Publications of the Astronomical Society of Australia (2023), **40**, e028, 24 pages
doi:10.1017/pasa.2023.24

CAMBRIDGE UNIVERSITY PRESS

Research Article

Hydra I: An extensible multi-source-finder comparison and cataloguing tool

M. M. Boyce[1], A. M. Hopkins[2], S. Riggi[3], L. Rudnick[4], M. Ramsay[1], C. L. Hale[5], J. Marvil[6], M. T. Whiting[7], P. Venkataraman[8], C. P. O'Dea[1], S. A. Baum[1], Y. A. Gordon[1,9], A. N. Vantyghem[1], M. Dionyssiou[8], H. Andernach[10], J. D. Collier[11,12], J. English[1], B. S. Koribalski[12,13], D. Leahy[14], M. J. Michałowski[15], S. Safi-Harb[1], M. Vaccari[11,16,17], E. L. Alexander[18], M. Cowley[19,20], A. D. Kapinska[6], A. S. G. Robotham[21] and H. Tang[22]

[1]Department of Physics and Astronomy, University of Manitoba, Winnipeg, Canada, [2]Australian Astronomical Optics, Macquarie University, North Ryde, NSW, Australia, [3]INAF, Osservatorio Astrofisico di Catania, Catania, Italy, [4]Minnesota Institute for Astrophysics, School of Physics and Astronomy, University of Minnesota, Minneapolis, MN, USA, [5]School of Physics and Astronomy, Institute for Astronomy, Royal Observatory, University of Edinburgh, Blackford Hill, Edinburgh EH9 3HJ, UK, [6]National Radio Astronomy Observatory, Socorro, NM, USA, [7]CSIRO Space & Astronomy, PO Box 76 Epping, NSW 1710, Australia, [8]Dunlap Institute for Astronomy and Astrophysics, University of Toronto, Toronto, ON, Canada, [9]Department of Physics, University of Wisconsin-Madison, Madison, WI, USA, [10]Departamento de Astronomía, DCNE, Universidad de Guanajuato, Guanjuato, CP, GTO, Mexico, [11]Inter-University Institute for Data Intensive Astronomy (IDIA), Department of Astronomy, University of Cape Town, Rondebosch, South Africa, [12]School of Science, Western Sydney University, Penrith, NSW, Australia, [13]Australia Telescope National Facility, CSIRO Astronomy and Space Science, Epping, NSW, Australia, [14]Department of Physics and Astronomy, University of Calgary, Calgary, Canada, [15]Astronomical Observatory Institute, Faculty of Physics, Adam Mickiewicz University, Poznań, Poland, [16]Department of Physics and Astronomy, Inter-University Institute for Data Intensive Astronomy (IDIA), University of the Western Cape, Bellville, Cape Town, South Africa, [17]INAF - Istituto di Radioastronomia, Bologna, Italy, [18]Jodrell Bank Centre for Astrophysics, Department of Physics and Astronomy, University of Manchester, Manchester, UK, [19]School of Chemistry and Physics, Queensland University of Technology, Brisbane, QLD, Australia, [20]Centre for Astrophysics, University of Southern Queensland, West Street, Toowoomba, QLD 4350, Australia, [21]ICRAR, M468, University of Western Australia, Crawley, WA 6009, Australia and [22]Department of Astronomy, Tsinghua University, Beijing 100084, China

Abstract

The latest generation of radio surveys are now producing sky survey images containing many millions of radio sources. In this context it is highly desirable to understand the performance of radio image source finder (SF) software and to identify an approach that optimises source detection capabilities. We have created Hydra to be an extensible multi-SF and cataloguing tool that can be used to compare and evaluate different SFs. Hydra, which currently includes the SFs Aegean, Caesar, ProFound, PyBDSF, and Selavy, provides for the addition of new SFs through containerisation and configuration files. The SF input RMS noise and island parameters are optimised to a 90% 'percentage real detections' threshold (calculated from the difference between detections in the real and inverted images), to enable comparison between SFs. Hydra provides completeness and reliability diagnostics through observed-deep ($\mathcal{D}$) and generated-shallow ($\mathcal{S}$) images, as well as other statistics. In addition, it has a visual inspection tool for comparing residual images through various selection filters, such as S/N bins in completeness or reliability. The tool allows the user to easily compare and evaluate different SFs in order to choose their desired SF, or a combination thereof. This paper is part one of a two part series. In this paper we introduce the Hydra software suite and validate its $\mathcal{D}/\mathcal{S}$ metrics using simulated data. The companion paper demonstrates the utility of Hydra by comparing the performance of SFs using both simulated and real images.

Keywords: methods: data analysis – radio continuum: general – techniques: image processing

(Received 15 October 2021; revised 30 January 2023; accepted 26 April 2023)

1. Introduction

With the advent of new facilities, radio surveys are becoming larger and deeper, providing fields rich in sources, in the tens of millions (Norris 2017), and delivering data at increasing rates, in the hundreds of gigabytes per second (Whiting & Humphreys 2012). The Evolutionary Map of the Universe (EMU, Norris et al. 2011, 2021) is expected to detect up to 40 million sources, expanding our knowledge in areas such as galaxy evolution and star formation. This outstrips surveys like the Karl G. Jansky Very Large Array (JVLA, or VLA) Sky Survey (VLASS, Lacy et al. 2020; Gordon et al. 2021) and the Rapid Australian Square Kilometre Array (SKA) Pathfinder (ASKAP) Continuum Survey (RACS, McConnell et al. 2020; Hale et al. 2021) by a factor of up to 30. Furthermore, the Variable and Slow Transients (VAST, Banyer et al., 2012; Murphy et al., 2013, 2021) survey, operating at a cadence of 5*s*, surpasses VLASS transient studies by several orders of magnitude, opening up areas of variable and transient research: e.g., flare stars, intermittent pulsars, X-ray binaries, magnetars,

Corresponding author: M. M. Boyce; Email: michelle.boyce2@umanitoba.ca.

Cite this article: Boyce MM, Hopkins AM, Riggi S, Rudnick L, Ramsay M, Hale CL, Marvil J, Whiting MT, Venkataraman P, O'Dea CP, Baum SA, Gordon YA, Vantyghem AN, Dionyssiou M, Andernach H, Collier JD, English J, Koribalski BS, Leahy D, Michałowski MJ, Safi-Harb S, Vaccari M, Alexander EL, Cowley M, Kapinska AD, Robotham ASG and Tang H. (2023) Hydra I: An extensible multi-source-finder comparison and cataloguing tool. *Publications of the Astronomical Society of Australia* **40**, e028, 1–24. https://doi.org/10.1017/pasa.2023.24

extreme scattering events, interstellar scintillation, radio supernovae, and orphan afterglows of gamma-ray bursts (Murphy et al., 2013, 2021). This places complex requirements on source finder (SF) software in order to reliably handle compact,[a] extended, complex, and faint or diffuse sources, along with demands for high data throughput for radio transients (*e.g.*, Hancock et al., 2012; Hopkins et al., 2015; Riggi et al., 2016; Hale et al., 2019; Boyce, 2020; Bonaldi et al., 2021). No current SF fits all of these requirements.

Hydra is an attempt to get the best of all worlds: it is an extensible multi-SF comparison and cataloguing tool, which allows users to choose the appropriate SF for a given survey, or take advantage of its collectively rich statistics by combining results. The Hydra software suite[b] currently includes Aegean (Hancock et al. 2012; Hancock, Cathryn, & Hurley-Walker 2018), Caesar (Compact And Extend Source Automated Recognition, Riggi et al. 2016, 2019), ProFound (Robotham et al. 2018; Hale et al. 2019), PyBDSF (Python Blob Detector and Source Finder, Mohan & Rafferty 2015), and Selavy (Whiting & Humphreys 2012).

This paper is part one of a two part series, (referred to hereafter as Papers I and II). Here we provide a brief overview of SFs, relevant to our implementation of Hydra. This is then followed by a description of the Hydra software suite. The software produces new metrics for handling real source components (or sources, herein), such as, completeness ($\mathcal{C}$) and reliability ($\mathcal{R}$), based on sources detected in a shallow ($\mathcal{S}$) image (e.g., a real image with artificial noise added) wherein real (sometimes referred to as 'deep' or $\mathcal{D}$) image detections are considered as true sources. We use simulated data, where the true sources are known, to validate these metrics. In Paper II we use the simulated images along with real data to evaluate the performance of the six different SFs included with Hydra. A preliminary discussion on SF performance is presented in this paper.

2. Source finders

The growing sizes and data rates of modern radio surveys have increased the need for automated source finding tools with fast processing speeds, and high completeness and reliability. One impetus for this came through the ASKAP EMU source finding data challenge (Hopkins et al. 2015), which explored a community-submitted set of eleven SFs: Aegean (Hancock et al. 2012), Astronomical Point source EXtractor (APEX, Makovoz & Marleau 2005), BLOBCAT (Hales et al. 2012), Curvature Threshold Extractor (CuTEx, Molinari et al. 2011), Duchamp (Whiting 2012), IFCA (International Federation of Automation Control) Biparametric Adaptive Filter (BAF, López-Caniego & Vielva 2012)/Matched Filter (MF, López-Caniego et al. 2006), PyBDSF (Mohan & Rafferty 2015), Python Source Extractor (PySE, Spreeuw 2010; Swinbank et al. 2015), Search and Destroy (SAD, Condon et al. 1998, with an honourable mention of its variant HAPPY, White et al. 1997), Selavy (with Duchamp at its core, Whiting & Humphreys 2012), Source Extractor (SExtractor, Bertin & Arnouts 1996), and SOURCE_FIND (Arcminute Microkelvin Imager (AMI) pipeline, AMI Consortium: Franzen et al. 2011). More recent SFs include Caesar (Riggi et al. 2016, 2019) and ProFound (Robotham et al. 2018; Hale et al. 2019; Boyce 2020). APEX, CuTEx, ProFound, and SExtractor have their origins in optical astronomy. Our focus will be on 2D SFs, such as those above, although there are also 3D packages like SoFiA (Source Finding Application, Serra et al. 2015; Koribalski et al. 2020; Westmeier et al. 2021) optimised for detecting line emission in data cubes, which can also function as 2D SFs.

Table 1. SF general design characteristics (re., Hopkins et al. 2015; Hale et al. 2019; Bonaldi et al. 2021). NxGen indicates multiprocessing capabilities.

SF	Source type			NxGen
	Compact	extended	diffuse	
Aegean	√			√
APEX[a]	√			
BLOBCAT	√	√		
Caesar	√	√	√	√
CuTEx[a]	√			
IFCA BAF/MF		√		
Selavy	√			√
ProFound[a]	√	√	√	
PyBDSF	√			
PySE	√			
SAD	√			
SExtractor[a]		√		
SOURCE_FIND		√		

[a]Optical SF.

By and large there is no 'one SF fits all' solution. Each is typically optimised for specific tasks (Hopkins et al. 2015; Hale et al. 2019; Bonaldi et al. 2021). In the broadest sense, there are SFs designed to handle sources that are compact, or extended and diffuse, see Table 1. They also have their specialisations: for example, BLOBCAT for linear polarisation data (Hales et al. 2012), Duchamp for HI observations (Whiting 2012), CuTEx for images with intense background fluctuations (Molinari et al. 2011), and PySE for transients (Fender et al. 2007; van Haarlem et al. 2013). There are also *'Next Generation'* (NxGen) SFs (see Table 1), which utilise multiple processors for handling high data throughput (Hancock et al. 2012; Riggi et al. 2016; Whiting & Humphreys 2012). Qualitatively different types of source finding and characterisation tools are being developed that use machine learning approaches (e.g., Bonaldi et al. 2021; Lao et al. 2021; D. Magro et al., in preparation), as well as citizen science approaches to classifying radio sources (e.g., Banfield et al. 2015; Alger et al. 2018), although it is beyond the scope of Hydra to attempt to incorporate all such efforts.

In general, SFs typically analyse an image in 3 stages: (1) background and noise estimation, (2) island detection, and (3) component modelling.

For the background estimation most SFs used in radio astronomy such as Aegean, PyBDSF, and Selavy, tend to use a sliding box method, where background noise estimates are calculated at a specific location using neighbouring pixels within a given box size, and estimated again for adjacent locations based on the sliding-step size. It is important that the box size be set so as not to be too small around bright sources, which would overestimate the background noise, or too large, so as to wash out any varying background structure that is important for reliable detection of faint

[a]Herein, compact refers to point sources.

[b]Hydra is available, along with the data products presented in this paper, by navigating through the CIRADA portal at https://cirada.ca. A more permanent home is expected, once VLASS data product development has been completed.

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

sources (e.g., Huynh et al 2012). This is discussed further below in Section 3.1.3.

Background noise estimation can be performed through various metrics such as the inter-quartile range (IQR) used by Aegean (with median background and IQR noise spread, Hancock et al. 2012), mean background (μ) and RMS noise (σ) used by PyBDSF (Mohan & Rafferty 2015), or median background and Mean Absolute Deviation From the Median (MAD, or MADFM herein: e.g., Riggi et al. 2016; Hopkins et al. 2015) noise used by Selavy (which also has a μ/σ option, Whiting & Humphreys 2012). SExtractor, on the other hand, uses $\kappa.\sigma$-clipping and mode estimation (see Section 3.1.1; Da Costa 1992; Bertin & Arnouts 1996; Huynh et al. 2012; Akhlaghi & Ichikawa 2015; Riggi et al. 2016) over the entire image; while PySE performs σ-clipping locally (see Hopkins et al. 2015). ProFound, an optical SF shown to be useful for radio images (Hale et al. 2019), also uses a σ-clipping schema (via. the MAKESKYGRID routine, Robotham et al. 2018) Caesar provides several options: μ/σ, median/MADFM, biweight and σ-clipped estimators (Riggi et al. 2016). The final stage typically involves bicubic interpolation to obtain the background noise estimates as a function of pixel location. It is important that these estimates are optimal as they have a significant effect on SF performance (Huynh et al. 2012).

There are various methods for island detection within an image. Perhaps the simplest is thresholding, in which the pixel with the highest flux is chosen along with neighbouring pixels down to some threshold above the background noise, defining an island. Variants of this method are used by Duchamp (Whiting & Humphreys 2012), ProFound (Robotham et al. 2018), Selavy (Whiting 2012), and SExtractor (Bertin & Arnouts 1996). Once the initial set of islands are chosen, they are sometimes then grown down to a lower threshold according to certain rules. For instance, ProFound uses a Kron/Petrosian-like dilation kernel, that is, it uses an island-shaped aperture (Kron 1980; Petrosian 1976) to grow the islands according to a surface brightness profile, in an iterative process, until the desired profile or lower threshold limit is reached (Robotham et al. 2018). It then separates out the islands into segments, through a watershed deblending technique.[c] Another method is flood-fill, wherein islands are seeded above some threshold and then grown down to a lower threshold, according to a set of rules. Aegean (Hancock et al. 2012), BLOBCAT (Hales et al. 2012), Caesar (Riggi et al. 2016), PyBDSF (Mohan & Rafferty 2015), and PySE (Swinbank et al. 2015) use variations on this theme.

