boloSource()

- Motivation, concept & implementation
- First results
- Perspectives

Roland Vavrek (HSC) and Gábor Marton (Konkoly)

Map-Maker Workshop @ ESAC, 31 January 2013

Acknowledgements

Csaba Kiss (Konkoly)
Davide Elia (IFSI)
Alessio Traficante (IFSI)
Lorenzo Piazzo (La Sapienza)

Clean maps for analysis of diffuse background

- Extended emission analysis requires clean maps
- Compact objects contribute to the image power spectra with a significant power at a broad range of spatial frequencies:
  - (1) modify the image properties at frequencies comparable to the beam-size
  - (2) depending on the surface density and clustering strength, lower spatial frequencies are contaminated with a smaller power density but typically at large bandwidth
- Image analysis techniques are difficult to compare if sources are not subtracted because their sensitivity to discrete sub-structures may be quite different
  - techniques using sparsity information (i.e. singularity skeleton) could be disturbed by even a few point-sources
  - Techniques analysing full intensity maps are more sensitive to clustering

Motivation for boloSource()

- For diffuse emission analysis we need a technique to subtract sources what fall within a well defined range of spatial frequencies
- A major requirement: preserve noise properties of the image!

➔ Classical way: try modeling the source intensity $I_{(x,y)}$ in the position-position space and subtract from the image

➔ …this is *not easy* but one could reduce the problem to 1D in the detector timeline

- **Subtract sources from the detector timeline and re-project the image**

Source subtraction on the timeline

- Reduced dimensionality in $I_{(t)}$ vs. $I_{(x,y)}$ but noise spectrum in timeline is more complex:

$$I_{(t)} = N_{(t)}^{1/f} + N_{(t)}^{D} + N_{(t)}^{\det} + I_{(t)}^{S}$$

$$I_{(x,y)} \cong N_{(x,y)}^{\det} + I_{(x,y)}^{S}$$

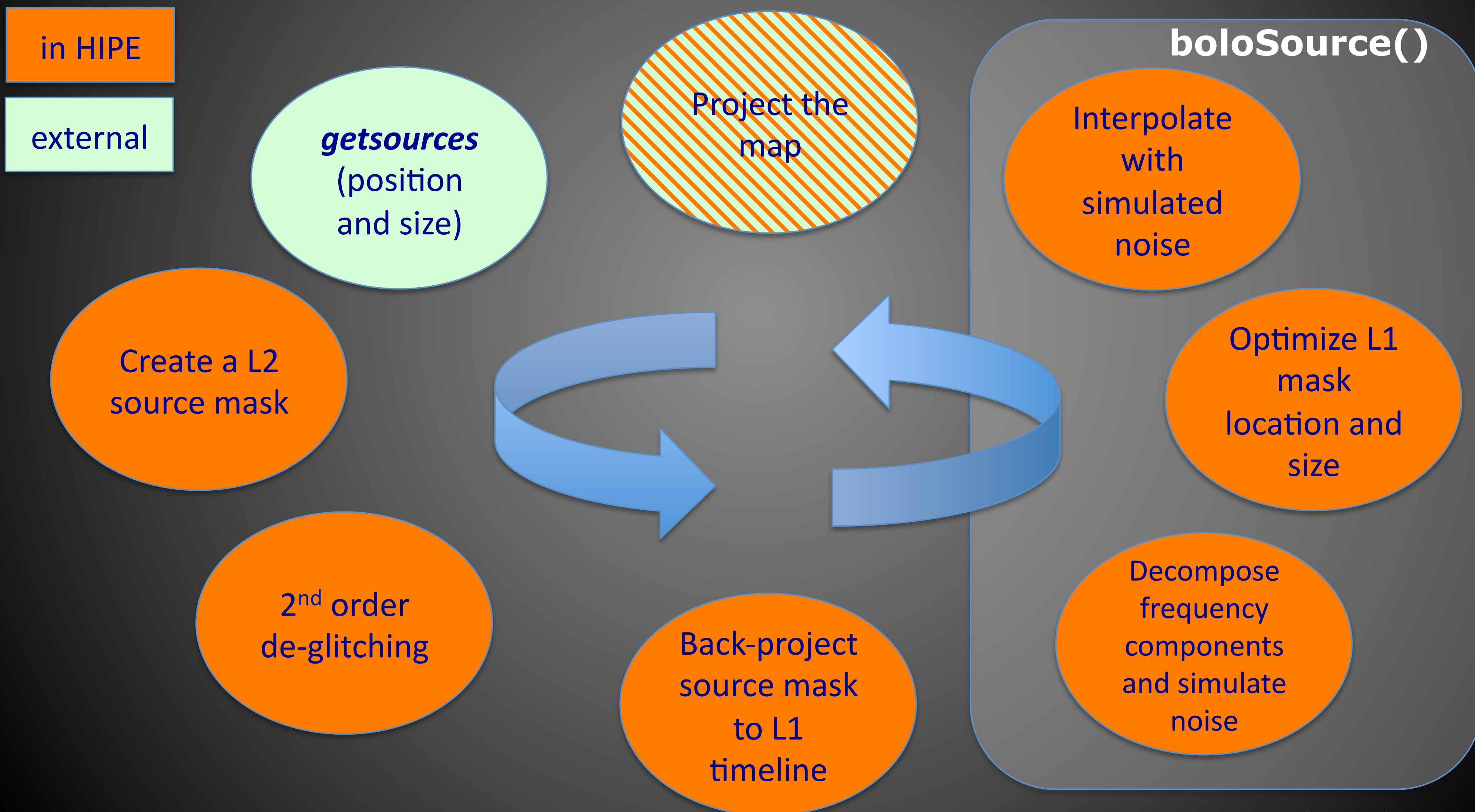
- Lower S/N in L1 timeline than in L2 reconstructed map

- *Looks difficult* but we mainly interested in to subtract high-frequency components. **In the masked part of the timeline one could interpolate with simulated noise plus sky background:**

Interpolated intensity in masked timeline →

$$I_{(t)} = \underbrace{N_{(t)}^{1/f} + N_{(t)}^{D} + N_{(t)}^{\det}}_{\text{Simulated noise}} + \underbrace{I_{(t)}^{S(lowfreq)}}_{\text{Baseline estimate from data}}$$

Workflow

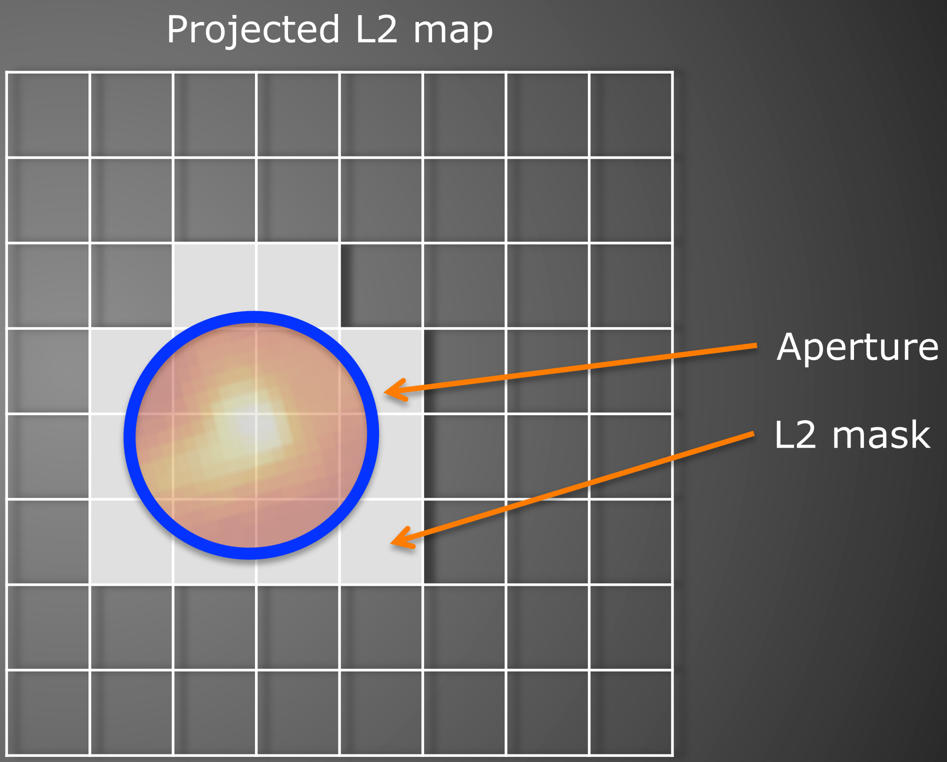


in HIPE
external
getsources (position and size)
Project the map
boloSource()
Interpolate with simulated noise
Create a L2 source mask
Optimize L1 mask location and size
2nd order de-glitching
Back-project source mask to L1 timeline
Decompose frequency components and simulate noise

L2 mask

Create a L2 source mask

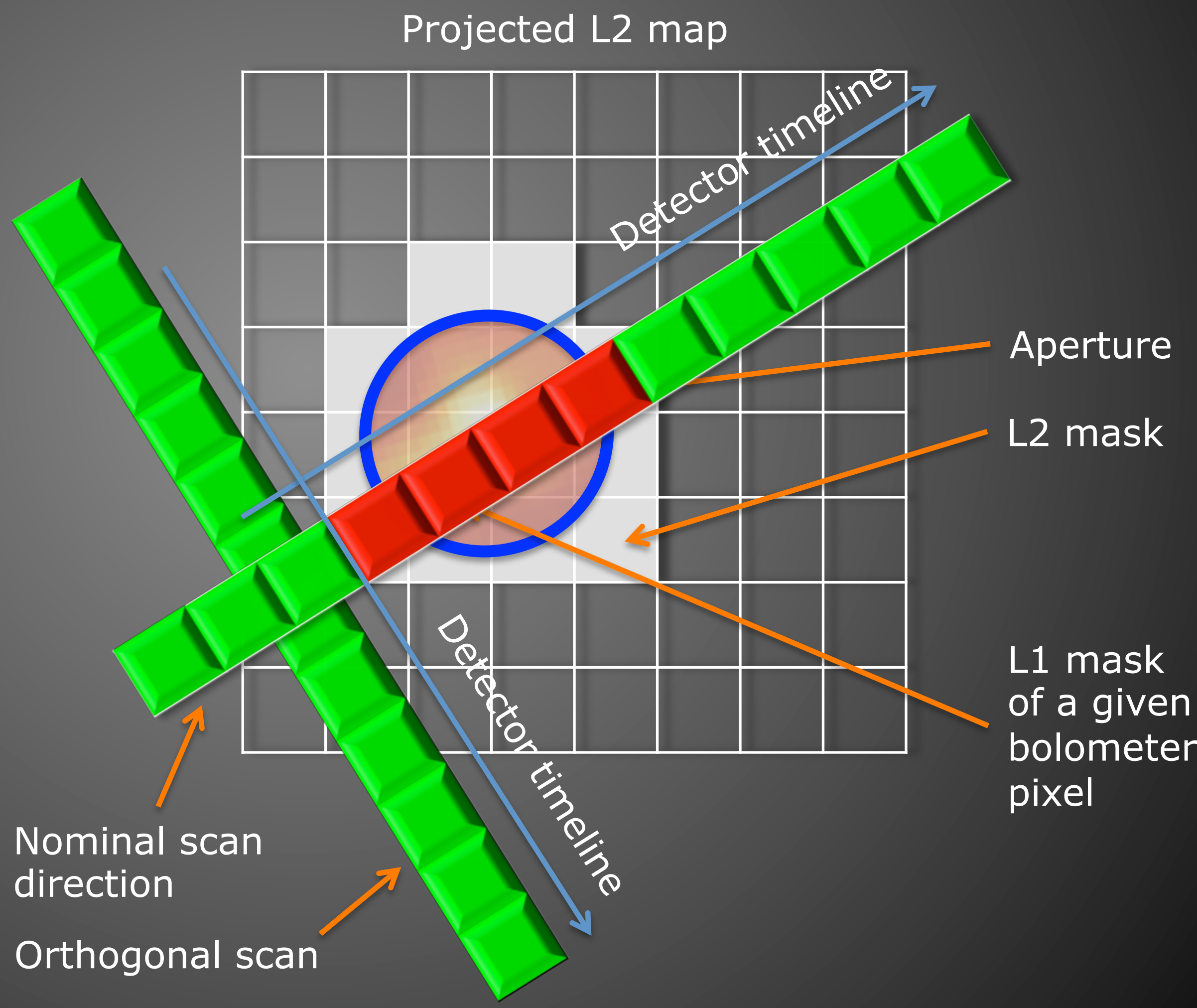
- Get source centroid & size from getsources
- Create a L2 mask for circular aperture
- Aperture radius defines the cutoff frequency



L1 mask

Back-project source mask to L1 timeline

- Get source centroid & size from getsources
- Create a L2 mask for circular aperture
- Aperture radius defines the cutoff frequency
- Backproject L2(x,y) mask to L1(t) timeline mask
- L1 mask size depends on L2 aperture size ***AND*** scan-speed



Signal decomposition

Decompose frequency components and simulate noise

Stationary wavelet transform SWT (à trous algorithm)

