


Research Paper

CAESAR source finder: Recent developments and testing

S. Riggi¹ , F. Vitello¹, U. Becciani¹, C. Buemi¹, F. Bufano¹, A. Calanducci¹, F. Cavallaro¹, A. Costa¹, A. Ingallinera¹, P. Leto¹, S. Loru¹, R. P. Norris^{2,3}, F. Schillirò¹, E. Sciacca¹, C. Trigilio¹ and G. Umana¹

¹INAF-Osservatorio Astrofisico di Catania, Via Santa Sofia 78, 95123 Catania, Italy, ²CSIRO, P.O. Box 76, Epping, NSW 1710, Australia and ³Western Sydney University, Penrith, NSW, Australia

Abstract

A new era in radio astronomy will begin with the upcoming large-scale surveys planned at the Australian Square Kilometre Array Pathfinder (ASKAP). ASKAP started its Early Science programme in October 2017 and several target fields were observed during the array commissioning phase. The SCORPIO field was the first observed in the Galactic Plane in Band 1 (792–1 032 MHz) using 15 commissioned antennas. The achieved sensitivity and large field of view already allow to discover new sources and survey thousands of existing ones with improved precision with respect to previous surveys. Data analysis is currently ongoing to deliver the first source catalogue. Given the increased scale of the data, source extraction and characterisation, even in this Early Science phase, have to be carried out in a mostly automated way. This process presents significant challenges due to the presence of extended objects and diffuse emission close to the Galactic Plane.

In this context, we have extended and optimised a novel source finding tool, named CAESAR, to allow extraction of both compact and extended sources from radio maps. A number of developments have been done driven by the analysis of the SCORPIO map and in view of the future ASKAP Galactic Plane survey. The main goals are the improvement of algorithm performances and scalability as well as of software maintainability and usability within the radio community. In this paper, we present the current status of CAESAR and report a first systematic characterisation of its performance for both compact and extended sources using simulated maps. Future prospects are discussed in the light of the obtained results.

Keywords: Galactic Plane – radio astronomy – source-finding – software

(Received 16 February 2019; revised 02 August 2019; accepted 02 August 2019)

1. Introduction

The Square Kilometre Array (SKA) precursor era has finally come with the opening of the Australian SKA Pathfinder (ASKAP) Early Science programme in October 2017. While the deployment phase is still ongoing, a number of target fields are being observed with the commissioned antennas to demonstrate ASKAP scientific capabilities, validate imaging pipeline, and facilitate the development of analysis techniques in view of the operations with the full 36-antenna array. In particular, the SCORPIO survey field (~ 40 deg² in size, centred on $l = 343.5^\circ$, $b = 0.75^\circ$) was observed in January 2018 in ASKAP Band 1 (912 MHz) with 15 antennas. Details on the observation strategy and data reduction will be presented in a forthcoming paper.

The SCORPIO survey (Umana et al. 2015), started in 2011 with a pilot programme conducted with the Australian Telescope Compact Array (ATCA), has a clear scientific goal, which is the study and characterisation of different types of Galactic radio sources, from stars to circumstellar regions (HII regions, planetary nebulae, luminous blue variables, Wolf–Rayet stars) and stel-

lar relics (e.g. supernova remnants). Besides its scientific goals, it represents an important test bench for imaging and analysis techniques in the Galactic Plane in view of the upcoming ASKAP Evolutionary Map of the Universe (EMU) survey (Norris et al. 2011), planned to start at the end of 2019.

In this context, the accuracy of source finding algorithms is still a concern considering that the size of the EMU survey in terms of surveyed area and number of expected sources will severely limit a manual intervention on the source cataloguing process.

Significant efforts have been spent within the ASKAP EMU Collaboration^a to systematically compare different source finders, evaluating their performances on simulated data samples (Hopkins et al. 2015). These analysis pointed out strong and weak features of the tested finders, triggering new developments in specific areas, such as source deblending and fitting [e.g. see Hancock et al. (2018) and Carbone et al. (2018) for recent works]. Existing works, however, concentrate on compact sources, and well-known source finders, such as Aegean (Hancock et al. 2012), PyBDSF (Mohan et al. 2015), and BLOBCAT (Hales et al. 2012), have been shown not to perform well on extended sources, revealing the need for further developments in the characterisation of complex extended sources and for a systematic testing with simulations.

The CAESAR source finder (Riggi et al. 2016) was developed to overcome some of these issues and to provide missing features, particularly for the analysis of radio maps in the Galactic Plane.

Author for correspondence: Simone Riggi, E-mail: simone.riggi@inaf.it

Cite this article: Riggi S, Vitello F, Becciani U, Buemi C, Bufano F, Calanducci A, Cavallaro F, Costa A, Ingallinera A, Leto P, Loru S, Norris RP, Schillirò F, Sciacca E, Trigilio C and Umana G. (2019) CAESAR source finder: Recent developments and testing. *Publications of the Astronomical Society of Australia* 36, e037, 1–13. <https://doi.org/10.1017/pasa.2019.29>

This paper has multiple goals. Firstly we report the status of CAESAR and recent developments made since Riggi et al. (2016) in Section 2. Secondly, we resume the ongoing efforts to systematically characterise and evaluate the source detection accuracy and computational performances with simulated data. In Section 3, we describe the simulated data sample produced to test CAESAR performances. The analysis carried out on simulated data are reported in Section 4. Performance results (completeness, reliability, etc.) obtained on both compact and extended sources are presented and discussed. In Section 5, we analyse the computational performance (CPU and memory usage, scalability, etc.) obtained in multithreaded and parallel runs performed over a test computing infrastructure. Finally, in Section 6, we discuss the CAESAR road map and further analysis to be carried out, taking into consideration the results obtained in this paper.

This work constitutes also part of the ongoing analysis for the preparation of ASKAP SCORPIO Early Science source catalogue. Besides the SCORPIO and ASKAP EMU Galactic programmes, this work is well suited in the context of SKA *OurGalaxy* key science project and the European SKA Regional Data Centre (ESDC) design.^b Indeed, it is anticipated that the SKA Science Data Processor (SDP) will invest limited resources for the development, optimisation, and testing of science algorithms particularly for the Galactic science (Johnston-Hollitt et al. 2016). These activities have therefore to be largely lead in synergy by science and ESDC working groups.

2. CAESAR: Status and recent developments

CAESAR (Riggi et al. 2016) is a C++ tool for extraction of compact and extended sources from astronomical images developed in the context of the SCORPIO project and ASKAP EMU survey. It is based on third-party libraries and software frameworks, among them ROOT (Brun & Rademakers 1997), OpenCV (Bradski 2000), and MPI library.^c

A number of improvements and developments have been done in distinct areas since the original work (Riggi et al. 2016), summarised as follows:

- **Code refactoring:** The software code was updated and reorganised to improve modularity and maintainability and to lower memory demand. Dependency on some of the external libraries (R, OpenMP, MPI) was made optional.
- **Algorithm optimisation and speed-up:** Recurrent tasks, including image reading, statistics and background estimation, and image filtering, were optimised and parallelised using OpenMP^d directives, whenever a benefit in speed-up was identified. For example, computation of image median estimators was optimised to improve the original time complexity from $O(N \log(N))$ to $O(N)$. Statistical moments (up to fourth order) are computed using online parallel formulas (e.g. while reading and filling the image in different threads). Benchmark tests were carried out against corresponding python implementations (mostly based on python numpy module) and a speed-up ~ 12 was found on sample images of size $32\,000 \times 32\,000$ pixels. Newer parallel algorithms available in the standard C++ library (e.g. parallel *nth_element*) were also tested and benchmarked

^bDetails on the SKA ESDC design and AENEAS EU H2020 project available at <https://www.aeneas2020.eu/>

^cwww.open-mpi.org

^dwww.openmp.org

against the corresponding non-parallel version. No significant improvements were found in this case.

- **Distributed processing:** A parallel MPI-based version of the source finder application was implemented to support distributed processing of large maps on different computing nodes. Multithread processing per node, based on OpenMP, is also available and configurable. A serialiser, based on the Google Protocol Buffer library,^e was added to allow serialisation/deserialisation of CAESAR objects when exchanging data across computing nodes.
- **Logging:** Custom logging macros were added to all components and applications using log4cxx library.^f Logging levels can be customised from a configuration file or programmatically.
- **Algorithm improvements and extensions:** Compact source finding was improved in different aspects with respect to previous version. Details are reported in the following sections. Additional applications, besides source finding, were added to ease post-processing tasks, such as source cross-matching and analysis.
- **Distribution and usability:** Efforts have been made to make CAESAR publicly available at <https://github.com/SKA-INAF/caesar.git>, portable and usable in different systems with limited effort. To this aim, we provide recipe files to build and run CAESAR applications in a Singularity^g container. Details on how to use CAESAR are given in the online documentation at <https://caesar-doc.readthedocs.io/en/latest/>.