The component extraction phase is perhaps the most varied in terms of modelling. The simplest is the top down raster-scan within an island to find flux peaks given some step size, or tolerance level. This method is utilised by Duchamp (Whiting & Humphreys 2012), and, in turn, is also employed by Selavy. These peaks are then fitted by elliptical Gaussians producing a component catalogue. The choice of elliptical Gaussians is motivated by the fact that point sources, or sources that are only very slightly extended, are well-modelled in this way as it corresponds to the shape of the telescope's synthesised beam. More complex source structures, on the other hand, tend to be poorly fit by this choice, leading to variations in fitting approach. Some SFs, for example, use multiple Gaussians to fit to an island, using various criteria. PyBDSF (Mohan & Rafferty 2015) and PySE (Spreeuw 2010; Swinbank et al. 2015) fall into this category.

There are also a class of SFs that use curvature maps to determine radio source components. Aegean searches for local maxima within an island which in turn are fitted by Gaussians, constrained by negative curvature valleys (Hancock et al. 2012). Caesar is rather unique in that it first searches for peaks and then uses watershed deblending to create sub-islands, for seeding and constraining Gaussian fits, respectively (Riggi et al. 2016). Extended sources are then extracted from the resulting residual image, using wavelet transform, saliency, hierarchical-clustering, or active-contour filtering. Consequently, Caesar is capable of extracting extended sources with complex structure.

The aforementioned SF algorithms are just the tip of the iceberg of possibilities (c.f., Hancock et al. 2012; Hopkins et al. 2015; Wu et al. 2018; Lukic, de Gasperin, & Brüggen 2019; Sadr et al. 2019; Bonaldi et al. 2021; D. Magro et al., in preparation). In our initial implementation of Hydra we have chosen to explore a representative set of commonly used SFs: Aegean, Caesar, ProFound, PyBDSF, and Selavy

3. Hydra

Hydra is a software tool capable of running multiple SFs. It is extensible, in that other SFs can be added in a containerised fashion by following a set of straightforward template-like rules. It provides diagnostic information such as $\mathcal{C}$ and $\mathcal{R}$. Statistical analysis can be based on injected ($\mathcal{J}$) source catalogues from simulated images or on real $\mathcal{D}$-images used as ground truths for detections in their $\mathcal{S}$ counterparts. Hydra is innovative in that it minimises the False Detection Rate (FDR, Whiting 2012) of the SFs by adjusting their detection threshold and island growth (and optionally RMS box) parameters. This is an essential step in automation, especially when dealing with large surveys such as EMU (Norris 2017; Norris et al. 2021).

3.1. The hydra software suite

> *Upon Heracles shield wrought Homados (Tumult), the din of battle noise, and riding alongside Cerberus, the unruly master of mayhem. Only the wrath of Cerberus's father Typhon, a controlling force, can temper their chaos. And hitherto, Typhon's son Hydra, was tasked with bringing the chaos to bear fruit, while his mother Echidna, a hidden force, plucked the fruit from the vines to make wine. (Inspired from Powell, 2017, and Buxton, 2016.)*

Fig. 1 shows an overview of the Hydra software suite, which consists of the following software components: Homados, Cerberus, Typhon, and Hydra. Homados is used for providing image statistics such as μ/σ, and image manipulation such as inversion and adding noise. Cerberus is an extensible multi-SF software suite. It currently includes Aegean (Hancock et al. 2012, 2018), Caesar (Riggi et al. 2016, 2019), ProFound (Robotham et al. 2018; Hale et al. 2019), PyBDSF (Mohan & Rafferty 2015), and Selavy (Whiting & Humphreys 2012). Typhon is a tool for optimising the SF parameters and then producing output catalogues. It uses Homados and Cerberus to do this task. Hydra is the main tool which uses Typhon to produce data products, including

[c]The term 'watershed' refers to drainage basins formed from streams running between mountains (islands), during a rainfall, following the steepest descent (Beucher & Lantuejoul 1979).

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

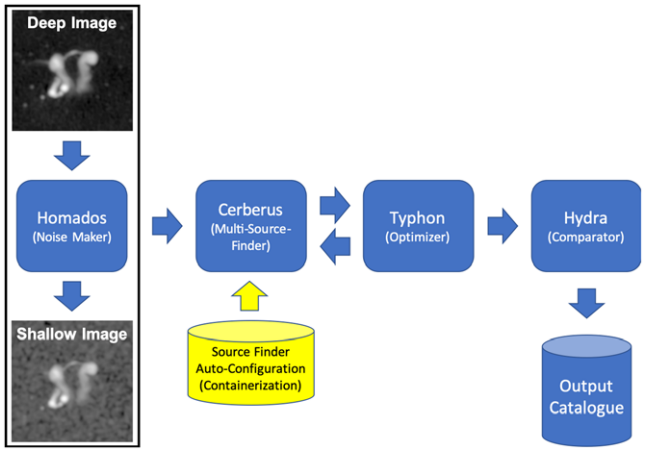


Figure 1. High level schematic representation of the Hydra software suite workflow. Homados provides $\mathcal{D}$ and $\mathcal{S}$-image channels for simulated/real images ('dancing ghosts' example image, see Norris et al. 2021). Each channel is run separately through the Typhon optimiser, which uses the SF interface provided by Cerberus. Hydra coordinates all of these activities, building catalogues and compiling statistics at the end of the process.

catalogues, residual images, and region files.[d] Echidna is a planned catalogue stacking and integration tool, to be added to Hydra.

3.1.1. Homados

The main purpose of Homados is to add noise to images. We shall often refer to the original image, as the $\mathcal{D}$-image, and the noise-added image, as the $\mathcal{S}$-image. These 'deep-shallow' ($\mathcal{DS}$) image pairs can be used to create statistics such as $\mathcal{DS}$-completeness ($\mathcal{C}_{\mathcal{DS}}$) and $\mathcal{DS}$-reliability ($\mathcal{R}_{\mathcal{DS}}$), based on the assumption that the sources detected in the $\mathcal{D}$-image are real. These statistics are used for real images, where the source inputs are unknown.

An $\mathcal{S}$-image is created by adding to the $\mathcal{D}$-image a Gaussian noise map that has been convolved with the corresponding synthesised beam (i.e., `BMIN`, `BMAJ`, and `BPA`). The noise map is created with mean noise, μ_{image}, and RMS noise, $n\sigma_{image}$ ($\equiv \sigma_{noise\ map}$), where n is the desired noise level (i.e., factor), and μ_{image} and σ_{image} are obtained from the $\mathcal{D}$-image using σ-clipping (Akhlaghi & Ichikawa 2015). This is then convolved with the synthesised beam, from which its RMS noise, $\sigma_{convolved}$, is computed. For convergence, this process is repeated using the convolved image as input, but with n replaced by $\sigma_{noise\ map}\ n/\sigma_{convolved}$. The final convolved image is then added to the $\mathcal{D}$-image, obtaining the $\mathcal{S}$-image.

Fig. 2 shows an example of an $\mathcal{S}$-image generated by Homados from an Australia Telescope Large Area Survey (ATLAS) Chandra Deep Field South (CDFS) Data Release 1 (DR1) tile (Norris et al. 2006). The noise level was scaled by a factor of $n = 5$. This factor is assumed for the $\mathcal{S}$-image generation in the rest of this paper.

In addition, Homados uses σ-clipping to compute image statistics, such as, m (median), μ (mean), σ (RMS), $I_{\rm min}$ (minimum pixel value), and $I_{\rm max}$ (maximum pixel value). It also does image inversion for FDR calculations.

[d]Hydra refers to both the software suite and the software tool `hydra.py`. There should be no source of confusion in this regard, as only `hydra.py` is used to create the data products.

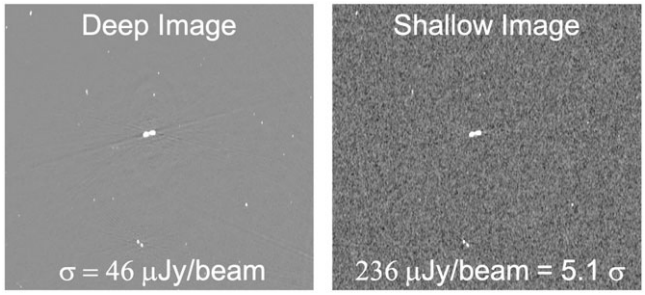


Figure 2. Homados $\mathcal{S}$-image generation example, using an image cutout sample from an ATLAS CDFS DR1 2.2° × 2.7° tile (Norris et al. 2006). The figures show $\mathcal{D}$ (left) and $\mathcal{S}$ (right) images, zoomed in. The noise level scale factor, n, was set to 5 to generate the shallow image.

3.1.2. Cerberus

Cerberus is an extensible interface for running SF modules within the Hydra software suite. It currently supports Aegean, Caesar, ProFound, PyBDSF, and Selavy, as indicated by its command-line interface.[e]

```
$ python cerberus.py --help
Usage: cerberus.py [OPTIONS] COMMAND [ARGS]...

  Runs a collection of source finder modules.

Options:
  --help  Show this message and exit.

Commands:

  process   Use multiple source finders.
  aegean    Use aegean source finder.
  caesar    Use caesar source finder.
  profound  Use profound source finder.
  pybdsf    Use pybdsf source finder.
  selavy    Use selavy source finder.
$
```

New modules are added through code generation, using Jinja template-code[f] in conjunction with Docker[g] and YAML[h] configuration files. The workflow is as follows:

- Create a containerised SF wrapper:
  - Create a SF wrapper script
  - Create a Docker build file wrapper
  - Update the master docker-compose build file
  - Build the container image

[e]The command-line interface for all Hydra tools is standardised using Click, https://click.palletsprojects.com. Click allows direct interface calls within a script, through its `standalone_mode` (=`True`) flag, which allows for interoperability between Hydra tools while preserving their user interfaces.

[f]Jinja (https://jinja.palletsprojects.com) is similar to Django (https://www.djangoproject.com) templates for dynamically creating webpages, but can also be applied to software.

[g]Docker (https://www.docker.com) is used to hide the complexity of installing and operating SFs, by wrapping them inside their own mini-operating system environment.

[h]YAML (https://yaml.org) configuration files are used to store Python-like data structures, but in human readable form.

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

Table 2. Cerberus RMS and Island parameter definitions in units of σ with respect to the background, with soft constraint $\sigma_{Island} < \sigma_{RMS}$.

Source	RMS parameter			Island parameter		
Finder	Name	Default	Description	Name	Default	Description
Aegean[a]	`seedclip`	5.0	The clipping value for seeding islands	`floodclip`	4.0	The clipping value for growing islands
Caesar[b]	`seedThr`	5.0	Blob finding threshold	`mergeThr`	2.6	Blob growth threshold
ProFound[c]	`skycut`	2.8	Island threshold	`tolerance`	4.0	Defines island separation height
PyBDSF[d]	`thresh_pix`	5.0	Source detection threshold	`thresh_isl`	3.0	Threshold for the island boundary
Selavy[e]	`snrCut`	4.0	Detection threshold	`growthCut`	3.0	Threshold value to grow detections down to

[a]https://github.com/PaulHancock/Aegean/wiki/Simple-usage.
[b]https://caesar-doc.readthedocs.io/en/latest/usage/app_options.html#input-options.
[c]https://cran.r-project.org/web/packages/ProFound/ProFound.pdf.
[d]https://pybdsf.readthedocs.io/en/latest/process_image.html.
[e]https://www.atnf.csiro.au/computing/software/askapsoft/sdp/docs/current/analysis/selavy.html.

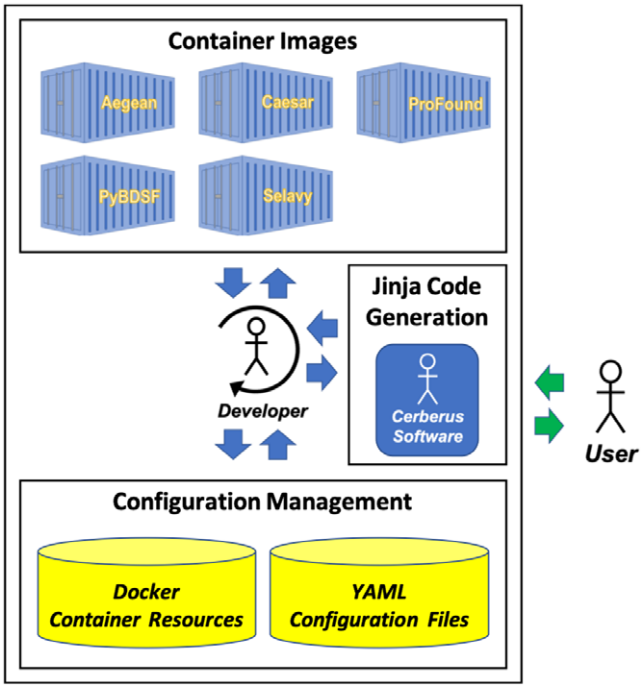


Figure 3. Cerberus code generation workflow.

- Update the `cerberus.py` script:
  - Create a YAML metadata file
  - Update the master YAML metadata file
  - Run the Jinja script generator tool
- Test the Hydra software suite
- Update the Git[i] repository

Fig. 3 summarises this high-level workflow: containers for each SF are shown under Container Images and the Docker and YAML configuration files are shown under Configuration Management. The developer must follow a fixed set of rules when adding a new SF, in order for the Jinja template-driven script generator to update Cerberus. (For the purpose of reproducibility, Appendix A provides architectural design notes, using Aegean as an example.) All of this is transparent to the user, who has access to a simple interface, so one does not have to be an expert at using SFs in order to use Hydra.

Hydra's modular design requires that the user has access to the key elements of a SF's interface; in particular, access to its 'RMS-like' and 'Island-like' parameters. In the case of Aegean, for example, this would be `seedclip` and `floodclip` (Hancock et al. 2012), respectively. It is important to note that the parameters are not necessarily equivalent between SFs;[j] regardless, they do affect thresholding and island formation. Consequently, they have the strongest influence on FDR calculations. Table 2 summarises the parameters for the currently supported SFs. These parameters are used by Typhon to baseline the SFs, by minimising their FDRs.

Hydra also requires that SF modules provide optional RMS box and step size parameters, even if they are dummies. Some SF software manuals recommend these parameters be externally optimised, under certain conditions. PyBDSF is an example of such a case.[k] Regardless, this is also a good way of baselining (i.e., calibrating, Huynh et al. 2012; Riggi et al. 2016) SFs for comparison purposes.

3.1.3. Typhon

Typhon is a tool for optimising the SFs to a standard baseline that can be used for comparison purposes. We have adopted the Percent Real Detections (PRD) metric, as used by Hale et al. (2019) in a comparative study of Aegean, ProFound, and PyBDSF: that is,

$$\mathrm{PRD} = \frac{N_{image} - N_{inv.\,image}}{N_{image}}\,100 \tag{1}$$

where N_{image} is the number of detections in the original image, and $N_{inv.\,image}$ is the number of detections in the inverted image.[l] Basically, if one assumes the image noise is predominately Gaussian, then the peaks detected in the inverted image should statistically match the noise-peaks detected in the non-inverted image. Thus the FDR can be reduced by optimising the PRD. This approach is not suitable for non-Gaussian (or non-symmetric) noise properties, such as the Ricean noise distribution in polarisation images.

Typhon uses the RMS and island parameters to optimise the PRD for each SF. Fig. 4 shows Typhon generated PRD curves for a $2^\circ \times 2^\circ$ simulated $\mathcal{D}$-image along with its corresponding $\mathcal{S}$-image.

