SKA Our Galaxy Meeting
11-12 July 2018
Catania

Development of analysis software for SKA surveys

Simone Riggi - INAF-OACT

Outline

- **Why should we invest time in analysis software?**
  - Most analysis (.e.g cataloguing) still requiring significant manual intervention
    - Not feasible already at the scale of precursors (>10^6 sources expected in EMU)
    - Analysis reproducibility
  - Some areas not fully covered in SDP Science Pipeline (i.e. left to either SWGs or RCs)
  - Regional Center activities (at least in Europe) more focused on design (i.e. no development)
  - Analysis tools currently in use have limitations
    - Not scaling well vs image size (even at the scale of precursors) or source density?
    - Not well tested or poorly performing on Galactic fields
    - Delivering poor output for post-processing analysis (e.g. classification studies, etc)
    - Need adaptation or re-designing for Regional Centers
  - Because all medium- and large-scale experiments did (and continue to do so...)

- **What kind of software (not only for analysis) do we need?**
  - Extraction of compact and extended sources
  - Improved imaging on the GP (single dish + SKA combination) (see Ingallinera's talk)
  - Source correlation across frequency channels, observations or surveys
  - Object classification
  - Visualization and exploration tools (see Vitello's talk)
  - Many others...

- **Software tools for source extraction and characterization**

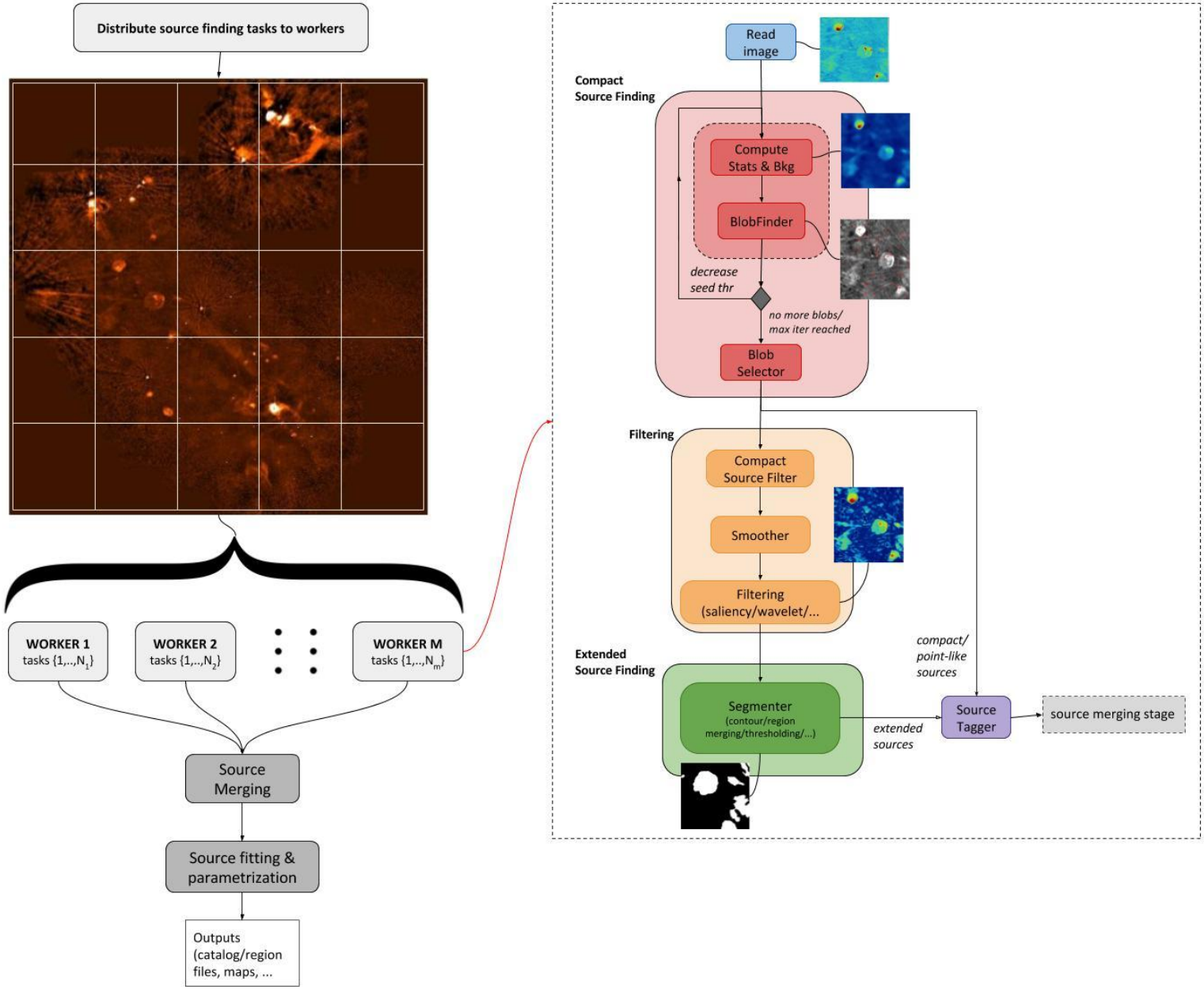
Status of radio source finders

- **Several finders publicly available for compact source extraction**
  - Sample tools: Selavy (ASKAP), PySE/PyBDSF (LOFAR), Aegean, Blobcat, CuTEx (...)
  - Mostly based on flood-fill algorithms
  - Tested with simulations with extragalactic-like fields (ASKAP Data Challenge)
    - ➢ Comparable performances (reliability, completeness, position & flux, ...)
  - Current limitations and areas of development
    - ➢ Reliability, deblending, fit robustness to be improved
    - ➢ Few able to run in parallel on different processors (e.g. PySE) or nodes (e.g. Selavy)
    - ➢ Few providing an API
    - ➢ No detailed testing in presence of significant diffuse/extended emission