- Wavelet coefficients contains the same number of samples as input signal (redundant representation similar to CWT)
- Inverse transform is trivial and provide full control over frequency components
- Residual signal conserves flux
- Filter width scales with power of 2
- Similar performance (in speed) to DWT algorithms

Baseline estimation

Decompose frequency components and simulate noise

- Take L2 mask size (i.e. effective diameter of input mask aperture) for a given source, this determines the high-frequency cutoff for the baseline estimate on the L1 timeline (Note, the L1 adaptive mask size cannot be used as it requires a priori baseline estimate!)
- Decompose signal with SWT
- Flag high-frequency outliers above a given sigma level on the entire scan leg
- Interpolate flagged outliers with baseline estimate
- Solve inverse transform, reproduce intensities
- Create output products:
  - Baseline
  - Noise dataset
  - Outlier mask

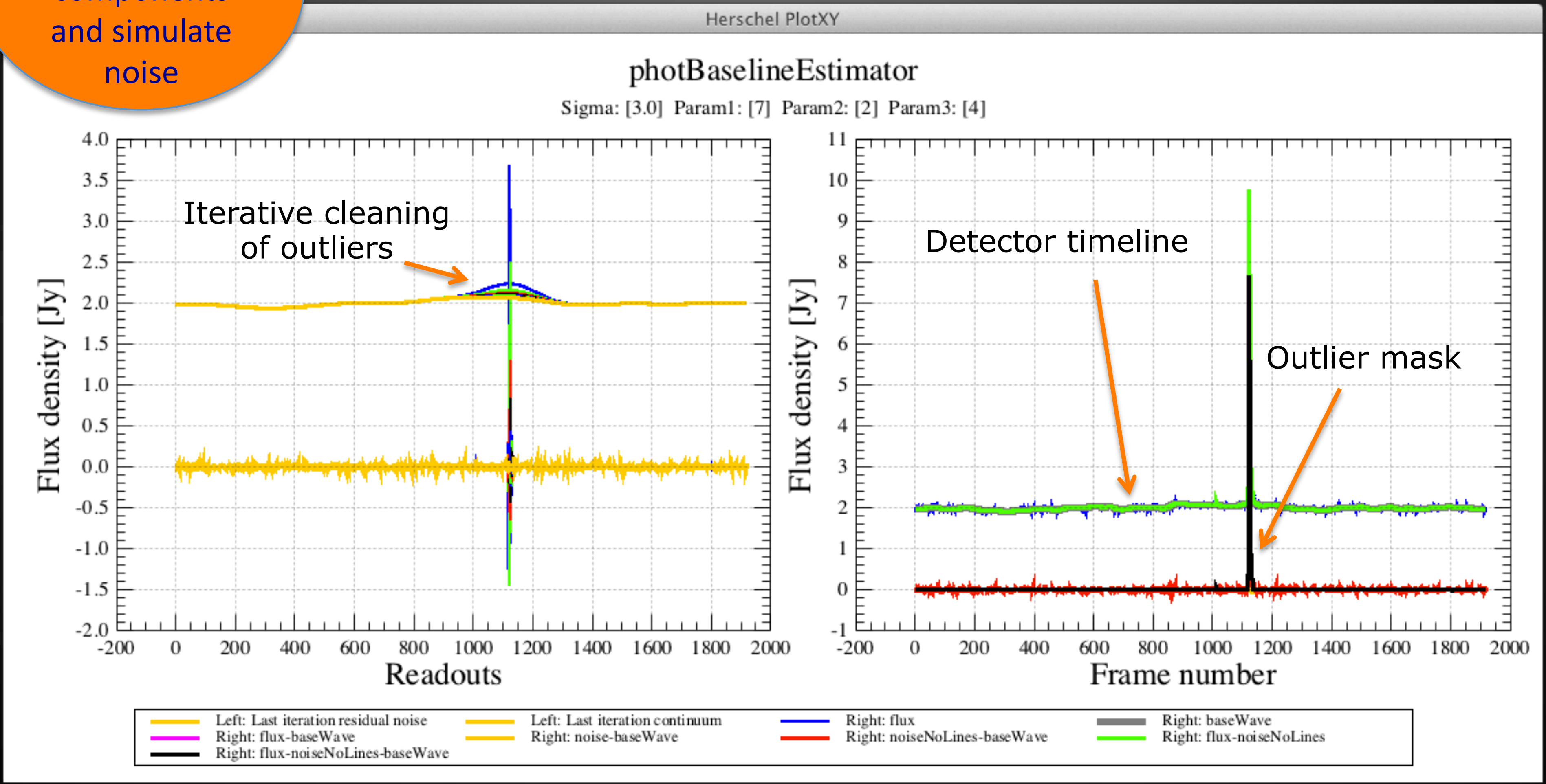
Iterate until outliers disappear

Interpolated intensity in masked timeline →

$$I_{(t)} = \underbrace{N_{(t)}^{1/f} + N_{(t)}^{D} + N_{(t)}^{\det}}_{\text{Simulated noise}} + \underbrace{I_{(t)}^{S(lowfreq)}}_{\text{Baseline estimate from data}}$$