[i]https://git-scm.com.

[j]Appendix B delineates these differences.
[k]See `rms_box` discussion at URL in footnote *d* of Table 2.
[l]that is, negative pixel values.

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

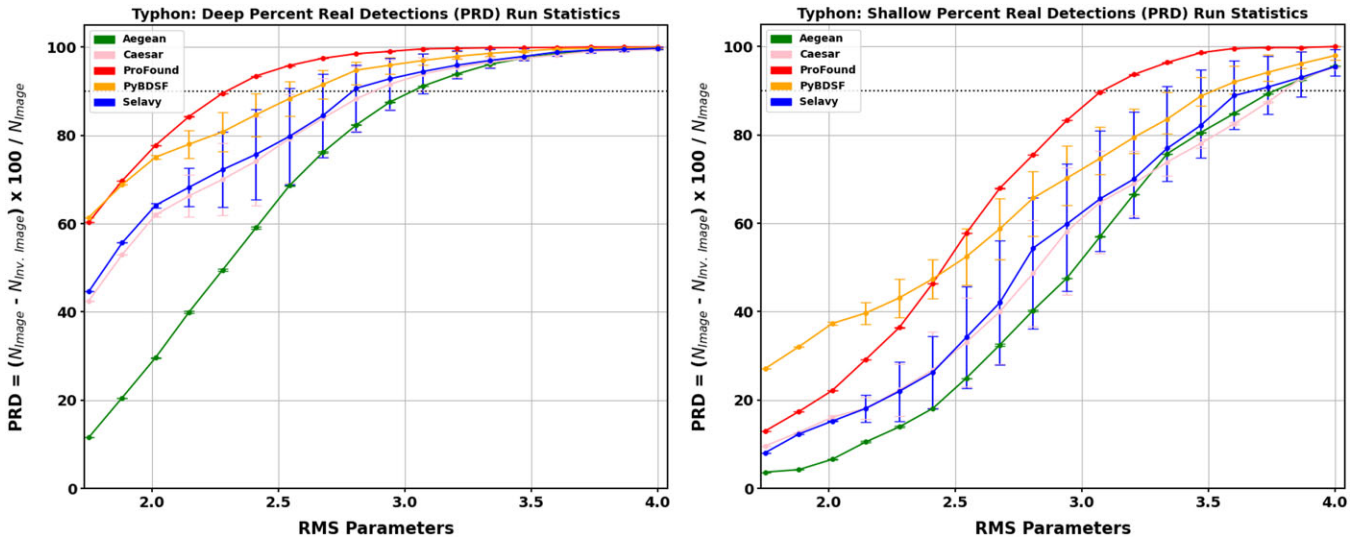


Figure 4. Example Typhon PRD of a 2° × 2° simulated $\mathcal{D}$-image (left) and its corresponding $\mathcal{S}$-image (right). The variation in the PRD with the RMS parameters (in σ units) is represented along the horizontal axis, and the variation in the PRD with the island parameters is represented by the error bars. The data points represent average values: that is, Aegean and ProFound indicate the true shape of the curves, due their insensitivity to their island parameters. The SF parameters are listed in Table 2. The dotted horizontal lines indicate the 90% PRD levels.

Typhon identifies the optimal parameters to be those that correspond to the 90% PRD threshold. This threshold is motivated by the desire to use the knee of the PRD curve, whose position appears to be scale-invariant above a certain image size. Although the shape of the curve is not always guaranteed to be smooth, this crude method appears to be quite effective at framing the region of interest around the desired 90% PRD. The 90% to 98% PRD range has been investigated and the former threshold seems to provide reasonable results. Hale et al. (2019) use a 98% PRD to baseline their SFs, beyond which the detection rate degrades. At that cut-off, however, we tend to find a non-scale-invariant increase in the RMS threshold with image size.

Typhon uses the image statistics output from Homados to determine the RMS parameter range over which to optimise the PRD:

$$1.5\sigma \leq \mathrm{RMS} \leq \mathrm{RMS}_{\max}\,, \tag{2}$$

where

$$\mathrm{RMS}_{\max} = \left\lceil \frac{I_{\max} - \mu}{\sigma} \right\rceil \sigma\,.$$

with μ, σ, and $I_{\max}$ determined through σ-clipping (re. Section 3.1.1). The 1.5σ lower limit is where the FDR starts to degrade. In general, Typhon samples the PRD from high values to low values in the RMS parameter, while varying the island parameter at each step, until the 90% threshold is reached.

The island parameters are SF-specific, and are typically defined over a finite range. Table 3 shows the parameter ranges used by Hydra, which are stored in its Configuration Management (Fig. 3). Typhon uses this information along with the constraint $\sigma_{island} < \sigma_{RMS}$ (otherwise, $\sigma_{island} = 0.999\,\sigma_{RMS}$), as it searches the parameter space.

Typhon will also perform an initial RMS box optimisation before optimising the PRD, if it is configured to do so. This is of particular importance for extended objects or around bright sources (Mohan & Rafferty 2015), especially for Gaussian-based extraction SFs such as Aegean, PyBDSF, and Selavy. Typhon uses Aegean's background/noise image generation tool, BANE (Hancock et al. 2018), to search the RMS box size (`box_size`) and step size (`step_size`) parameter space,

$$\left.\begin{aligned} 3 \leq \frac{\text{box\_size}}{[4(\texttt{BMAJ} + \texttt{BMIN})/2]} \leq 6 \\ \frac{1}{4} \leq \frac{\text{step\_size}}{\text{box\_size}} \leq \frac{1}{2} \end{aligned}\right\}\,, \tag{3}$$

for the lowest background level, μ (c.f., Riggi et al. 2016). The $4(\texttt{BMAJ} + \texttt{BMIN})/2$ factor represents the BANE default box size, where we assume a square box, for simplicity. The limits 3 and 6 are consistent with the rule of thumb that the box size should be 10–20 times larger than the beam size (Riggi et al. 2016). The 1/4 and 1/2 bounds are used for providing a smoothly sliding box (Mohan & Rafferty 2015).

Table 3. Configured island parameters.

SF	Island parameter	Range
Aegean	`floodclip`	[2,5]
Caesar	`mergeThr`	[2,3]
ProFound	`tolerance`	[2,5]
PyBDSF	`thresh_isl`	[2,5]
Selavy	`growthCut`	[2,5]

The Typhon optimisation algorithm can be summarised as follows.

- If the RMS box μ-optimisation is desired:
  - Minimise μ over a 6 × 3 box_size by step_size search grid, constrained by Equation (3)
- Select a centralised $n \times n$ image sample-cutout:
  - Use an n^2-area rectangle, if non-square image
  - Use the full area, if image is too small

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

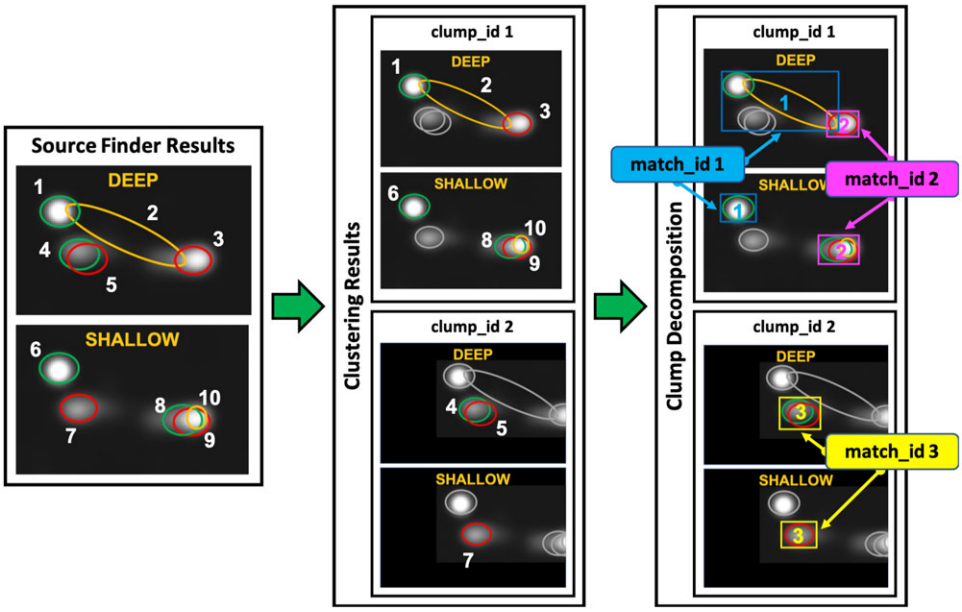


Figure 5. Clustering Algorithm Infographic: The left panel shows the results of 3 hypothetical SFs (red, green, and gold). The middle panel shows the results after clustering, resulting in two clumps, assigned `clump_id` 1 (upper panel) and `clump_id` 2 (lower panel). A clump is defined through the spatial overlap between SF detections (i.e., components), in the $\mathcal{D}$ and $\mathcal{S}$-images together. The components are numbered independently and can be associated with the clump they end up in. For instance, components 1, 2, and 3 are linked together in the $\mathcal{D}$-image, and, in addition, 1 overlaps with 6, and 3 overlaps with 8, 9, and 10 in the $\mathcal{S}$-image. Together this set of components populate `clump_id` 1. Similarly, `clump_id` 2 is composed of components 4, 5, and 7. Clumps are centred in the Hydra Viewer (Fig. 7), with unassociated components greyed out. The right panel shows the results after the clumps are decomposed into closest (i.e., overlapping centre-to-centre) matches between SFs, in the $\mathcal{D}$ and $\mathcal{S}$-images, such that there is only one SF with a match in the $\mathcal{D}$ and $\mathcal{S}$-images. These matched sets are assigned `match_ids`, with boxes enclosing the extremities of the components. The Hydra Viewer displays these numbers at the centre of the boxes (which are coloured differently here, for emphasis). So `clump_id` 1 contains `match_id` $1 = \{1, 2, 6\}$ and `match_id` $2 = \{3, 8, 9, 10\}$, while `clump_id` 2 contains `match_id` $3 = \{4, 5, 7\}$.

- Determine the RMS parameter bounds (Equation (2))
- For each SF:
  - If applicable, set the RMS box parameters to the optimised values
  - Extract the island parameter bounds from Configuration Management (re. Table 3)
  - Optimise the PRD of the sample-cutout:
    - Iterate the RMS parameter backwards from $\mathrm{RMS}_{\mathrm{max}}$
    - At each RMS step, iterate the island parameter, such that, $\sigma_{island} < \sigma_{RMS}$, otherwise set $\sigma_{island} = 0.999\,\sigma_{RMS}$
    - For each RMS-island parameter pair compute the PRD
    - Terminate iterations just before the PRD passes below 90%

      If the PRD is always below 90% choose the highest value.
- Run the SFs on the full image using their optimised parameters
- Archive the results in a tarball

For our initial studies, we have chosen to set $n = 2.5°$ to provide a sufficiently large region of sky to ensure the finder parameters are not biased by small-scale structure in a given image. Also Aegean, PyBDSF, and Selavy are configured to use the RMS box optimisation inputs from BANE, with Selavy only accepting the RMS box size. Appendix B provides details of the SFs and their settings used herein.

For the purposes of placing the SFs on equal footing, we have chosen to restrict Aegean, Caesar and Selavy to single threaded mode, so as to keep the background/noise statistics consistent, at the cost of computational speed. In addition, we keep all of the internal parameters of all of the SFs fixed, instead of tweaking them by hand for each use case. Every effort has been made to keep each SF module as generic as possible.

3.1.4. Hydra

Hydra is the main tool that glues everything together, by running Typhon for $\mathcal{D}$ and $\mathcal{S}$ images, and producing data products such as diagnostics plots and catalogues. One of main underlying features of Hydra is that it uses a clustering algorithm (Boyce 2018) to relate information between SF components in both $\mathcal{D}$ and $\mathcal{S}$ images. In addition, Hydra will also accept simulated catalogue input, under a source-finder pseudonym.

Fig. 5 shows an example of how the clustering algorithm works. All components between all $\mathcal{D}$ and $\mathcal{S}$ SF detections (i.e., catalogue rows) are spatially grouped together, with their overlaps forming clumps with unique `clump_ids`. The clumps are also decomposed into the closest $\mathcal{DS}$ matches, and assigned unique `match_ids`. The matches are further broken down by SF into subclumps (not shown), and assigned unique `subclump_ids`. All of this information is compiled into a cluster catalogue (or table), containing the following key reference elements (columns): cluster catalogue ID, clump ID, match ID, subclump ID, SF $\mathcal{D}/\mathcal{S}$ catalogue cross-reference ID, and image depth ($= \mathcal{D}$, $\mathcal{S}$). In addition, the catalogue contains common SF output parameters, such as RA, Dec, flux density, etc. There is also a clump catalogue, consisting of rows by unique `clump_id`, of cluster centroid positions, cutout sizes, total number of components, number of components per SF, SFs with the best residual statistics, etc.

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

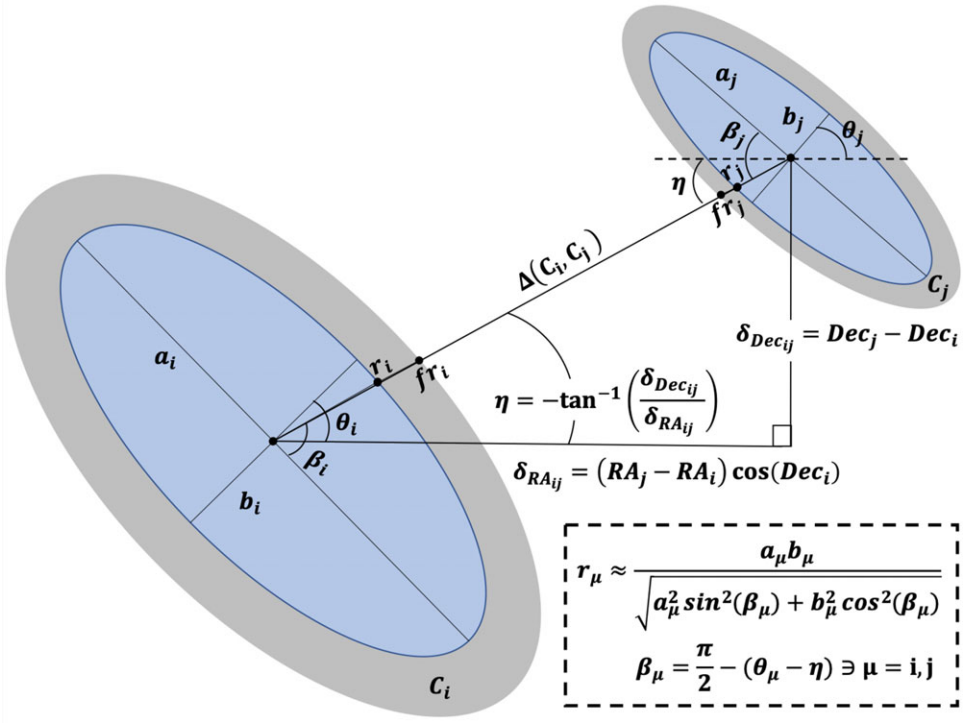


Figure 6. Derivation of the distance metric used for clustering. Here we assume that the space is locally flat, so that $\Delta(C_i, C_j) \approx (\delta^2_{\mathrm{RA}_{ij}} + \delta^2_{\mathrm{Dec}_{ij}})^{1/2}$, where $\delta_{\mathrm{RA}_{ij}} = (\mathrm{RA}_j - \mathrm{RA}_i)\cos(\mathrm{Dec}_i)$ and $\delta_{\mathrm{Dec}_{ij}} = \mathrm{Dec}_j - \mathrm{Dec}_i$. The distances from the centres of components C_i and C_j to their edges, along a ray between them, is given by r_i and r_j, respectively: that is, r_μ, is a standard geometrical expression in terms of angle $\beta_\mu = \pi/2 - (\theta_\mu - \eta)$ with respect to the ray and the semi-major axis a_μ, where θ_μ is the position angle, $\eta = -\tan^{-1}(\delta_{\mathrm{Dec}_{ij}}/\delta_{\mathrm{RA}_{ij}})$, and $\mu = i, j$. The grey area outside the ellipses is the skirt, whose extent is determined by f.