- **Lack of tools for both compact and extended sources**
  - Different algorithms proposed (e.g. active contours, region segmentation, Hough Transform, Wavelet Transform, ...) but detailed testing required

- **CAESAR Source Finder**
  - CAESAR: *Compact And Extended Source Automated Recognition*
  - Developed for SCORPIO and ASKAP EMU surveys
  - Ref: *S. Riggi et al, MNRAS 460, 1486-1499 (2016), https://github.com/SKA-INAF/caesar.git*
  - Implemented in C++, based on ROOT/OpenCV (+ other libraries)
  - Recent development: logging, parallel implementation, algorithm improvements, ...
  - Tested with SCORPIO data, detailed testing with simulations ongoing

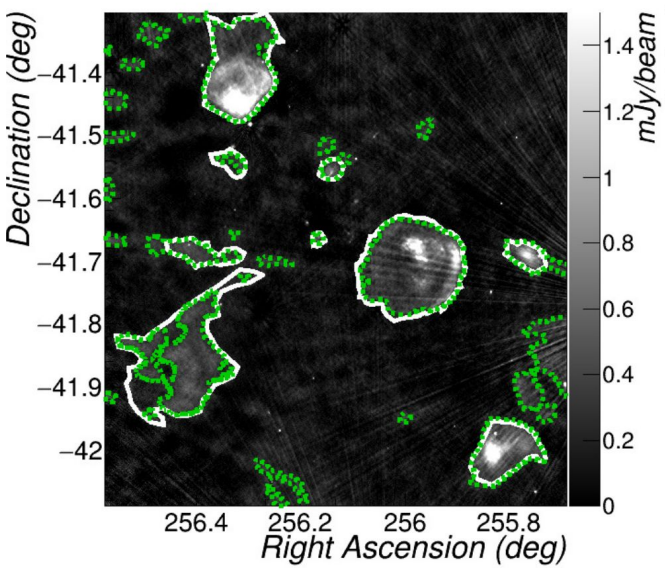
CAESAR processing pipeline



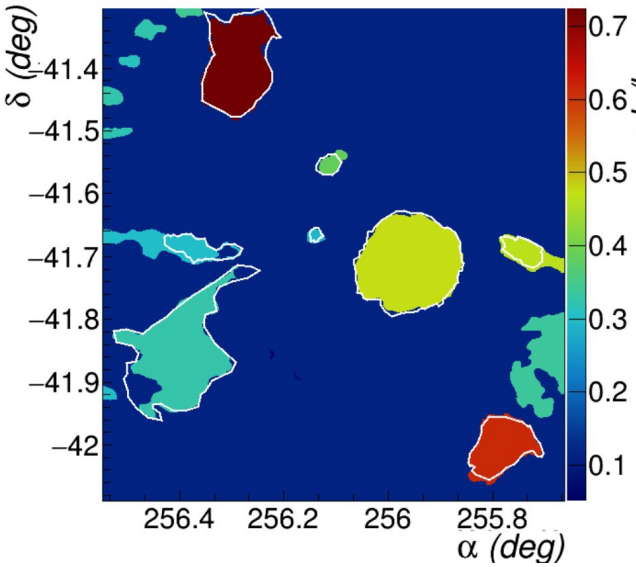
Testing with SCORPIO fields

Results on sample SCORPIO field (1600x1850 pixels): centred on SNR G344.7-0.1, MSC 345.1-0.2 SNR Candidate

