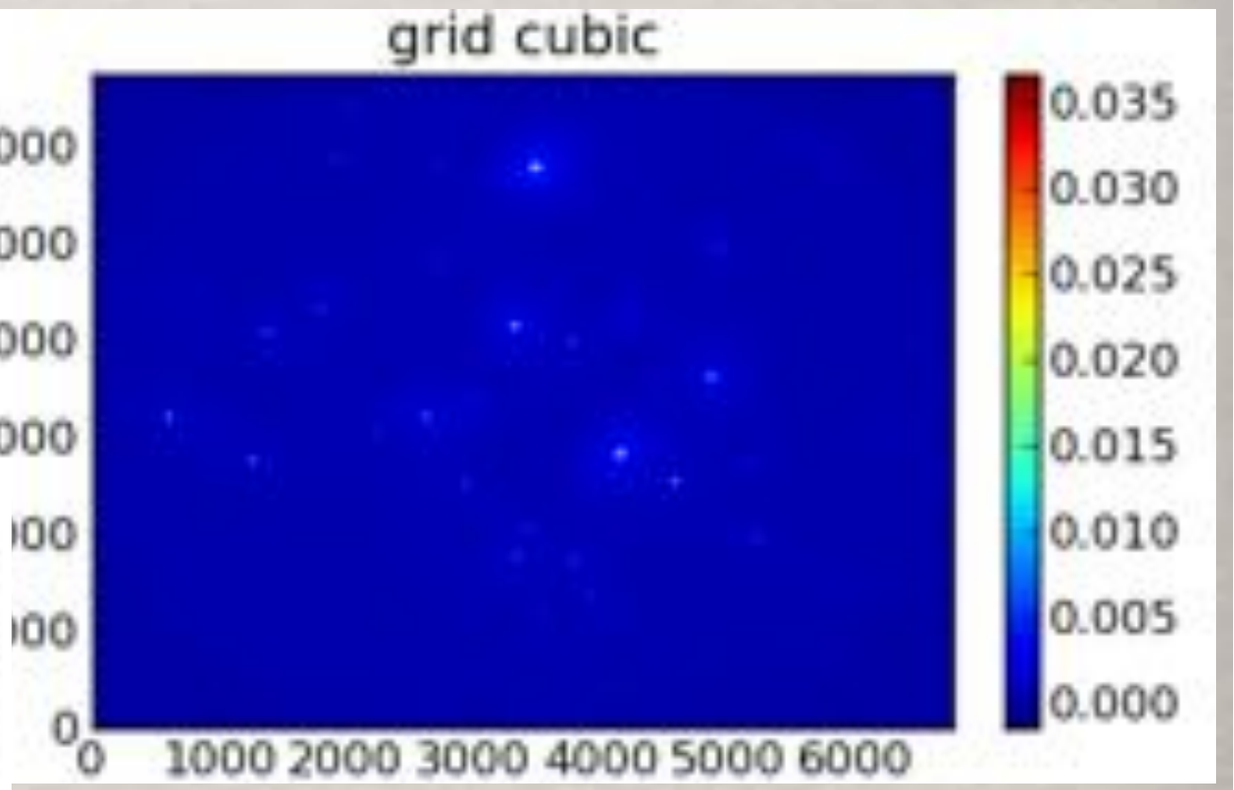
Blue and red : new smaller boxes

Centres of boxes to calculate rms in

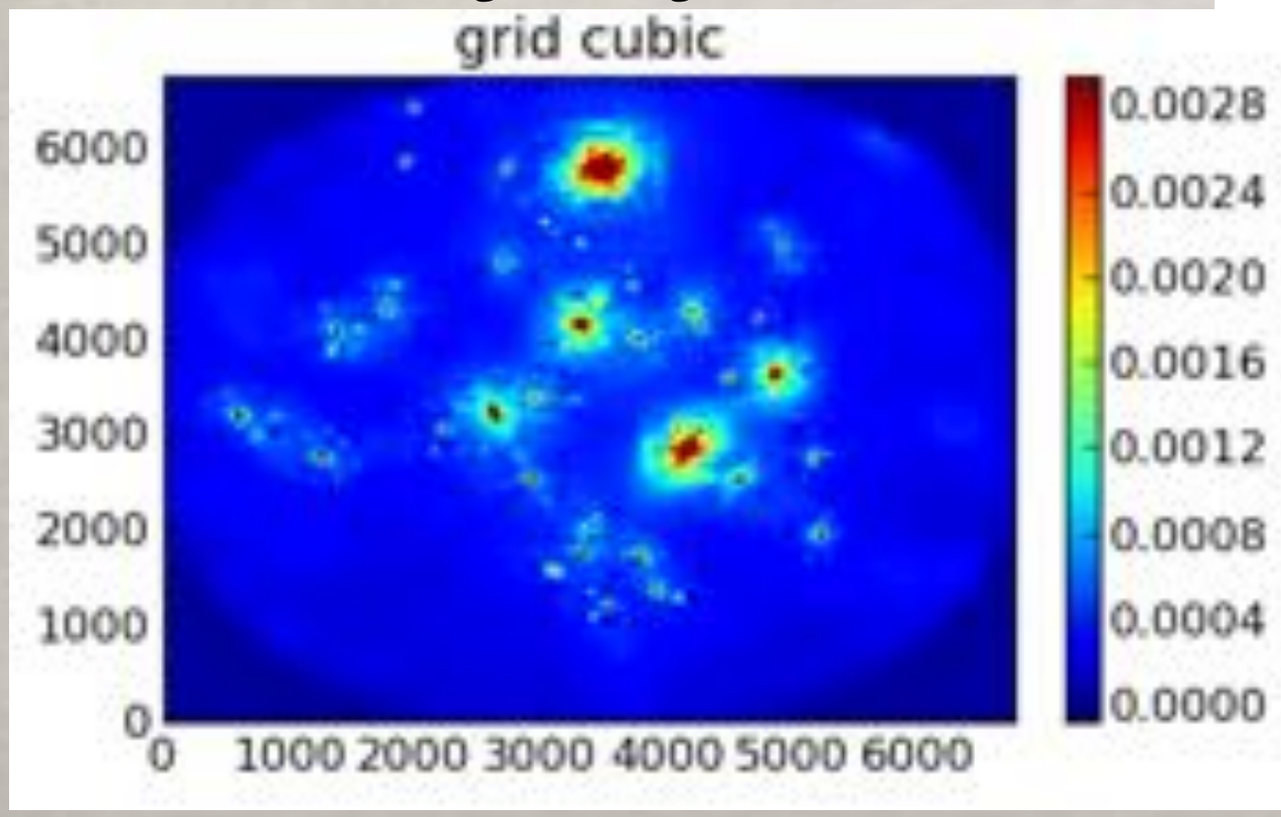
No one size to fit them all



RMS image, single box size



RMS image, adaptive boxes

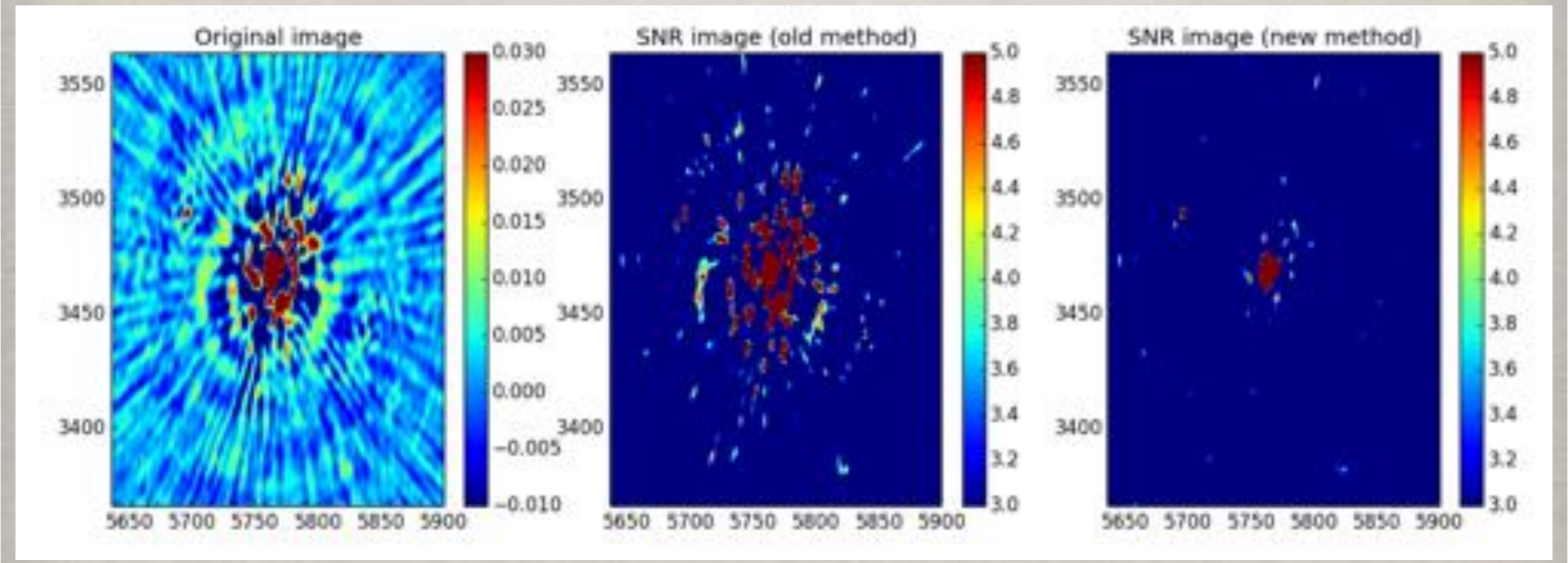


RMS image, adaptive boxes, same scale as single box image

No one size to fit them all

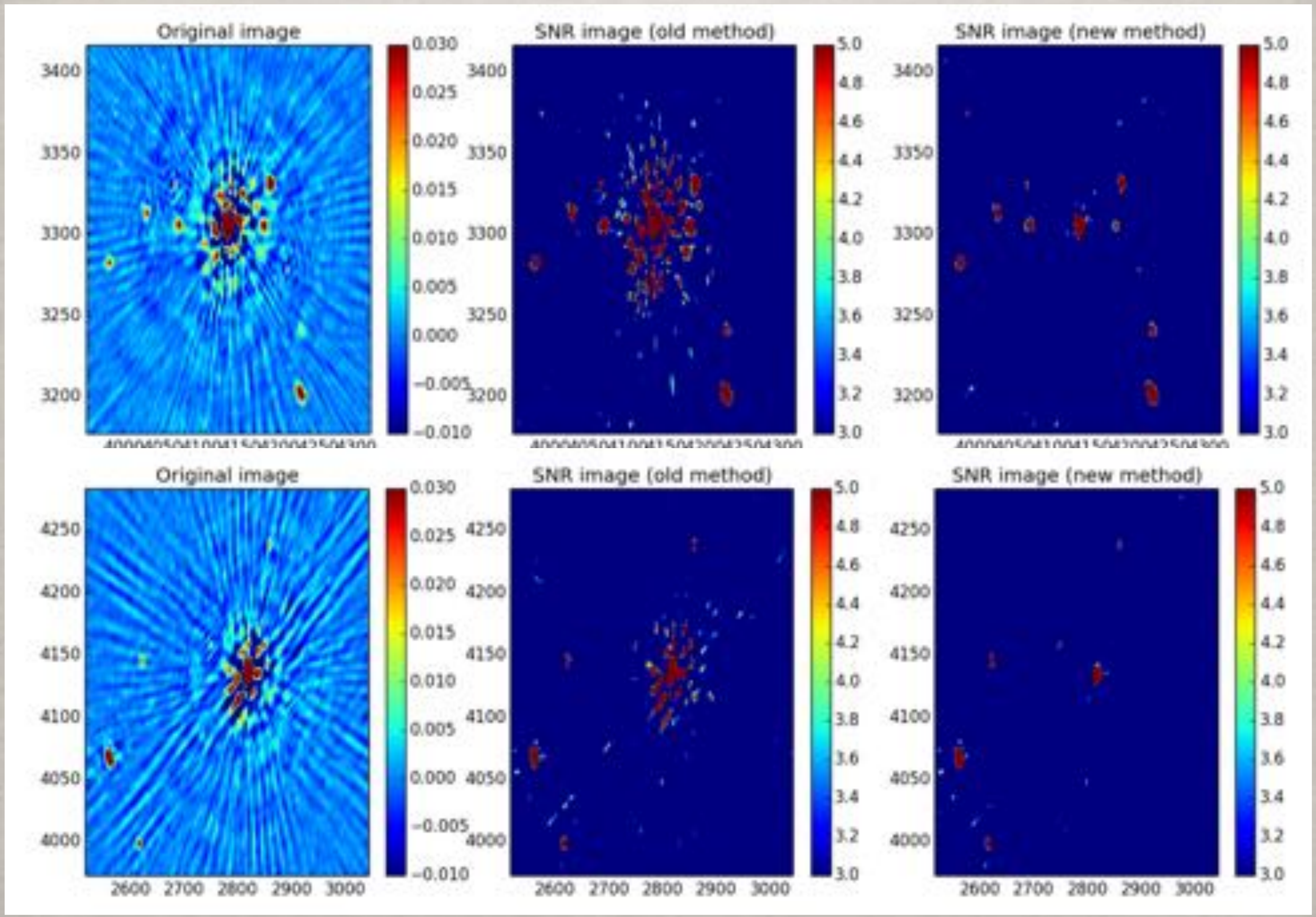
We can now construct the interpolated rms image that samples the rms variation differentially based on source flux