Fig. 6 shows the derivation of the distance metric used in the clustering algorithm. The algorithm uses the following distance metric constraint to determine the overlap between two elliptical components, C_i and C_j, with centre-to-edge distances, r_i and r_j, along an adjoining ray.

$$\Delta(C_i, C_j) \leq r_i + r_j \tag{4}$$

where

$$\Delta(C_i, C_j) = \sqrt{(\mathrm{RA}_j - \mathrm{RA}_i)^2 \cos^2(\mathrm{Dec}_i) + (\mathrm{Dec}_j - \mathrm{Dec}_i)^2},$$

is the distance metric, and

$$r_\mu = \frac{a_\mu b_\mu}{\sqrt{a_\mu^2 \cos^2(\theta_\mu - \eta) + b_\mu^2 \sin^2(\theta_\mu - \eta)}},$$

are the centre-to-edge distances, for $\mu = i, j$, with

$$\eta = -\tan^{-1}\left[\frac{\mathrm{Dec}_j - \mathrm{Dec}_i}{(\mathrm{RA}_j - \mathrm{RA}_i)\cos(\mathrm{Dec}_i)}\right],$$

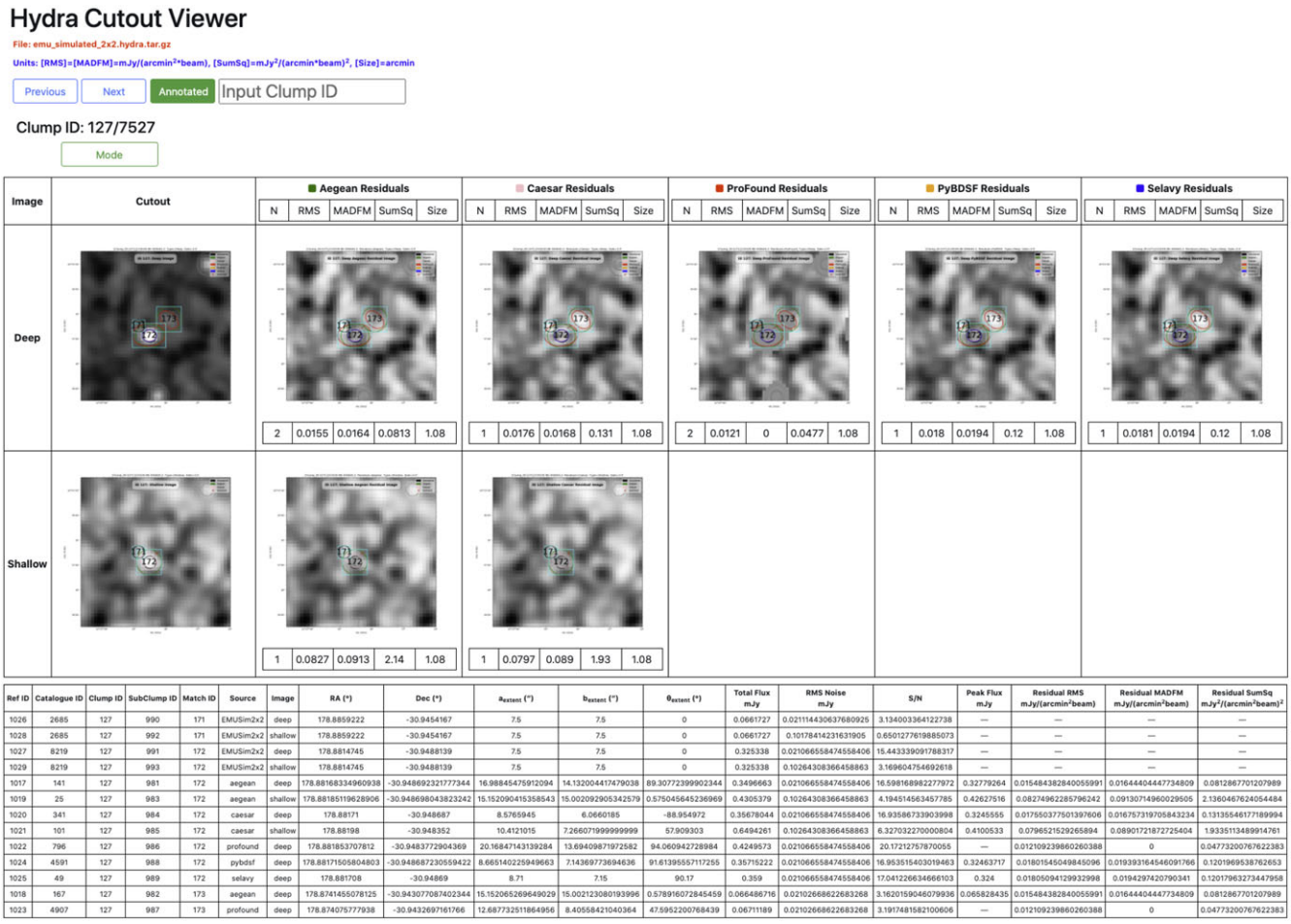
a_μ is the semi-major axis, b_μ is the semi-minor axis, and θ_μ is the position angle (defined in the same manner as the beam PA, Greisen 2017). So components satisfying this constraint are clumped together.

Hydra also provides a web-viewer (known as the Hydra Viewer) for exploring image and residual image cutouts by `clump_id`, along with corresponding cluster table information. Fig. 7 provides a detailed description of the Hydra Viewer's cutout interface. As indicated in the figure, the viewer has radio component annotations that can be toggled on/off. Fig. 8 provides a more detailed example. Table 4 describes the annotation colours, which are stored as metadata in Hydra's Configuration Management (Fig. 3).

The following is a list of the data products produced by Hydra:

- Typhon Metrics
  - $\mathcal{D}/\mathcal{S}$ Diagnostic Plots of
    - PRD
    - PRD CPU Times
    - Residual RMS
    - Residual MADFM
    - Residual ΣI^2
  - Table of $\mathcal{D}$ and $\mathcal{S}$ optimised RMS and island parameters
- $\mathcal{D}/\mathcal{S}$ Catalogues
  - SF Catalogues
  - Cluster Catalogue
  - Clump Catalogue
- Optional Simulated Input Catalogue
- $\mathcal{D}/\mathcal{S}$ Cutouts
  - Un/annotated Images
  - Un/annotated Residual Images
- $\mathcal{D}/\mathcal{S}$ Diagnostic Plots
  - Clump Size Distributions
  - Detections vs S/N

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press



Ref ID	Catalogue ID	Clump ID	SubClump ID	Match ID	Source	Image	RA (°)	Dec (°)	a_{extent} (°)	b_{extent} (°)	θ_{extent} (°)	Total Flux mJy	RMS Noise mJy	S/N	Peak Flux mJy	Residual RMS mJy/(arcmin²beam)	Residual MADFM mJy/(arcmin²beam)	Residual SumSq mJy²/(arcmin²beam)²
1026	2685	127	990	171	EMUSim2x2	deep	178.8859222	-30.9454167	7.5	7.5	0	0.0661727	0.021114430637680925	3.134003364122738	—	—	—	—
1028	2685	127	992	171	EMUSim2x2	shallow	178.8859222	-30.9454167	7.5	7.5	0	0.0661727	0.10178414231631905	0.6501277619885073	—	—	—	—
1027	8219	127	991	172	EMUSim2x2	deep	178.8814745	-30.9488139	7.5	7.5	0	0.325338	0.021066558474558406	15.443339091788317	—	—	—	—
1029	8219	127	993	172	EMUSim2x2	shallow	178.8814745	-30.9488139	7.5	7.5	0	0.325338	0.10264308366458863	3.169604754692618	—	—	—	—
1017	141	127	981	172	aegean	deep	178.88168334960938	-30.948692321777344	16.98845475912094	14.132004417479038	89.30772399902344	0.3496663	0.021066558474558406	16.598168982277972	0.32779264	0.015484382840055991	0.016444044477348 09	0.0812867701207989
1019	25	127	983	172	aegean	shallow	178.88185119628906	-30.948698043823242	15.152090415358543	15.002092905342579	0.5750456455236969	0.4305379	0.10264308366458863	4.194514563457785	0.42627516	0.08274962285796242	0.09130714960029505	2.1360467624054484
1020	341	127	984	172	caesar	deep	178.88171	-30.948687	8.5765945	6.0660185	-88.954972	0.35678044	0.021066558474558406	16.935867339039998	0.3245555	0.017550377501397606	0.016757319705843234	0.13135546177189994
1021	101	127	985	172	caesar	shallow	178.88198	-30.948352	10.4121015	7.266071999999999	57.909303	0.6494261	0.10264308366458863	6.327032270000804	0.4100533	0.0796521529265894	0.08901721872725404	1.9335113489914761
1022	796	127	986	172	profound	deep	178.88185370781 2	-30.9483772904369	20.168471431392 84	13.69409871972582	94.060942728984	0.4249573	0.021066558474558406	20.172127578700 55	—	0.012109239860260388	0	0.04773200767622383
1024	4591	127	988	172	pybdsf	deep	178.88171505804803	-30.948687230559422	8.665140225949663	7.143697736946 36	91.613955571172 55	0.35715222	0.021066558474558406	16.953515403019463	0.32463717	0.018015450498459 96	0.019393164546091766	0.1201969538762653
1025	49	127	989	172	selavy	deep	178.881708	-30.94869	8.71	7.15	90.17	0.359	0.021066558474558406	17.041226634666103	0.324	0.018050941299329 98	0.019429742079034 1	0.1201796327344795 8
1018	167	127	982	173	aegean	deep	178.87414550781 25	-30.943077087402344	15.152065269649029	15.002123080193996	0.578916072845459	0.066486716	0.021026668622683268	3.162015904607993 6	0.065828435	0.015484382840055991	0.016444044477348 09	0.0812867701207989
1023	4907	127	987	173	profound	deep	178.874075777938	-30.943269716176 6	12.687732511864956	8.405584210403 64	47.595220076843 9	0.06711189	0.021026668622683268	3.191748158210060 6	—	0.012109239860260388	0	0.04773200767622383

Figure 7. Hydra Viewer Infographic: The cutout viewing section of Hydra's local web-viewer interface. At the top is the navigation bar, which allows the user to navigate by clump ID, go to a specific clump, turn on/off cutout annotations, or examine S/N bins of diagnostic plots such as $\mathcal{C}$ and $\mathcal{R}$, by using the Mode button. The main panel contains $\mathcal{D}$ (top) and $\mathcal{S}$ (bottom) square image cutouts (first column) and SF residual image cutouts (remaining columns), centred about a given clump's centroid. Here the annotation is turned on, with the neighbouring clumps greyed out. The table at the bottom (not to scale) is the cluster table rows for the clump, with the following columns: cluster catalogue ID, SF catalogue cross-reference ID, clump ID, subclump ID, match ID, SF or $\mathcal{J}$ catalogue name, image depth, RA (°), Dec (°), semi-major axis (″), semi-minor axis (″), position angle (°), total flux density (mJy), BANE RMS noise (mJy), S/N (total flux over BANE RMS noise), peak flux (mJy beam^{-1}), normalised-residual RMS (mJy (arcmin^2 beam)$^{-1}$), normalised-residual MADFM (mJy (arcmin^2 beam)$^{-1}$), and normalised-residual ΣI^2 ((mJy (arcmin^2 beam))$^{-2}$). The normalised-residual statistics are normalised by the cutout area (arcmin^2). This statistical information is also shown below each cutout, along with the number of components (N), and cutout size (Size, in arcmin). This figure is to illustrate the layout of the Hydra viewer, not the details. It shows screen shots from the Hydra Viewer pasted together, hence the fonts appear small. The data at the bottom is raw output from the cluster table, which is not rounded in this version of the software.

  - $\mathcal{C}$ vs S/N
  - $\mathcal{R}$ vs S/N
  - Flux-Ratios: S_{out}/S_{in} vs S/N
  - False-Positives vs S/N
- Hydra Viewer: Local Web-browser Tool

All of this information is stored in a tarball. The Hydra Viewer allows the user to view all of these data products. The cutout viewer portion is linked only to the cluster catalogue. It is accessible through an `index.html` file in the main `tar` directory.

4. Completeness and reliability metrics

Completeness and reliability metrics can be generated through various combinations of deep, shallow, and injected sources. Fig. 9 shows a Venn diagram of the overlapping possibilities. In addition, we need to be careful in our definitions of these metrics.

Here we use a clustering approach to spatially match our detections (Equation (4)). An alternative method is to use a cut-off distance in a catalogue cross-match (e.g., Huynh et al. 2012; Hopkins et al. 2015; Hale et al. 2019).[m] Depending on the cutoff distance adopted, 'locally,' this approach may lead to associations between adjacent clumps that may not actually be related. The clustering approach aims to mitigate against this effect.

Completeness ($\mathcal{C}$) is the fraction of real detections to real sources, and reliability ($\mathcal{R}$) is the fraction of real detections to detected sources (Fig. 9). Here we define these metrics in terms of 'real' injected (simulated) sources ($\mathcal{J}$) vs deep ($\mathcal{D}$) and shallow ($\mathcal{S}$) detections and, 'assumed-real' deep detections vs shallow detections. In the case where the sources are known (i.e., injected),

[m] Bonaldi et al. (2021) also requires a level of consistency in flux density to associate independently detected sources as being in common. We do not use this in our metric as it can lead to falsely rejecting true source associations where flux densities are poorly measured.

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

Table 4. SF annotation colours.