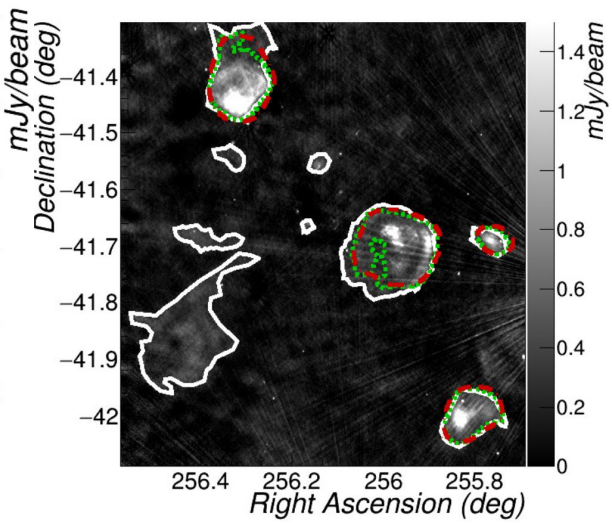
White lines: manual segmentation



Chan-Vese contour finder



Hierarchical clustering



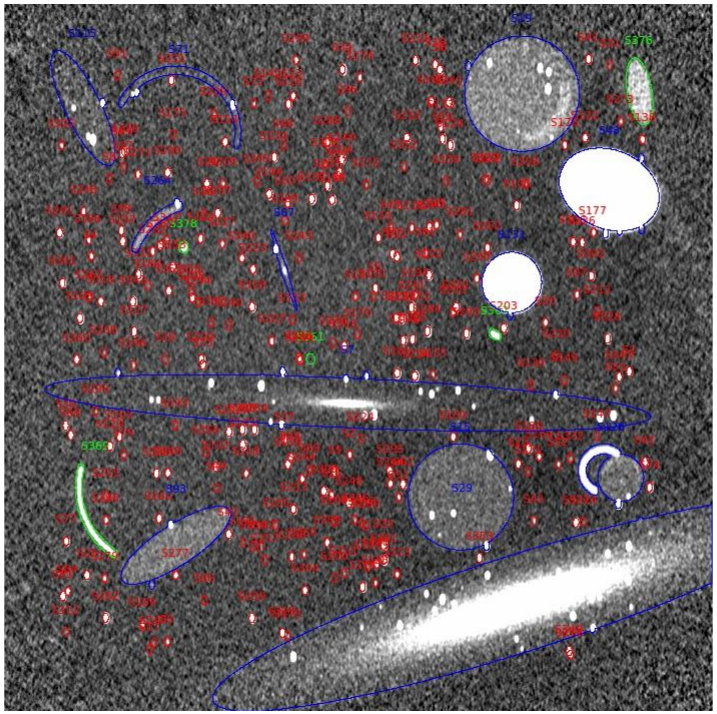
Wavelet Transform

NB: Algorithms for compact sources completely misses extended object in most cases

Testing with simulations

NB: Ideal simulations

- **Sky model generation**
  - Map size: 2560x2560 pixels, pix size= 1 arcsec
  - Point-sources
    - density=1000/deg^2, uniform spatial distr.
    - S=[100 μJy,1 Jy] exp distr.
  - Extended sources
    - density=50 deg^2, uniform spatial distr.
    - Flux ampl=[1 μJy,100 μJy] exp distr.
    - 4 different types (disk+shell, ring, gaus, sersic profile), max scale 10'
  - Data size <~ 100 MB/map
- **Simulation** (done with CASA)
  - 12 h ATCA all baselines configuration
  - Raw data ~ 2GB/map
- **Imaging** (done with CASA)
  - Fields cleaned up to 10 mJy using true source masks and combined with linearmosaic
  - rms ~ 300-400 μJy
  - Raw data ~ 1.5 GB/map, final map ~ 30 MB/map

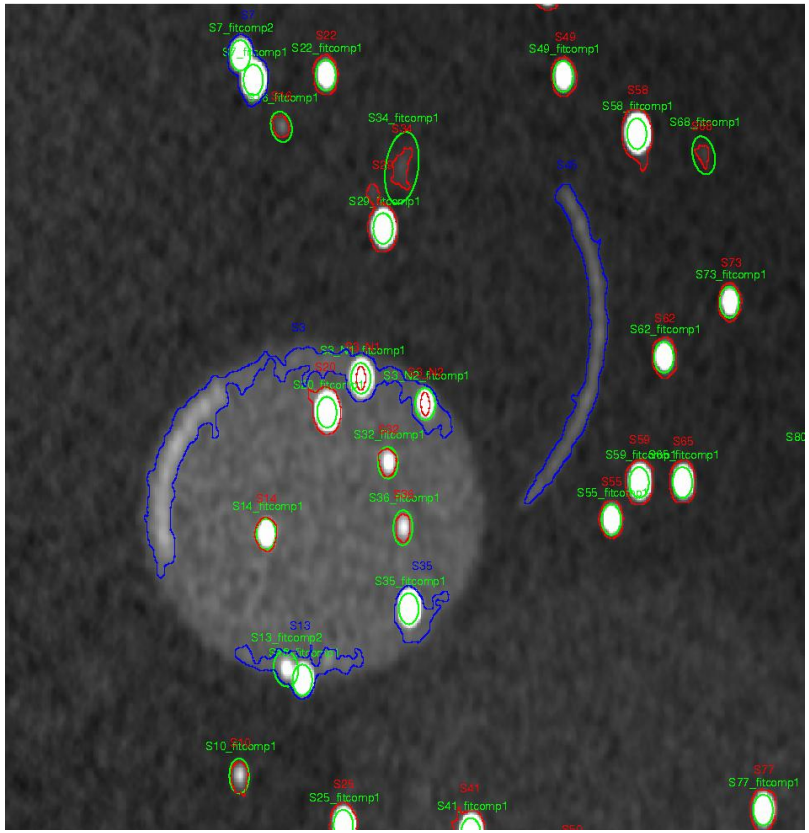


Sample map with generated (convoluted) sources (see next slide)

100 simulated maps generated up to now
~ 5 CPU hours per map on OACT cluster

50000 CPU hours available (proposal approved) on INAF computing facilities

Compact source extraction



Sample simulated sub-field
Blue: compact (not point-like)
Red: point-like
Green: point-like fitted