Original image *SNR (old method)* *SNR (new method)*



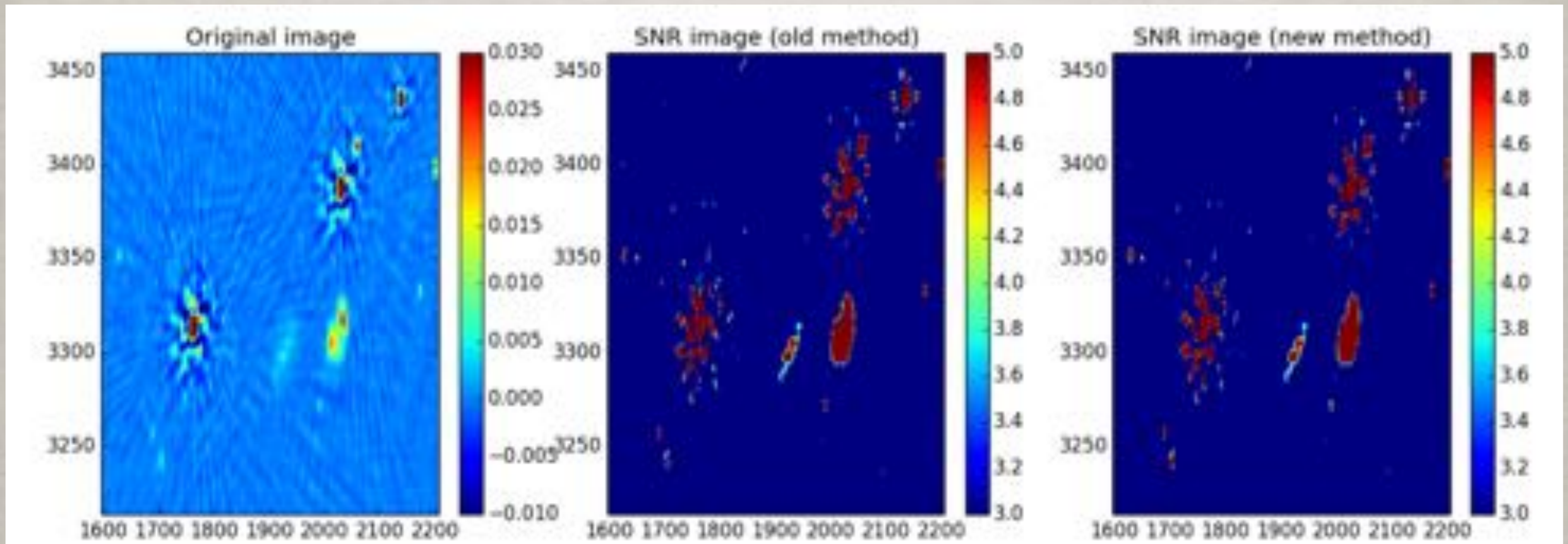
P1 image from the LOFAR HBA survey (image courtesy Shimwell)

No one size to fit them all



No one size to fit them all

One needs to run this analysis down to low SNR (~50)
and small sizes (< 5 beams)



We need to fine tune this part of PyBDSM and re-run on the LOFAR HBA survey images to recreate more accurate catalogues and calculate completeness as a function of flux (and size)

What else is needed for SKA?