SF	Colour
Aegean	Green
Caesar	Magenta
ProFound	Red
PyBDSF	Orange
Selavy	Blue
Simulated	Black

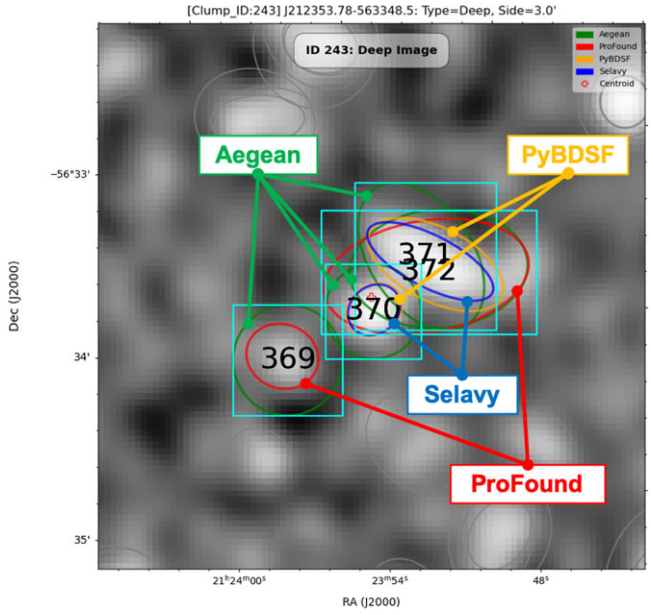


Figure 8. An example of a $\mathcal{D}$-image cutout, with annotations turned on, consisting of 4 Aegean, 2 ProFound, 2 PyBDSF, and 2 Selavy overlapping $\mathcal{D}$-image catalogue components. The label at the top indicates it corresponds to `clump_id` 243, and the numbers at the centres of the cyan boxes are the `match_ids` (369 through 372).

we take the fraction of real–deep ($\mathcal{D} \cap \mathcal{J}$) or real–shallow ($\mathcal{S} \cap \mathcal{J}$) detections to the injected sources for our completeness,

$$\mathcal{C}_{\mathcal{D}} = \frac{\mathcal{D} \cap \mathcal{J}}{\mathcal{J}} \tag{5}$$

or

$$\mathcal{C}_{\mathcal{S}} = \frac{\mathcal{S} \cap \mathcal{J}}{\mathcal{J}}, \tag{6}$$

respectively.[n] Similarly, the fraction of real–deep or real–shallow detections to the corresponding deep or shallow detections give the reliability

$$\mathcal{R}_{\mathcal{D}} = \frac{\mathcal{D} \cap \mathcal{J}}{\mathcal{D}} \tag{7}$$

or

$$\mathcal{R}_{\mathcal{S}} = \frac{\mathcal{S} \cap \mathcal{J}}{\mathcal{S}}, \tag{8}$$

respectively. In the case where no true underlying sources are known we use the deep detections as a proxy, and take the fraction

[n] In general, in our notation, the length of a set is implicitly assumed: for example, $\mathcal{D} \cap \mathcal{J} \equiv |\mathcal{D} \cap \mathcal{J}|$.

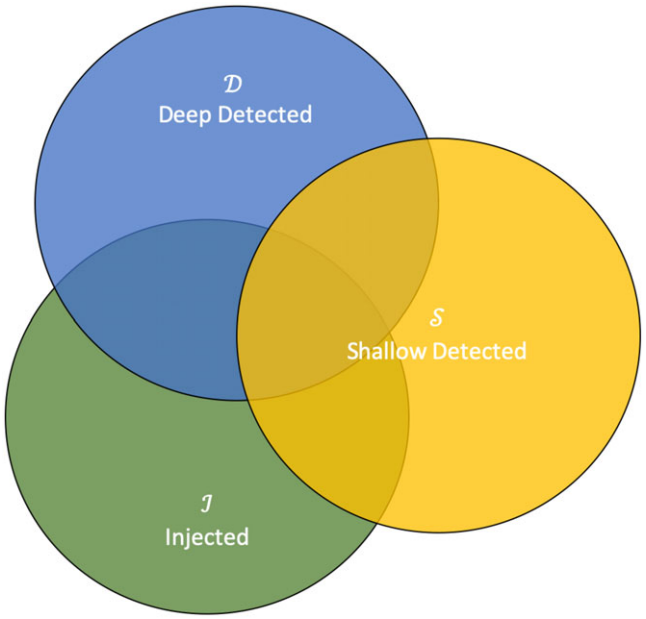


Figure 9. Venn diagram of completeness and reliability, for sets of deep ($\mathcal{D}$), shallow ($\mathcal{S}$), and injected ($\mathcal{J}$) sources.

of real–shallow detections to deep detections for our completeness,

$$\mathcal{C}_{\mathcal{DS}} = \frac{\mathcal{S} \cap \mathcal{D}}{\mathcal{D}}, \tag{9}$$

and the fraction of real–shallow to shallow detections for our reliability,

$$\mathcal{R}_{\mathcal{DS}} = \frac{\mathcal{S} \cap \mathcal{D}}{\mathcal{S}}. \tag{10}$$

We can take this one step further by asking the question, *'Given our knowledge of injected sources, how good are our measures of deep–shallow completeness ($\mathcal{C}_{\mathcal{DS}}$) and reliability ($\mathcal{R}_{\mathcal{DS}}$)?'* From this, we define the fraction of real–deep–shallow detections, $(\mathcal{D} \cap \mathcal{J}) \cap (\mathcal{S} \cap \mathcal{J})$, to real–deep detections, $\mathcal{D} \cap \mathcal{J}$, as our goodness of completeness,

$$\tilde{\mathcal{C}}_{\mathcal{DS}} = \frac{(\mathcal{D} \cap \mathcal{J}) \cap (\mathcal{S} \cap \mathcal{J})}{\mathcal{D} \cap \mathcal{J}}, \tag{11}$$

and the fraction of real–deep–shallow detections to real-shallow detections, $\mathcal{S} \cap \mathcal{J}$, as our goodness of reliability,

$$\tilde{\mathcal{R}}_{\mathcal{DS}} = \frac{(\mathcal{D} \cap \mathcal{J}) \cap (\mathcal{S} \cap \mathcal{J})}{\mathcal{S} \cap \mathcal{J}}. \tag{12}$$

Table 5 summarises all of our completeness and reliability metrics (Equations (5) through (12)).[o]

Fig. 10 shows examples of deep–shallow source component overlaps, $\mathcal{S} \cap \mathcal{D}$, to illustrate the calculation of $\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$. Matches are done pair-wise, within clumps, between the closest centres of overlapping deep–shallow components. This method is more precise than a typical fixed separation nearest neighbour approach (Hopkins et al. 2015; Riggi et al. 2019), as it ensures source components always overlap. The $|\mathcal{S} \cap \mathcal{D}|$:$|\mathcal{D}|$ and

[o] The current version of the Hydra Viewer (Fig. 7) only supports filtering the S/N bins of its $\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$ diagnostic plots, through its Mode button.

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

Table 5. Completeness/reliability metrics (see Fig. 9) in terms of deep ($\mathcal{D}$), shallow ($\mathcal{S}$), and injected ($\mathcal{J}$) sources.

Inputs	Detections	Real detections	Completeness	Reliability
$\mathcal{J}$	$\mathcal{D}$	$\mathcal{D}\cap\mathcal{J}$	$\mathcal{C}_{\mathcal{D}}=\frac{\mathcal{D}\cap\mathcal{J}}{\mathcal{J}}$	$\mathcal{R}_{\mathcal{D}}=\frac{\mathcal{D}\cap\mathcal{J}}{\mathcal{D}}$
$\mathcal{J}$	$\mathcal{S}$	$\mathcal{S}\cap\mathcal{J}$	$\mathcal{C}_{\mathcal{S}}=\frac{\mathcal{S}\cap\mathcal{J}}{\mathcal{J}}$	$\mathcal{R}_{\mathcal{S}}=\frac{\mathcal{S}\cap\mathcal{J}}{\mathcal{S}}$
$\mathcal{D}$	$\mathcal{S}$	$\mathcal{S}\cap\mathcal{D}$	$\mathcal{C}_{\mathcal{DS}}=\frac{\mathcal{S}\cap\mathcal{D}}{\mathcal{D}}$	$\mathcal{R}_{\mathcal{DS}}=\frac{\mathcal{S}\cap\mathcal{D}}{\mathcal{S}}$
$\mathcal{D}\cap\mathcal{J}$	$\mathcal{S}\cap\mathcal{J}$	$(\mathcal{D}\cap\mathcal{J})\cap(\mathcal{S}\cap\mathcal{J})$	$\tilde{\mathcal{C}}_{\mathcal{DS}}=\frac{(\mathcal{D}\cap\mathcal{J})\cap(\mathcal{S}\cap\mathcal{J})}{\mathcal{D}\cap\mathcal{J}}$	$\tilde{\mathcal{R}}_{\mathcal{DS}}=\frac{(\mathcal{D}\cap\mathcal{J})\cap(\mathcal{S}\cap\mathcal{J})}{\mathcal{S}\cap\mathcal{J}}$

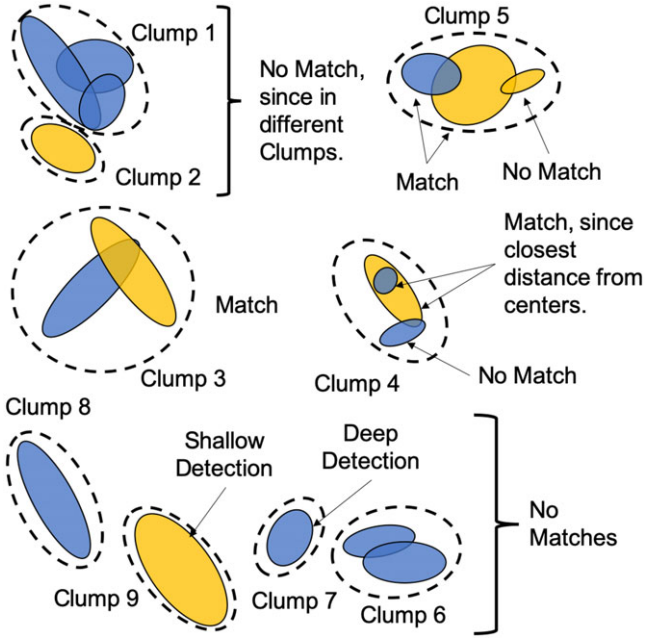


Figure 10. Examples of deep (blue) and shallow (amber) source component overlaps, $\mathcal{C}_{\mathcal{DS}}=(\mathcal{S}\cap\mathcal{D})/\mathcal{D}$ and $\mathcal{R}_{\mathcal{DS}}=(\mathcal{S}\cap\mathcal{D})/\mathcal{S}$. Real-shallow detections are indicated by overlapping pair-wise deep-shallow detections ($\mathcal{S}\cap\mathcal{D}$), whose centres are closest. The dash-lines indicate clumps of component extent overlays.

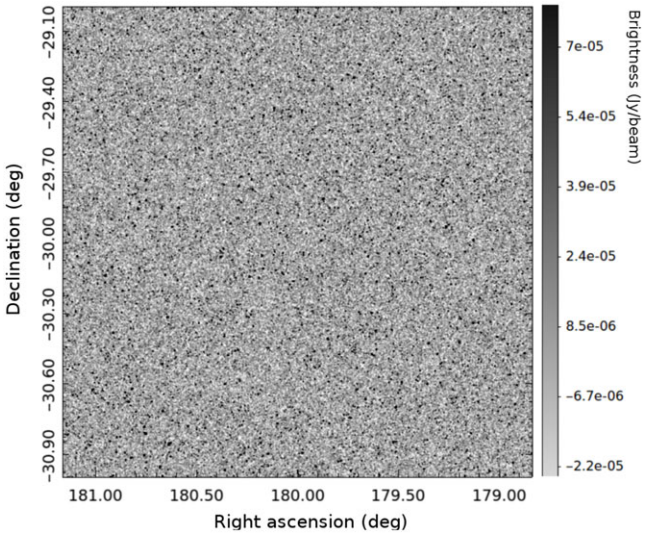


Figure 11. Simulated map with point-like (compact) sources. The coordinates are arbitrarily set, and the FWHM is set to 15".

$|\mathcal{S}\cap\mathcal{D}|:|\mathcal{S}|$ ratios are then binned with respect to the $\mathcal{D}$ and $\mathcal{S}$ flux densities (or S/N), respectively, producing $\mathcal{C}_{\mathcal{DS}}$ vs $\mathcal{D}$ completeness and $\mathcal{R}_{\mathcal{DS}}$ vs $\mathcal{S}$ reliability histograms.

5. Validation

In this section we use $2^\circ\times2^\circ$ simulated-compact (CMP) and simulated-extended (EXT) image data to characterise the performance of Hydra, and validate some new metrics. In particular, the simulated data are used to explore and develop metrics that can be used for real images where the ground truth is unknown. A preliminary discussion on SF performance is also presented. Paper II is focused on cross-SF comparison, using our simulated data along with some real data.

5.1. Image data

5.1.1. Simulated compact sources, CMP

The simulated image, shown in Fig. 11, is produced in two steps; generation of a noise image, followed by the addition of artificial sources. We use MIRIAD (Sault, Teuben, & Wright 1995) to generate the artificial noise image, using the following steps. The IMGEN task was used to produce a 1800×1800 pixel image, with 4" pixels, (i.e., a $2^\circ\times2^\circ$ field) populated by random Gaussian noise of RMS 20 μJy beam^{-1}. This image was convolved using CONVOL to mimic a 15" FWHM beam, which has the effect of increasing the noise level by a factor of 2.8, so the resulting image is then scaled using MATHS to divide by this factor, restoring the original noise level of 20 μJy beam^{-1}.

To generate the properties of the artificial sources, we use the 6th order polynomial fit to the 1.4 GHz source counts from Hopkins et al. (2003), which is consistent with more recent source count determinations for the flux density range considered here (e.g., Gürkan et al. 2022, and references therein). A sequence of 34 exponentially spaced bins in flux density was defined, ranging from 50 μJy to 1 Jy, and within each bin the number of sources was calculated from the source count model. The flux density for each artificial source was assigned randomly between the extrema of the bin in which it lies. Source positions were also assigned randomly, with no attempt to mimic the clustering properties of real sources. For the $2^\circ\times2^\circ$ field with a flux density limit of 50 μJy, we end up with a list of 9075 artificial sources.

These sources were added to the noise image using the Python module Astropy (Astropy Collaboration et al. 2013, 2018) by constructing 2D Gaussian models with the FWHM of the restoring

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

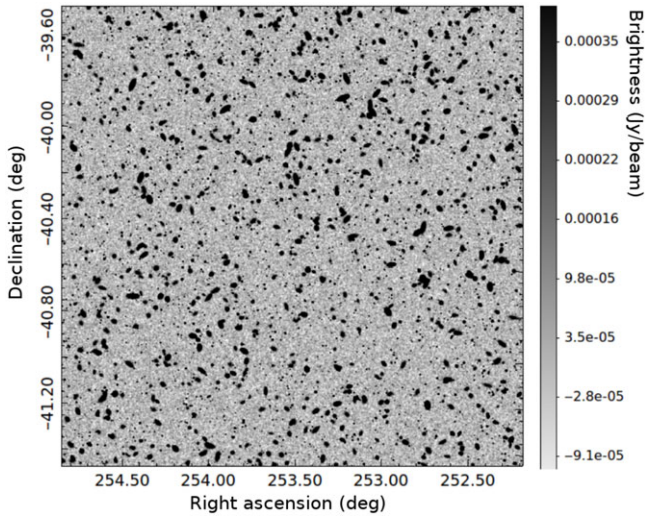


Figure 12. Simulated image with both point-like (compact) and extended sources. The sky coordinates are arbitrarily chosen.

beam (15”) and scaling the amplitude to represent the randomly assigned peak flux density of the source. Given the sources modelled here are assumed to be point-like (compact), the peak flux density for a source has the same amplitude as the integrated flux density. Using this Gaussian model for each source, we generated an image array for each source to be added into the simulated image. We used Astropy again to convert the RA/Dec location of the source to pixel locations and each source was added to the simulated image at the desired location.