Typical parameters

- *Background*
  - box size=20 x beam size (beam size~12'')
  - 3-sigma clipped stats
- *Compact source search*
  - seed/merge thr= 5/2.5 sigma
  - pix min=5
  - niters=3
- *Nested sources*
  - Method: LoG
  - min/max scales: 1/2 x beam
  - nested/mother>20 x beam area
  - seed/merge curv thr= 5/2.5 sigma
- *Fitting*
  - max components=5
  - source area<10 beam (for joint fit)
  - peak significance thr>1 (wrt source)

Compact source extraction

~35640 compact sources in simulated sample (cumulated over 100 maps)

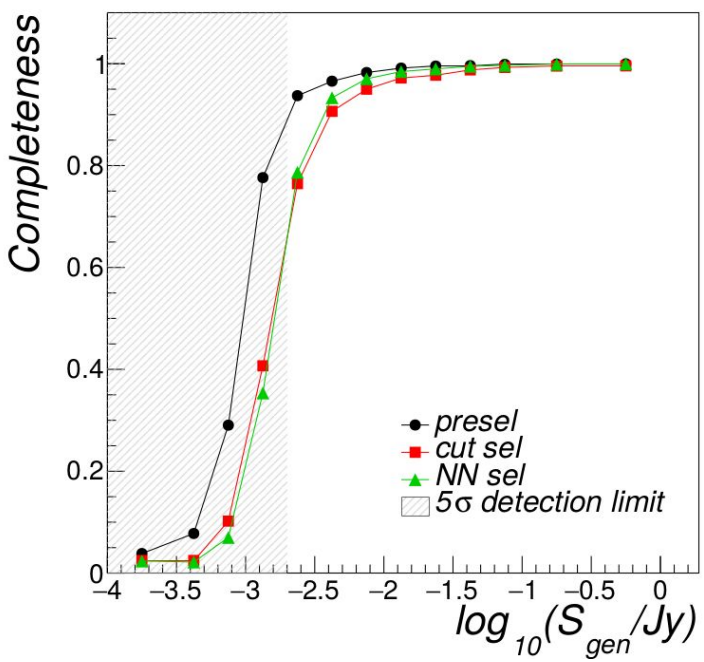
VERY PRELIMINARY

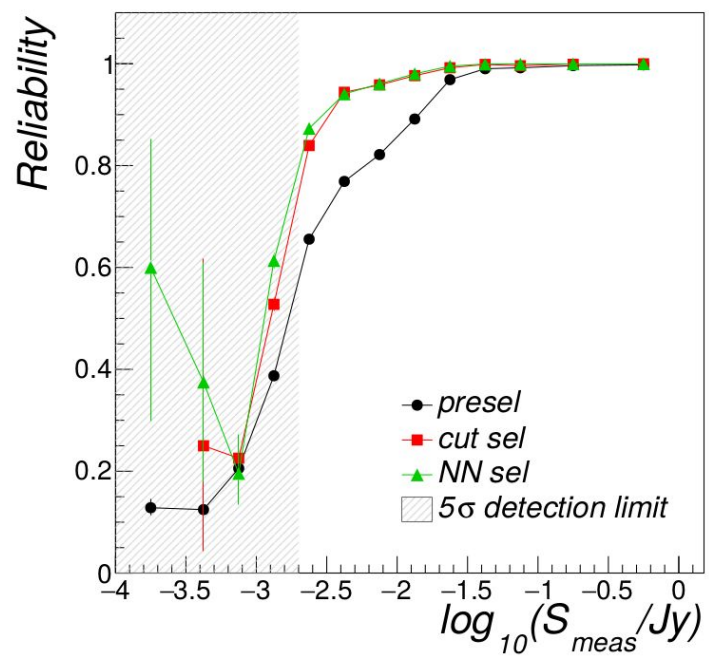
Match criteria

- Positional difference<10"

Selection cuts

- *Preselection*
  - ➢ Point source tag + fit available
  - ➢ Only sources tagged as point-like used for reliability evaluation
  - ➢ Exclude sources at the image borders
- *Selection*
  - ➢ Identified as point-source by simple cut or neural network classifiers





Completeness/ Reliability vs generated/measured source flux density

Compact source extraction

~35640 compact sources in simulated sample (cumulated over 100 maps)