- Algorithms need to be tuned better, with schema to estimate optimal parameters from the image itself
- Estimate astrometric and photometric quality factors from analysis
- PyBDSM has a separate path for A team sources, but needs improvement
- PSF variation algorithm to be adapted for various calibration techniques
- Machine Learning to recognise morphologies and for classification
- More accurate errors on fitted parameters (move beyond Condon)
- Explore other basis sets, e.g. directional wavelets (ridgelets, curvelets) for diffuse structures
- Make the runtime faster

Testing the vanilla version

Simulate sources from known counts, gaussian fits from PyBDSM and SAD/AIPS

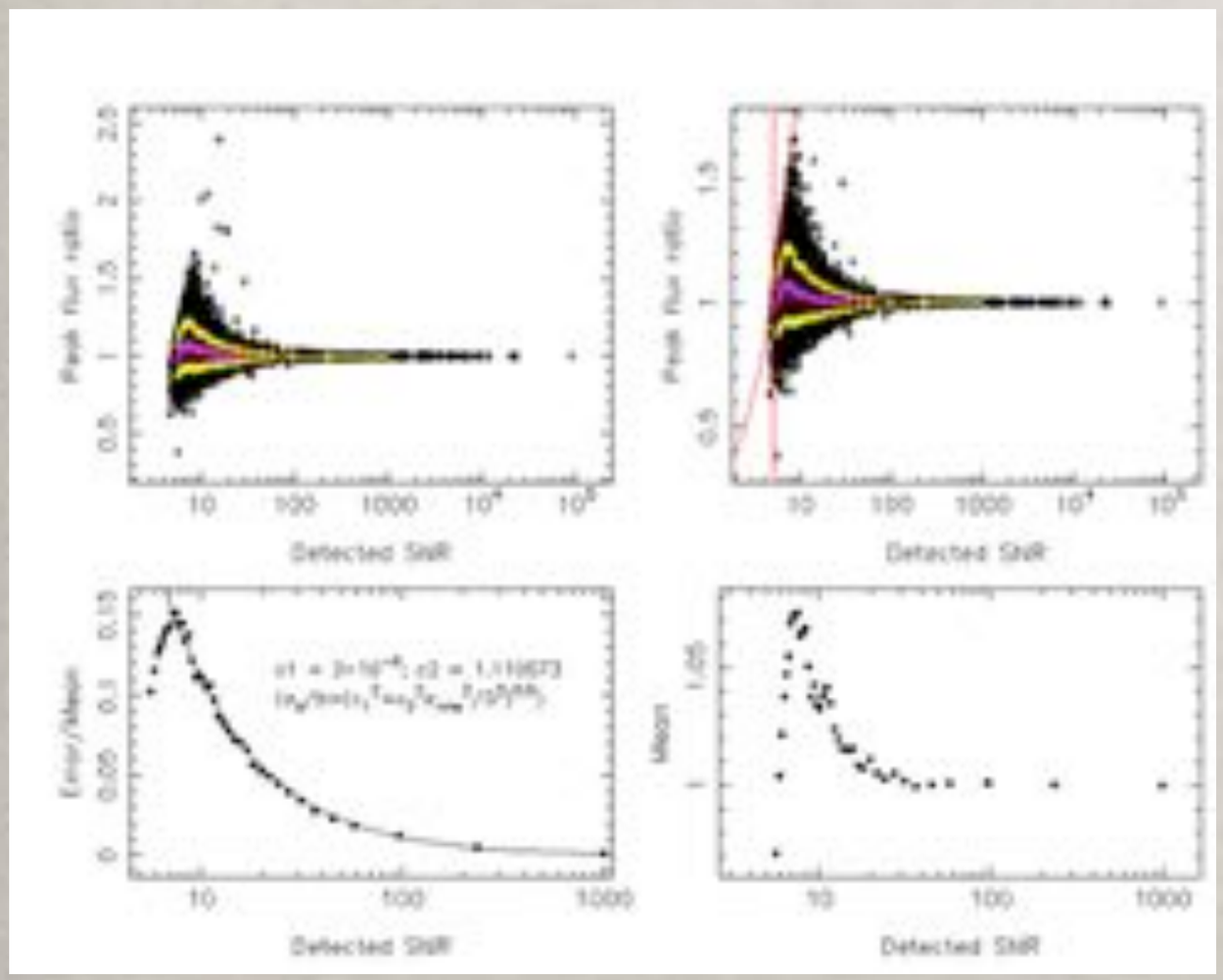
Compare performance by parametrising departure from input flux, posn, width etc vs SNR

For example, peak flux ratios S_1/S_2

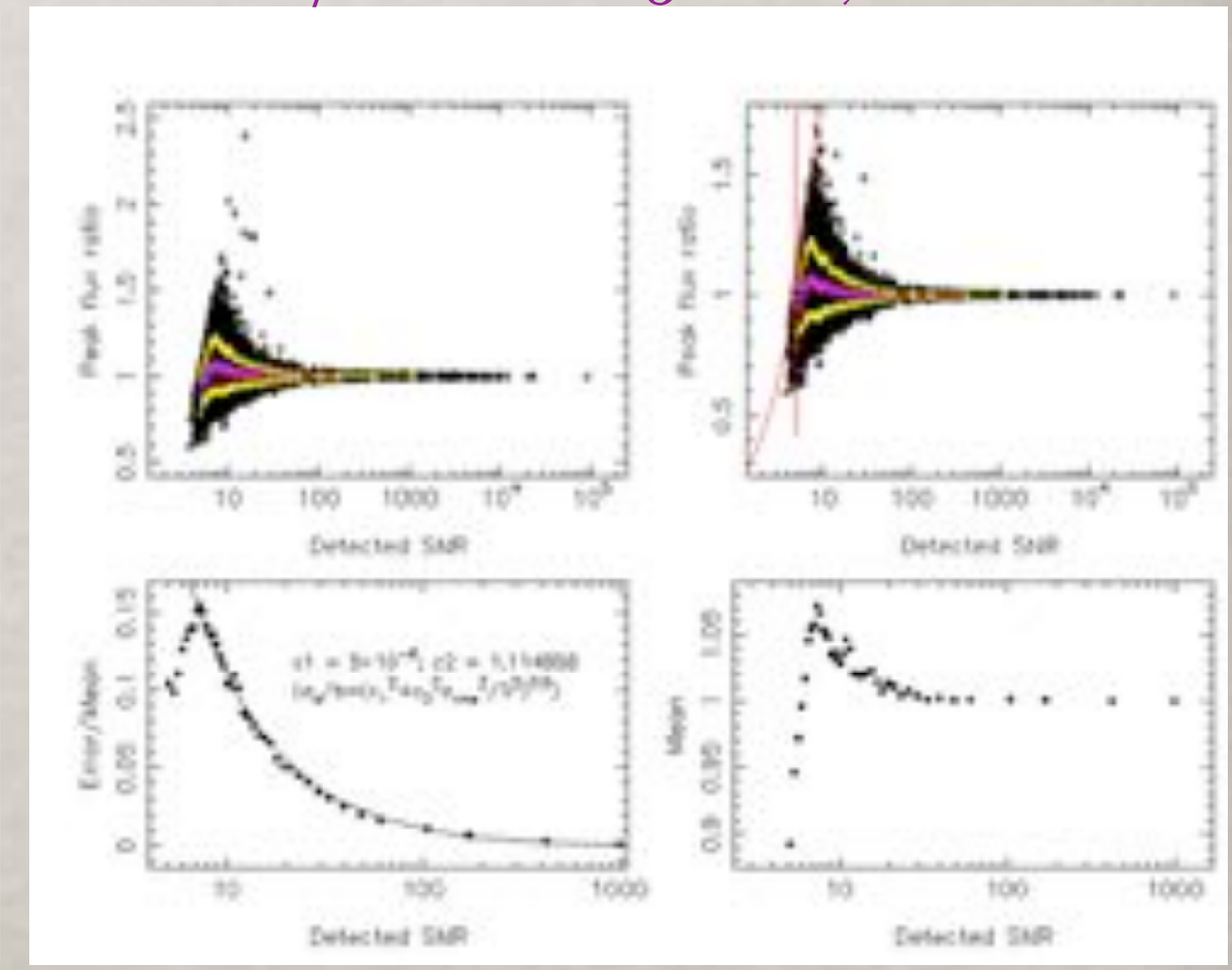
$$var(\frac{S_1}{S_2}) = [c_1^2 + \frac{c_2^2}{SNR^2}]^{\frac{1}{2}}$$