5.1.2. Simulated extended sources, EXT

Following a similar procedure as in Section 5.1.1, we generated a sky model of size 1800× 1800 pixels (4″ pixel size, 2°×2° field of view) with both point-like and extended sources injected. The noise level is again set to 20 μJy beam^{-1}. Extended sources are 2D elliptical Gaussians with randomised position angle and axis ratio, with axis ratio varying between 0.4 to 1. A maximum major axis size was set at three times the synthesised beam size (45″, with a 15″ FWHM beam, as in Section 5.1.1).

A total of 9974 artificial sources were injected, corresponding to a source density of 2500 deg^{-2}, with 10% being extended sources. The peak flux densities S of both point-like and extended sources were set to follow an exponential distribution $10^{-\lambda S}$ with λ=1.6, consistent with that seen in early ASKAP observations of the Stellar Continuum Originating from Radio Physics In Our Galaxy (SCORPIO, Umana et al. 2015) field (Riggi et al. 2021). The minimum peak flux density for all sources was set at 50 μJy and the maximum fixed at 1 Jy for compact sources and 1 mJy for extended sources. The final simulated image, shown in Fig. 12, was produced by convolving this input sky model using the CASA[p] *imsmooth* task and a target resolution of 15″.

It is important to note here that, unlike the compact source simulation above where the faintest injected source lies at $(S/N)_{min} \sim 2.5$, in the extended source simulated image 20.6% of the injected sources fall below S/N = 1. This is by design, to provide a more realistic image, and to test the impact on the SFs of having real sources lying below the noise level (re. M. M. Boyce et al., in preparation).

[p]Common Astronomy Software Applications, https://casa.nrao.edu/.

Table 6. Hydra μ-optimised `box_size` and `step_size` inputs for Aegean, PyBDSF, and Selavy,[a] using CMP and EXT $\mathcal{D}/\mathcal{S}$-image data.

Image	Image depth	μ (μJy beam^{-1})	`box_size` (″)	`step_size` (″)
CMP	$\mathcal{D}$	21.81	240	120
	$\mathcal{S}$	108.2	180	88
EXT	$\mathcal{D}$	68.01	480	240
	$\mathcal{S}$	325.3	240	120

[a]Selavy only accepts `box_size`.

5.2. Typhon statistics

5.2.1. Optimisation run results

Hydra uses Typhon to set the RMS box and island parameters for each SF (Aegean, Caesar, ProFound, PyBDSF, and Selavy) to ensure a 90% PRD cutoff. The RMS box parameters, obtained from Typhon’s μ-optimisation routine, were used by Aegean, PyBDSF, and Selavy. An $\mathcal{S}$-image was generated by Homados, and RMS box and island parameters similarly estimated. Tables 6 and 7 summarise these results for our CMP and EXT images.

For the CMP source $\mathcal{D}$-image, $\mu \sim 22$ μJy beam^{-1} (Table 6) which is consistent with the design RMS noise of 20 μJy beam^{-1} (re. Section 5.1). For the EXT source $\mathcal{D}$-image, $\mu \sim 68$ μJy beam^{-1} which is higher due to the inclusion of extended structures with a slightly higher source density, and the fact that the box size has doubled. This is consistent with an average source size increase from 15″ to 30″. We also note that $\mu_{\mathcal{S}}/\mu_{\mathcal{D}} \sim 5$ for all images, which is consistent with the factor of 5 noise increase for the $\mathcal{S}$-images (Section 3.1.1). So the RMS box statistics are consistent with what might be expected from the simulated images.

In Table 7 we notice, in the broadest sense, that the RMS and island parameters are similar for the $\mathcal{D}/\mathcal{S}$-images, row-wise. Fig. 13 shows stacked plots of $\mathcal{D}/\mathcal{S}$ source counts for the images. In the CMP and EXT $\mathcal{D}$-images, there is some variability in the source counts, that is, $N \sim 4\,650 \pm 890$, except for Selavy being consistently lower by $N \sim 1\,770 \pm 950$. In the $\mathcal{S}$-image case, the variability is fairly tight, with $N_{\rm CMP} \sim 661 \pm 54$ and $N_{\rm EXT} \sim 1\,258 \pm 70$, except for Selavy which is consistently low with $N_{\rm CMP} = 427$ and $N_{\rm EXT} = 787$.

For the CMP $\mathcal{D}$-image, the RMS and MADFM residual statistics are all ∼19 μJy beam^{-1}, with the exception of Selavy having a significantly higher RMS (∼45 μJy beam^{-1}), likely the primary cause for the reduction in numbers of detected sources it reports. For the EXT $\mathcal{D}$-image, the RMS values increase in order from ProFound, Caesar, Aegean, PyBDSF, to Selavy, with the latter three being comparable to the former being at extreme end. This is also reflected in their MAFDMs, although here the values are somewhat comparable. For the $\mathcal{S}$-image case, everything is similar within each image data set, except for Selavy having a higher RMS in the EXT image case. These observed discrepancies for Selavy are likely due to its use of robust statistics in the background estimation (Section B.5). In contrast, the MADFM is similar in both cases for all SFs.

Table 8 shows ratios of deep-to-injected ($\mathcal{D}:\mathcal{J}$) and shallow-to-deep ($\mathcal{S}:\mathcal{D}$) source counts: that is, $\mathcal{D}:\mathcal{J}$ indicates the fraction

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

Table 7. Typhon run statistics for CMP and EXT images, with SF, image depth, SF RMS parameter (n_{rms} [σ]), SF island parameter (n_{island} [σ]), source counts (N), residual RMS (μJy beam^{-1}), and residual MADFM (μJy beam^{-1}) columns.[a]

SF	Image	CMP sources					EXT sources				
	depth	n_{rms}	n_{island}	N	RMS	MADFM	n_{rms}	n_{island}	N	RMS	MADFM
Aegean	$\mathcal{D}$	3.074	3.070	6016	20.0	19.0	3.074	3.070	4112	67.0	56.0
Caesar	$\mathcal{D}$	3.074	3.000	4084	19.0	16.0	3.206	3.000	3618	54.0	44.0
ProFound	$\mathcal{D}$	2.412	2.409	4997	18.0	16.0	2.412	2.409	3730	52.0	43.0
PyBDSF	$\mathcal{D}$	2.809	2.806	5991	22.0	19.0	2.809	2.806	4688	106	54.0
Selavy	$\mathcal{D}$	3.206	3.203	3225	45.0	20.0	3.206	3.203	2544	982	58.0
Aegean	$\mathcal{S}$	3.868	3.864	747	110	110	3.603	3.599	1287	321	317
Caesar	$\mathcal{S}$	4.000	3.000	657	109	106	3.603	3.000	1297	310	295
ProFound	$\mathcal{S}$	3.074	3.070	642	109	107	2.941	2.938	1138	311	298
PyBDSF	$\mathcal{S}$	3.735	3.732	598	111	109	3.338	3.335	1312	316	313
Selavy	$\mathcal{S}$	4.000	3.996	427	114	110	3.735	3.732	787	623	320

[a]The MADFM estimators are normalised by 0.6744888 (Whiting 2012).

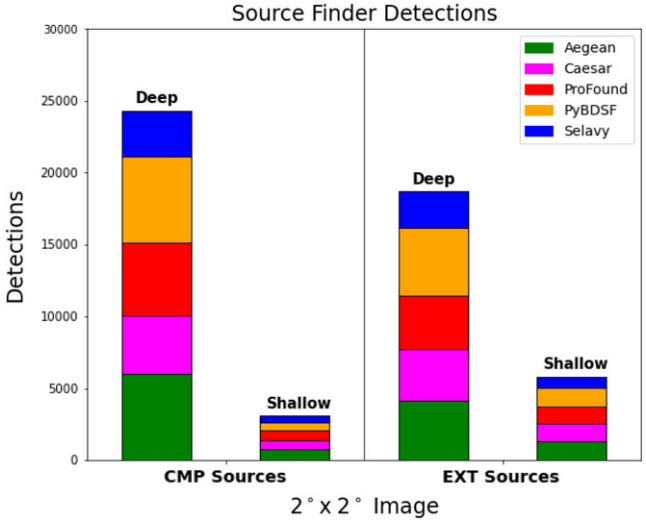


Figure 13. SF CMP and EXT $\mathcal{D}/\mathcal{S}$-image detection stacked plots (re. the N columns of Table 7).

of sources recovered in the simulated images, whereas $\mathcal{S}:\mathcal{D}$ indicates the recovery rate assuming the deep detections are *'real'*. Also included are $\mathcal{S}:\mathcal{J}$ recovery rates, for comparison.[q] The $\mathcal{D}:\mathcal{J}$ recovery rates are not expected to reach 100%, as the simulations includes low S/N sources, and for EXT sources, some lying below S/N= 1. The $\mathcal{S}:\mathcal{D}$ recovery rate is lower for CMP sources than EXT, which both track consistently with $\mathcal{D}:\mathcal{J}$.[r] This may be due to some confusion in the EXT image (Fig. 12), given the $\mathcal{S}:\mathcal{J}$ recovery rates are similar.

The consistency of the RMS box, MADFM, $\mathcal{D}:\mathcal{J}$, $\mathcal{S}:\mathcal{J}$, and $\mathcal{S}:\mathcal{D}$ statistics provides a good indication that Hydra's optimisation routines are performing robustly.

5.2.2. Source size distributions

Fig. 14 shows the major-axis distribution for our simulated image data. Both the $\mathcal{D}$ and $\mathcal{S}$ source detections are combined, as the $\mathcal{S}$ provides no contrasting information and its statistics are quite low (Fig. 13). Note that the size estimates for different SFs use different methods and are not necessarily directly comparable. Those SFs that fit Gaussians to source components report size as a FWHM, while others (such as ProFound) use different measures, such as a flux-weighted fit (see Appendix B for details).

For the CMP source case, ProFound and PyBDSF tend to overestimate the source sizes. This is likely due to deblending issues or the incorporation of noise spikes. As for EXT sources, PyBDSF tends to most accurately recover the extended source sizes, but, along with Caesar also has the most outliers. These could be attributed to size overestimates due to inclusion of noise spikes or adjacent sources in the fitted sizes. All other SFs tend to marginally underestimate the EXT source sizes. Components smaller than 15″ are attributed primarily to noise spikes, but also occasionally to underestimating the source sizes.

Table 8. Ratios of deep-to-injected ($\mathcal{D}:\mathcal{J}$) shallow-to-injected ($\mathcal{S}:\mathcal{J}$), and shallow-to-deep ($\mathcal{S}:\mathcal{D}$) sources. The $\mathcal{D}/\mathcal{S}$ source counts (N) are provided in Table 7, and the injected source counts are 9075 and 9974 for CMP and EXT sources, respectively (re. Section 5.1).

Source	CMP			EXT		
finder	$\mathcal{D}:\mathcal{J}$	$\mathcal{S}:\mathcal{J}$	$\mathcal{S}:\mathcal{D}$	$\mathcal{D}:\mathcal{J}$	$\mathcal{S}:\mathcal{J}$	$\mathcal{S}:\mathcal{D}$
Aegean	66.3%	8.2%	12.4%	41.2%	7.5%	31.3%
Caesar	45.0%	7.2%	16.1%	36.3%	6.6%	35.8%
ProFound	55.1%	7.1%	12.8%	37.4%	6.4%	30.5%
PyBDSF	66.0%	6.6%	10.0%	47.0%	6.0%	28.0%
Selavy	35.5%	4.7%	13.2%	25.5%	4.3%	30.9%

5.3. Completeness and reliability

In this section, we discuss deep ($\mathcal{D}$) and shallow ($\mathcal{S}$) completeness ($\mathcal{C}$) and reliability ($\mathcal{R}$) (see Section 4). We then provide justification for using deep-shallow ($\mathcal{DS}$) completeness ($\mathcal{C}_{\mathcal{DS}}$) and reliability ($\mathcal{R}_{\mathcal{DS}}$) metrics for real images, and qualifications on their use, through goodness of completeness ($\tilde{\mathcal{C}}_{\mathcal{DS}}$) and goodness of reliability ($\tilde{\mathcal{R}}_{\mathcal{DS}}$) results.

[q]As $\mathcal{S}:\mathcal{D}=\mathcal{S}:\mathcal{J}/\mathcal{D}:\mathcal{J}$.
[r]that is, $(\mathcal{S}:\mathcal{J})_{\text{CMP \& EXT}} \sim 6.5 \pm 1.1\%$ implies $\mathcal{S}:\mathcal{D}/\mathcal{D}:\mathcal{J}$ is roughly constant.[q]

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

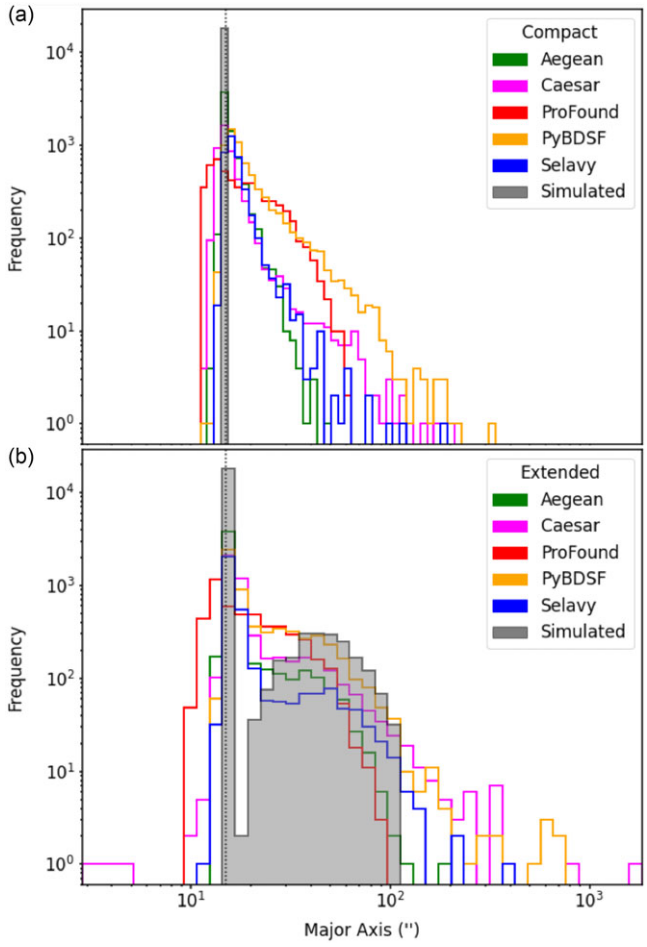


Figure 14. Major-axis distributions for (a) CMP and (b) EXT sources (with $\mathcal{D}$ and $\mathcal{S}$ both included). The vertical dashed-line represents the beam size. The distributions of the injected sources are shown in grey. Recall that size estimates between SFs are not necessarily directly comparable as they are estimated using different methods.

5.3.1. Simulated sources

Fig. 15 shows $\mathcal{C}_{\mathcal{D}}$ vs S/N (top) and $\mathcal{R}_{\mathcal{D}}$ vs S/N (bottom) for CMP (left) and EXT (right) $\mathcal{D}$-images, where the signal-to-noise (S/N) is the ratio of the $\mathcal{D}$-signal to $\mathcal{D}$-noise. Fig. 16 shows the corresponding results for $\mathcal{S}$-images, where S/N is the ratio of the $\mathcal{S}$-signal to $\mathcal{S}$-noise.