2.1. Processing pipeline

A schema of the processing pipeline is shown in Figure 1. The input image is partitioned into sub-images or tiles according to configurable parameters (e.g. tile size and overlap). Tiles are then distributed among available workers for processing. Each processor, thus, effectively reads and keeps in memory only a portion of the input image, corresponding to the assigned tiles. Each worker executes the source finding pipeline on the assigned tiles in sequence. This includes a series of steps, shown in Figure 1 for one representative processor and tile, summarised as follows:

1. Extract compact sources from input tile through the following stages (see 2.1.1 for more details):
 - (a) Compute image statistic estimators, global and local background, noise, and significance maps.
 - (b) Iteratively extract blobs using significance map and nested blobs using a blob-sensitive filter map.
 - (c) Reject anomalous blobs and promote blobs to source candidates, tagging them as compact or point-like.
 - (d) Fit and parametrise source candidates present in the tile and tag sources located at the borders.
2. Compute a residual map, obtained from input map by applying one or more filters (smoothing, filters to enhance diffuse emission) after removal of compact bright sources (see 2.1.2).
3. Extract extended sources from residual map according to the selected algorithm and tag them accordingly (see 2.1.3).
4. Merge adjacent and overlapping compact and extended sources found in the tile (see 2.1.4).

^e<https://developers.google.com/protocol-buffers/>

^f<https://logging.apache.org/log4cxx/index.html>

^g<https://singularity.lbl.gov/>

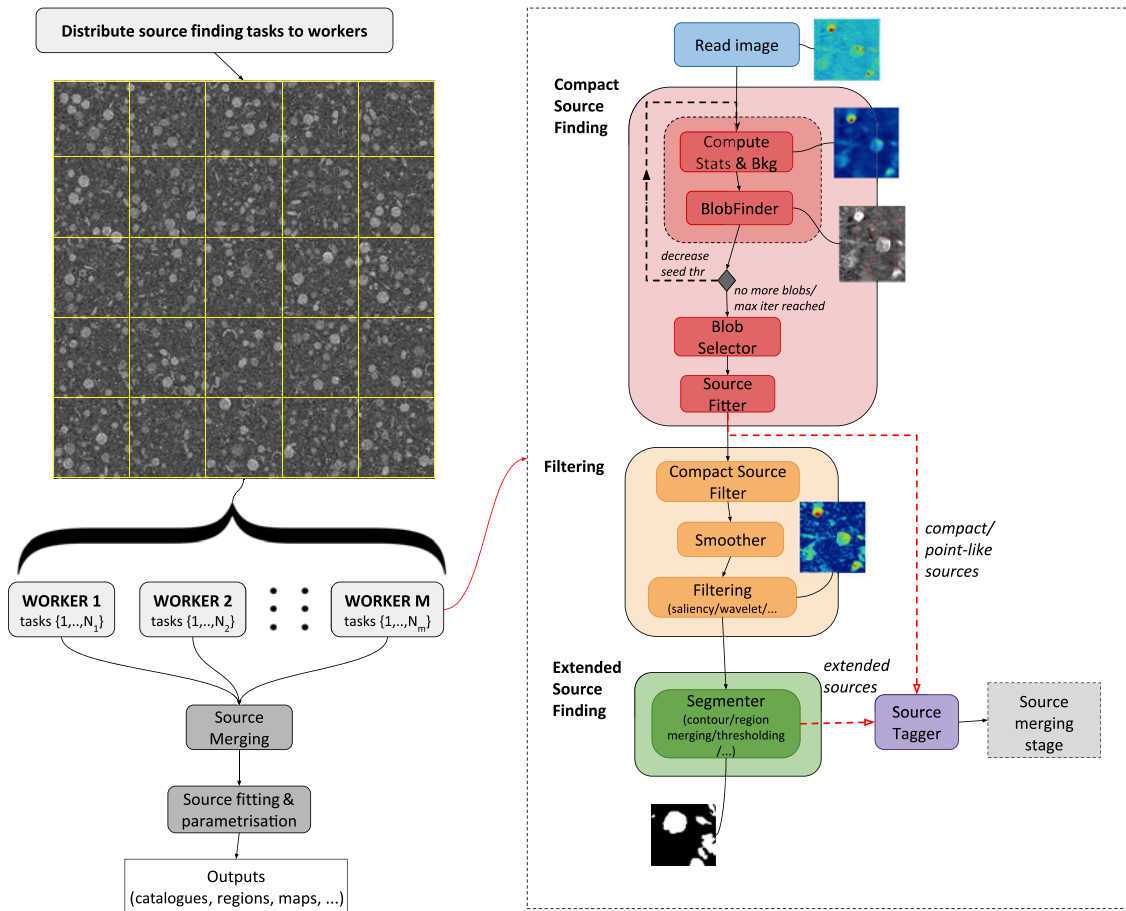


Figure 1. A schema of CAESAR source finding pipeline. See text for a description of pipeline stages. Compact and extended source finding stages are described in 2.1.1 and 2.1.3, respectively. Filtering and source merging stages are described in 2.1.2 and 2.1.4.

A master processor aggregates sources extracted by other workers, merging them if overlapping or adjacent at tile borders. Source fitting and parametrisation are finally performed on merged sources (if any), and outputs (e.g. catalogues with sources and fitted components, regions, etc.) are delivered as final results.

Details on the computing stages and specific algorithms can be found in Riggi et al. (2016). In the following sections, we limit the discussion on the improvements made in the new version, mainly relative to compact source extraction.

2.1.1. Compact source extraction

Compact source extraction is based on four stages:

1. *Blob search:* Blobs are extracted from the input map with a flood-fill algorithm using a pixel significance detection threshold $Z_{thr,d}$ (usually equal to 5) and a lower aggregation threshold $Z_{thr,m}$ (usually equal to 2.5). Pixel significance level Z is computed as

$$Z = \frac{S - \mu_{bkg}}{\sigma_{rms}}, \tag{1}$$

where S is the pixel flux and μ_{bkg} and σ_{rms} are the estimated background level and noise rms, respectively.

Blob extraction can now be performed using an iterative procedure in which the background and noise maps are re-computed at each iteration without pixels belonging to sources extracted

in the previous iterations. Detection thresholds can be progressively lowered by a configurable amount ΔZ (0.5 by default) at each j th step until a maximum number of iterations is reached:

$$Z_{thr,d}^{(j)} = Z_{thr,d}^{(0)} - j \times \Delta Z.$$

2. *Nested blob search:* A blob detector algorithm can be applied on the input map to search for ‘nested’ (or ‘child’) blobs inside the ‘primary’ (or ‘mother’) blobs extracted in the previous step. Nested blobs are used in the image residual and source fitting stages (described in the following paragraphs). When enabled, the algorithm proceeds as follows:

- A primary blob mask is obtained using blobs detected with the flood-fill approach.
- A blob-sensitive filter is applied to the input map, and blobs are searched in the resulting filtered map using flood-fill method around detected peaks above a specified significance threshold (typically equal to 5). Extracted blobs are then used to build a secondary blob mask.
- The secondary blob mask is cross-matched against the primary one to extract nested blobs and associate them to primary blobs.

Two alternative blob-sensitive filter models are provided [multiscale Laplacian of Gaussian (LoG), elliptical Gaussian] with customisable kernel size and scale parameters (first/last scale,

scale increment). The first method can be computationally demanding if the chosen kernel size is large (e.g. say above 9–11 pixels) and several scales are requested. The second approach is relatively fast as it employs only one scale, that is, the elliptical beam of the input map.

Nested blob search can be disabled if not explicitly needed (typically in the absence of extended sources) or, alternatively, customised. For example, nested blobs can be searched only on sources tagged as extended, that is, exceeding a certain area-to-beam threshold factor (usually set to 10–20).