PyBDSM : c1 = 2 x 10^{-6}, c2 = 1.11

SAD/AIPS : c1 = 5 x 10^{-6}, c2=1.11

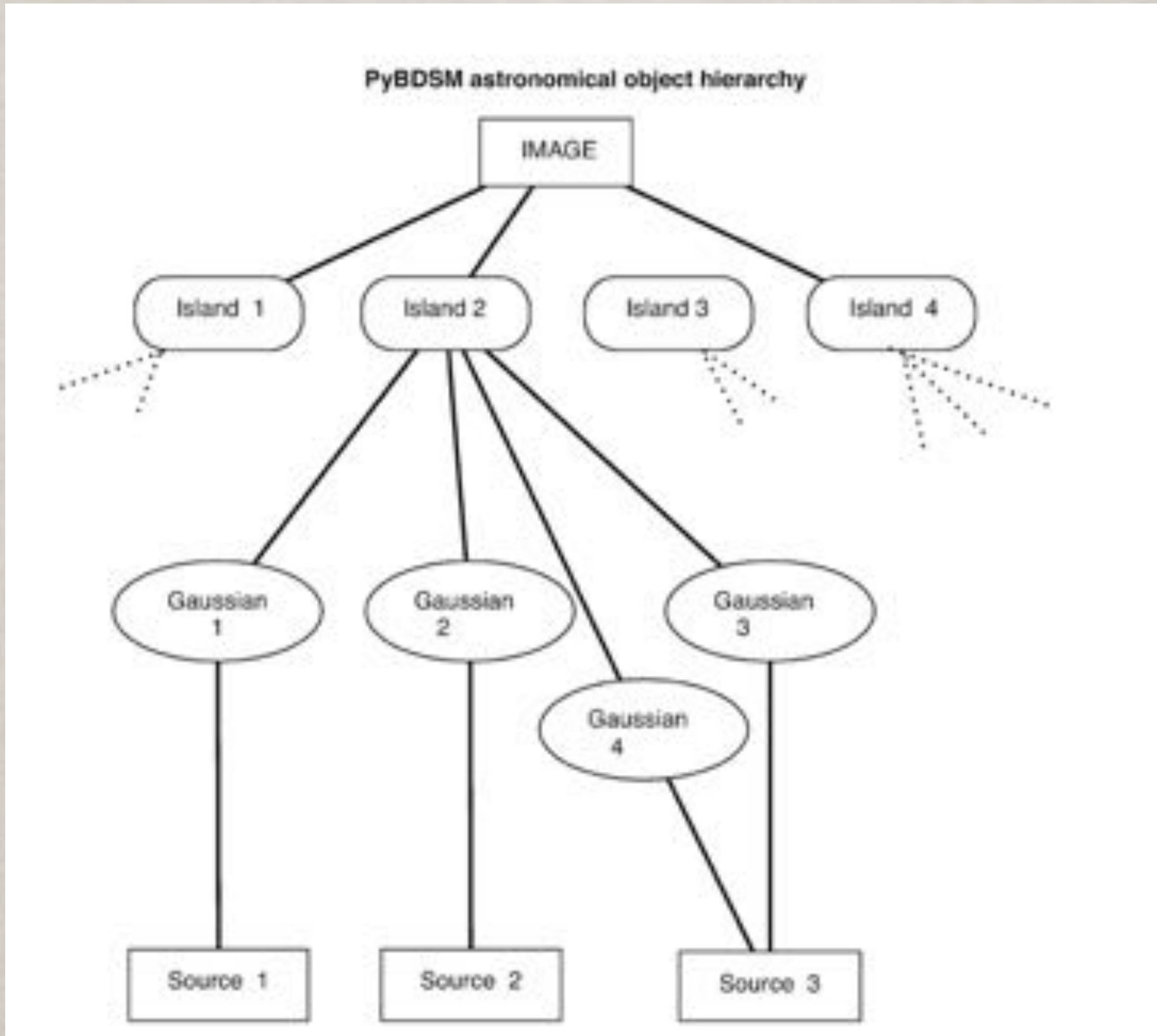


PyBDSM



SAD/AIPS

Bells and whistles : reblending

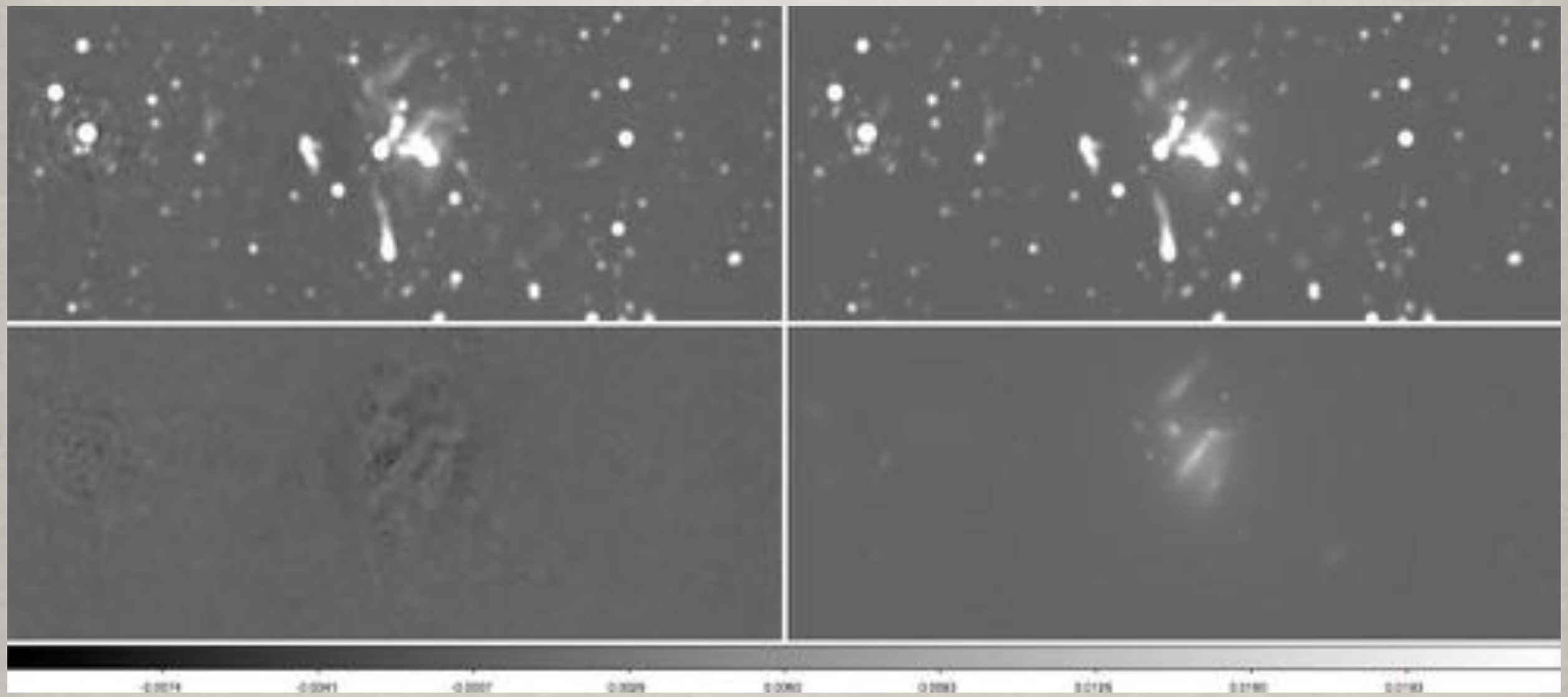


Wavelets for extended emission

- Wavelet transforms are used to characterise extended emission

Original image

PyBDSM model



Residual

Wavelet model

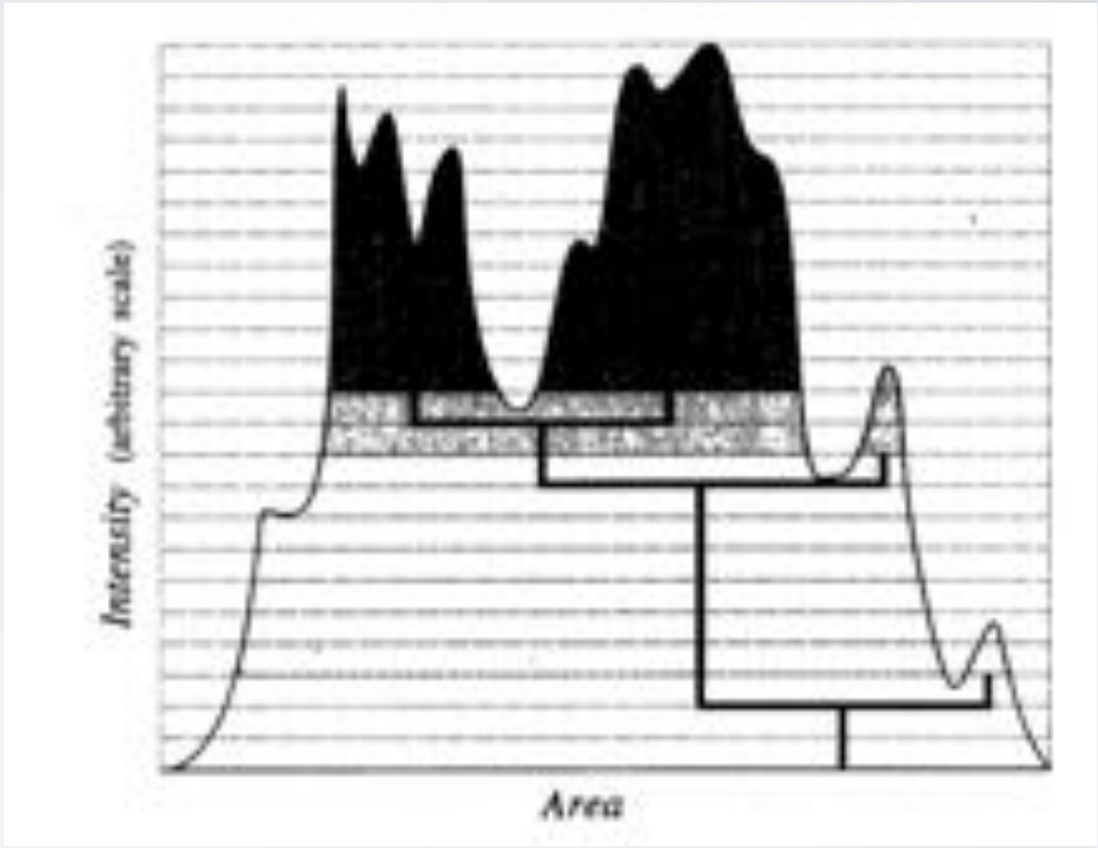
PyBDSM multi-scale wavelet model of Abell 2255

Deblending Of Gaussians

For islands which are extended, have many peaks of emission and are larger than around 5 beams, the initial guess for the gaussian fitting procedure is fairly important. This deblending of extended sources can be done in many ways.

SExtractor does deblending by producing a tree structure with flux contrast criterion. AIPS fits multiple gaussians based on successive chi square, with initial guesses. MOPEX uses multiple thresholds.

PyBDSM has three method for determining the initial guess for the gaussian set, to be used for active deblending while fitting.



Deblending scheme for SExtractor

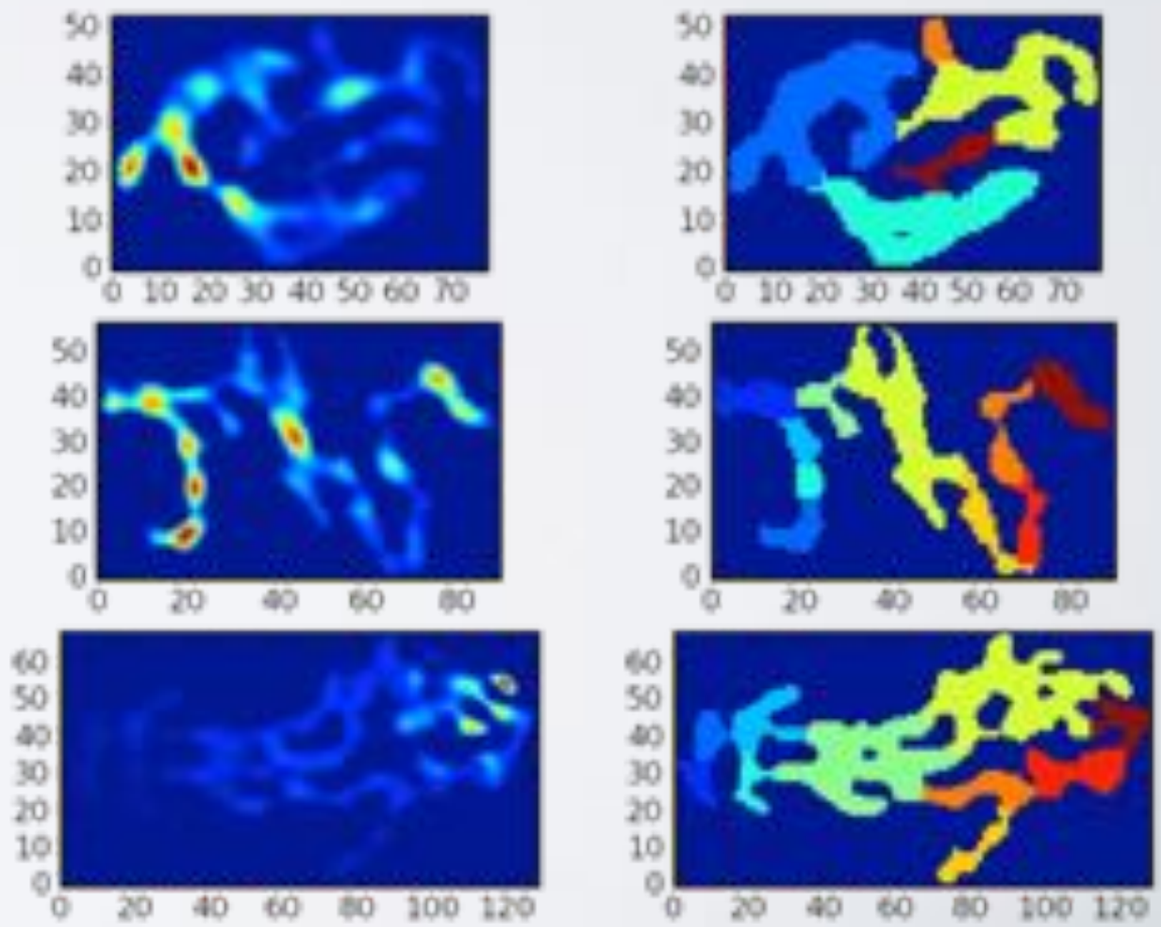
1. Assume a synthesized beam at the peak pixel and fit a gaussian. If the brightest peak in the residual image is higher than a threshold, add another gaussian (shaped like the beam) at this position and iterate.
    - Used in SAD/AIPS
    - Is usually sufficient for most islands
    - Doesn't work for very large sources, with components much bigger than beam

Deblending Of Gaussians

2.Obtain all distinct peak pixels, assume a gaussian (like a beam) at each location. Calculate the residual image. If the peak value and rms of the residual image are large enough and the peak pixel is not too close to the existing list of gaussians, place another gaussian (like a beam) at the peak pixel and iterate. This is the default option.

3.Scale-free guess, (used for wavelet transforms). This is not trivial. For a single peak, we take the moments. For more than one peak, the *watershed* algorithm is used to divide the island into distinct areas around each peak. For each such area, a mask is created and a gaussian is fit using its moment as the initial guess. The set of all such gaussians, one for each area, is the set of initial guesses for the island. Finally, the diffuse emission is also similarly fit, to provide one more guess solution.

Fitting very large islands (typically > 50 beams) which need 10s of gaussians takes forever. These islands are split into smaller ones for fitting purposes using the *Opening* operator (either 3x3 or 5x5) and some constraints including convex deficiency