For CMP sources all SFs show a similar behaviour in general. The completeness metric starts to decline slowly from 100% at S/N$\lesssim$ 30, and drops rapidly toward zero below S/N$\sim$ 5. False detections are typically limited to about 10% of the sample down to S/N$\sim$ 5, but can appear in some SFs as high as S/N$\sim$ 30. In general terms, Aegean provides the highest level of completeness at any given S/N, with a well-behaved decline in reliability below S/N$\sim 5-6$. At the other end of the scale, Selavy has the lowest level of completeness at any S/N, but the best reliability (fewest false detections).

All SFs seem to miss some bright sources, with Selavy standing out as the poorest in this regard. This is likely due in part to the handling of overlapping sources. Both PyBDSF and ProFound report the largest numbers of false sources (seen in $\mathcal{R}_{\mathcal{D}}$ and $\mathcal{R}_{\mathcal{S}}$) at high S/N. For PyBDSF this is a consequence of overestimating source sizes (Fig. 14), especially in the presence of closely neighbouring sources, or nearby noise spikes, by quite significant amounts in some cases (see Paper II). For ProFound this arises due to the blending of neighbouring sources (see Paper II).

For EXT sources there is generally poorer performance overall compared to those for the CMP sources, most clearly seen in the $\mathcal{D}$-image results (Fig. 15). There are also several artefacts appearing at unphysically low S/N (S/N <1) arising from spurious faint detections. Even at reasonable S/N ($10<$ S/N <100) there is measurable incompleteness, and reliability that dips as low as 80% for some SFs. Here Aegean appears to perform the best, with Selavy showing much poorer performance.

5.3.2. Metrics for real sources

Fig. 17(a) and (b) show $\mathcal{C}_{\mathcal{DS}}$ vs S/N and $\mathcal{R}_{\mathcal{DS}}$ vs S/N, respectively, for CMP sources. As we know what the true sources are, we can explore the validity of $\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$ by removing any false detections in the $\mathcal{D}$ and $\mathcal{S}$-images, that is, we can compute $\tilde{\mathcal{C}}_{\mathcal{DS}}$ (Equation (11)) and $\tilde{\mathcal{R}}_{\mathcal{DS}}$ (Equation (12)). The results are shown in Fig. 17(c) and (d), respectively.

Comparing $\tilde{\mathcal{C}}_{\mathcal{DS}}$ and $\tilde{\mathcal{R}}_{\mathcal{DS}}$ with $\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$, respectively, $\tilde{\mathcal{C}}_{\mathcal{DS}}$ is largely unchanged, while $\tilde{\mathcal{R}}_{\mathcal{DS}}$ does not show the dip in $\mathcal{R}_{\mathcal{DS}}$ around S/N $\sim$ 10 seen for most SFs. It also excludes the decline seen by Aegean at low S/N. This suggests that the apparently poorer estimated reliability in $\mathcal{R}_{\mathcal{DS}}$ arises from the existence of spurious sources detected in the $\mathcal{D}$-image that are (not surprisingly) missed in the $\mathcal{S}$-image.

To quantify these results we define residuals between our unconstrained ($\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$) and constrained ($\tilde{\mathcal{C}}_{\mathcal{DS}}$ and $\tilde{\mathcal{R}}_{\mathcal{DS}}$) metrics: that is, the residual completeness,

$$\delta\mathcal{C}_{\mathcal{DS}} = \tilde{\mathcal{C}}_{\mathcal{DS}} - \mathcal{C}_{\mathcal{DS}}\,, \tag{13}$$

and residual reliability,

$$\delta\mathcal{R}_{\mathcal{DS}} = \tilde{\mathcal{R}}_{\mathcal{DS}} - \mathcal{R}_{\mathcal{DS}}\,. \tag{14}$$

These quantities are shown in Fig. 17(e) and (f), respectively. In general, they are expected to be positive, as there should be an excess of false detections in $\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$ compared to $\tilde{\mathcal{C}}_{\mathcal{DS}}$ and $\tilde{\mathcal{R}}_{\mathcal{DS}}$, by construction. Negative values may appear when real input sources are detected but poorly fit, leading to inconsistent flux densities between the $\mathcal{D}$ and $\mathcal{S}$-images. The figure shows $\delta\mathcal{C}_{\mathcal{DS}}$ predominantly highlighting a small number of spurious PyBDSF detections, and $\delta\mathcal{R}_{\mathcal{DS}}$ emphasising the distribution in S/N of the false detections.

Recognising these limitations, while also noting that for several SFs $\delta\mathcal{C}_{\mathcal{DS}}$ and $\delta\mathcal{R}_{\mathcal{DS}}$ are small, the approach of estimating completeness and reliability for a given finder based on real images is not unreasonable. Clearly it is not as robust as doing so using known injected sources as a reference, but it may be a useful addition to analyses comparing finder performance on real data containing imaging artefacts and other hard to simulate systematics. Similar conclusions can be drawn for the EXT source case (Fig. 17(g) through (l)).

Finally, we note that $\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$ in Fig. 17(a) and (b), respectively, for CMP sources fare much better than their EXT counterparts, (g) and (h), respectively. The metrics in the latter case perform even more poorly than their $\mathcal{C}_{\mathcal{D}}$, $\mathcal{R}_{\mathcal{D}}$, $\mathcal{C}_{\mathcal{S}}$, and $\mathcal{R}_{\mathcal{S}}$ counterparts (Figs. 15 and 16). This could perhaps be due to confusion, given the source density (Fig. 12) and high MADFM statistics (Table 7).

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

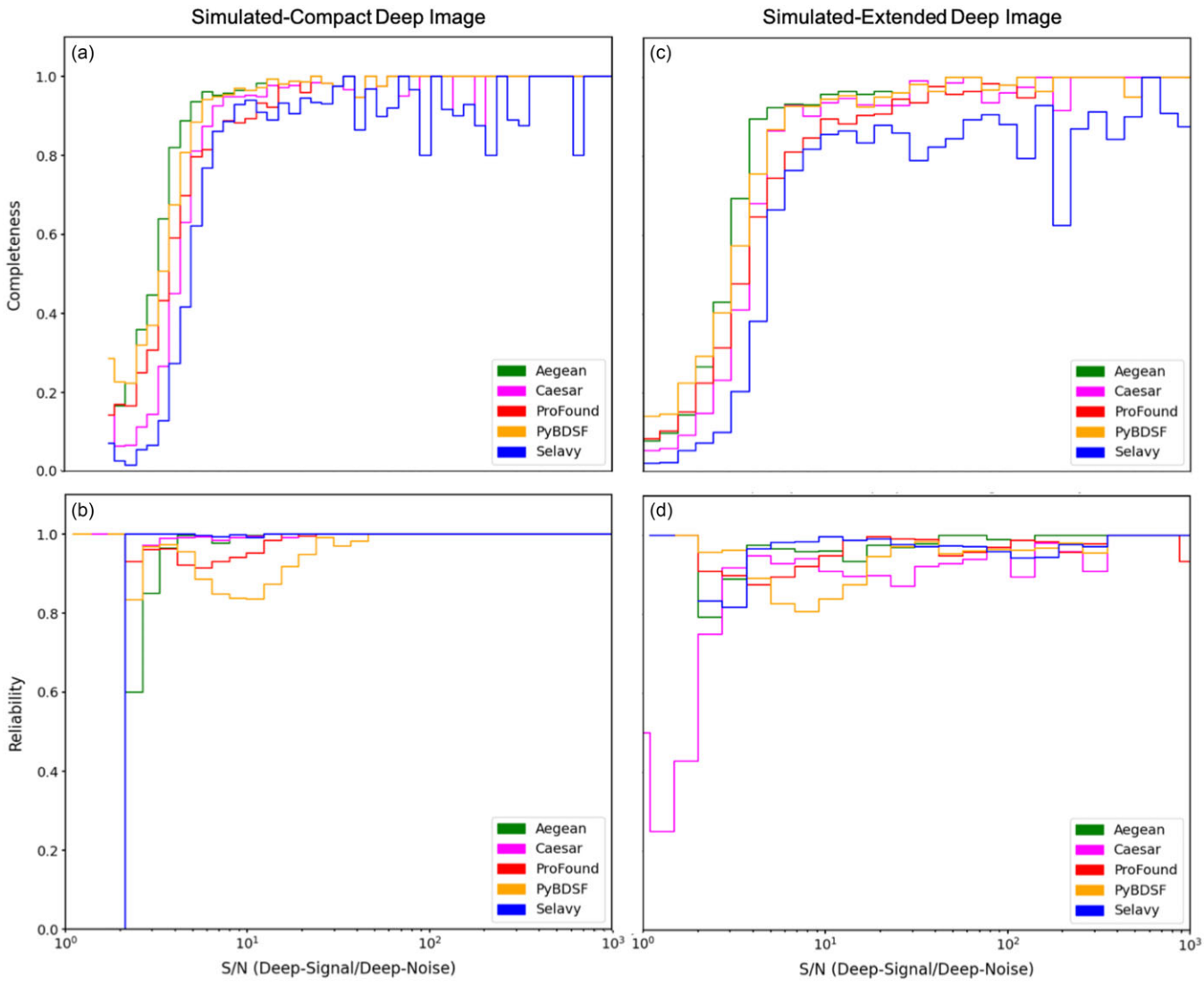


Figure 15. Simulated $\mathcal{D}$-image compact (left) and extended (right) source $\mathcal{C}_{\mathcal{D}}$ (top) and $\mathcal{R}_{\mathcal{D}}$ (bottom) vs S/N ($\mathcal{D}$-signal/Deep-noise) plots. The $\mathcal{D}$-noise is computed using BANE.

5.4. Summary of validation tests

The Hydra software was tested using the Aegean, Caesar, ProFound, PyBDSF, and Selavy SFs, comparing them by first minimising the FDR based on a 90% PRD cutoff, through Typhon. This process was done for $2^\circ \times 2^\circ$ CMP and EXT $\mathcal{D}/\mathcal{S}$-images. The RMS box, MADFM, and source detection statistics were also shown to be consistent with the simulated data, thus validating Hydra's performance. The source size distributions also provided an indication of its performance.

The simulated data was used to develop $\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$ metrics for deep/shallow image pairs, treating $\mathcal{D}$ detections as true sources. This was done by examining these statistics with the erroneous detections filtered out, using knowledge of the underlying injected sources, leaving goodness of completeness, $\tilde{\mathcal{C}}_{\mathcal{DS}}$, and goodness of reliability, $\tilde{\mathcal{R}}_{\mathcal{DS}}$, metrics. Contrasting $\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$ with $\mathcal{C}_{\mathcal{D}}$ (or $\mathcal{C}_{\mathcal{S}}$) and $\mathcal{R}_{\mathcal{D}}$ (or $\mathcal{R}_{\mathcal{S}}$), respectively, a notable degradation in the former was observed for EXT images, most likely due to $\mathcal{DS}$-confusion. That being said, in general, the form of $\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$ remains relatively unchanged, although the $\mathcal{S}$ recovery rate, $\mathcal{S}:\mathcal{D}$, is significantly reduced (Table 8). This suggests that these metrics are good for studying SF performance in real images, given the ability to quantify incompleteness $(1-\mathcal{C})$ and FDR $(1-\mathcal{R})$.

In passing, some observations of SF performance were also made, and are explored in more detail in Paper II. The source detection numbers were comparable between all SFs except for Selavy, which was consistently low (Table 8). Some variability in the residual RMS estimated for the $\mathcal{D}$-mages was observed, with Selavy having unusually high values. For CMP sources the values were comparable except for Selavy, whereas for the EXT case there was significant variation except for Caesar and Profound. As for the $\mathcal{S}$-images the values were all consistent, except for Selavy being significantly high in the EXT source case. The MAFDM statistics were consistent in both cases for all SFs.

For CMP and EXT sources, Aegean had the best $\mathcal{C}_{\mathcal{D}}$ (and $\mathcal{C}_{\mathcal{S}}$) statistics followed by PyBDSF, ProFound, Caesar, and Selavy. Selavy, followed to a lesser degree by Caesar, tends to miss bright sources, more so than the other SFs. For $\mathcal{R}_{\mathcal{D}}$ (and $\mathcal{R}_{\mathcal{S}}$) they also

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

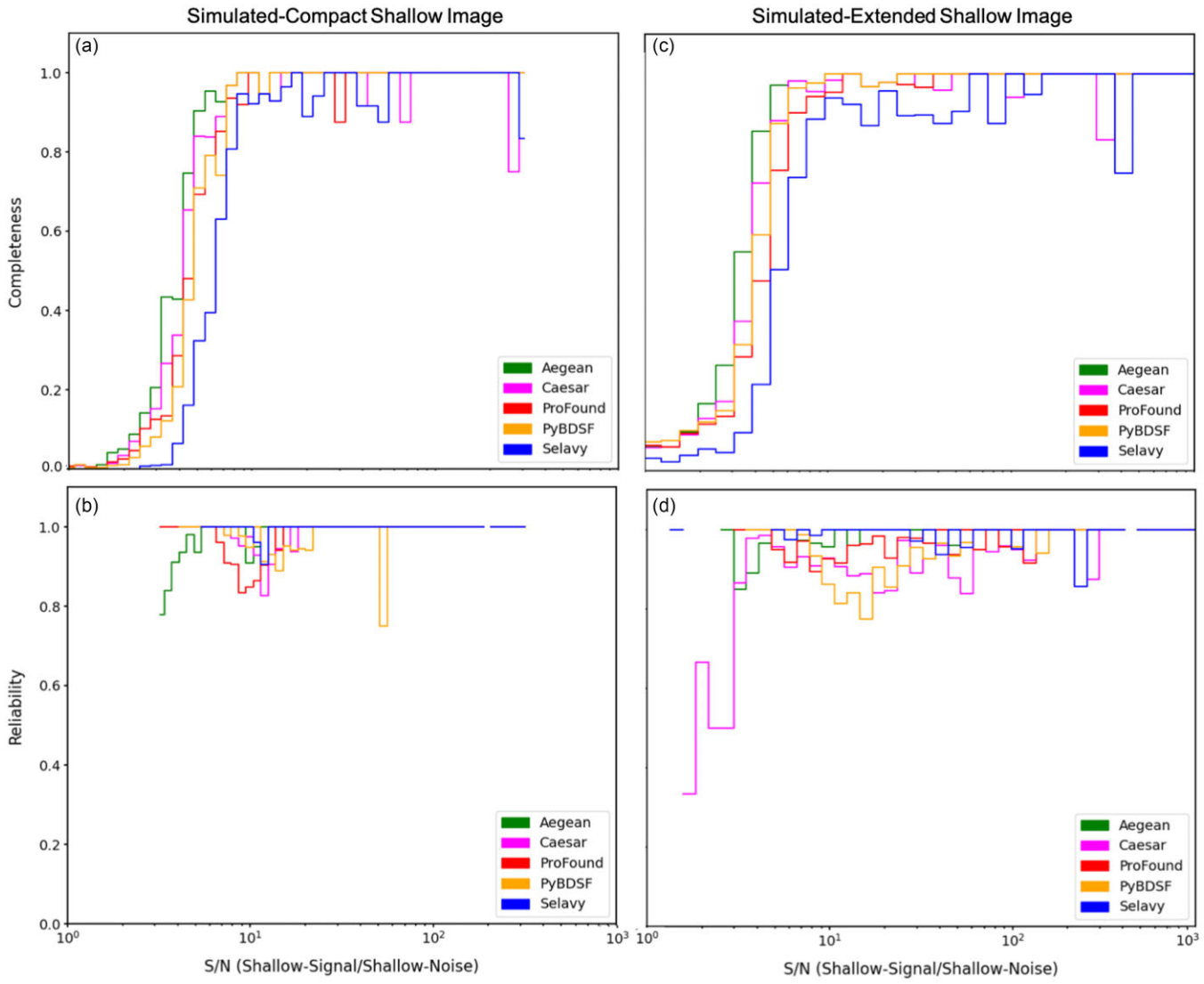


Figure 16. Simulated $\mathcal{S}$-image compact (left) and extended (right) source $\mathcal{C}_{\mathcal{S}}$ (top) and $\mathcal{R}_{\mathcal{S}}$ (bottom) vs S/N ($\mathcal{D}$-signal/$\mathcal{S}$-noise) plots. The $\mathcal{S}$-noise is computed using BANE.

reported the largest number of false sources at high S/N. The quality of the $\mathcal{C}$ and $\mathcal{R}$ statistics is poorer for EXT than CMP sources which is mainly attributed to confusion (see Fig. 12).