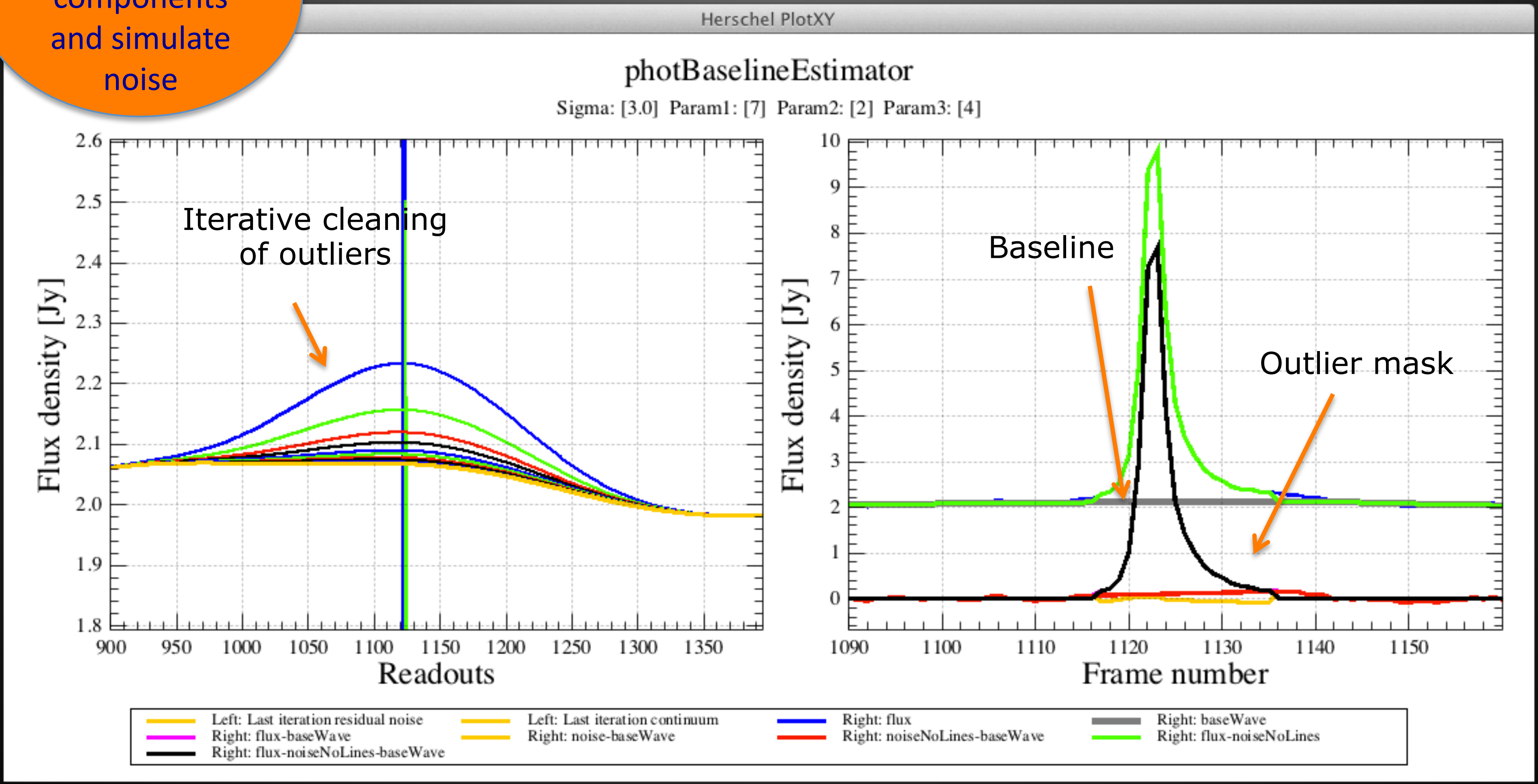
Baseline estimation

Decompose frequency components and simulate noise



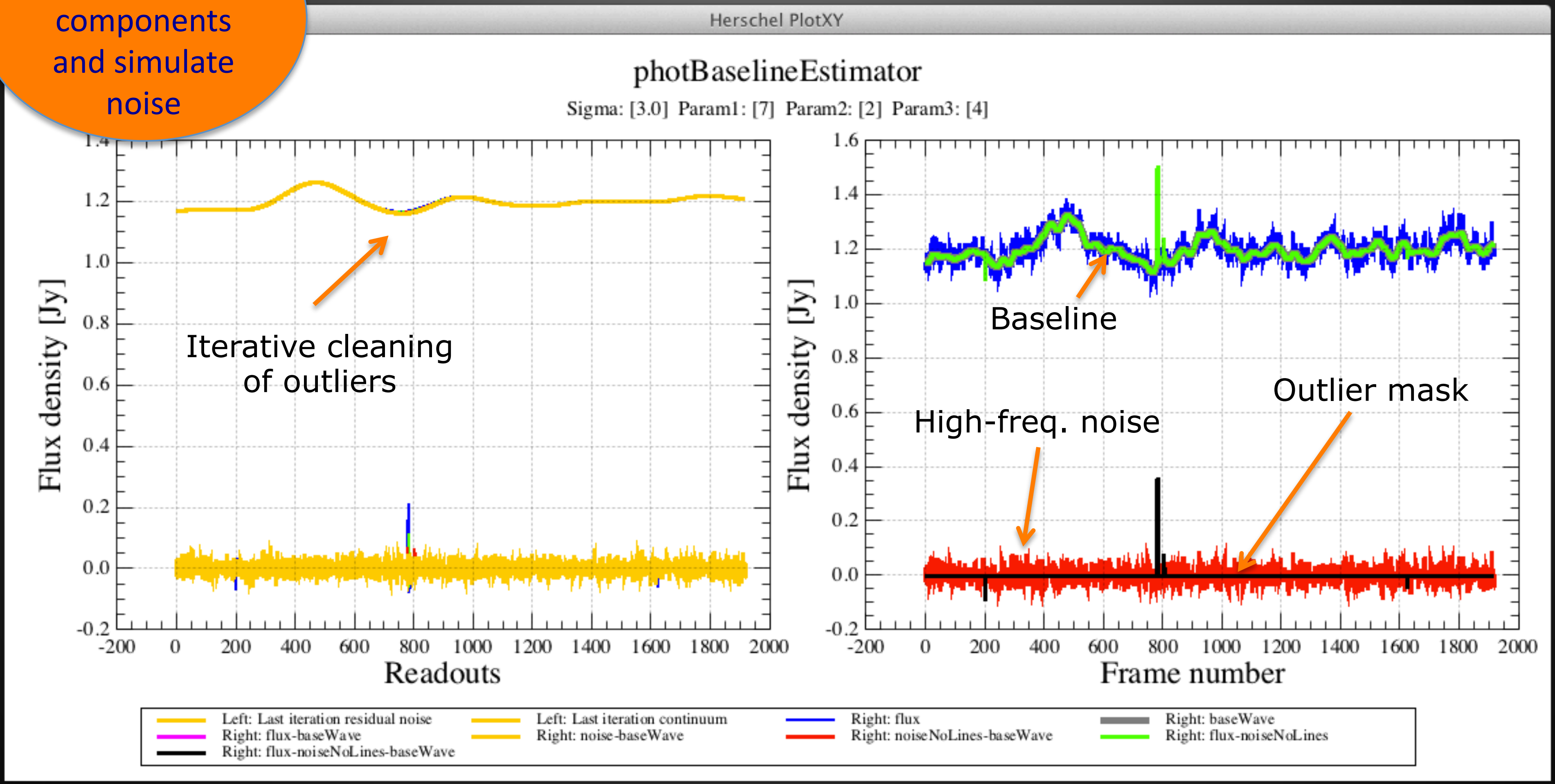
Baseline estimation

Decompose frequency components and simulate noise



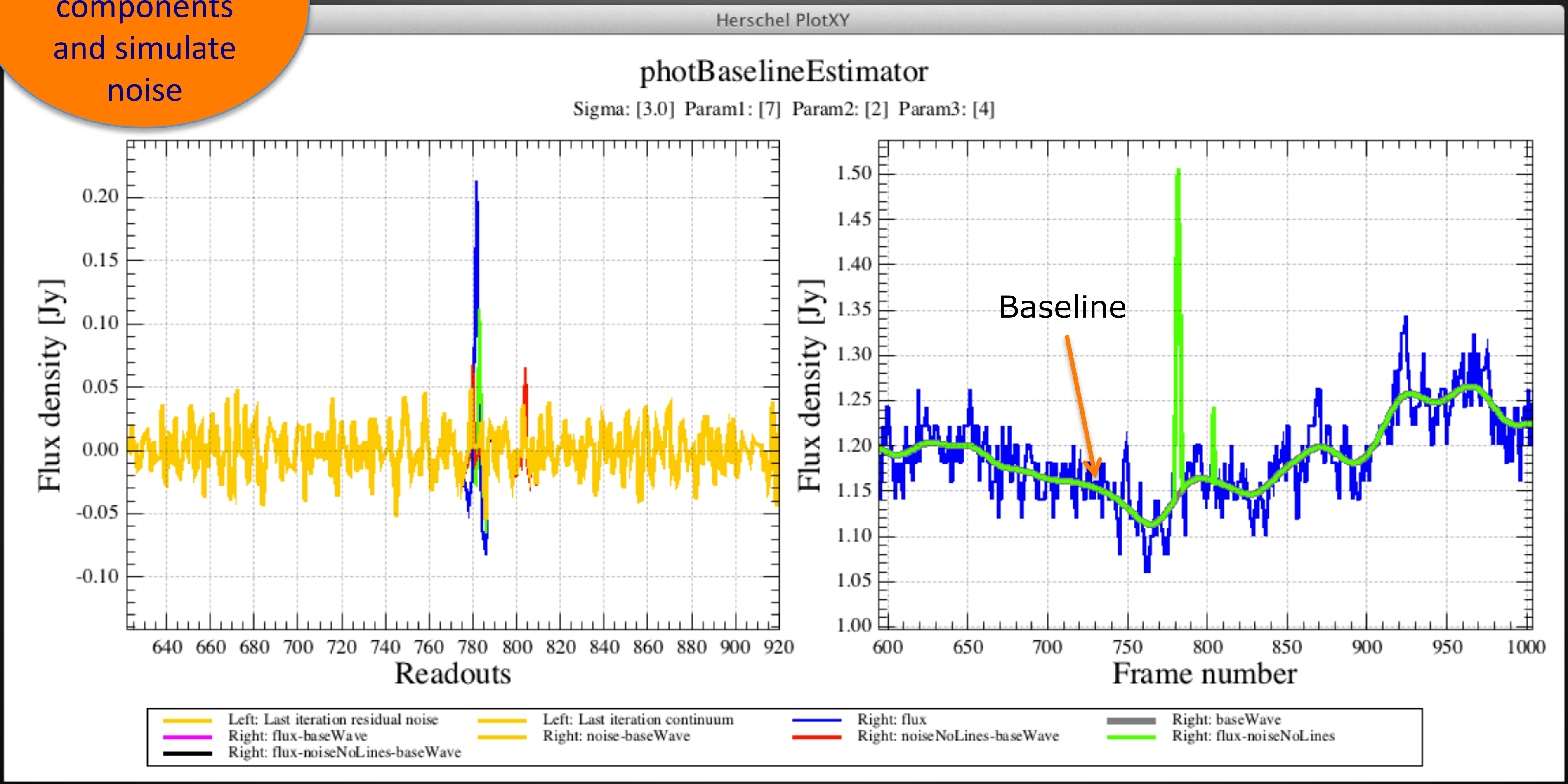
Baseline estimation

Decompose frequency components and simulate noise



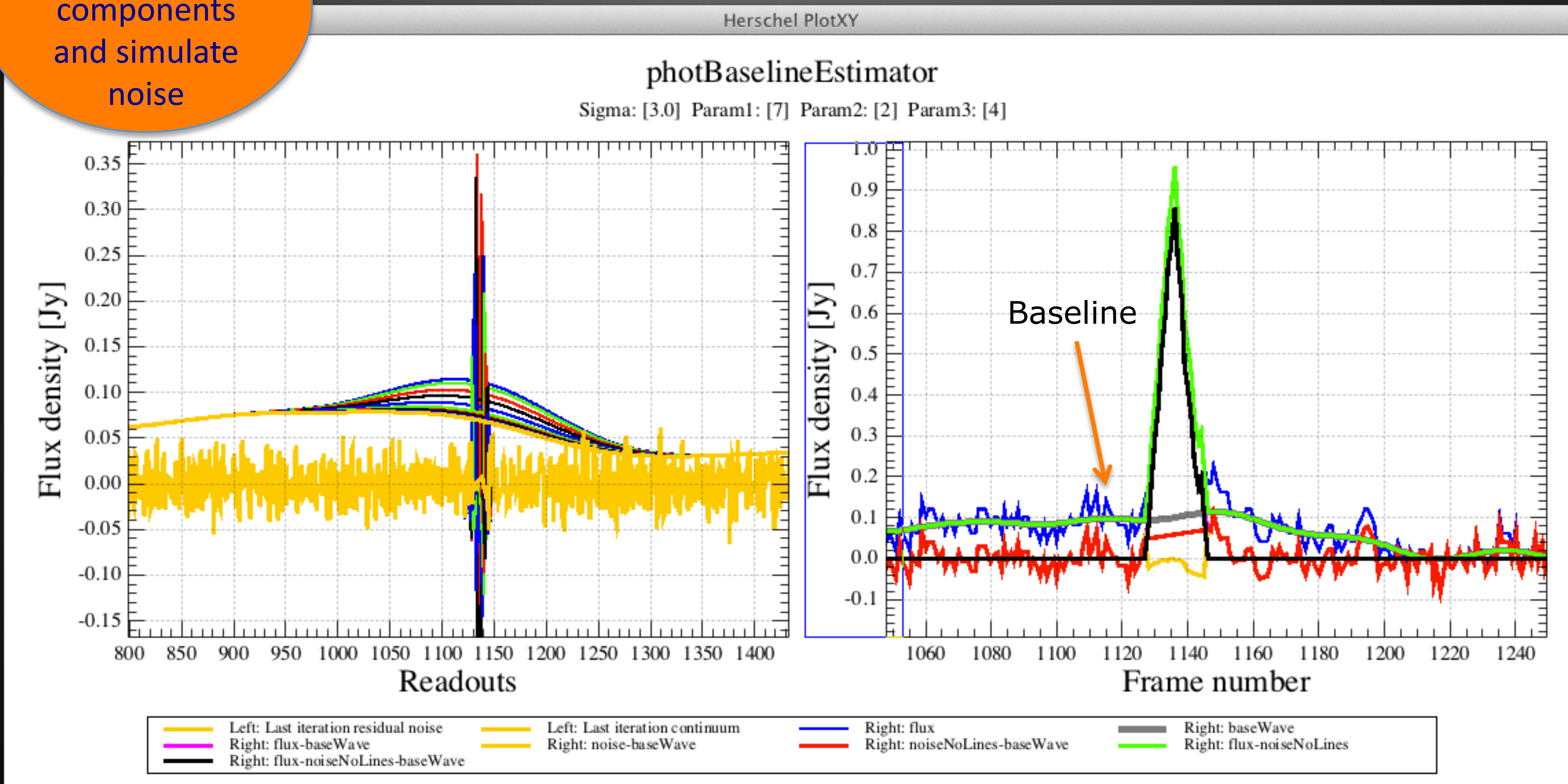
Baseline estimation

Decompose frequency components and simulate noise



Baseline estimation

Decompose frequency components and simulate noise



Noise simulation

Decompose frequency components and simulate noise

- The objective is to simulate intensity distribution of a single scan-leg with similar noise power spectrum as we experience in the observed data.
- In small sections of the timeline the noise is considered being stationary
- Non-parametric Monte-Carlo simulation in the wavelet space:
  - Measure STDDEV of outlier filtered wavelet coeffs after last iteration
  - Get random noise and scale each wavelet frequency coefficients to the measured STDDEV
  - The inverse wavelet-transform of coefficients at frequencies equal and higher than the mask size gives the noise simulation

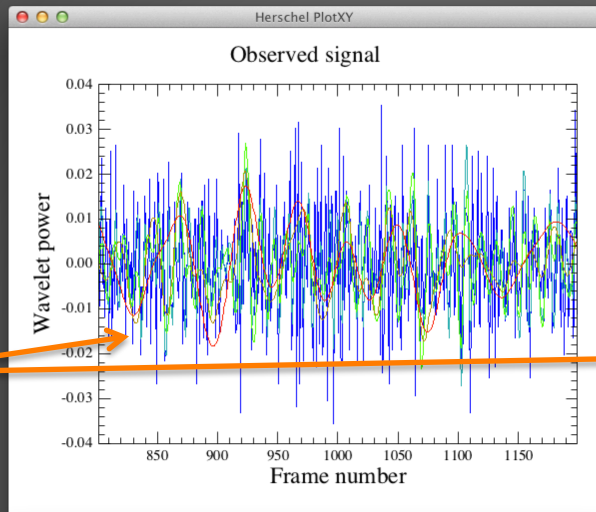
Interpolated intensity in masked timeline →

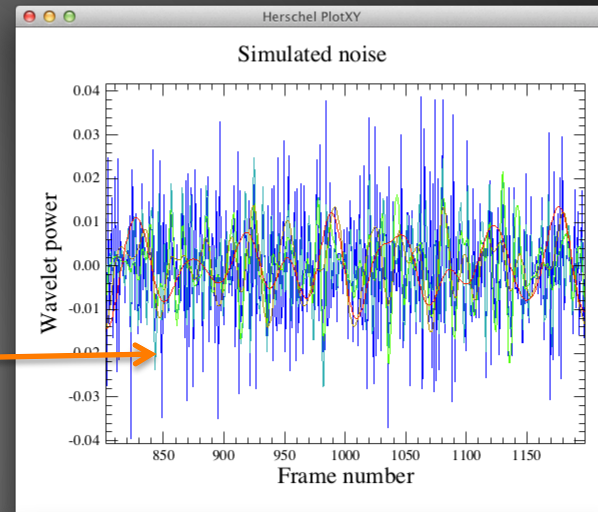
$$I_{(t)} = \underbrace{N_{(t)}^{1/f} + N_{(t)}^{D} + N_{(t)}^{\det}}_{\text{Simulated noise}} + \underbrace{I_{(t)}^{S(lowfreq)}}_{\text{Baseline estimate from data}}$$

Noise simulation

Decompose frequency components and simulate noise

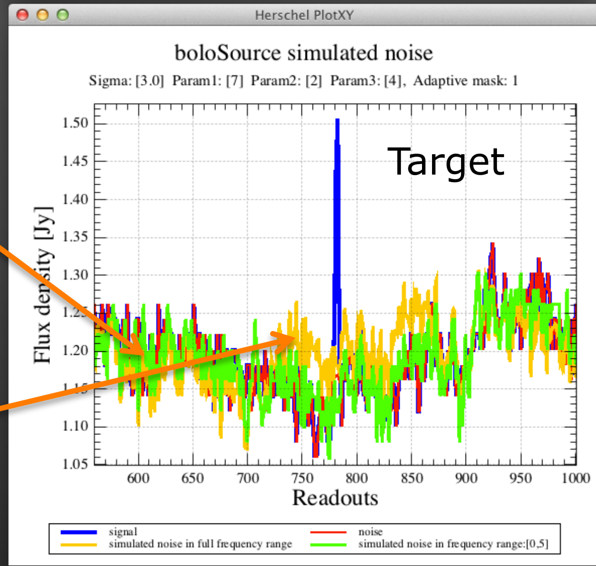
Wavelet coeffs. Identical STDDEV for observed and simulated cubes

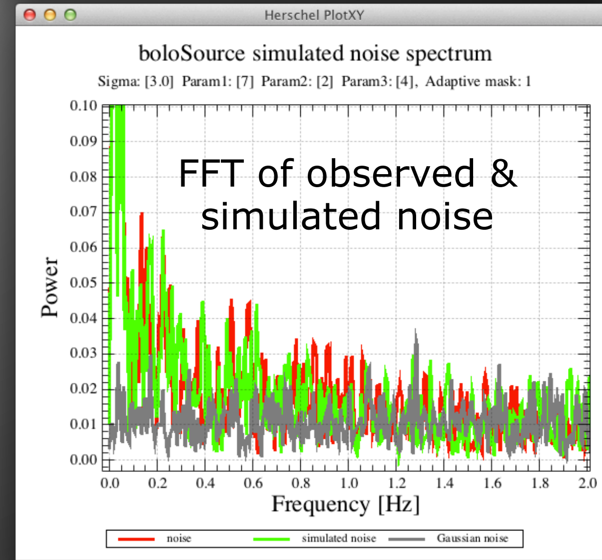




Green: simulated noise adjusted to the source cutoff-frequency

Yellow: simulated noise including all frequency components





Timeline interpolation

Optimize L1 mask location and size

Interpolate with simulated noise

- Optimize L1 mask size and location through the adaptive mask option:
  - Extend the L1 mask to 2x larger working mask and find a maximum:
    - defined by the baseline estimator outlier mask (this is the only criteria if the "autodetect" option is active)
    - or if outlier mask is empty within the working mask then try to find local peak of at least 3 readouts broad (for faint features)
  - Find the 1st and last readouts below the baseline around the peak
  - This section defines the final adaptive L1 mask
- Take the previously identified baseline and add simulated noise components beyond the cutoff frequency within the section of L1 adaptive mask
- Do bit-rounding (2-bit < OD 1005 and 1-bit > OD 1005)