3. *Blob selection*: Extracted blobs are selected using simple morphological parameters (blob area-beam ratio, roundness, elongation, bounding box, etc.) to tag candidate point-like sources and exclude anomalous blobs with elongated shapes, most likely due to imaging artefacts.
4. *Source deblending and fitting*: Source fitting is performed by workers on sources that are not located at the tile borders and by the master processor on merged edge sources. The adopted fitting procedure depends on the detected source size. For extended sources, that is, above a configured area-to-beam ratio, only nested blobs (if any) are individually fitted. Compact sources (e.g. nested or not and below the area-to-beam ratio threshold) are fitted with a mixture of M Gaussian components, plus a background offset parameter S_0 . The following χ^2 is minimised with respect to $(M + 1)$ fitting parameters $\Theta = \{S_0, \Theta_1, \dots, \Theta_M\}$, where $\Theta_k = \{\bar{x}_k, \bar{y}_k, \sigma_{x_k}, \sigma_{y_k}, \theta_k\}$:

$$\chi^2 = \sum_{i=1}^N \frac{[S_i(x_i, y_i) - \hat{S}_i(x_i, y_i; \Theta)]^2}{\sigma_i^2}, \quad (2)$$

where

$$\hat{S}(x_i, y_i, \Theta) = S_0 + \sum_{k=1}^M f_k(x_i, y_i; \Theta_k), \quad (3)$$

$$f_k(x_i, y_i, \Theta_k) = A_k \exp[-a_k(x_i - \bar{x}_k)^2 - b_k(x_i - \bar{x}_k)(y_i - \bar{y}_k) - c_k(y_i - \bar{y}_k)^2], \quad (4)$$

$$a_k = \frac{\cos^2(\theta_k)}{2\sigma_{x_k}^2} + \frac{\sin^2(\theta_k)}{2\sigma_{y_k}^2}, \quad (5)$$

$$b_k = \frac{\sin(2\theta_k)}{2\sigma_{x_k}^2} - \frac{\sin(2\theta_k)}{2\sigma_{y_k}^2}, \quad (6)$$

$$c_k = \frac{\sin^2(\theta_k)}{2\sigma_{x_k}^2} + \frac{\cos^2(\theta_k)}{2\sigma_{y_k}^2}, \quad (7)$$

with N number of source pixels, S_i and \hat{S}_i the data and the predicted flux of the i th source pixel, respectively, and σ_i^2 variance of the measurements. We assumed σ_i equal to the estimated noise averaged over fitted source pixels. A_k denotes the peak brightness of the k th fitted component.

Total source flux density I is computed as

$$I = \sum_{k=1}^M I_k, \quad I_k = 2\pi A_k \sigma_{x_k} \sigma_{y_k}, \quad (8)$$

with I_k flux density of the k th component. The flux density error δI is computed by error propagation:

$$\delta I = \sqrt{\mathbf{D}\Sigma\mathbf{D}^T}, \quad \mathbf{D} = \frac{\partial I}{\partial \Theta}, \quad (9)$$

where \mathbf{D} is the derivative matrix of flux density with respect to fit parameters Θ and Σ is the fit parameter covariance matrix. χ^2 numerical minimisation is performed with the ROOT minimiser libraries. Different minimisers^h [e.g. *Minuit* (James 1972), *Minuit2* (Hatlo et al. 2005), and *RMinimiser*] and minimisation algorithms (e.g. *Migrad*, *Simplex*, *BFGS*) are available to the user, all of them providing estimated errors on the fitted parameters as well as the fit parameter covariance matrix Σ .

The approach followed to determine the optimal starting number of fitted components and relative parameters is usually denoted as the deblending process. Details are provided in Appendix A.

All model parameters can be kept fixed or left free to vary in the fit and limits can be applied around parameter starting values. To guide fit convergence, the fit procedure is first performed with some parameters fixed (e.g. offset and component amplitudes) to the initial values. Fixed parameters are released afterwards and a full fit is performed. If one or more fitted parameters are found close or at the specified limits, the fit procedure can be iteratively repeated, progressively enlarging the parameter range, until no more parameters are found at limits or a maximum number of retry iterations are exceeded. If the fit does not converge, it can be repeated by progressively removing fainter fit components until convergence or until no more components are left.

2.1.2. Residual image and filtering

Algorithms to extract faint extended sources are almost ineffective in presence of very bright point sources and noise artefacts in the field. For these reasons, the search is carried out on a residual image in which sources with peak flux above a configurable significance threshold with respect to the background (usually equal to 10) are removed from the map. Subtraction can be done in two alternative ways. The first method simply replaces all pixels belonging to bright sources with the estimated background.ⁱ The advantage of this approach, proposed in Peracaula et al. (2011), is that it can be performed with only background information computed. A second, more refined, method subtracts the fitted model of bright sources from the input map. This, on the other hand, requires fit information to be available and accurate enough for the subtraction to be effective.

A series of filters can be applied to the residual image to limit the impact of small-scale artefacts and enhance the faint diffuse emission. A guided or Gaussian smoothing filter is employed in CAESAR for the former scope, while a Wavelet transform or saliency filter [see Riggi et al. (2016)] can be finally applied to produce the optimal input map for extended source search.

2.1.3. Extended source extraction

Four classes of algorithms are currently available in CAESAR to extract extended sources from a suitable input map (typically a residual map):

1. *Wavelet transform*: Input map is decomposed in J Wavelet scales (typically $J = 6-7$) and extended sources are extracted from higher scales by thresholding (e.g. employing the same algorithm used for compact sources).

^hFor multithreaded fitting, *Minuit2* has to be used as the other minimisers are not thread-safe.

ⁱThe algorithm uses a dilation filter to replace also pixels surrounding the source according to a configurable kernel size.

2. *Saliency filtering*: A multiscale saliency filter is applied to the input map and extended sources are extracted from the filtered image by thresholding (e.g. employing the same algorithm used for compact sources).
3. *Hierarchical clustering*: Input map is oversegmented into a series of superpixels (or regions) on the basis of a spatial and flux similarity measure. Neighbouring regions are then adaptively merged by mutual similarity and a final segmentation into background and sources is obtained. The method was presented in Riggi et al. (2016) and it is currently being updated to lower its computing resource demand;
4. *Active contour*: The method iteratively determines the contour that best separates objects from the background in the input image, starting from an initial curve and evolving it by minimising an energy functional until convergence. Two different algorithms are available, one based on the Chan–Vese active contour model (Chan & Vese 2001) and the other based on the localising region-based active contour model (Lankton & Tannenbaum 2008).

2.1.4. Source merging

Two source merging steps can be optionally included in the pipeline. The first is performed by workers at the end of each tile processing task to merge overlapping extended and compact sources found by different algorithms. This step was introduced to allow full detection of faint extended sources with compact brighter components. Indeed, the compact source finder typically detects only the bright regions, while the extended finder detects only the diffuse part, particularly if the former was removed/subtracted in the input residual map.

A second merging step is performed by the master process after gathering all sources detected by workers. Sources located at the edge or in overlapping regions of neighbouring tiles are merged if adjacent or coincident.

3. Simulated data

In order to test source finding performances, we generated simulated sky models (2560×2560 pixels, $1''$ pixel size) with both point and extended sources uniformly distributed in (α, δ) . A source density of 1000 deg^{-2} was assumed for point-sources and 50 deg^{-2} for extended sources. Source densities assumed in the simulation correspond to values measured in the SCORPIO ATCA survey (Umana et al. 2015). Source peak brightness S_{peak} was randomly generated with a uniform distribution in $\log(S_{peak})$ in the range $S_{peak} = [0.1, 1000] \text{ mJy/pixel}$ for point sources and $S_{peak} = [1, 100] \mu\text{Jy/pixel}$ for extended sources. The peak brightness distribution assumed was driven by the need of having a sufficient number of simulated sources for statistical analysis over the entire flux range, rather than by physical considerations or existing observations. Extended sources were generated with equal proportion weights from five different shape models (disc + shell, ring, ellipse, Gaussian, Sérsic profile) with a maximum angular scale of 10 arcmin.^j

For each sky model, we simulated 12 h observations with the Australia Telescope Compact Array (ATCA) using CASA tool (McMullin et al. 2007). All available ATCA configurations were

used. Eight pointings were needed to cover the sky model area given the ATCA primary beam.

The imaging stage was performed in an automated way assuming a $100\text{-}\mu\text{Jy}$ clean threshold and cleaning mask boxes around each generated source. Simulated fields were imaged singularly and combined afterwards to produce the final simulated mosaic. To limit computing time, the imaging process was not fully optimised. In fact, the focus was put in achieving a sufficient imaging of both compact and extended objects to carry out source finding. A number of 200 simulated mosaics are available to test source finding. The average noise level is $300\text{--}400 \mu\text{Jy}$ with the chosen imaging parameters and mosaic strategy. The synthesised beam of simulated maps is $b_{maj} = 13.3''$, $b_{min} = 8.4''$, and $b_{pa} = 0^\circ$. Although a number of effects have been neglected or ideally modelled (e.g. perfect calibration is considered), the simulated maps include typical interferometric noise patterns and can be used as a valid test bench for existing source finders. To this aim, the entire simulated data set was made publicly available at <http://doi.org/10.5281/zenodo.3257594>.

4. Analysis

In this section, we report the detection performances for compact and extended sources obtained on the simulated data sample described in Section 3.

4.1. Validation of simulated data

To test the imaging quality, we compared generated and imaged sources following this approach. Each generated source was convolved with the synthesis beam and the resulting image thresholded to keep 99% of the total source flux. The convolved source mask obtained represents the ground truth. Imaged sources are obtained by applying the convolved source mask to the simulated mosaic. Overlapping compact and extended sources were merged so that three classes of sources (point, extended + point, extended) have to be inspected.