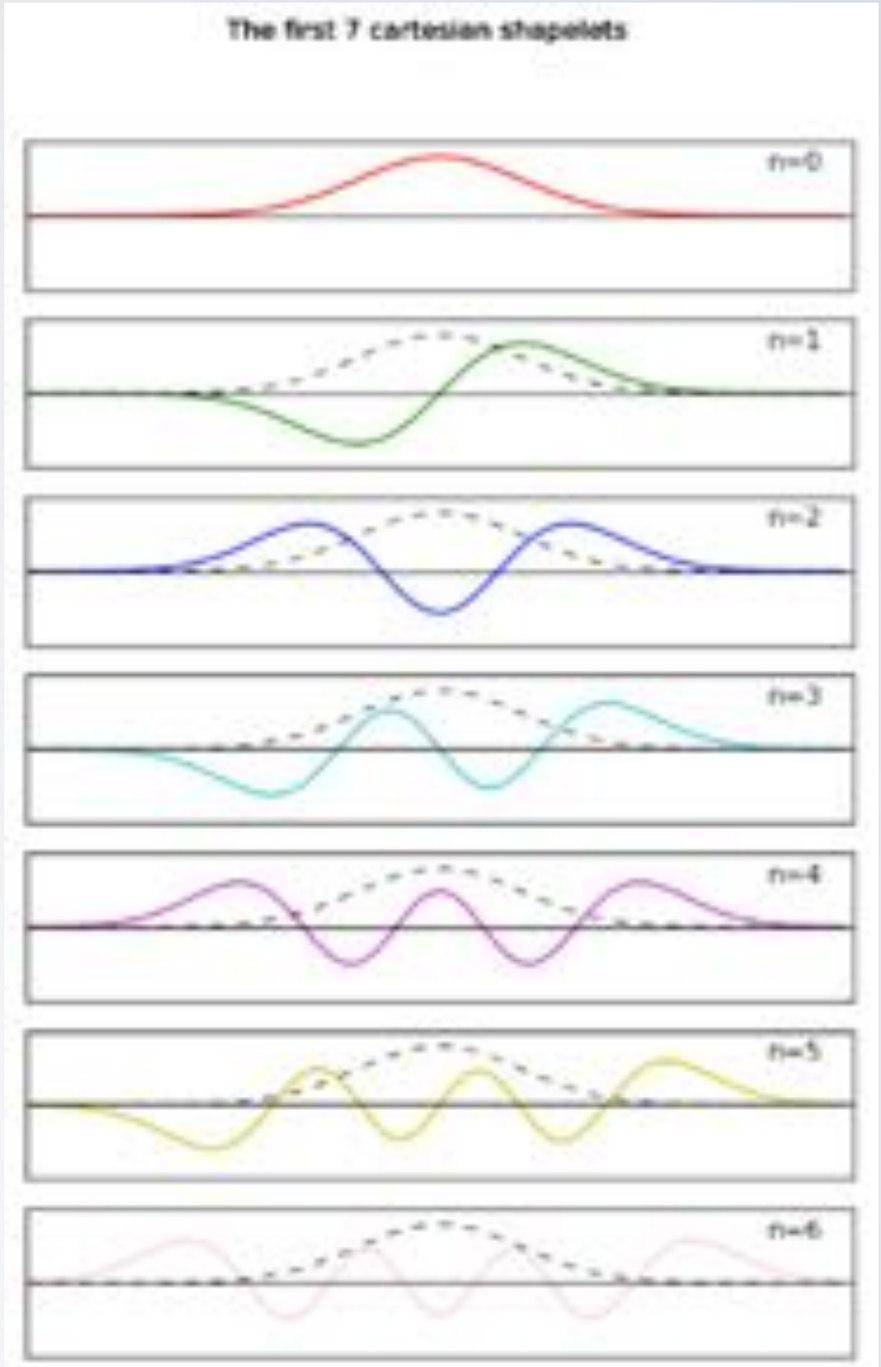
Three islands in the w1 wavelet transform of Cygnus A 327 MHz image, and their splitting into sub-islands.

Shapelets As A Basis Set

Shapelets are a complete orthonormal basis set.

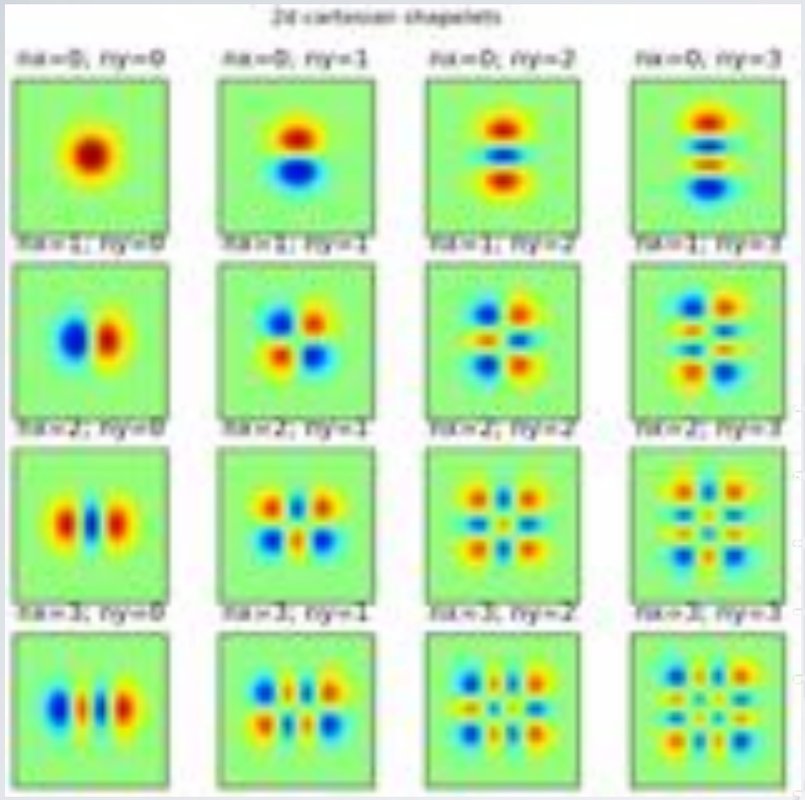
Cartesian shapelets are Hermite polynomials, weighted by a gaussian. Polar shapelets use Laguerre polynomials instead.

- Shapelets are highly localised
- The first order shapelet is a gaussian
- They are orthonormal and have simple FT and convolution properties
- They are defined by a centre and a scale
- The inner products of 2 shapelets in x and y yields an orthonormal 2d shapelet basis
- Localised images are extremely well decomposed by shapelets (for a given maximum order)
- Shapelets have been used in weak lensing and now, in radio (and optical) astronomy



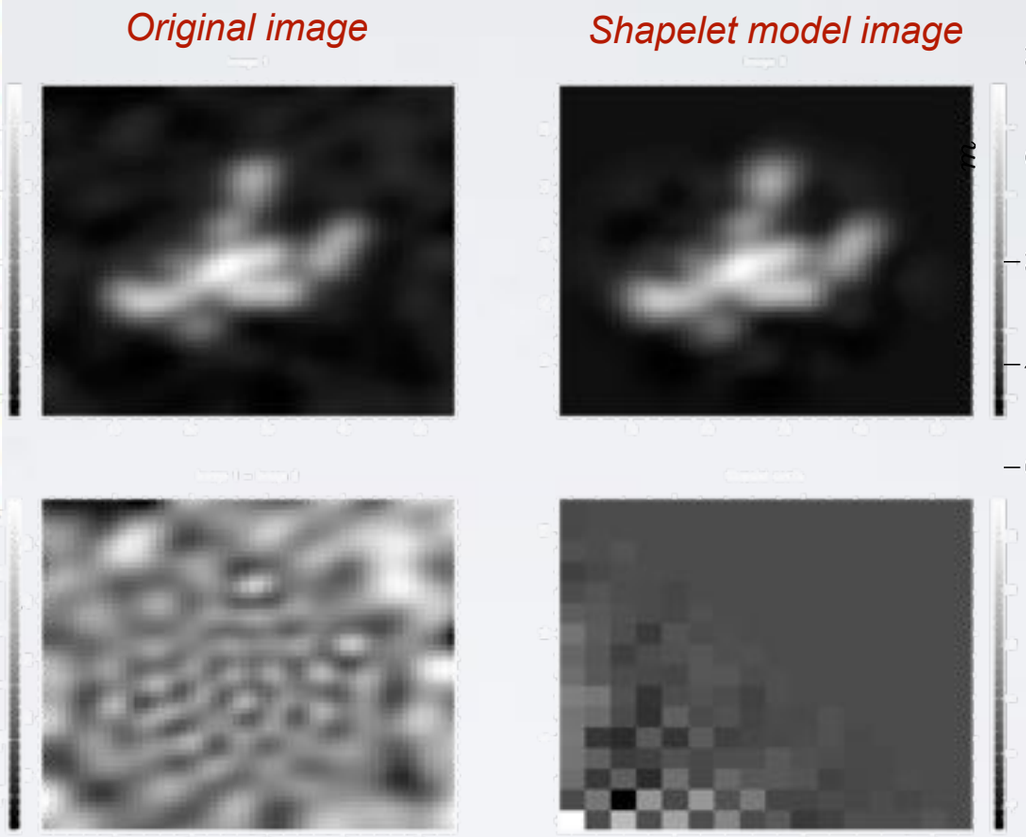
(See papers by Refregier, Bacon, Jarvis, Kuijken etc)