Timeline interpolation

Optimize L1 mask location and size

Interpolate with simulated noise

L0 bolometer timeline example (red band, pixel [8,8])

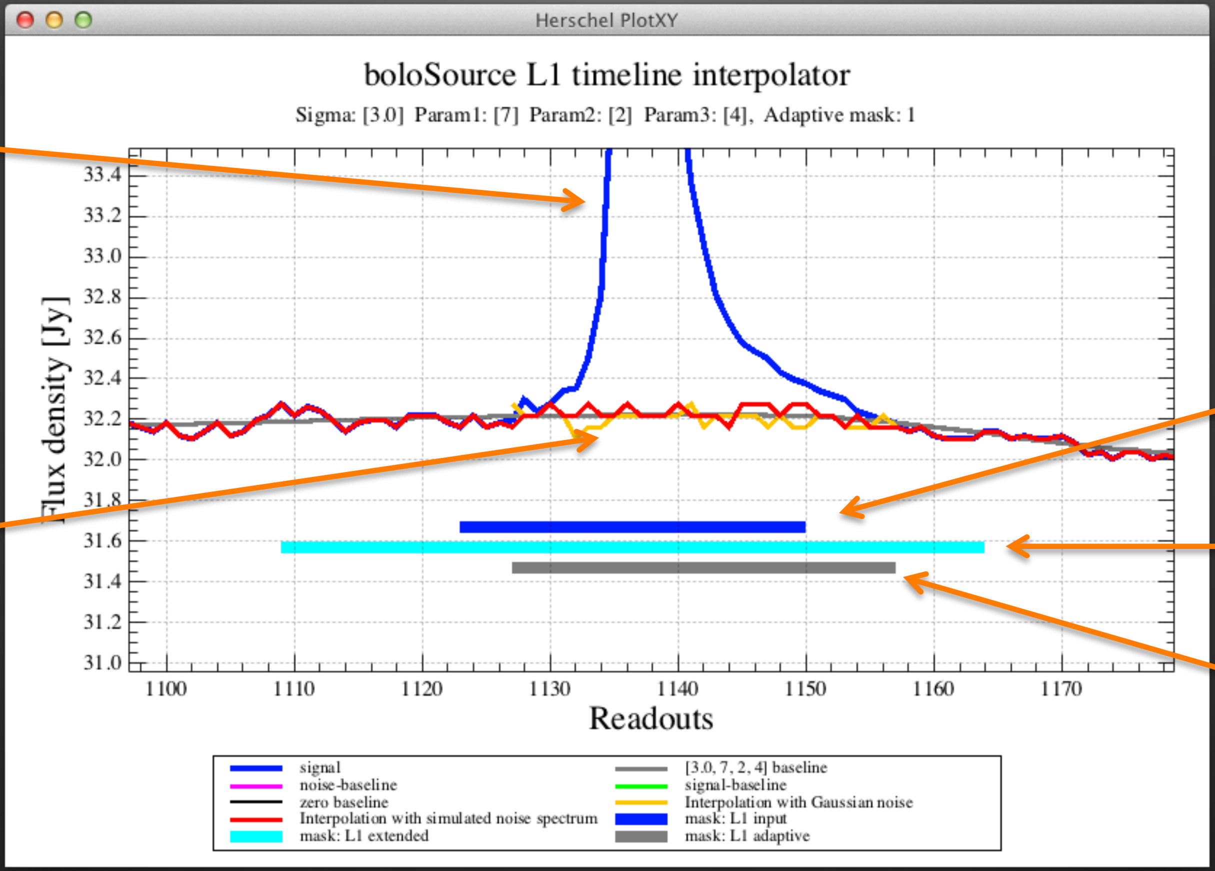
L1 bolometer signal in the timeline

Interpolated signal

Back-projected input mask

Intermediate working mask

Optimized adaptive mask



Timeline interpolation

Optimize L1 mask location and size

Interpolate with simulated noise

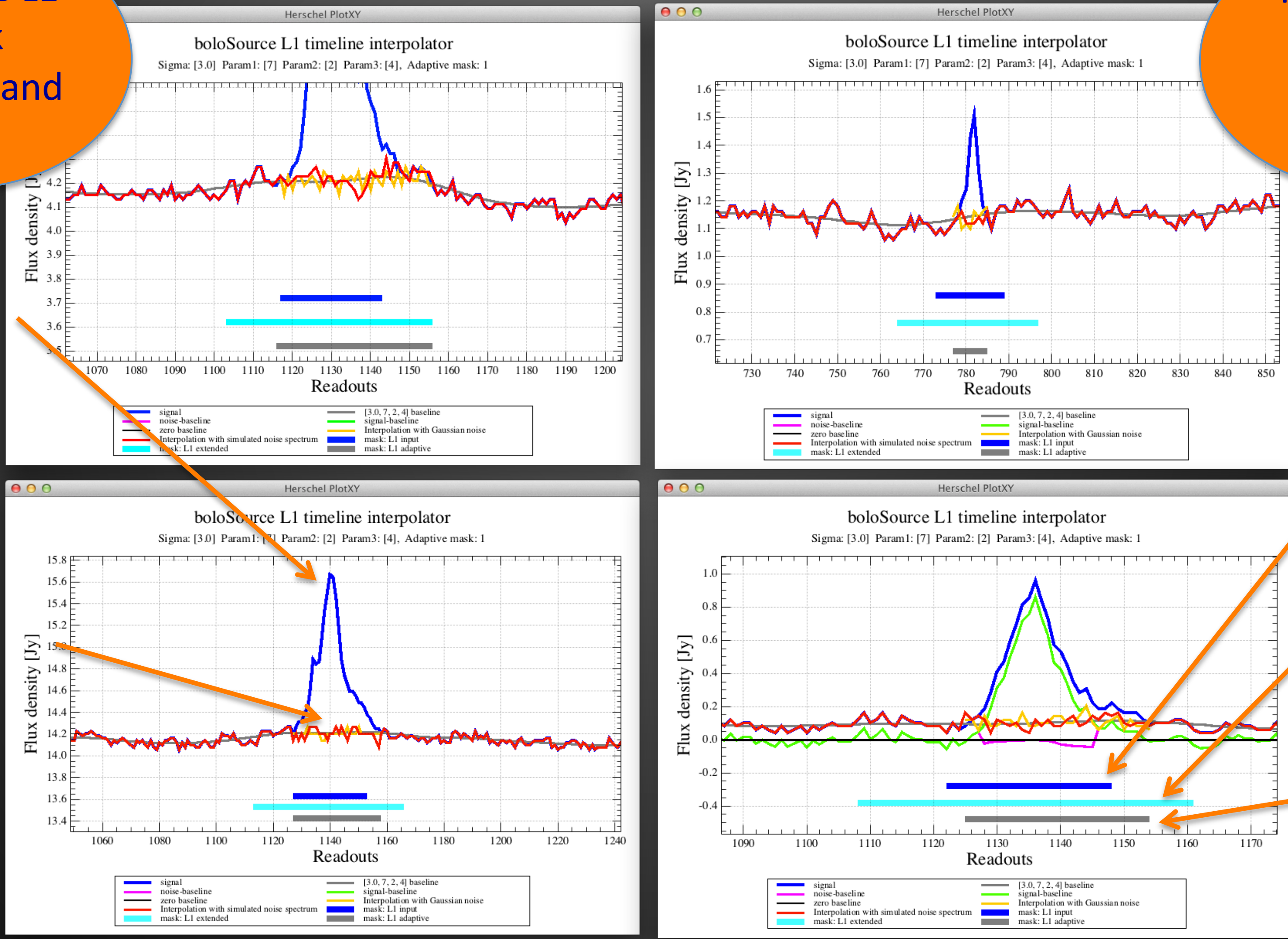
L1 bolometer signal in the timeline

Interpolated signal

Back-projected input mask

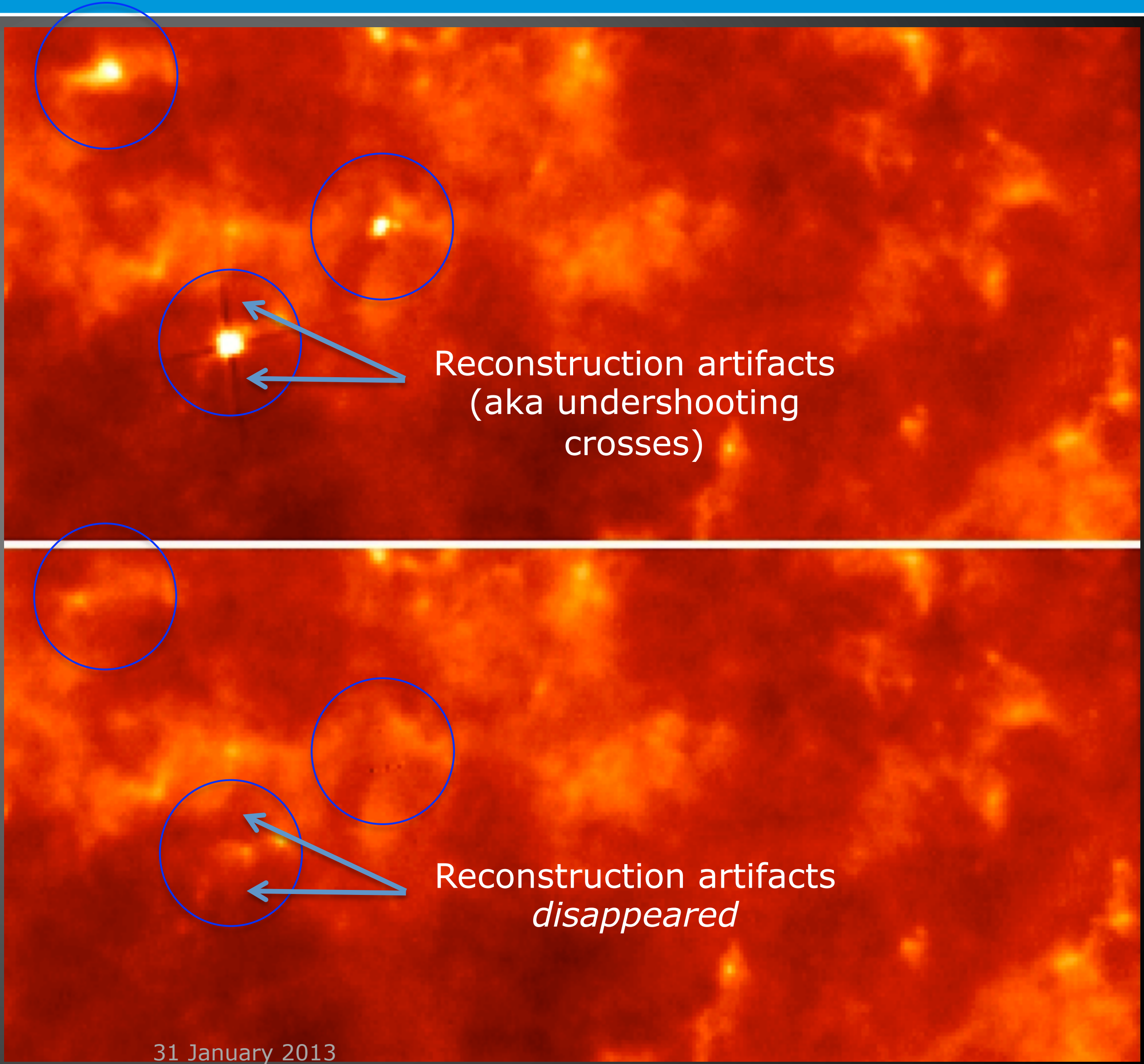
Intermediate working mask

Optimized adaptive mask



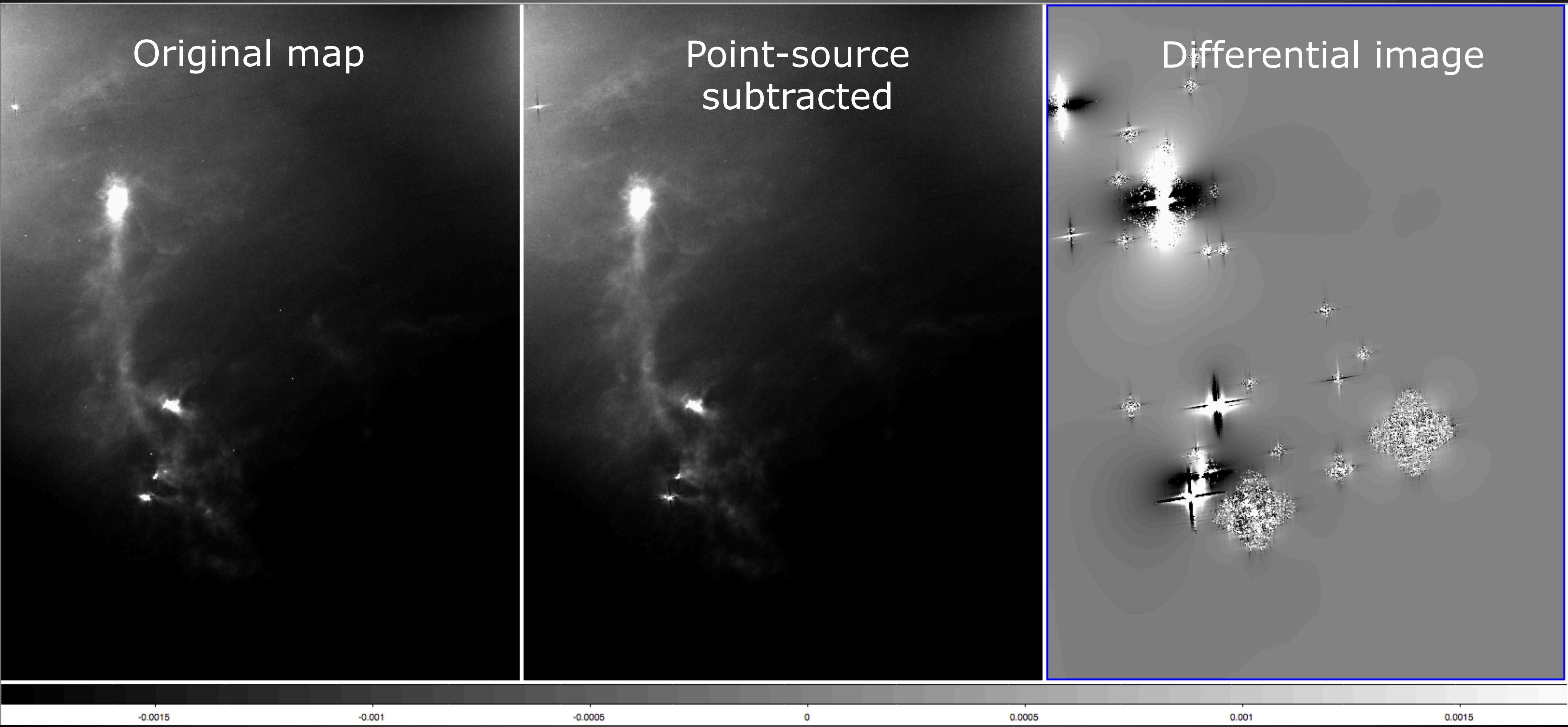
First run of boloSource()

- First run of boloSource (Hi-GAL l=297°)
- Background is quite well preserved
- By product: MadMap reconstruction noise (undershooting artifacts) could be eliminated
- For the analysis of extended emission there is no strict need for other cleaning techniques (L. Piazzo et al. in prep.)