A sample simulated map with convolved source contours for the three classes (point-like in red, extended in blue, and extended+point-like in green) is shown in Figure 2 (left panel). In Figure 2 (right panel), we compared the flux density of convolved and imaged sources (with background subtracted) for the three classes of sources and using the full simulated data set. As can be seen, fluxes are reconstructed with an accuracy better than 10% for bright sources, increasing to 40% for very faint sources. Systematic biases are found below 10%.

In the following analysis, we will take the imaged source flux as the reference when evaluating the flux uncertainty of the source finding process.

4.2. Detection of compact sources

Compact source finding was run on the $N = 200$ simulated maps using the set of parameters reported in Table 1. A number of 71 640 generated point-sources are available for analysis.

Sources tagged as point-like were cross-matched in position to generated sources. A generated source is labelled as ‘detected’ if the distance between its centroid and the one of a measured source is smaller than $10''$ (corresponding to 10 pixels and slightly less than the average of beam dimensions). If many measured source candidates are present, the matched source will be the one with the shortest distance.

^jFor Gaussian and Sérsic source generation models, the maximum angular scale assumed corresponds to the standard deviation and effective radius, respectively.

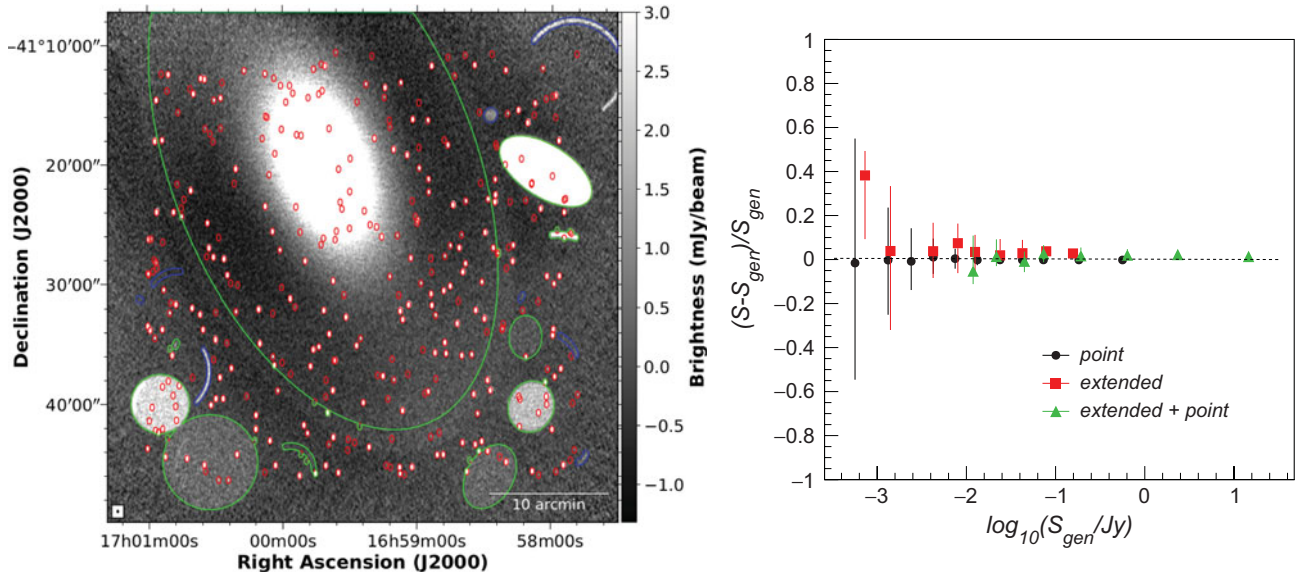


Figure 2. Left: Sample simulated map in mJy/beam units with convolved source contours superimposed (red = point-like, blue = extended, green = extended + point-like). Right: Imaging flux accuracy for point sources (black dots), extended sources with nested point sources (green triangles), isolated extended sources (red squares) obtained on the simulated data set. Each dot represents the median of the pull distribution $(S-S_{gen})/S_{gen}$ in $\log_{10} S_{gen}$ bins, being S_{gen} the generated source flux density (after convolution with the synthesis beam as described in the text), and S the imaged source flux density. Error bars are the interquartile range of the pull distribution.

Table 1. Compact source finder parameters.

Parameter	Value
<i>Bkg/Noise</i>	
Bkg	Median
Noise	MAD
Box size	10×beam
Grid step	20%box
<i>Blob detection</i>	
$Z_{thr,d}$	5
$Z_{thr,m}$	2.5
n_{pix}	5
n_{iter}	2
$\Delta\sigma_{seed}$	0.5
<i>Nested blob detection</i>	
Method	LoG
Min scale	1×beam
Max scale	2×beam
Scale step	1
$Z_{thr,d}$	5
$Z_{thr,m}$	2.5
n_{beams}^{thr}	20
<i>Source fitting</i>	
n_{beams}^{thr}	10
Max components	5
$Z_{thr,peak}$	1
Bkg offset	Fixed

4.2.1. Completeness and reliability

Following the described procedure, we computed the source completeness (or detection efficiency) and reliability metrics for the simulated data sample. Completeness, at a given level of quality selection, denotes the fraction of generated point-sources identified by the source finder, given the assumed match criteria, and passing the imposed selection cuts. Reliability is the fraction of detected sources passing the quality selection that corresponds to real sources. Completeness and reliability are reported in Figure 3 for four different selection cuts as a function of the source generated and measured flux density, respectively. The grey shaded area indicates a region of source significance below 5σ , assuming an average rms of $400 \mu\text{Jy}$. Black dots (labelled as ‘fit’) are obtained using a minimal set of quality cuts:

- Source match in position
- Source fit performed and converged
- Positive fitted amplitude parameters.

Red squares (labelled as ‘presel’) corresponds to high-quality fitted sources, passing the following preselection cuts:

- Fit $\chi^2/\text{ndf} < 10$
- Accurate fit error matrix (flag returned by fit minimiser).

Blue diamonds and green triangles correspond to two additional quality selection cuts applied to the detected sources after preselection (described in the following).

As can be seen, 90–95% of the generated sources at a 5σ flux significance are detected, assuming the finder parameters listed in Table 1 and the preselection cuts. The corresponding false detection rate is of the order of 20% at 5σ and 5–10% at larger significance levels. False detections are largely due to the overblending of imaging artefacts and extended sources present in the simulated maps. For instance, we report in Figure 4 examples

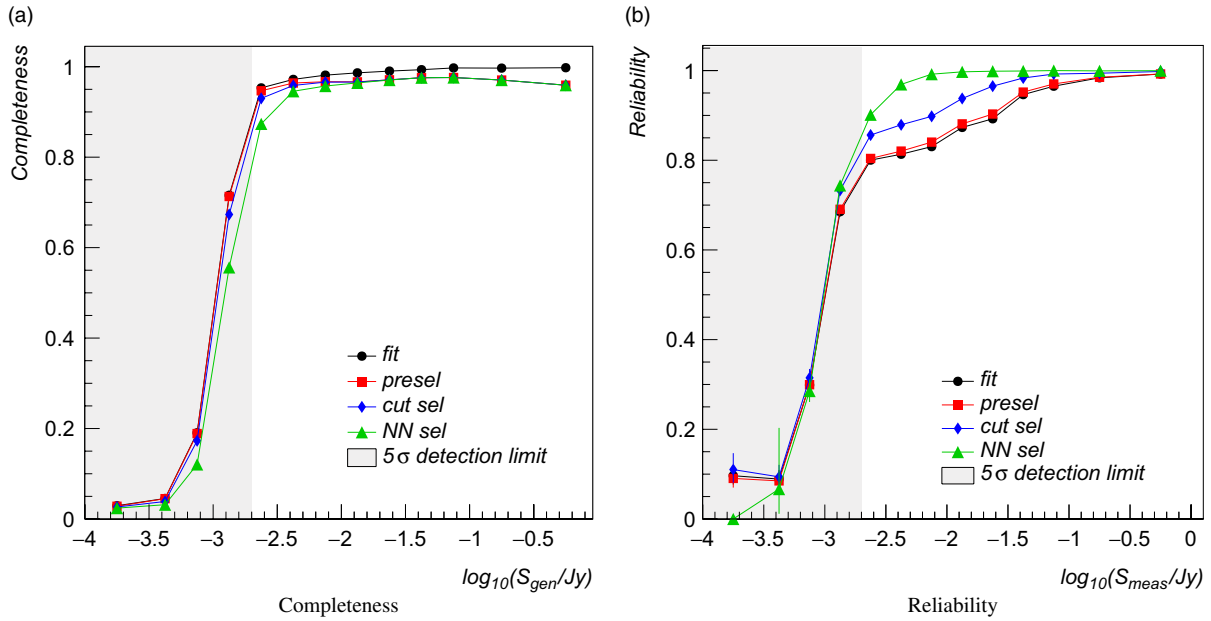


Figure 3. Left: Compact source detection efficiency as a function of the generated source flux density for four different source selections (described in the text): fit converged (black dots), preselection cuts (red squares), preselection + cut selection (blue diamonds), preselection + NN selection (green triangles). Right: Compact source detection reliability as a function of the measured source flux density for four different source selections (described in the text): fit converged (black dots), preselection cuts (red squares), preselection + cut selection (blue diamonds), preselection + NN selection (green triangles).