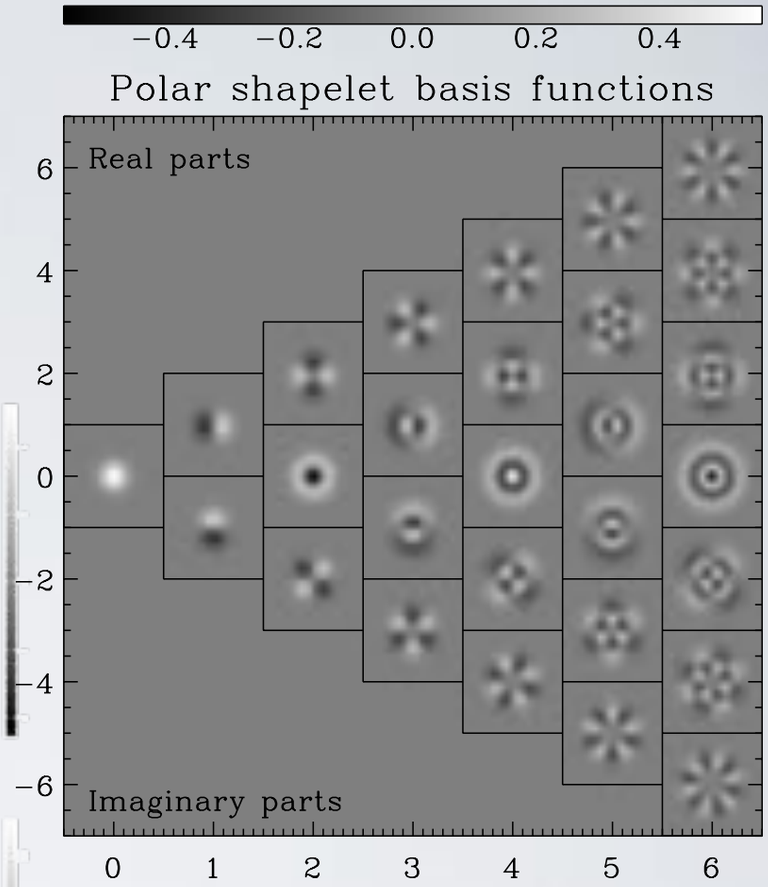
Shapelets As A Basis Set



2d cartesian shapelet basis set

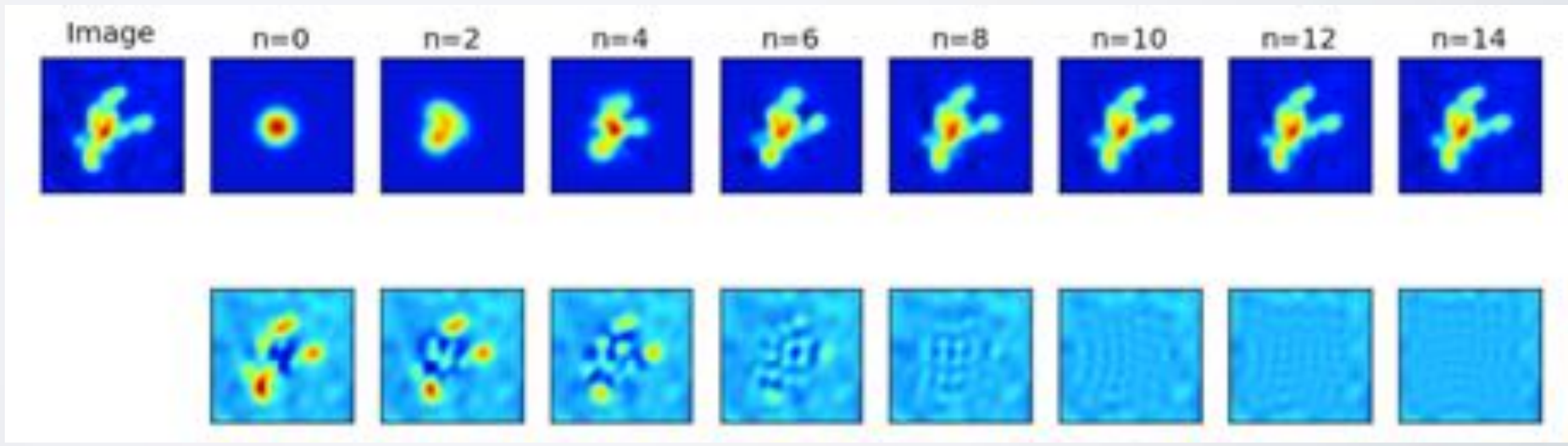
Cartesian shapelet decomposition of a simulated image





(Massey et al 2007)

Shapelet reconstructed (top row) and the residual (bottom row) for nmax as indicated on the figures.

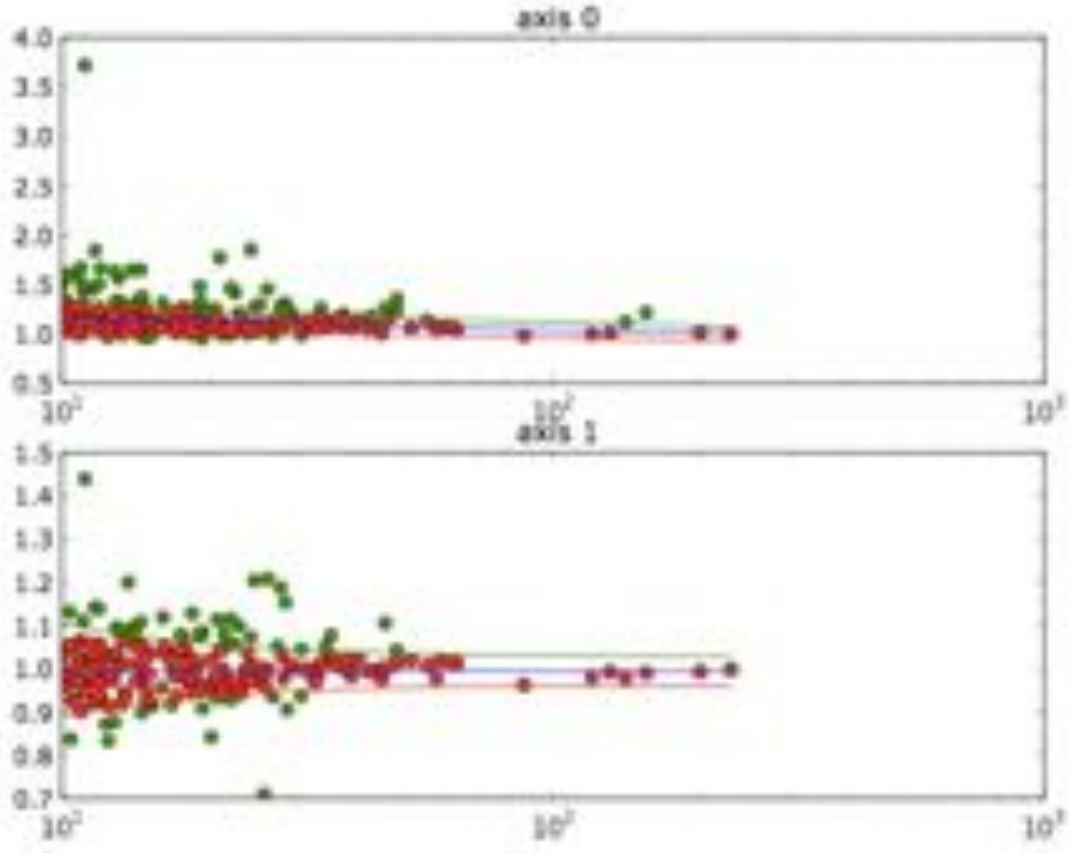


Identifying Point Sources

A point source is in principle, unresolved with the psf of the image. However, generating a catalogue of point sources depends on our knowledge of the psf, and for low frequencies, its variation across the image as well. Hence, given just an image with no extra information, we can only generate a statistically most probable set of point sources from the image itself.

A compact source is a source with only one peak (and is fit not very well by 1 gaussian).