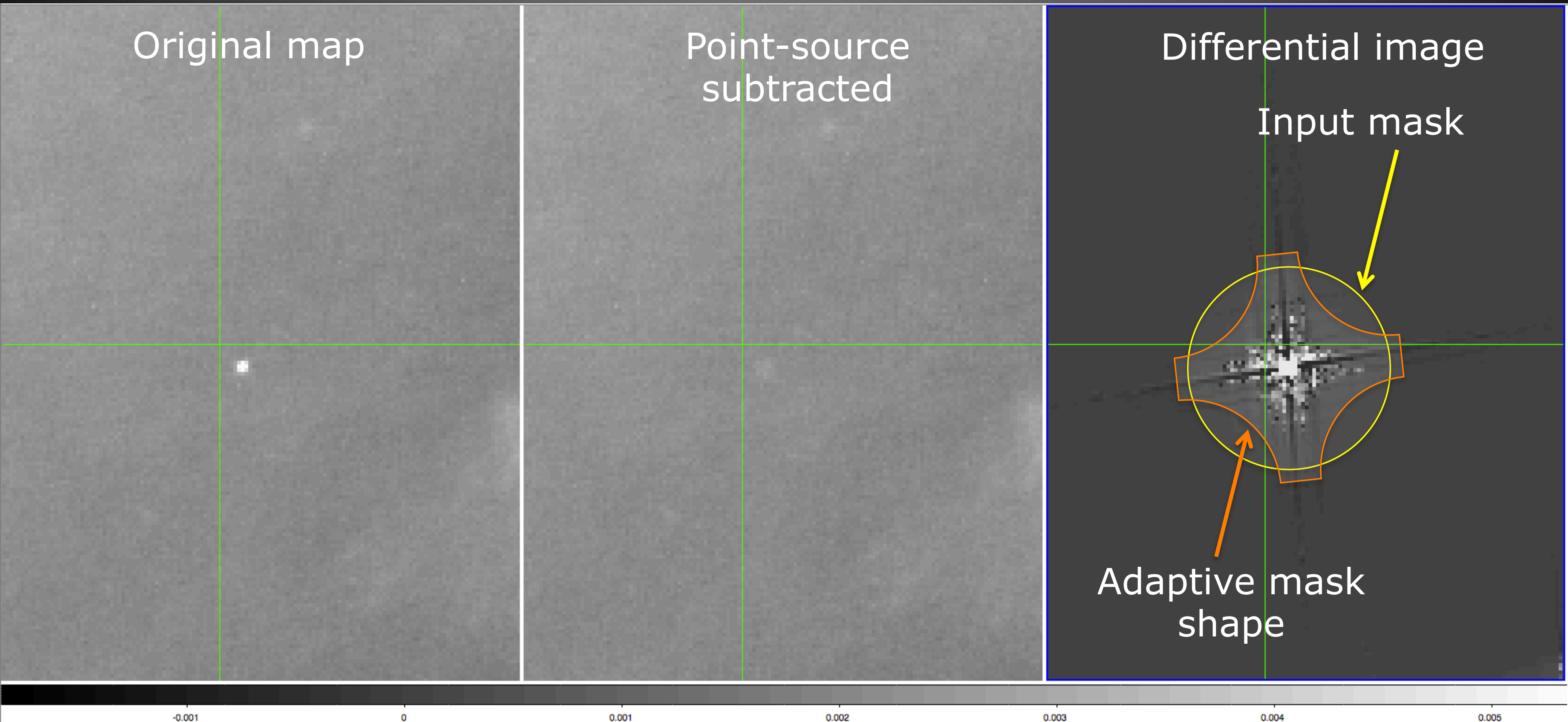
Chamaeleon-I

MadMap reduction



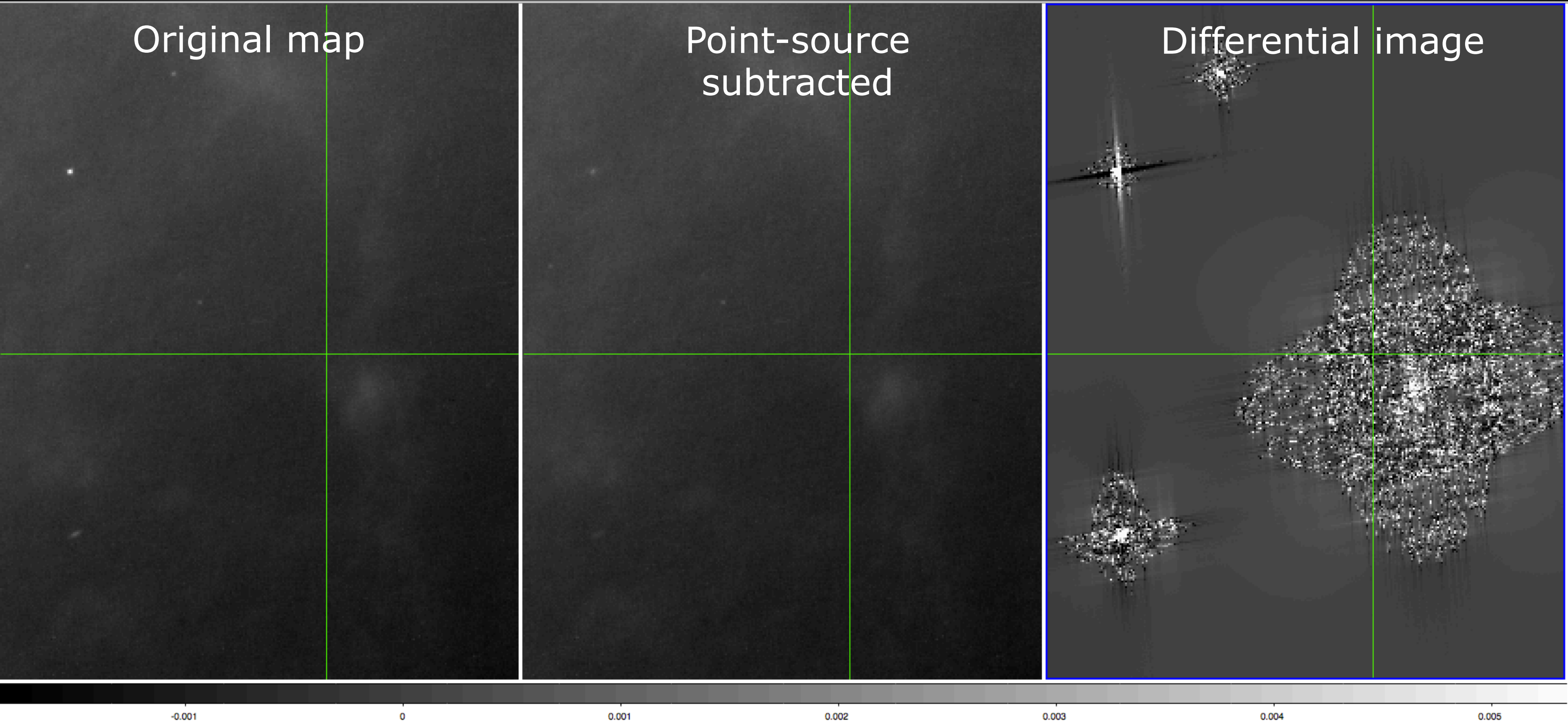
Chamaeleon-I

MadMap reduction



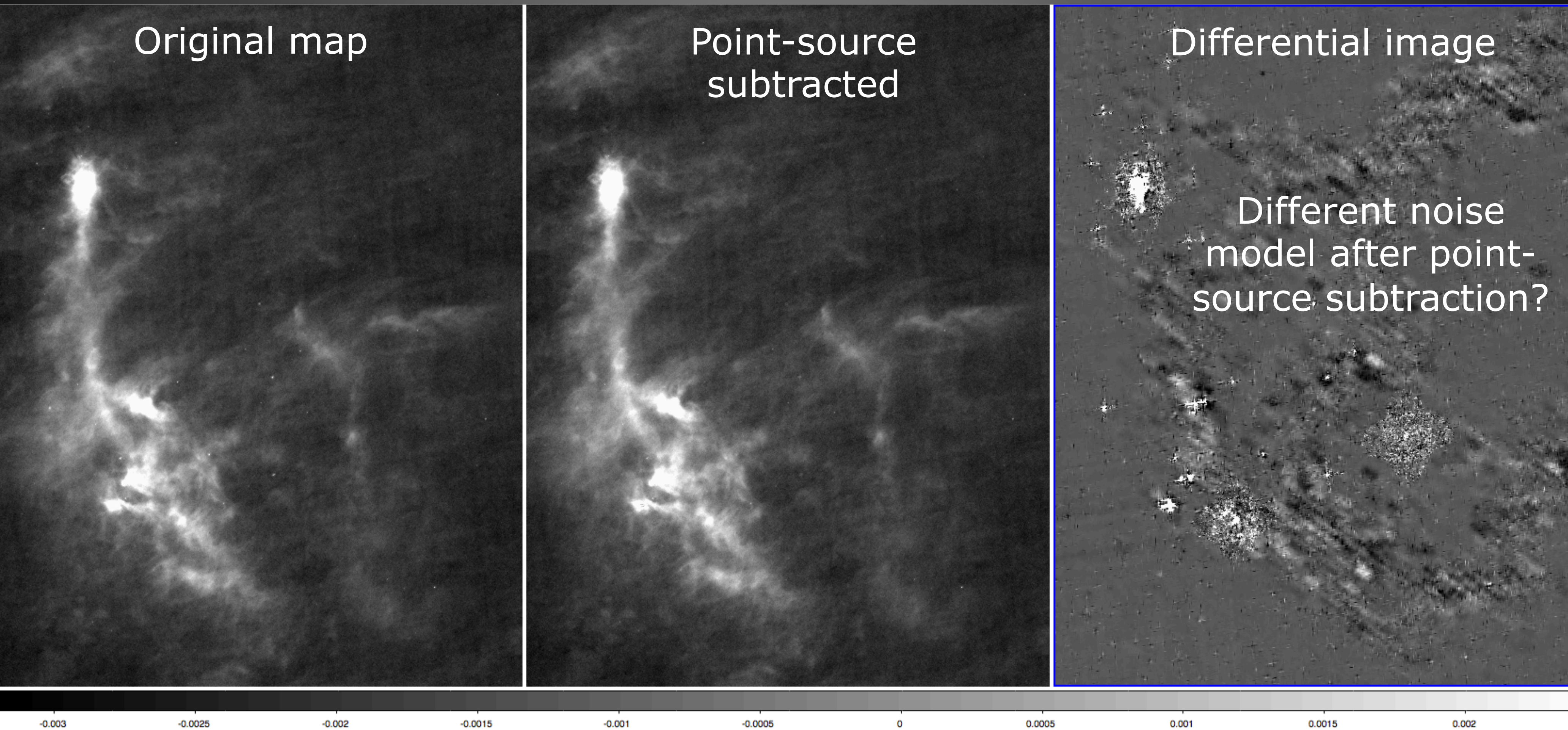
Chamaeleon-I

MadMap reduction



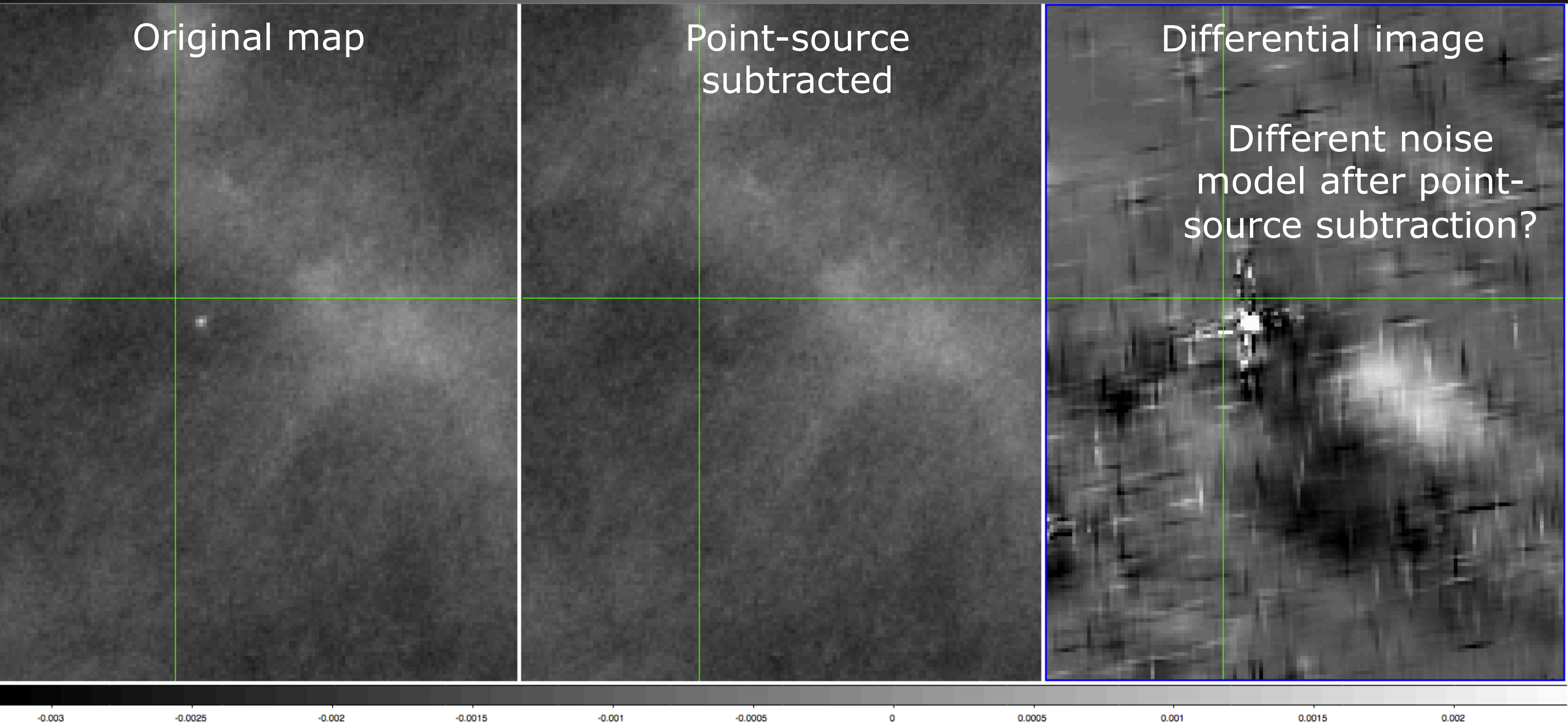
Chamaeleon-I

Scanamorphos reduction

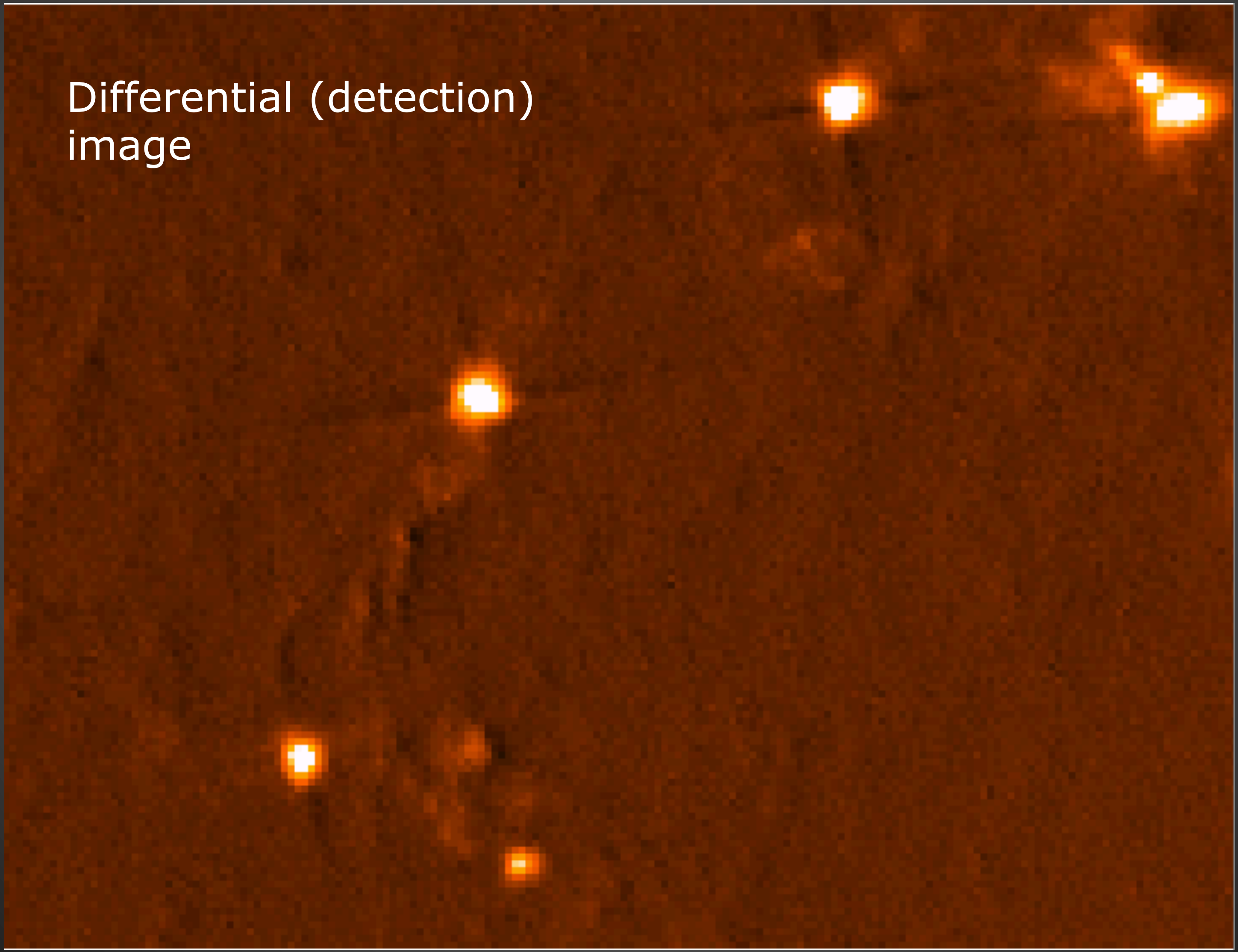


Chamaeleon-I

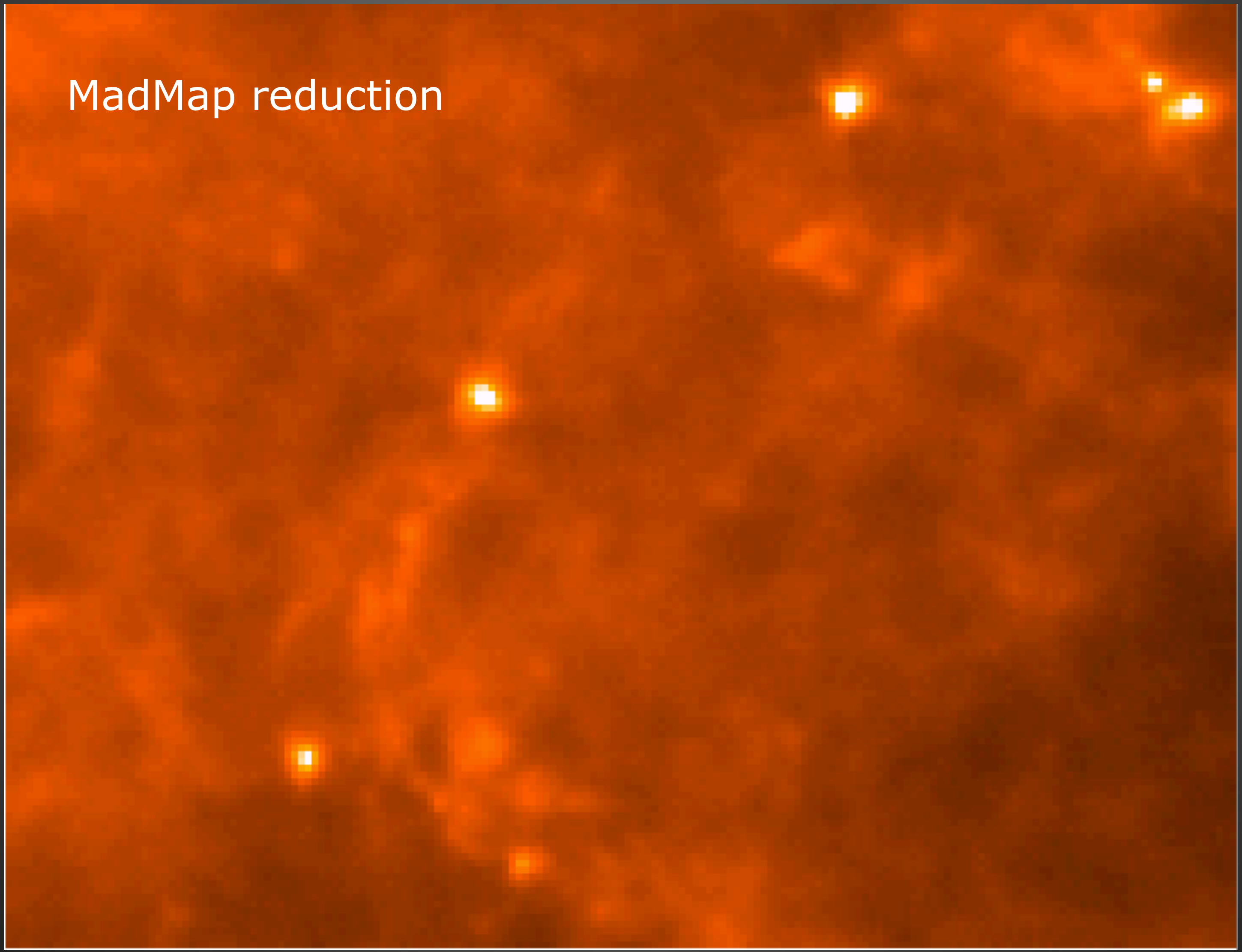
Scanamorphos reduction



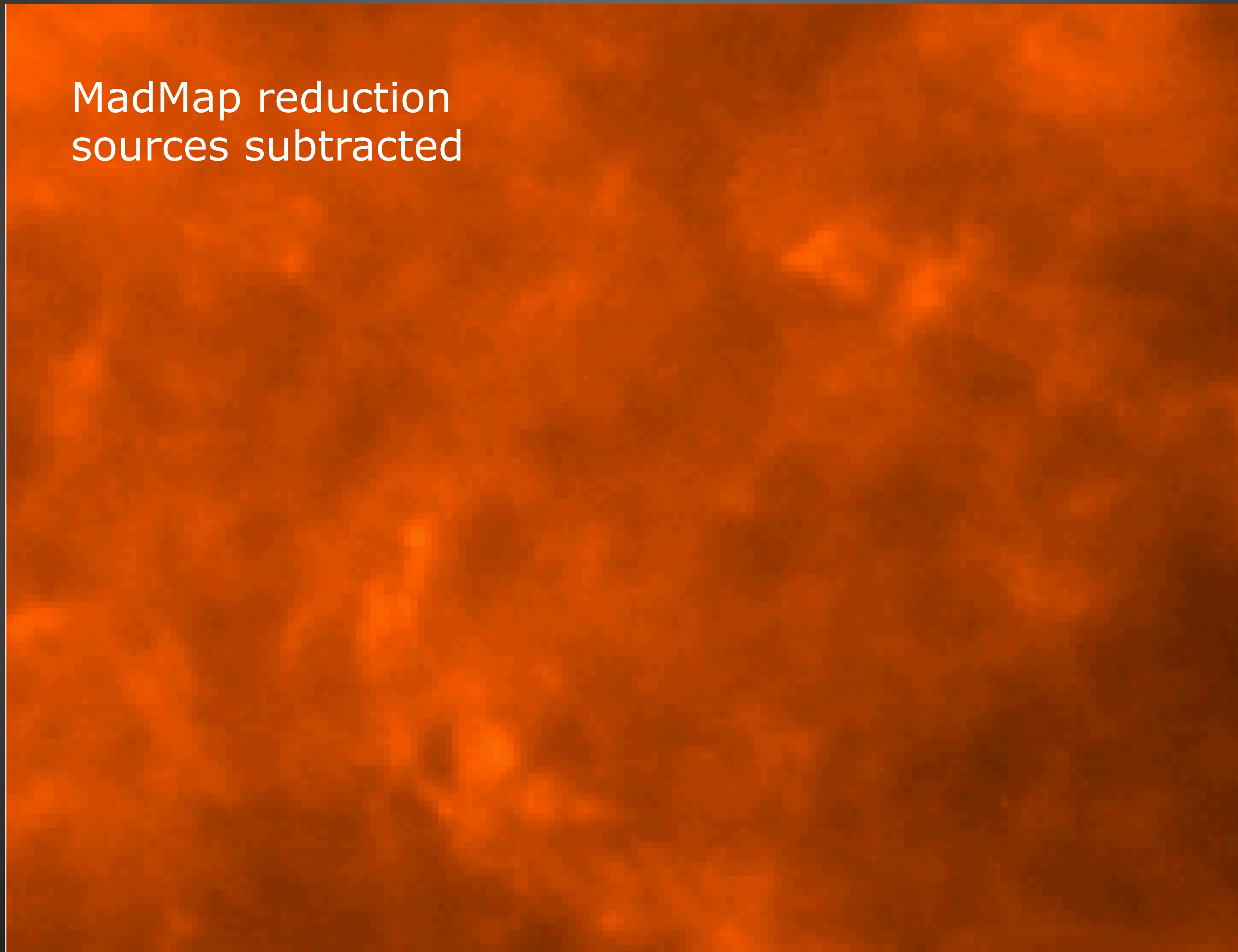
Hi-GAL l=297°



Hi-GAL l=297°



Hi-GAL l=297°

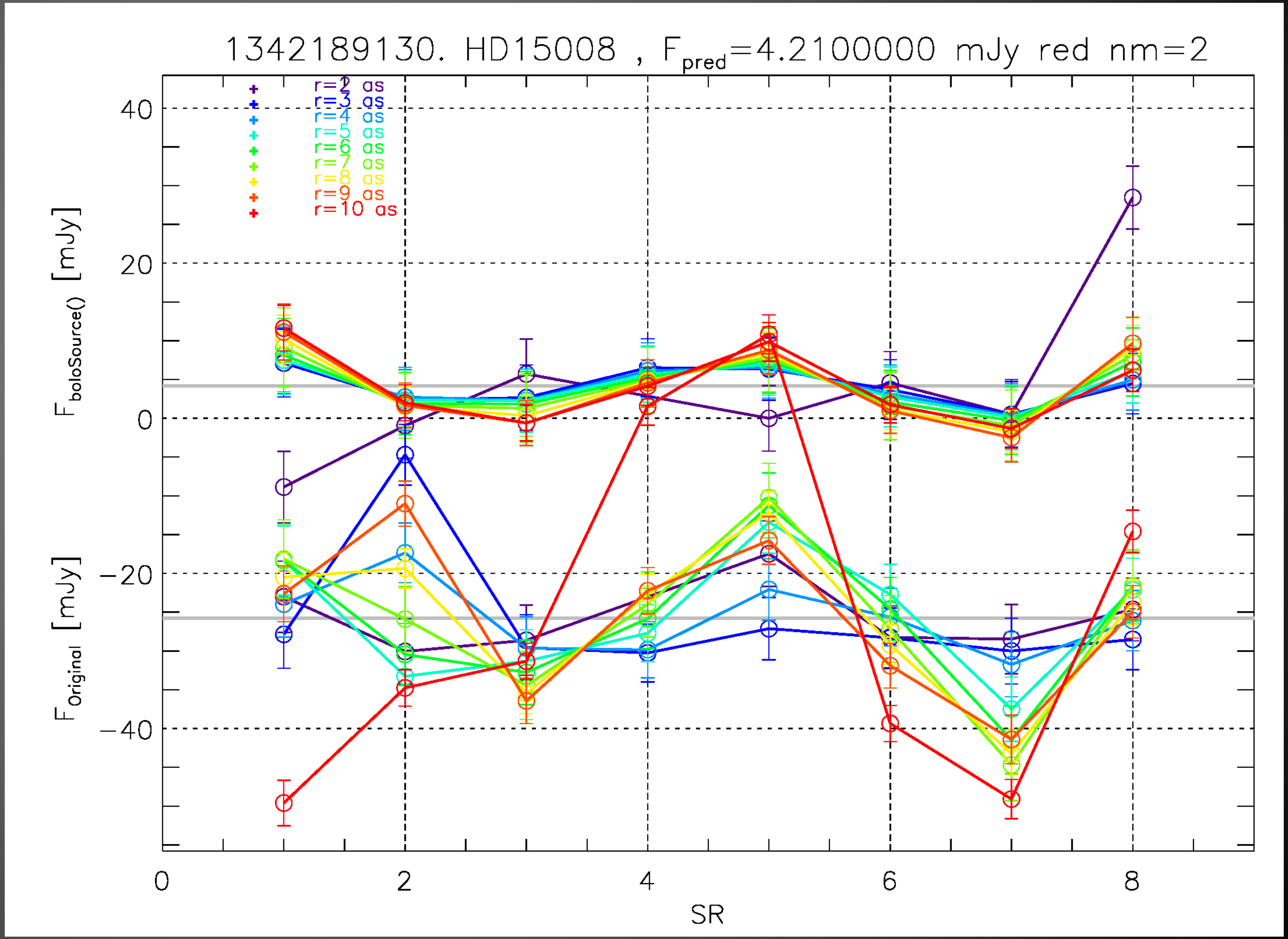


Point-source photometry

Early results on a 160 µm faint source