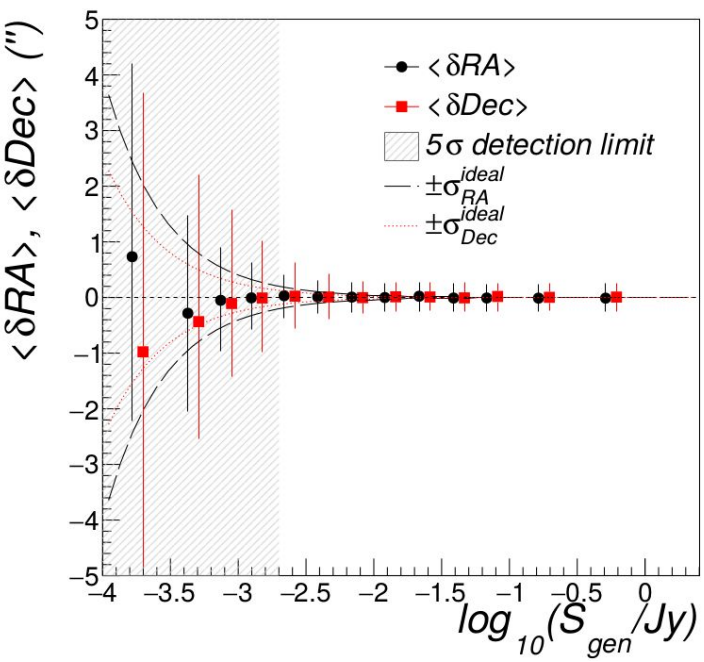
VERY PRELIMINARY

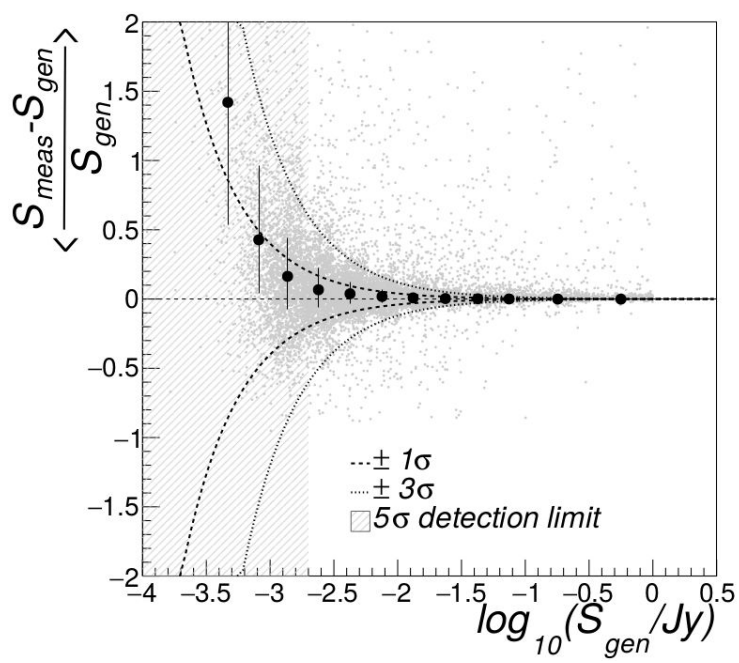
Match criteria

- Positional difference<10"

Selection cuts

- *Preselection*
  - Point source tag + fit available
  - Only sources tagged as point-like used for reliability evaluation
  - Exclude sources at the image borders
- *Selection*
  - Identified as point-source by simple cut or neural network classifiers





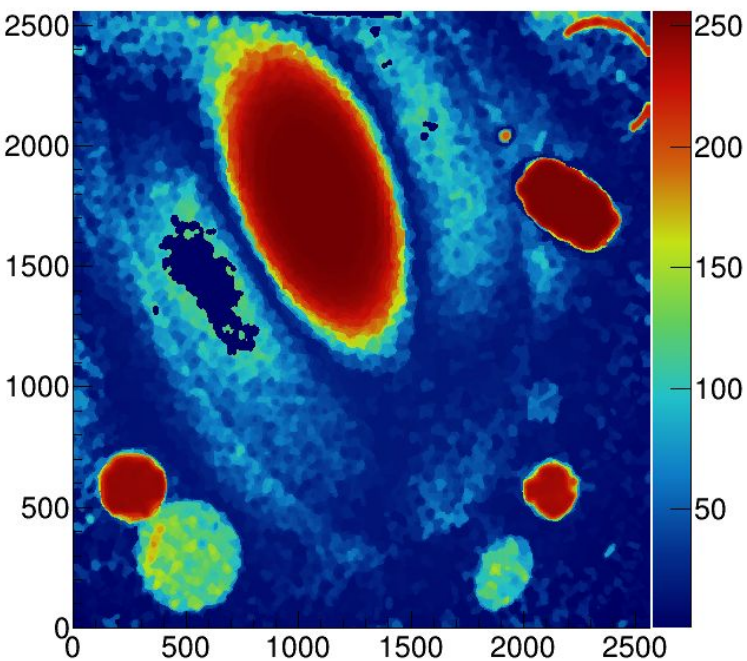
Positional/flux density accuracy vs generated source flux density