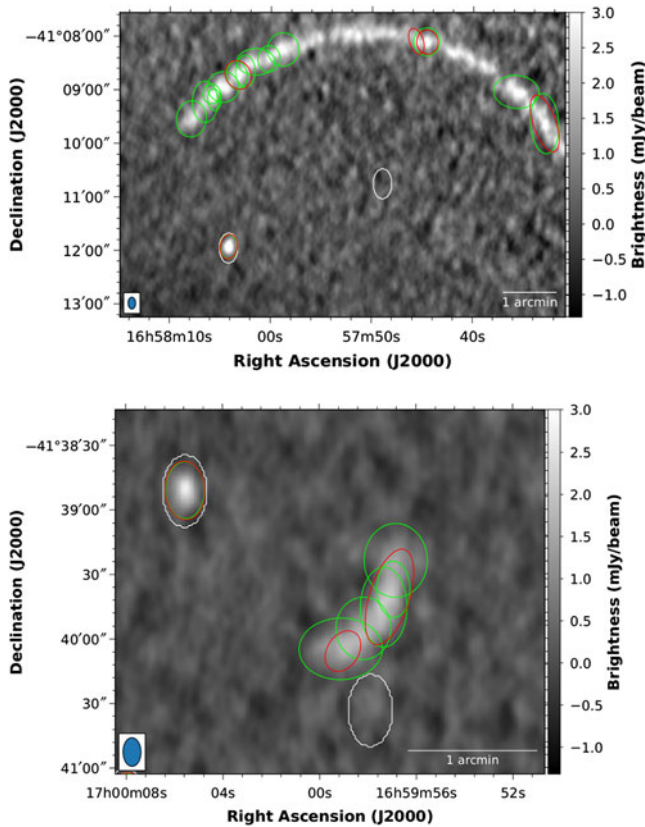


Figure 4. Sample false compact sources detected by CAESAR in simulated maps (red ellipses). Green ellipses represent sources detected by the AEGEAN source finder, while white ellipses represent generated point sources.

of false sources (shown with red contours) detected in two different simulated maps. White contours represents true point sources,

while green contours are the sources detected by the AEGEAN (Hancock et al. 2018) source finder for comparison. As can be seen, faint diffuse emission induces a large number of source components in the deblending process. This effect, expected to be observed in all finders implementing a deblending stage, is apparently less evident in similar analysis reported in the literature [e.g. see Hopkins et al. (2015)]. This is most likely due to a combinations of multiple factors: a better imaging in the (real or simulated) data, the absence of extended sources in the test samples, the usage of tighter quality cuts, etc.

Over-deblending can be partially prevented in CAESAR by increasing the source significance threshold and the deblending threshold parameters (peak threshold, n_{beams}^{thr} for fitting or the maximum number of fitted components). However, we have found that with a different choice of deblending parameters, the reliability can be slightly increased but no more than a few per cent. We therefore tried applying a further selection to the data to identify false sources. For this, we have exploited the physical consideration that true fitted point-like sources are expected to be morphologically similar to the beam and thus defined three classification parameters:

- $\delta\theta$: rotation angle (in degrees) of source fitted ellipse with respect to the beam ellipse, expected to be peaked around 0 for true sources, provided that the beam is elliptical in shape (as in this analysis);
- E_{source}/E_{beam} : source fitted ellipse eccentricity divided by the beam ellipse eccentricity, expected to be peaked around 1 for true sources;
- A_{source}/A_{beam} : source fitted ellipse area divided by the beam area, expected to be peaked around 1 for true sources;

In Figure 5, we report the distributions of the three parameters for real (red histograms) and false (black histograms) sources. Using these parameters, we have set up two different classifiers:

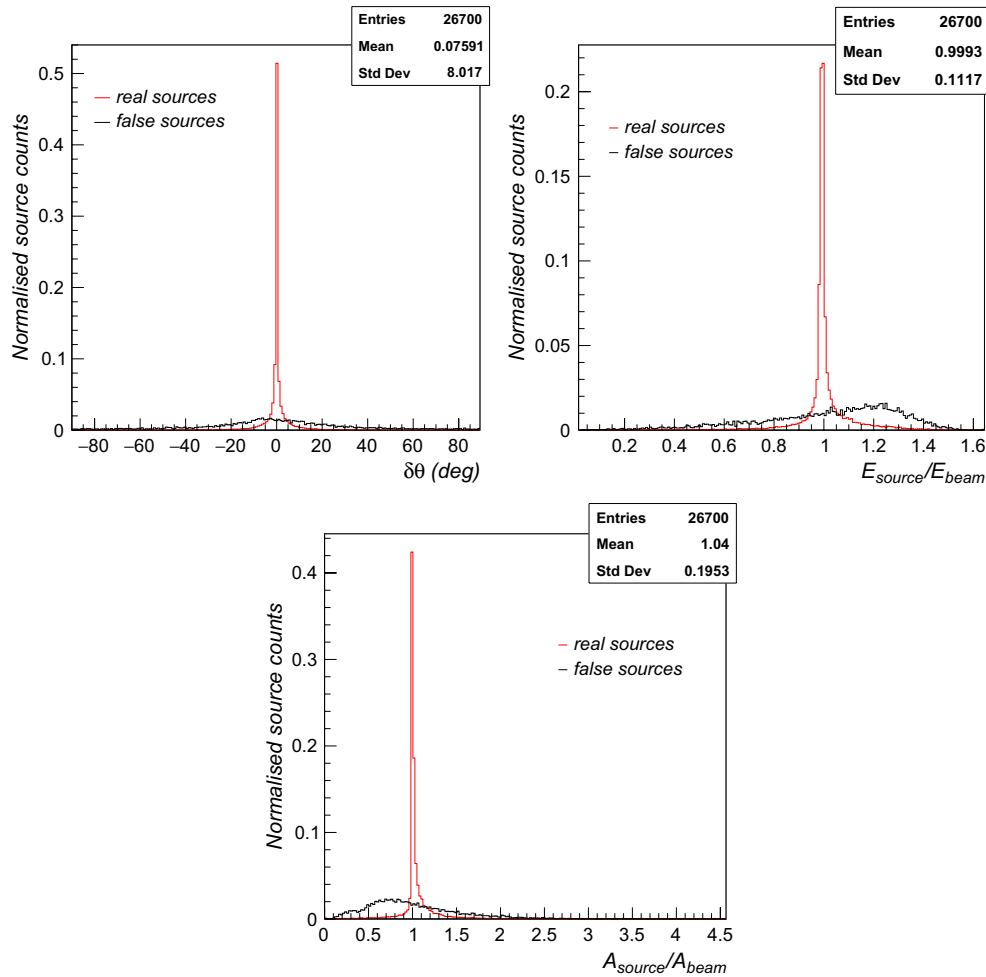


Figure 5. Distribution of the classification parameters for real (red histogram) and false sources (black histogram). $\delta\theta$ (upper panel) represents the rotation angle (in degrees) of source fitted ellipse with respect to the beam ellipse. E_{source}/E_{beam} (middle panel) represents the ratio between the source fitted ellipse eccentricity and the beam ellipse eccentricity. A_{source}/A_{beam} (bottom panel) represents the ratio between the source fitted ellipse area and the beam area. Histograms are normalised to unit area with normalised counts reported in the y-axis.

- *Cut-based classifier:* Sources passing these quality cuts on the three source parameters are selected as real:

- $|\delta\theta| < 45$,
- $0.5 < E_{source}/E_{beam} < 1.5$,
- $0.1 < A_{source}/A_{beam} < 10$.

Cuts are not fine-tuned and no correlation among variables is taken into account (e.g. cuts are derived separately for each parameter). Cut values can be customised in the finder configuration file.

- *Neural network classifier:* We trained a multilayer perceptron (MLP) neural network (NN) on 50% of the available source sample to identify real and false sources using the three parameters as input variables.^k

Both classifiers were applied to the full set of detected sources, and completeness and reliability were computed on the selected data

^kWe are deliberately employing in this paper the simplest NN architecture possible (i.e. MLP with two hidden layers) trained with only three input parameters. In the future, we plan to increase performances by employing more advanced deep learning network architectures (e.g. convolutional NNs) working on the full image pixel data. Moreover, additional simulated maps are planned to be generated to provide a completely independent training sample with respect to the one currently used for testing.

sample. We reported the obtained results in Figure 3: blue diamonds are relative to the cut-based classifier; green triangles are obtained using the NN classifier. As can be seen, both classifiers allow to increase the detection reliability by ~ 10 – 15% at the cost of a moderate completeness degrade. The NN approach, working on a joint set of classification variables and providing a non-linear decision boundary, outperforms, as expected, the simpler cut analysis.