(G. Marton)

boloSource() uncertainty compared with aperture photometry on HPF +photProject map

1 observation with 9 repetition, photometry is done on maps combined of 2 repetitions pairs



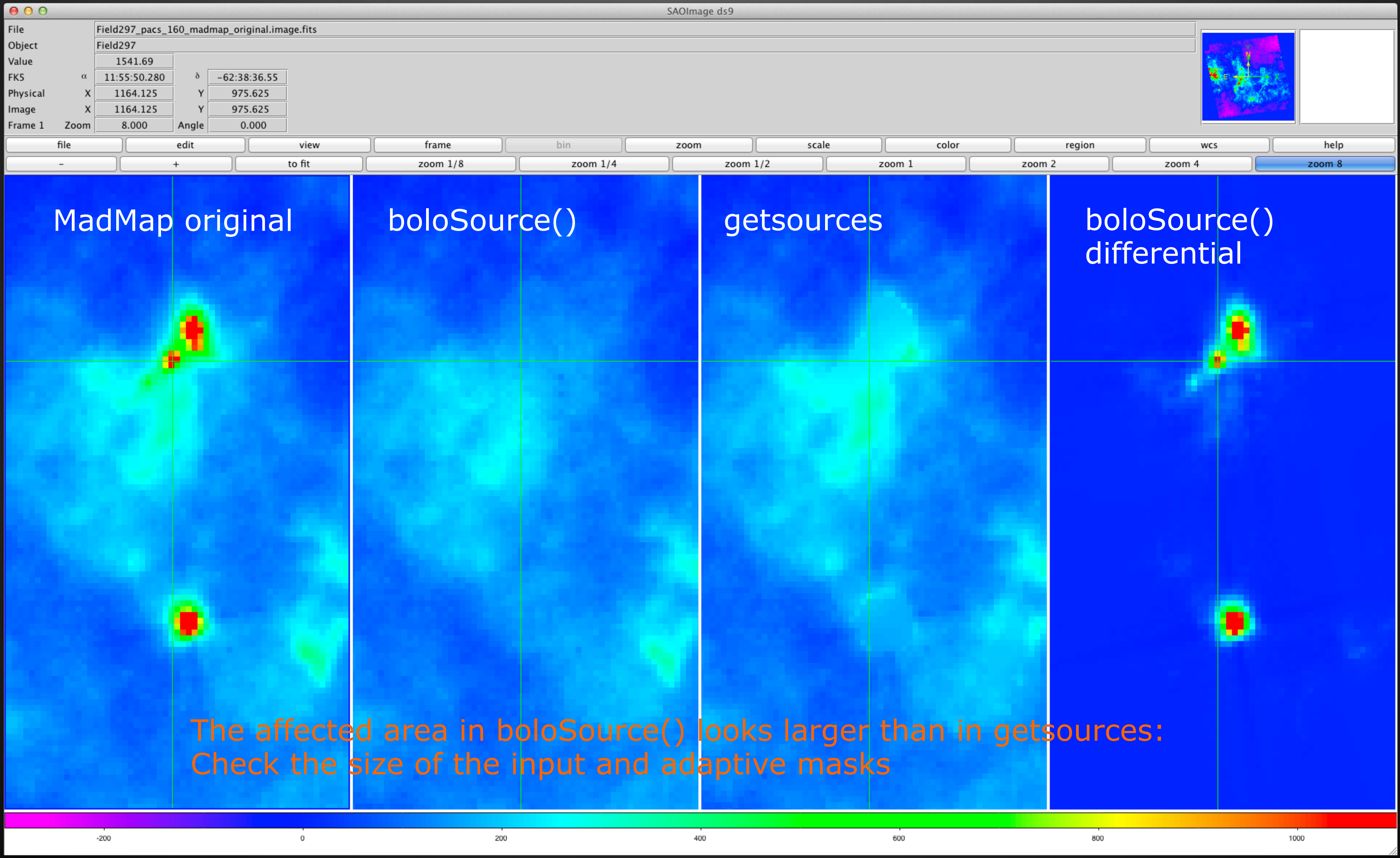
Techs

- boloSource() input:
  - source list (centroid coordinates)
  - Input L2 mask **OR** source cutoff frequency, i.e. aperture radius *(no range possible in the current version)*
  - PACS OBSID pair
- boloSource() output:
  - PACS L1 cubes with interpolated timeline
- Compatible with any projection algorithm what can use PACS L1 frames cube structure
- Full code in Jython, will be available in build HIPE v10.0+