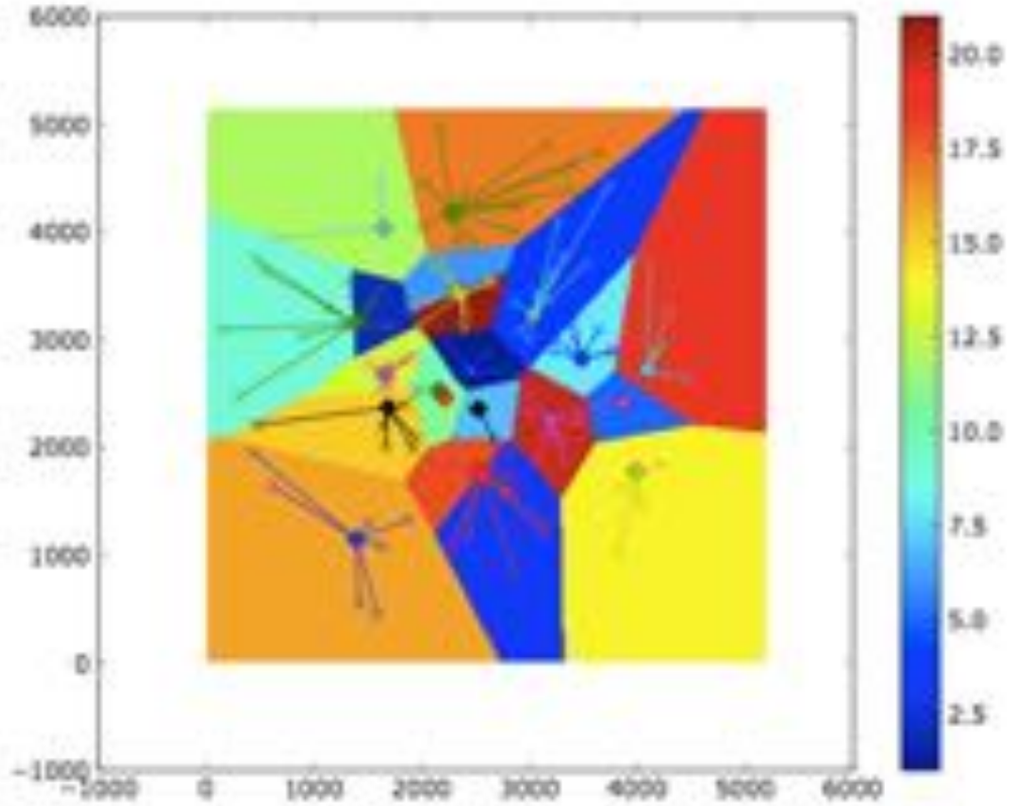
An example is the psf variation algorithm in PyBDSM. Assume that most of the sources are indeed unresolved.

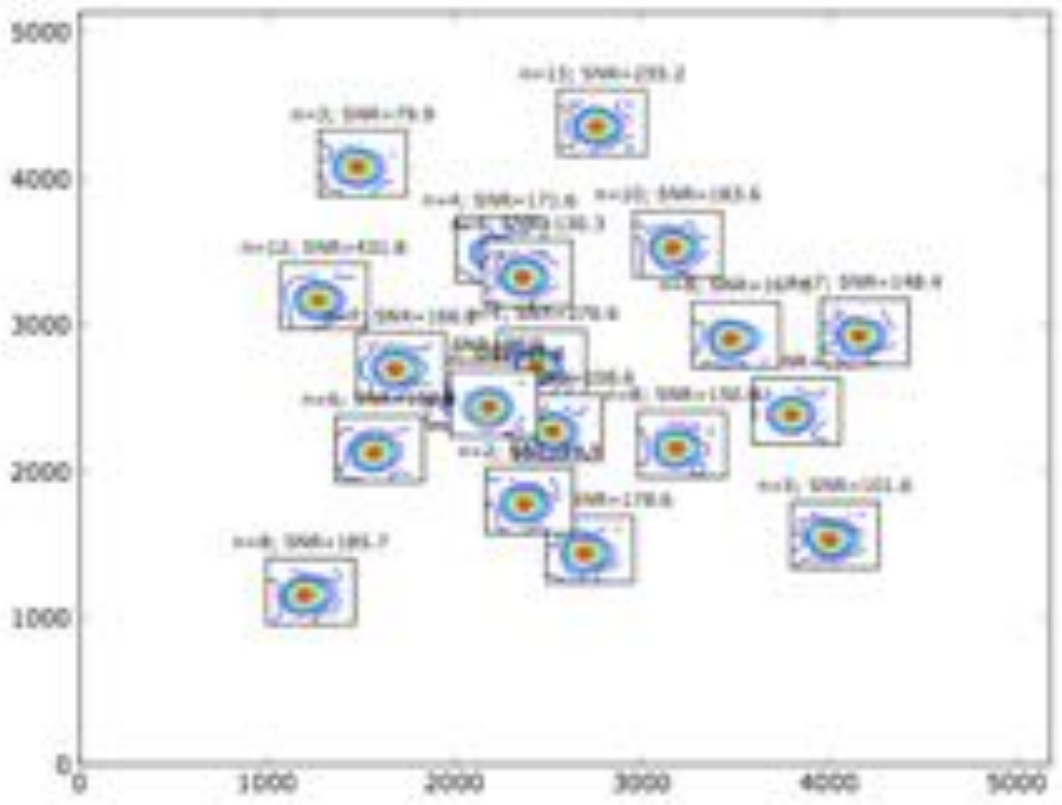


- Plot the distribution function of fitted gaussian widths
- Bin these quantities. In each bin, iteratively clip the outliers assuming a normal distribution
- Now iteratively fit a function to the median and rms of the binned values (2nd order poly for median and sqrt(a+b/x^2) for the rms, clipping outlier bins
- All sources within 2 sigma of the fitted function are deemed to be point sources.

PSF variation across the image

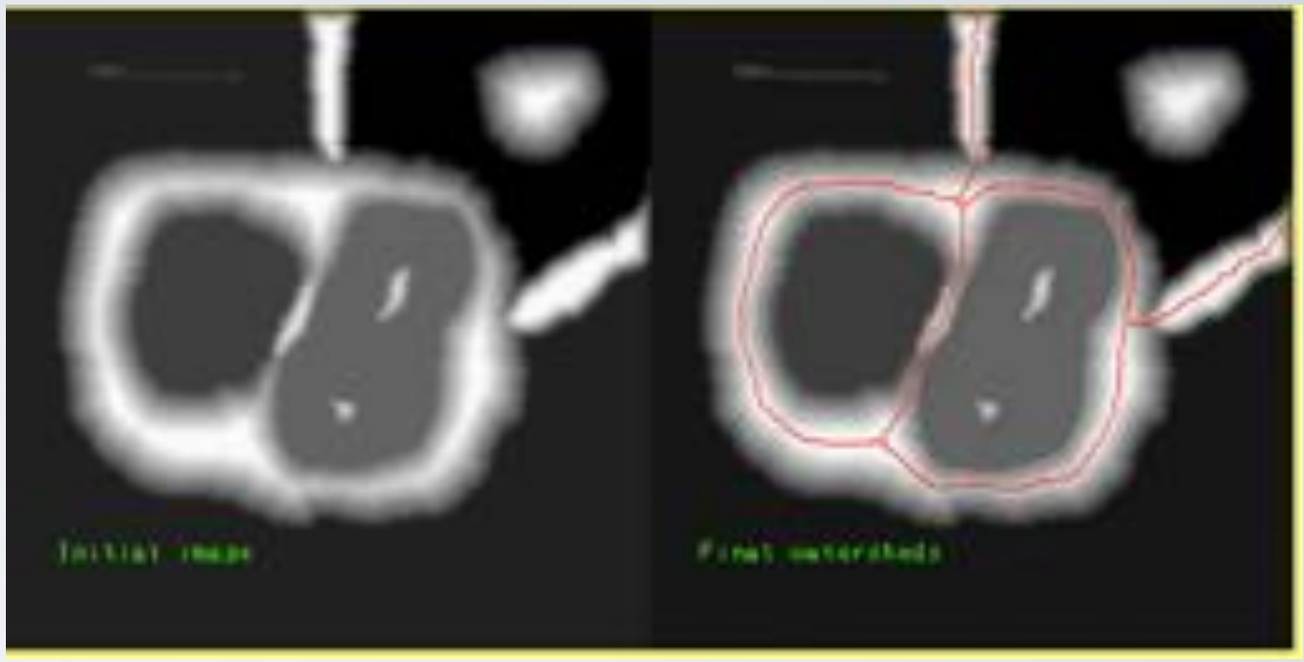
Right : Tiled image with all point sources
Left bottom: Co-added psf in each tile
Right bottom: gridded interpolated PSFs





Segmentation by Watershed algorithm or flood-filling

Imagine each minima (or predefined set of points) to be where water starts flowing from. We define ‘catchment areas’ bounded by maxima and when they intersect, we build ‘watershed’ or ‘bunds’. This way, the entire image is segmented.



Segmentation by Voronoi Tesselation

Each pixel in an image is assigned to a tile based on its ‘distance’ from a predefined set of points. These distances can be any metric, with weights iteratively defined on each tile. There are fuzzy tesselations as well. Voronoi tiles are dual to Delauney triangulation.