4.2.2. Position and flux accuracy

We report in Figure 6 the source position (left panel) and flux density (right panel) accuracy as a function of the source generated flux obtained over preselected source sample. Reconstruction bias is estimated using the sample median in each flux bin and reported in the top panels. Statistical resolution is estimated using the semi-interquartile range (SIQR) and reported in the bottom panels.

Ideal position resolutions in both coordinates are reported in the bottom left panel and given by [see (Condon 1997)]

$$IQR(x) = f \times \sqrt{\frac{2\sigma_x}{\pi\sigma_y} \frac{\sigma_{rms}}{A}} h,$$

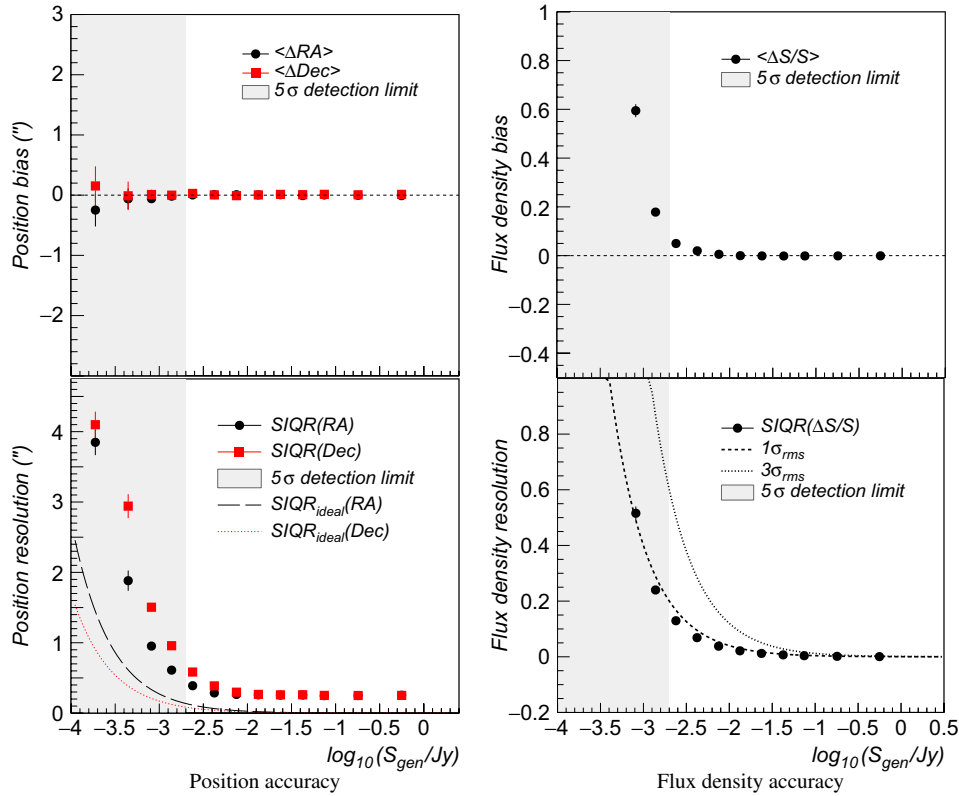


Figure 6. Left: Compact source position reconstruction bias (upper panel) and resolution (bottom panel) as a function of the source generated flux. Bias is estimated using sample median in each flux bin, while resolution is computed using the SIQR. Black dots and red squares indicate RA and Dec coordinates, respectively. Dashed and dotted lines denote the ideal resolution in both coordinates computed with expression 10 (see text). Right: Compact source flux density reconstruction bias (top panel) and resolution (bottom panel) as a function of the source generated flux. Dashed and dotted lines indicate the expected $1\sigma_{rms}$ and $3\sigma_{rms}$ flux density errors, respectively, with $\sigma_{rms} = 400\mu\text{Jy}$ rms noise level.

$$IQR(y) = f \times \sqrt{\frac{2\sigma_y}{\pi\sigma_x} \frac{\sigma_{rms}}{A}} h, \quad (10)$$

with $h = 1''$ map pixel scale size, A source peak flux, $\sigma_{x,y}$ source Gaussian sigma in the x and y directions ($b_{maj} \sim 13.3''$, $b_{min} \sim 8.4''$), $\sigma_{rms} = 400 \mu\text{Jy}$ image noise rms, and $f \sim 0.674$ factor to convert from Gaussian standard deviation to SIQR. Typical values are $\sim 0.2''$ at 5σ and $\sim 0.05''$ at 20σ source significance levels. The reconstructed position uncertainties above 5σ are found of the order of $0.4\text{--}0.5''$. No significant position bias is found even well below the source detection threshold.

Flux reconstruction presents a small positive bias ($\sim 5\text{--}10\%$) near the detection threshold. A similar trend was found also in other finders (Hopkins et al. 2015). Flux accuracy is found better than few per cents for bright sources and $\sim 10\%$ at the detection threshold.

4.3. Detection of extended sources

Extended source finding was run on the $N = 200$ simulated maps. A number of 3459 generated extended sources are available for this analysis. For this work, we considered the saliency filtering algorithm among those available in CAESAR. The algorithm steps were summarised in 2.1.3 and extensively described in Riggi et al. (2016). Algorithm parameters used in this analysis are reported in Table 2.

Sources tagged as extended or extended + compact were cross-matched to generated sources (convolved with the synthesis beam as described in Section 3) using overlap area and flux ratio parameters.

A generated source i is considered as ‘detected’ by a measured source j if

- $n_{i \cap j} / n_i > f_{thr}^{high}$,
- $n_{i \cap j} / n_j > f_{thr}^{high}$,

or, alternatively, if

- $n_{i \cap j} / n_i > f_{thr}^{low}$, $t_{thr}^{min} < S_{i \cap j} / S_i < t_{thr}^{max}$,
- $n_{i \cap j} / n_j > f_{thr}^{low}$, $t_{thr}^{min} < S_{i \cap j} / S_j < t_{thr}^{max}$,

where

- n_i : number of pixels in generated source i ,
- n_j : number of pixels in measured source j ,
- $n_{i \cap j}$: number of overlapping pixels between generated and detected sources,
- S_i : sum of pixel fluxes for generated source i ,
- S_j : sum of pixel fluxes for measured source j ,
- $S_{i \cap j}$: sum of pixel fluxes for measured source j , computed over pixels overlapping with generated source i .

f_{thr}^{high} , f_{thr}^{low} , t_{thr}^{min} , and t_{thr}^{max} are configurable thresholds, assumed equal to 60%, 10%, 80%, and 120%, respectively, in this work. The first condition imposes a large overlap area between generated and measured sources without any condition applied on their fluxes. The second condition, instead, requires a minimal overlap area plus a high match between fluxes.

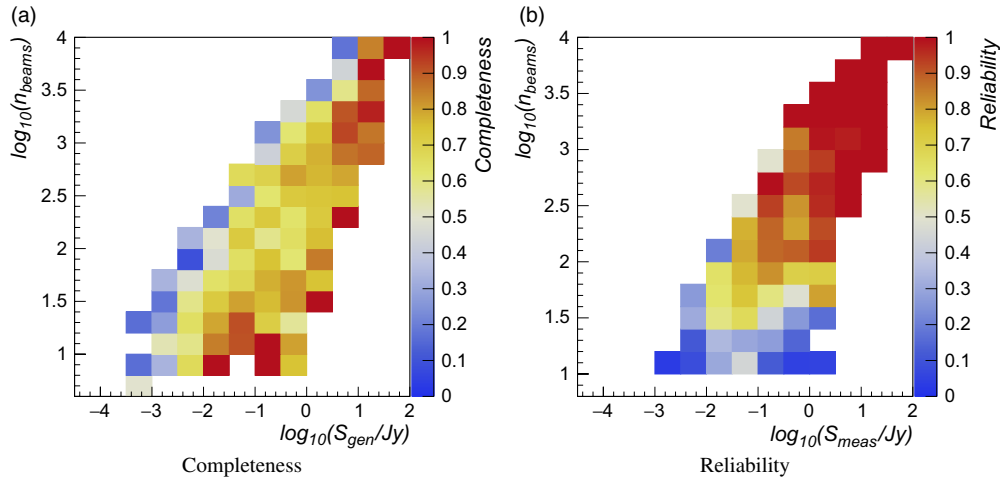


Figure 7. Left: Extended source detection efficiency as a function of the generated source flux density and n_{beams} (multiple of the synthesised beam size). Right: Extended source detection reliability as a function of the measured source flux density and n_{beams} .

Table 2. Extended source finder parameters.