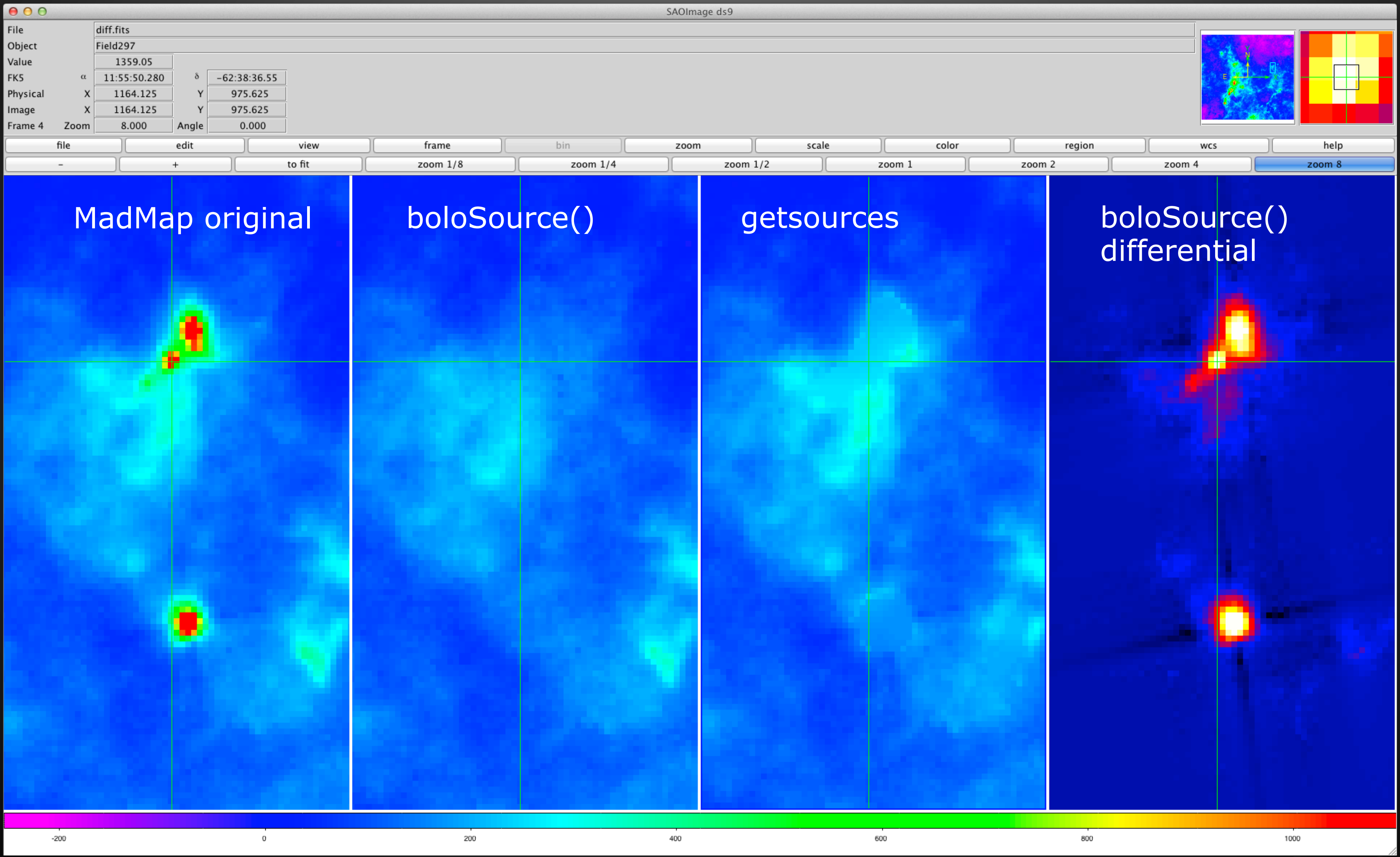
Perspectives

- Build bechmarking environment by using simulated sources back-projected onto the L1 detector timeline
- Photometry on differential detection image. The benchmarking environment should cover the [surface brightness, PWS slope] space
- Try in combination with map-makers and check the reconstruction artifact removal potential (and get source detection map in one go!)
- Mask detection layer can be added. Thresholding on unsharp-masked image looks very promising (H. Bouy)
- Try it for SPIRE, simple ROI back-projection of L2->L1 mask is possible