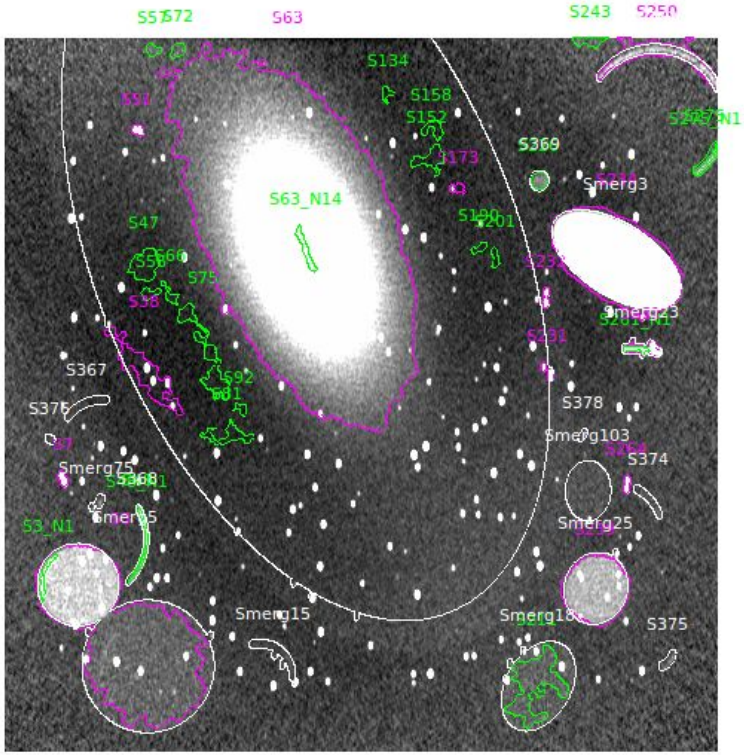
Extended source extraction

- **Saliency filter algorithm under test**
- **Parameters**
  - Residual map: kern size=9, high/low thr=10/5 sigma
  - Smooth filter (same of SCORPIO maps): radius=12 pixels, eps=0.04
  - Saliency map
    - ➢ Min/max/step scales: 20/60/20 pixels
    - ➢ Threshold=3 x median
    - ➢ No bkg/noise/curvature maps combined

Detected sources
Green: extended
Purple: extended+compact

Saliency filter map





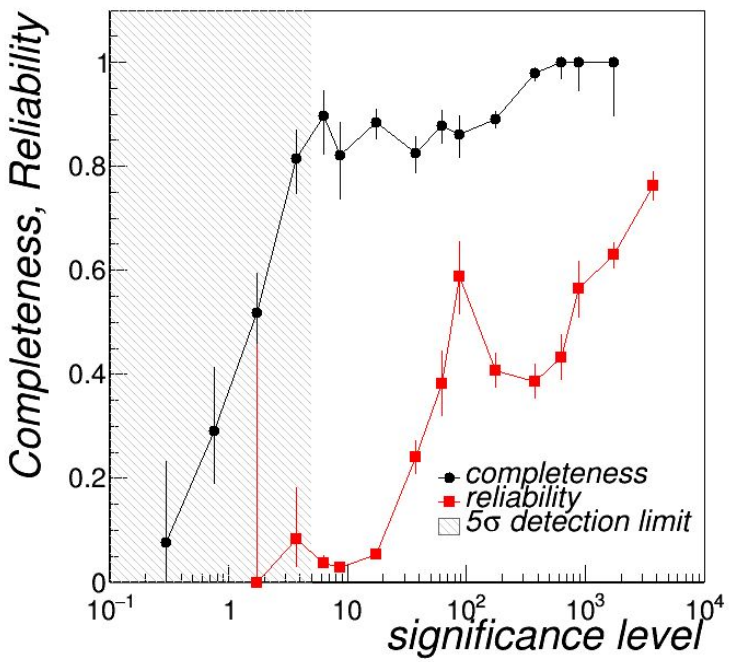
Extended source extraction

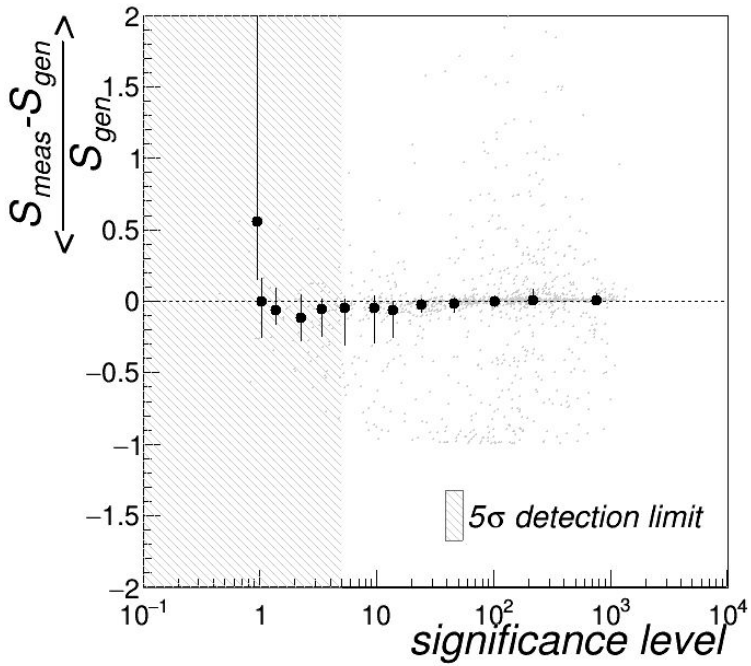
#1670 extended sources in simulated sample (cumulated over 100 maps)