Parameter	Value
<i>Compact source filter</i>	
$z_{thr,low}^{res}$	5
$z_{thr,high}^{res}$	10
Filter kernel size (#pix)	21
Removed sources	Point-like
<i>Smoothing filter</i>	
Filter model	Guided
Radius (# pix)	12
ε	0.04
<i>Saliency filter</i>	
spSize (# pix)	20
spBeta	1
spMinArea	10
saliencyResoMin (#pix)	20
saliencyResoMax (#pix)	60
saliencyResoStep (#pix)	20
saliencyNNFactor	1
saliencyThrFactor	2.5

4.3.1. Completeness and reliability

Similarly to what has been done for compact sources, we computed the completeness and reliability obtained for extended sources as a function of generated/measured source flux density and n_{beams} (multiple of the synthesised beam size). Results are reported in Figure 7. Completeness is on average 60–70% for fainter sources and $\sim 80\%$ for brightest sources. Reliability is found of the order of $\sim 70\%$ on average and above 90% for high flux densities. Detection efficiency slightly degrades for sources with size comparable with the minimum spatial scale assumed in the finding algorithm. A similar trend is observed for the largest

Table 3. Detection efficiency ε for different extended source types.

Source type	ε (%)
Ring	58
Disc + shell	81
Ellipse	82
Sérsic	61
Gaussian	69
Mixed	80

sources injected in the simulation. For a given flux density, this is due to their intrinsic smaller pixel detection significance.

Given the limited size of the simulated sample currently available, we are not able to disentangle the relative contributions of different simulated source types (ring, disc + shell, ellipse, Sérsic, Gaussian) in the above trends.

We therefore limit our report (see Table 3) to the detection efficiency obtained for different source classes irrespective of flux density and source size. Sources formed by a combination of different types have been labelled as ‘mixed’. Sources of a given pure class having point-like sources inside were still considered as belonging to the same class.

These results suggest that ring-shaped sources and sources with tailed flux profiles (Sérsic, Gaussian) are harder to be identified with respect to other types.

As expected, the detection performances are not at the same level of point sources. Nevertheless, as we have shown in Riggi et al. (2016) with real interferometric data, the outcomes would have been considerably worse or even close to a null detection efficiency if we had used the same algorithm used for compact source.

4.3.2. Flux accuracy

In Figure 8, we report the flux accuracy (bias and resolution) obtained for the detected extended sources. Flux density was computed using the sum of pixel fluxes divided by the beam area. A flux resolution below 10% was obtained on the selected source sample. No significant biases were found.

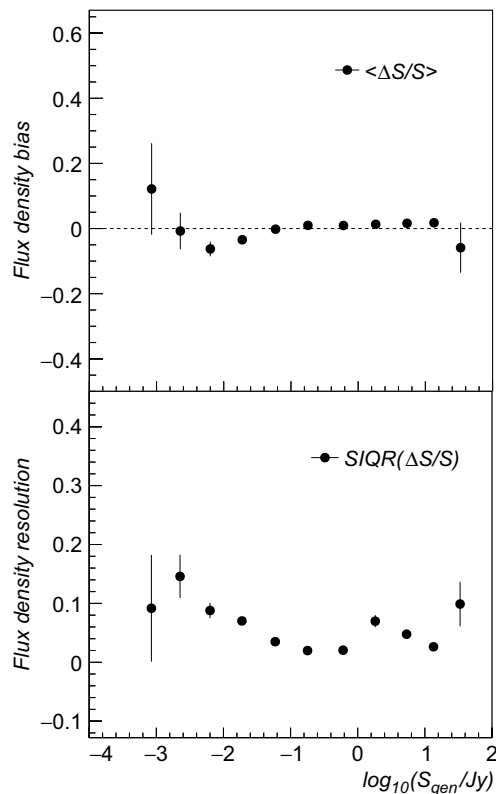


Figure 8. Extended source flux density reconstruction bias (top panel) and resolution (bottom panel) as a function of the source generated flux. Bias is estimated using sample median in each flux bin, while resolution is computed using the SIQR.

5. Computational performance

As discussed in Section 2, CAESAR was improved to support parallel processing. An hybrid programming model with two levels of parallelism was adopted. The outer MPI-based level enables to distribute source finding over image tiles on multiple processors in the available computing nodes. The inner OpenMP-based level distributes source finding tasks for a single image tile across multiple threads.

To validate the current implementation and estimate the achieved performances, we measured the computing times of compact source finding on large simulated images over this computing infrastructure:

- two computing nodes connected through a 10 Gbit network link,
- 4 sockets \times 10 Core Intel(R) Xeon(R) CPU E5-4627 2.60 GHz per node,
- 256 GB DDR4 2133 MHz memory per node.

In Figure 9 (left panel), we report the obtained speed-up as a function of the number of threads allocated by OpenMP for different source finding stages and overall (shown in black) over a simulated image of size $10\,000 \times 10\,000$ pixels. In the performed runs, the MPI processing was switched off and the image was not partitioned in tiles. We also imposed thread affinity on the basis of the Non-Uniform Memory Access (NUMA) architecture reported by the two computing nodes; for example, we bound threads to run on the same socket if fitting the available number of cores per socket. As can be seen, a computational speed-up ~ 3 – 4 is achieved overall up to a moderate number of threads (4–6), above which no significant improvement is observed for most

of the tasks. Some tasks (e.g. blob finding, background calculation, and source fitting) exhibit a better scalability (up to ~ 10 allocated threads) due to their embarrassingly parallel nature and implementation. Others (e.g. image statistic calculation and blob masking) are rather flat in speed-up, either because dominated by serial parts or because affected by thread management overhead (creation, synchronisation, etc.).

In Figure 9 (right panel), we report the percentage of total CPU time spent in different finding stage for two representative number of allocated threads: 1 (red histogram), 4 (blue histogram). A CPU time of $\sim 1.3/2.6$ h was spent in total with/without splitting the input image into tiles, improving by a factor of 3–4 using 4–6 threads. As expected, the largest contribution is due to source finding and fitting stages. Local background and rms map calculation contributes to less than 10% of the total CPU time. Image reading and computation of statistical estimators contribute to less than 1%.

In Figure 10, we report the speed-up obtained on a sample simulated image of size $32\,000 \times 32\,000$ pixels as a function of the number of MPI processes used. Runs were performed on an NFS file system mounted on both nodes. Input image was split into tiles of size $4\,000 \times 4\,000$ pixels. Red squares and black dots correspond to runs performed with 1 and 4 OpenMP threads per MPI process, respectively. Green triangles refer to the speed-up obtained using $n_{threads} = 4$ per MPI process, with all four threads running on the same computing core rather than in a dedicated core. As can be seen, a good speed-up is found using one single OpenMP thread. The speed-up with four OpenMP threads is superlinear up to ~ 8 processes, above which the effect of inter-process communication and data serialisation becomes dominant. This is expected given that the load is partitioned over a larger number of cores, with respect to the case $n_{threads} = 1$. However, when running OpenMP threads in a single core (green triangles) rather than in a separate one (black dots), the obtained speed-up is comparable to the single thread speed-up (red squares).

The performance degrade due to the network file system and log activity was investigated by comparing the computing times obtained when running on a local file system and disabling logging. We observed an increase of $\sim 5\%$ in the total computing time in the NFS file system. Tests will be performed in the future to evaluate the benefits of using a parallel file system such as Lustre^l or BeeGFS.^m Logging was also found to negatively impact performances with an increase of $\sim 40\%$ in the total computing time.

6. Summary and outlooks

We have presented in the paper the current status of CAESAR source finder. Considerable improvements were done since the first reference paper (Riggi et al. 2016), among them distributed source processing and algorithm improvements on compact sources.

We reported the performances achieved on both compact and extended source detection using simulated data. Results obtained on compact sources are comparable to similar analysis reported in the literature (Westmeier 2012; Hopkins et al. 2015), despite the presence of background emission from extended sources (typically not included in other analysis). We discussed also possible methods to discover and remove false source detections from the final catalogue.