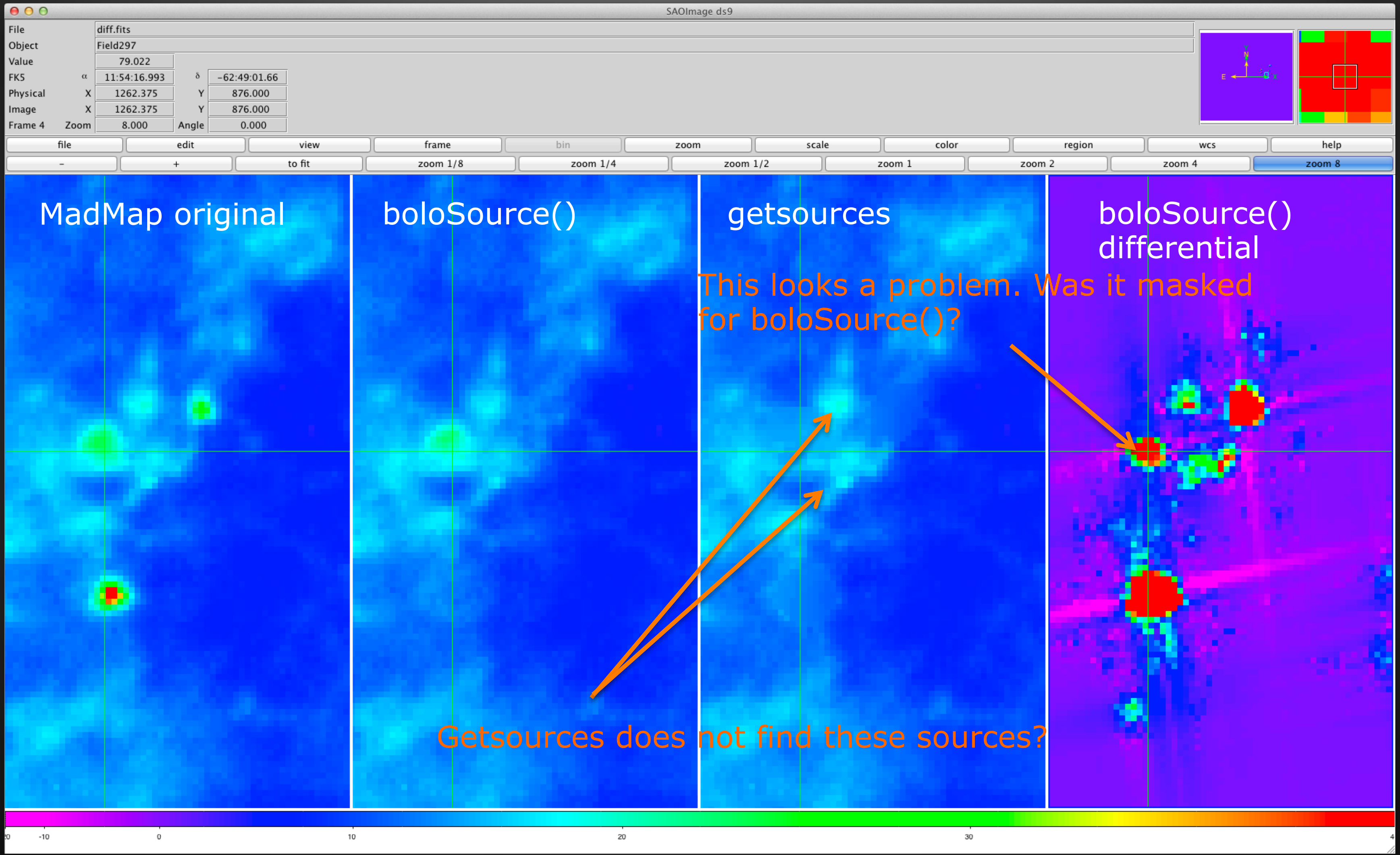
Some examples



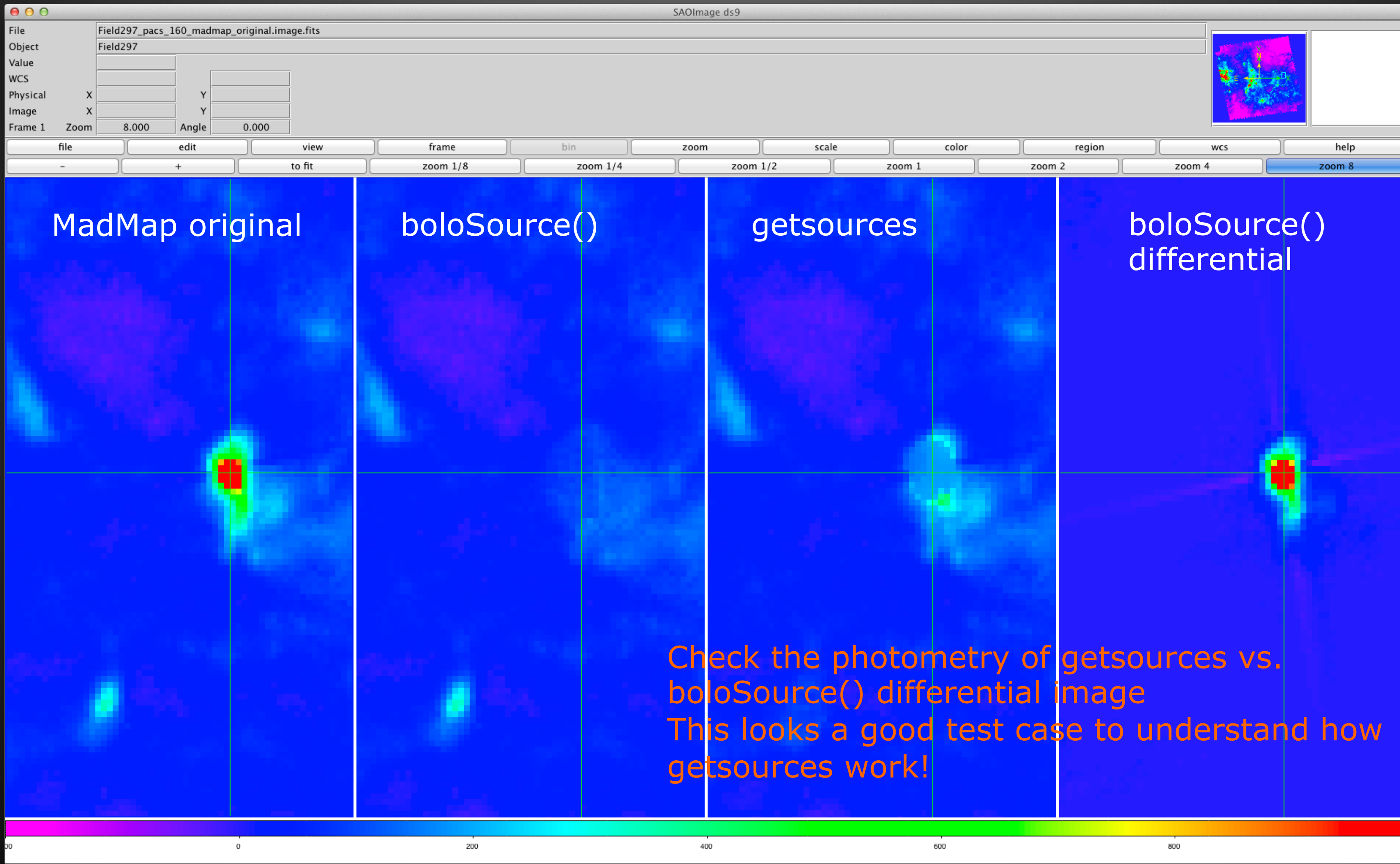
Some examples



Some examples

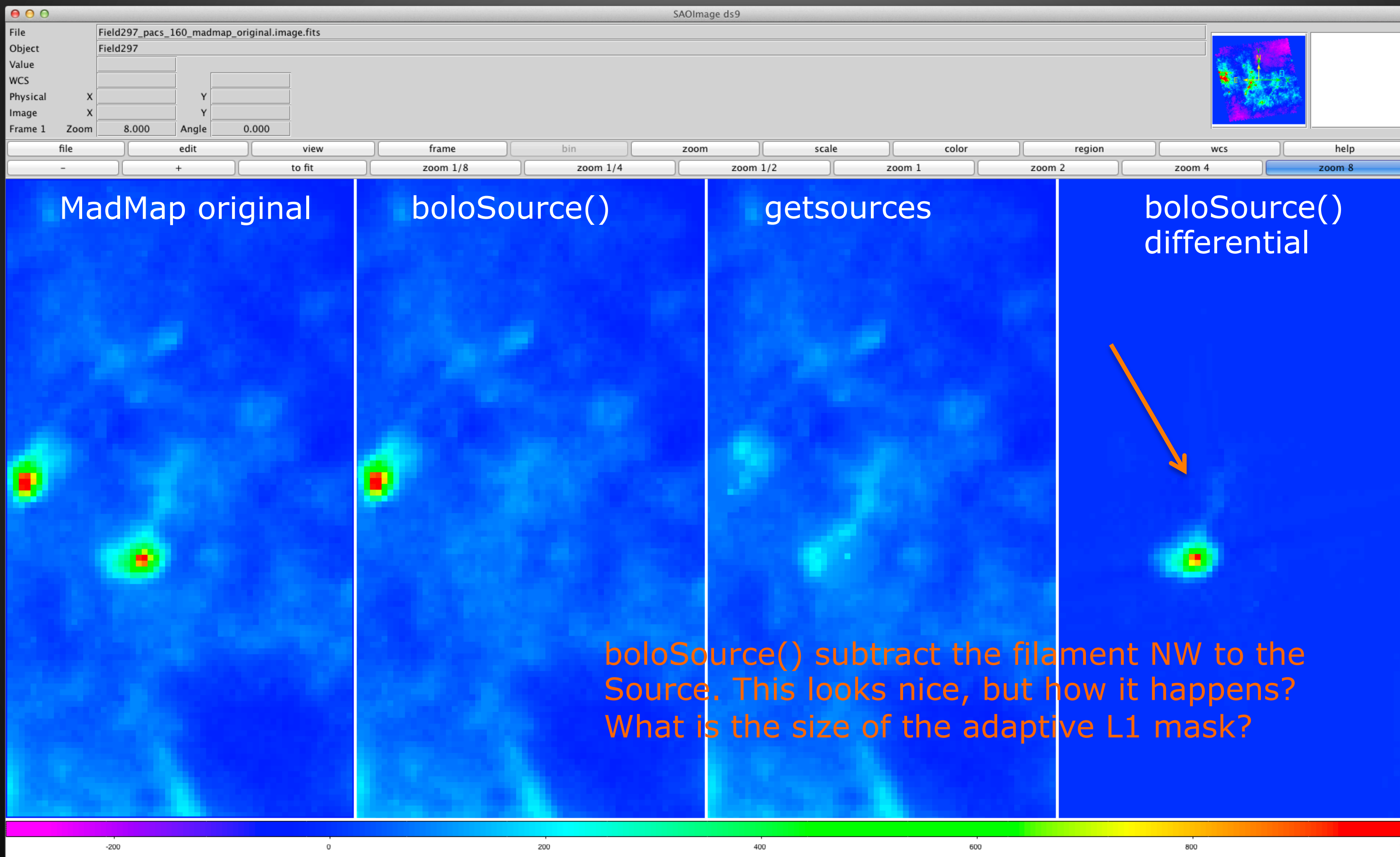


Some examples



Check the photometry of getsources vs. boloSource() differential image
This looks a good test case to understand how getsources work!

Some examples



boloSource() subtract the filament NW to the Source. This looks nice, but how it happens? What is the size of the adaptive L1 mask?