VERY PRELIMINARY

Match criteria

- pixel overlapThr=0.1



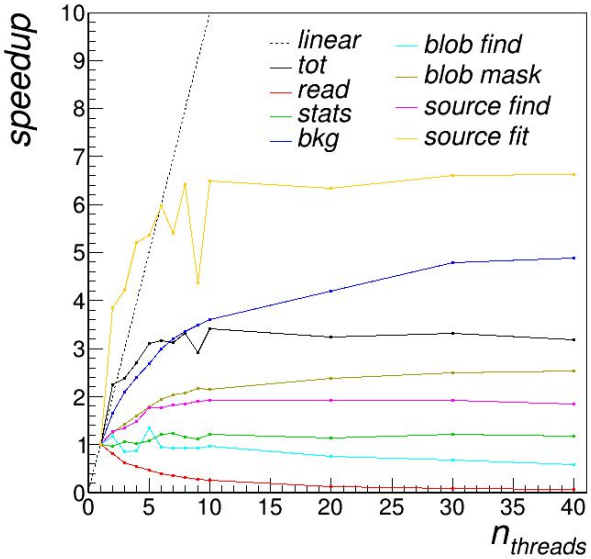


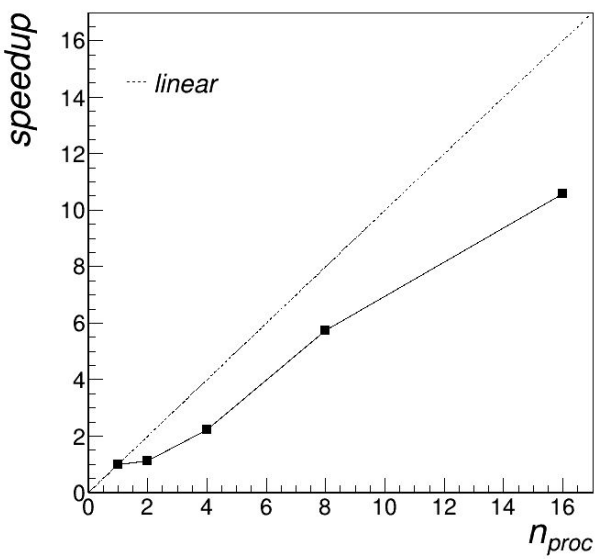
- ~ 20% source missed (majority are ring/arc-shaped sources)
- Many false detections, need a method to identify them
- Outliers in flux accuracy observed (possibly due to nested point-sources or wrong identifications)

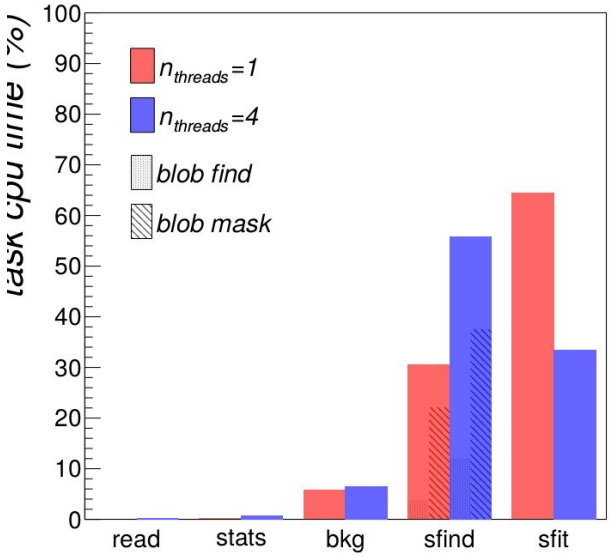
Computing performances

Parallel speedup vs number of threads (for different finding tasks)

- 1 Node: 4 sockets x 10 Core Intel(R) Xeon(R) CPU E5-4627 2.60GHz, 256 GB DDR4 2133 MHz
- Image size: 2560 x 2560 pixels (similar results for 10000 x 10000 pixels)
- Moderate speedup (up to 6-8 threads) for some tasks (bkg finding, source fitting)
- Threads allocated according to NUMA architecture
- Source fitting dominates in total cpu time







Parallel speedup vs number of MPI processes (for fixed OpenMP threads)

- 2 Nodes (same architecture as above)
- 4 threads per MPI process
- MPI processes allocated according to NUMA architecture
- Image size: 10000 x 10000 pixels partitioned into 2500x2500 tiles
- Good speedup observed

Summary

- **Some progress made since CAESAR reference paper**
  - Many to-do-list items were processed (e.g. public distribution, parallelization, ...)
  - Some areas requiring further work (e.g. memory usage, optimizations, source matching, extended source performances, etc)
  - Testing with simulations is ongoing and requiring a considerable effort

- **Road map & additional features**
  - *Short-term*
    - Additional Input/output formats (e.g. CASA, HDF5, VTK, VO, etc)
    - Complete task parallelization (e.g. blob masking, extended finders)
    - Algorithm improvements
    - Complete and improve documentation
  - *Medium term*
    - New extended source algorithms (once testing on existing is completed)
  - *Long term*
    - GPU version, distributed version with alternative programming models & frameworks

For discussion…

- **A SKA working group for data reduction and analysis software development?**
  - Should focus on all data reduction, analysis and computing aspects that:
    - Are to be performed at the Regional Centers
    - Are not deeply considered by SDP
    - Are common to different Science Groups
  - Is there a similar working group in SKA?
  - Possible issues
    - Working groups are geographically spread and not actively interacting
    - Open Observatory (e.g. PI-based projects) vs Wide Collaboration (like in LHC)
  - Activities to be carried out over a time scale of ~5 years
    - Using real data produced by precursors & SKA simulations
  - A SKA data analysis framework as possible goal
    - Tight coordination with regional centers

BACKUP SLIDES

Reading input maps, stats/bkg

- **Distributed source finding**
  - A series of tasks (=tiles to be processed) are assigned to each worker node
    - Raw task distribution, no intelligent scheduling on available resources & load
  - Assigned tiles can overlap each other