To the best of our knowledge, this paper reports also a first attempt to systematically test extraction of extended sources with

^l<http://lustre.org/>

^m<https://www.beegfs.io/content/>

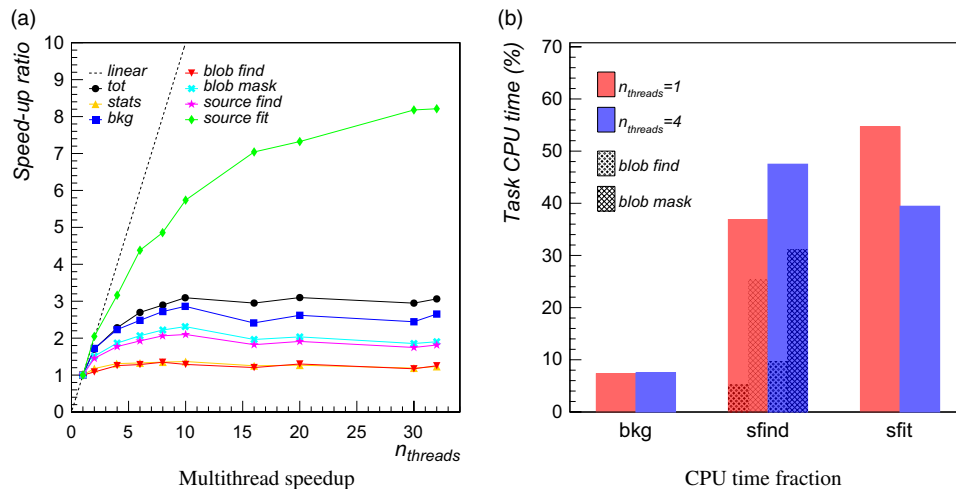


Figure 9. Left: Computational speed-up of multithreaded compact source finding over a 10 000 × 10 000 pixel map as a function of the number of allocated threads (black line) compared with the ideal speed-up (black dashed line). Coloured lines indicate the speed-up obtained on different tasks: image statistic calculation (orange line), image background calculation (blue line), source finding (purple line), and source fitting (green line). Source finding is further decomposed in two subtasks: blob finding (red line) and blob mask (light blue line). Right: Fraction of the total CPU time spent in different source finding tasks with $n_{threads} = 1$ (red histogram) and $n_{threads} = 4$ (blue histogram).

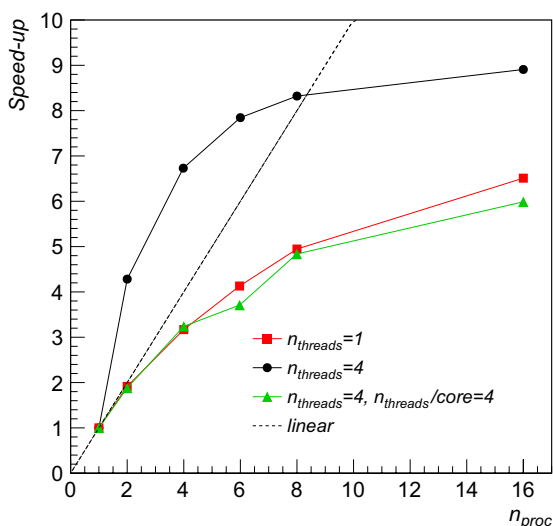


Figure 10. Computational speed-up of compact source finding in MPI+OpenMP runs over a 32 000 × 32 000 pixel simulated map as a function of the number of allocated MPI processes using $n_{threads} = 1$ (red squares) and $n_{threads} = 4$ (black dots) per MPI process. Green triangles refer to the speed-up obtained using $n_{threads} = 4$ per MPI process, with all four threads running on the same computing core rather than in a dedicated core.

different shapes, other than the standard Gaussian model used in other analysis. The overall performance achieved by the extended finder algorithm tested in this paper does not compete yet with those obtained by compact finder algorithms on point sources. Nevertheless, when considering the complete sample of sources (compact plus extended) present in the observed field, the results are encouraging since the combination of different algorithms in CAESAR allows to recover a significant fraction of sources that would have been undetected if only using the compact source finder.

Despite the progress made, there is still room to extend and improve CAESAR both at the code and algorithmic level. For future releases, we foresee additional refinements and optimisations in the code to improve memory usage, scalability, and

fault tolerance of the parallel implementation. In a shorter time scale, scalability can be slightly improved by exploiting parallelism on selected tasks of the pipeline that are still serially performed, particularly in extended source extraction. In a longer term, following the current trends in exascale computing, we expect a potential boost in performances if additional developments will be made to fully exploit new generations of HPC systems equipped with high-capacity memories and one or more accelerators (GPUs or FPGAs) per node.

The obtained results highlighted that additional efforts are to be spent to improve source finding performances in view of future large area surveys. For compact sources, we expect that improving the deblending stage and the spurious source identification will be the major area of investigation using deep NNs trained to identify real and false components in extracted sources. Extended source finding will instead require different and more refined algorithms to be tested. For this purpose, additional test campaigns are planned to be performed using the algorithms already implemented in CAESAR.

Acknowledgements. We acknowledge the computing centre of INAF - Osservatorio Astrofisico di Catania, under the coordination of the CHIPP project, for the availability of computing resources and support. The research leading to these results has received funding from the European Commissions Horizon 2020 research and innovation programme under the grant agreement No. 731016 (AENEAS). We thank the authors of the following software tools and libraries that have been extensively used for data analysis and visualisation: CASA, astropy, ROOT, ds9, and APLpy. The authors thank in particular the anonymous referee for helpful comments that led to the improvement of this paper.

References

- Bradski, G. 2000, Dr. Dobb's Journal of Software Tools, record: citeulike:2236121. See also <http://opencv.org/>
- Brun, R., & Rademakers, F. 1997, in Nucl. Inst. and Meth. in Phys. Res. A 389, Proc. of the AIHENP'96 Workshop, Lausanne, Switzerland, p. 81. See also <http://root.cern.ch/>, doi:10.1016/S0168-9002(97)00048-X
- Carbone, D., et al. 2018, Astronomy and Computing, 23, 92, doi:10.1016/j.ascom.2018.02.003
- Chan, T., & Vese, L. 2001, IEEE Transaction on Image Processing, 10, 266, doi:10.1109/83.902291

- Condon, J. J. 1997, *PASP*, 109, 166, doi:10.1086/133871
- Hales, C. A., et al. 2012, *MNRAS*, 425, 979, doi:10.1111/j.1365-2966.2012.21373.x
- Hancock, P. J., et al. 2012, *MNRAS*, 422, 1812, doi:10.1111/j.1365-2966.2012.20768.x
- Hancock, P. J., et al. 2018, *PASA*, 35, E011, doi:10.1017/pasa.2018.3
- Hatlo, M., et al. 2005, *IEEE Trans. Nucl. Sci.*, 52, 2818, doi:10.1109/TNS.2005.860152
- Hopkins, A. M., et al. 2015, *PASA*, 32, E037, doi:10.1017/pasa.2015.37
- Johnston-Hollitt, M., et al. 2016, SKA-TEL-SDP-0000031
- James, F. 1972, CERN Program Library Long Writeup D506
- Lankton, S., & Tannenbaum A. 2008, *IEEE Transaction on Image Processing*, 17, 2029, doi:10.1109/TIP.2008.2004611
- McMullin, J. P., et al. 2007, *Astronomical Data Analysis Software and Systems XVI (ASP Conf. Ser. 376)*, ed. R. A. Shaw, F. Hill, & D. J. Bell (San Francisco, CA: ASP), 127
- Mohan, N., & Rafferty, D. 2015, *Astrophysics Source Code Library*, record ascl:1502.007
- Norris, R. P., et al. 2011, *PASA*, 28, 215, doi:10.1071/AS11021
- Peracaula, M., et al. 2011, *Proc. of the 18th IEEE International Conference on Image Processing (ICIP)*, Brussels, Belgium, 2805, doi:10.1109/ICIP.2011.6116254
- Riggi, S., et al. 2016, *MNRAS*, 460, 1486, doi:10.1093/mnras/stw982
- Umana, G., et al. 2015, *MNRAS*, 454, 902, doi:10.1093/mnras/stv1976
- Westmeier, T. 2012, *PASA*, 29, 276, doi:10.1071/AS11041

Appendix A. Source deblending and fit initialisation

The number of components to be fitted for a detected source is set to the number of nested blobs, eventually ordered and selected by significance level and limited to a maximum number (5, for example). Starting values for component fit parameters are determined from blob moments.

If no nested blobs are present in the source, the number of components and relative starting parameters are estimated with the following algorithm:

1. Compute blob masks at different configurable scales (usually 1 to 3 times the beam size). A blob mask is obtained by thresholding the source image convolved by an LoG kernel at a given scale.
2. Find peaks in blob masks with a dilation filter using different kernel sizes (3, 5, 7 pixels by default).
3. Reject peaks below desired flux significance level.
4. Compare surviving peaks found at different scales. If multiple peaks match within a tolerance (1–2 pixels usually), consider the one with the largest intensity and select the blob optimal scale.
5. Set number of estimated components to selected peaks, again ordered by significance level and limited up to a maximum number.
6. Set initial fit component centroid and amplitude to the peak position and flux, respectively.
7. Estimate fit component shape from the previously computed masks at optimal blob scale using a Watershed segmentation algorithm seeded to the detected peak. Compute initial fit component sigma and position angle parameters from segmented blob moments. Fall back to beam parameters if segmentation fails.

The starting offset parameter can be either specified by the user from the configuration file or determined from the map, for example, set to the estimated background averaged over source pixels or computed in a box centred on the source. If desired, the offset parameter can be included as a free parameter in the fit. By default, however, it is kept fixed as pixel data included in the fit (down to 2.5σ significance) do not allow the possibility to fully constrain it.