- **Input maps supported**
  - Only 2D maps supported, formats: FITS, ROOT TH2, OpenCV, jpg/png/...
  - FITS images can be read in parallel on single- (each thread reads a tile) or multi-hosts (each host reads and keep only assigned tiles)
    - Single-host: each thread reads a tile
    - Multi-host: each host reads (in multithread if desired) and keep in memory only assigned tiles

- **Image stats calculation**
  - Stats moments computed while reading image with online parallel formulas
  - Robust estimators (median/mad) calculation speed-up wrt to original implementation
  - Benchmark tests done against python numpy

- **Background/noise maps**
  - Cubic interpolation on a 2D grid made with sampling image tiles (size=multiple of beam size)
  - Sampling grid calculation done in parallel (not for interpolation)
  - If needed, significant blobs (e.g. >5 sigmas) can be excluded from background estimation
  - Benchmark tests done against BANE

Image filtering & residual map

- **Different image filters & processing available to support finding**
  - Wavelet Transform
  - Compact source filter
  - guided/gaus
  - LoG/grad
  - morph/dilation
  - Saliency
  - Superpixels

- **Residual map**
  1. Compact source removal
     - ➢ Replace bright sources with given tag (e.g. point-like) and above significance levels with configurable dilation kernel size
     - ➢ Using a dilation filter with configurable ellipse kernel size
  2. Smooth image to limit texture-like features

Compact and extended finders

- **Compact source finder**
  1. Blob detector: flood-fill algorithm (ala Aegean/Blobcat) iterated up to N times and until no more blobs are found (threshold decreased at each cycle)
  2. Blob selection: good/bad/point-source selected according to simple parameters (area wrt to beam, roundness/elongation, bounding box)
  3. Nested blob search: based on image curvature thresholding

- **Extended source finder: different algorithms available**
  1. Filtering + thresholding: Multi-scale saliency, Wavelet Transform
     - Method: filter image to enhance diffuse emission and apply threshold
     - Pros: conceptually simpler, threshold-based
     - Cons: object boundary can be not so accurate, sensitive to residual map quality
  2. Region-merging: Hierarchical Clustering (not ready yet)
     - Method: hierarchical superpixel/region merging by similarity
     - Pros: dealing with disjoint areas
     - Cons: intensive computing, many parameters to be optimized, sensitive to residual map quality
  3. Contour finding: Chan-Vese/LRAC
     - Pros: relatively fast, object boundaries detection
     - Cons: highly dependent on initial contour, insensitive to nested objects, varying object intensity levels, sensitive to residual map quality

Source merging & parametrization

- **Source merging: 2 aggregation stages can be performed**
  1. The first is performed by each task on each tile to merge extended sources found to compact sources detected at the blob finder stage
     - ➢ Helpful if compact finder detects only bright regions and extended finder only the diffuse part
  2. The second is performed by master process after gathering all sources detected by workers
     - ➢ Aggregate sources tagged at edge by workers
     - ➢ Aggregate sources overlapping in the tile overlap region (if present)

- **Source parametrization**
  1. Source moments (standard, Hu), centroid, integrated flux
  2. Contour morphology parameters (e.g. Zernike pols)

- **Source fitting & deblending (only for compact sources)**
  1. Peak finder using a multi-kernel image dilation method
  2. Peaks sorted by significance and selected (e.g. up to a configurable number of components N, sufficiently far each other, above a given significance)
  3. Multi-component 2D gaussian fit (using MINUIT)
     - ➢ Model: const offset + N gaussians
     - ➢ Gaussian parameters limited around peak amplitude and beam (limit range configurable)
     - ➢ Offset fixed to user-provided value or estimated bkg average or free to vary

- **Delivered outputs**
  - FITS images, DS9 regions (colored & tagged by source type)
  - ROOT file with image & source object collections

How to run CAESAR?

- **Interactive mode (useful for simple tasks/testing/drawing/macros)**

ROOT cli

```
[...]$ root
root [0] Image* img= new Image
root [1] img->ReadFITS("recmap.fits")
root [2] img->ComputeStats(true)
root [3] ImgBkgData* bkgData=
img->ComputeBkg(eMedianBkg,true,100,100,10,10)
... ...
```

python cli

```
[...] $ ipython
In [1]: from ROOT import gSystem
In [2]: gSystem.Load('libCaesar')
In [2]: from ROOT import Caesar
In [3]: img= Caesar.Image()
In [4]: img.ReadFITS('recmap.fits')
In [5]: img.ComputeStats(True)
In [6]: bkgData=
img.ComputeBkg(Caesar.eMedianBkg,True,100,100,10,10)
... ...
```

- **Batch mode (standard or container run)**

Standard run

```
[...]$ FindSourceMPI --config=myconfig.cfg
... ...
```

Run inside Singularity container

```
[...]$ singularity apps caesar.simg  (to list available apps)
[...]$ singularity run --app sfinder caesar.simg
          --config=myconfig.cfg
... ...
```

- **Submission scripts available for running on batch systems**
- **Other apps provided for standard or container run**
  - fits2root/root2fits
  - skymodel, simulation, imaging, convolver
  - bkgfinder, saliencyfinder
  - catalogcorr