6. Conclusions

Radio astronomy has dramatically progressed in the lead-up to the SKA era (see, e.g., Fig. 1 of Norris et al. 2021). The past decades have seen development of technologies and facilities that improve the survey speed, survey depth, and a rapid growth in results from SKA precursors. Ongoing and planned surveys such as VLASS (Lacy et al. 2020; Gordon et al. 2021) with 82% sky-coverage, and EMU (Norris et al. 2011, 2021) with 75% sky-coverage, expect to produce catalogues with source numbers into the millions and tens of millions.

Increasing survey sizes drive a need for SF software with highly robust and well-characterised completeness and reliability statistics. This need has driven source-finding challenges (e.g., Hopkins et al. 2015; Bonaldi et al. 2021) for comparing the various tools and technologies. Some optical SFs have also been applied to radio images, such as SExtractor (Bertin & Arnouts 1996) and ProFound (Hale et al. 2019). Caesar was introduced for handling compact sources jointly with diffuse emission (Umana et al. 2015), through the reprocessing of its residual image (Riggi et al. 2016). These are atural extensions of traditional thinking on radio source extraction. Qualitatively different approaches are also being developed, including the application of machine learning to this field (e.g., Bonaldi et al. 2021; D. Magro et al., in preparation). In the SKA era, it may be that source detection and cataloguing will need to be done on the fly due to the data volume (Bonaldi et al. 2021). A Hydra-like tool may have substantial value in that context, encapsulating the strengths of multiple SFs run in parallel.

The optimum comparison between SFs requires the expertise of the originators to fine tune their performance for a given set of reference images, as pursued in such data challenges. For this reason it is necessary to fairly compare large numbers of SFs on an even footing. Hydra was developed to encapsulate this expertise, in a modular fashion, using Docker containers. Hydra is extensible and the user does not have to be an expert at every SF to

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

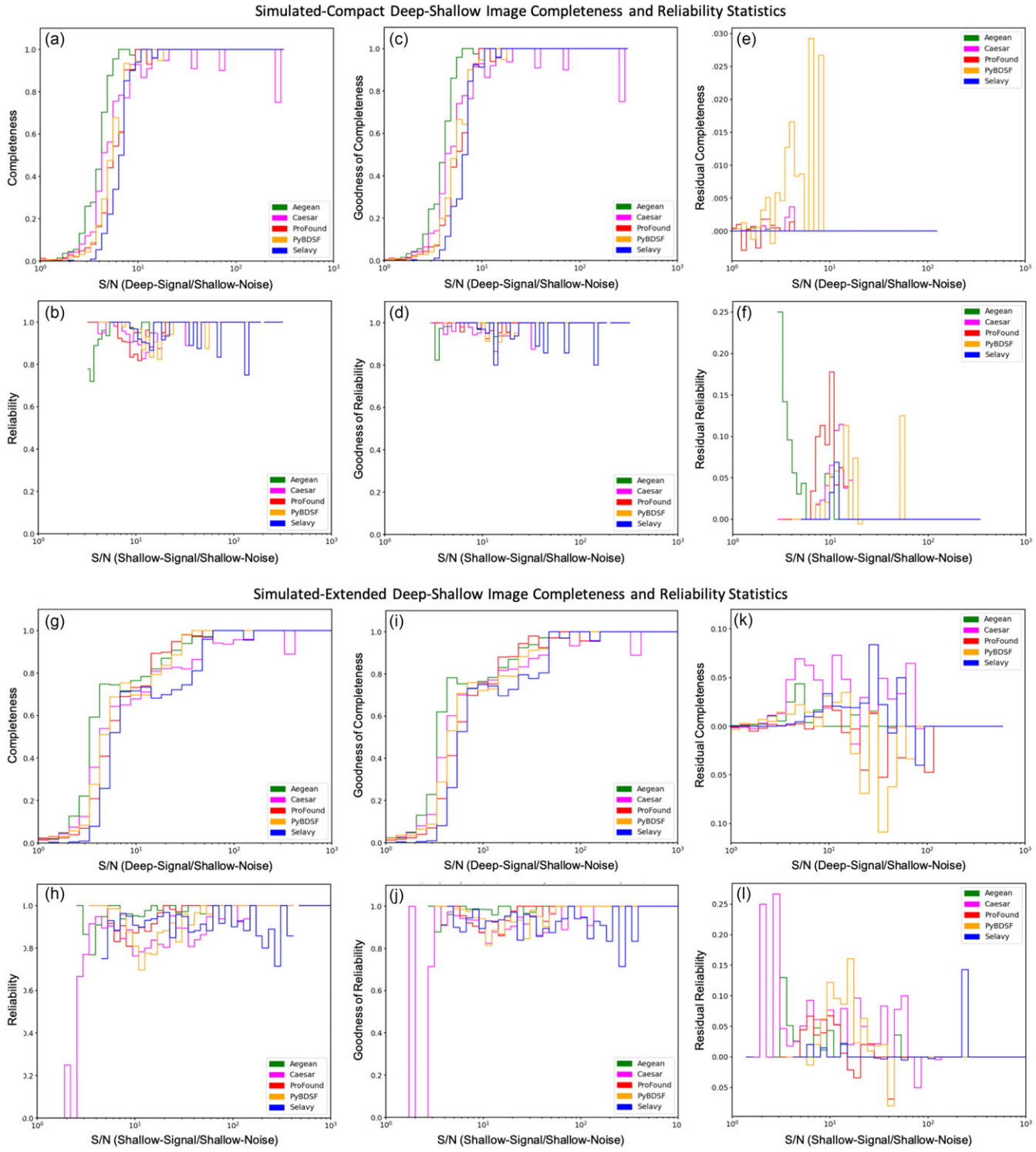


Figure 17. $\mathcal{C}_{\mathcal{DS}}$ (a and g), $\mathcal{R}_{\mathcal{DS}}$ (b and h), $\tilde{\mathcal{C}}_{\mathcal{DS}}$ (c and i), $\tilde{\mathcal{R}}_{\mathcal{DS}}$ (d and j), $\delta\mathcal{C}_{\mathcal{DS}}$ (e and k), and $\delta\mathcal{R}_{\mathcal{DS}}$ (f and l) vs S/N for CMP (top set) and simulated-extend (bottom set) sources. The S/N are expressed as $\mathcal{D}$-signal/$\mathcal{S}$-noise and $\mathcal{S}$-signal/$\mathcal{S}$-noise for completeness and reliability, respectively, where the $\mathcal{S}$-noise is computed using BANE.

use it. Hydra focuses specifically on optimising the RMS threshold and island growth parameters, common to the traditional class of SFs, through the percentage real detections metric, PRD (Equation (1)). This two-parameter optimisation technique is adopted following Hale et al. (2019), who did the optimisation by hand, in a study comparing Aegean, ProFound, and PyBDSF. Hydra also includes an optional background-estimation optimisation step to identify the RMS box and step size parameters, through the mean noise metric, μ (Equation (3)). It is clearly possible to improve Hydra in order to handle more SF-specific parameters.

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

Hydra provides deep ($\mathcal{D}$, i.e the input image) and shallow ($\mathcal{S}$, i.e., the D-image with 5σ noise added) catalogues for each SF, which are linked through a cluster table of overlapping components, or clumps (Fig. 5). The $\mathcal{S}$-image creation and analysis is motivated by a desire to assess a given finder's performance against itself, in the absence of simulated data, treating the $\mathcal{D}$-image version of the SF's catalogue as the ground truth. Each clump has an ID which can be used to locate the associated $\mathcal{D}/\mathcal{S}$-image and residual cutouts (with and without component annotations). $\mathcal{D}/\mathcal{S}$ region files for the full images are also available. Hydra also merges catalogues of known sources (simulated in our case), providing corresponding metrics such as completeness ($\mathcal{C}$) and reliability ($\mathcal{R}$). Hydra also comes with an HTML viewer that allows the user to explore the various data products (Fig. 7).

This paper is part one of a two part series, in which we have introduced the Hydra software, and validated its optimisation algorithms, using simulated-compact (CMP) and simulated-extended (EXT) image data. In addition to the traditional $\mathcal{D}$-image metrics, such as completeness ($\mathcal{C}_\mathcal{D}$) and reliability ($\mathcal{R}_\mathcal{D}$), Hydra introduces a whole new set of metrics, such as $\mathcal{S}$-image completeness ($\mathcal{C}_\mathcal{S}$) and reliability ($\mathcal{R}_\mathcal{S}$), and $\mathcal{DS}$-image completeness ($\mathcal{C}_{\mathcal{DS}}$) and reliability ($\mathcal{R}_{\mathcal{DS}}$), respectively. In this paper we also validated the $\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$ metrics for use with real images, by using our simulated data where the true sources are known. It was found that $\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$ are useful for characterising SF performance, provided one keeps in mind the $\mathcal{D}$ detections are incomplete with a slight degradation in $\mathcal{R}_{\mathcal{DS}}$ for low S/N. In Paper II we evaluate the performance of the SFs using our simulated images along with real data.

Acknowledgements. M. M. Boyce, S. A. Baum, Y. A. Gordon, D. Leahy, C. O'Dea, and A. N. Vantyghem acknowledge partial support from the NSERC of Canada. S. Riggi acknowledges INAF for financial support through the PRIN TEC programme 'CIRASA.' Partial support for L. Rudnick came from U.S. National Science Foundation Grant AST17-14205 to the University of Minnesota. M. I. Ramsay acknowledges support from NSERC and the University of Manitoba Faculty of Science Undergraduate Research Award (USRA). C. L. Hale acknowledges support from the Leverhulme Trust through an Early Career Research Fellowship. Y. A. Gordon is supported by US National Science Foundation grant 2009441. H. Andernach benefited from grant CIIC 138/2020 of Universidad de Guanajuato, Mexico. D. Leahy acknowledges support from NSERC 10020476. M. J. Michaowski acknowledges the support of the National Science Centre, Poland through the SONATA BIS grant 2018/30/E/ST9/00208. S. Safi-Harb acknowledges support from NSERC through the Discovery Grants and the Canada Research Chairs programs, and from the Canadian Space Agency. M. Vaccari acknowledges financial support from the Inter-University Institute for Data Intensive Astronomy (IDIA), a partnership of the University of Cape Town, the University of Pretoria, the University of the Western Cape and the South African Radio Astronomy Observatory, and from the South African Department of Science and Innovation's National Research Foundation under the ISARP RADIOSKY2020 Joint Research Scheme (DSI-NRF Grant Number 113121) and the CSUR HIPPO Project (DSI-NRF Grant Number 121291). E. L. Alexander gratefully acknowledges support from the UK Alan Turing Institute under grant reference EP/V030302/1 and from the UK Science & Technology Facilities Council (STFC) under grant reference ST/P000649/1. A. S. G. Robotham acknowledges support via the Australian Research Council Future Fellowship Scheme (FT200100375). H. Tang greatfully acknowledges the support from the Shuimu Tsinghua Scholar Program of Tsinghua University, the fellowship of China Postdoctoral Science Foundation 2022M721875, and long lasting support from JBCA machine learning group and Doa Tsinghua machine learning group.

Hydra is written primarily in Python, with some elements of Cython (Behnel et al. 2011) and R, along with their standard libraries. Hydra uses alphashape (Bellock, Godber, & Philip 2021), APLpy (Robitaille & Bressert 2012), Astropy (Astropy Collaboration et al. 2013, 2018), Matplotlib (Hunter 2007), NumPy (Harris et al. 2020), and pandas (McKinney 2010; Pandas Development Team 2020) Python libraries commonly used in astronomy. Hydra utilises click, gzip, Jinja, tarfile, and YAML Python libraries as part its overall architectural infrastructure. Hydra encapsulates Aegean (Hancock et al. 2012, 2018), Caesar (Riggi et al. 2016, 2019), ProFound (Robotham et al. 2018; Hale et al. 2019), PyBDSF (Mohan & Rafferty 2015), and Selavy (Whiting & Humphreys 2012) source finder software using Docker. The technical diagrams in this paper we created using macOS Preview and Microsoft PowerPoint. We acknowledge the authors of the aforementioned software applications, languages, and libraries.

M. M. Boyce would like to thank T. Galvin for discussions regarding clustering techniques (re. Section 3.1.4) and extending these concepts to potential computational geometry applications (re., Edelsbrunner 2013) in multiwavelength astronomy.

CIRADA is funded by a grant from the Canada Foundation for Innovation 2017 Innovation Fund (Project 35999) and by the Provinces of Ontario, British Columbia, Alberta, Manitoba and Quebec, in collaboration with the National Research Council of Canada, the US National Radio Astronomy Observatory and Australia's Commonwealth Scientific and Industrial Research Organisation.

The National Radio Astronomy Observatory is a facility of the National Science Foundation operated under cooperative agreement by Associated Universities, Inc.

ASKAP is part of the ATNF which is managed by the CSIRO. Operation of ASKAP is funded by the Australian Government with support from the National Collaborative Research Infrastructure Strategy. ASKAP uses the resources of the Pawsey Supercomputing Centre. Establishment of ASKAP, the Murchison Radio Astronomy Observatory and the Pawsey Supercomputing Centre are initiatives of the Australian Government, with support from the Government of Western Australia and the Science and Industry Endowment Fund. We acknowledge the Wajarri Yamatji people as the traditional owners of the Observatory site.

Data Availability. Hydra is available, along with the data products presented in this paper, by navigating through the CIRADA portal at https://cirada.ca.

References

AMI Consortium, et al. 2011, MNRAS, 415, 2699
Akhlaghi, M., & Ichikawa, T., 2015, ApJS, 220, 1
Alger, M. J., et al. 2018, MNRAS, 478, 5547
Astropy Collaboration, et al. 2013, A&A, 558, A33
Astropy Collaboration, et al. 2018, AJ, 156, 123
Banfield, J. K., et al. 2015, MNRAS, 453, 2326
Banyer, J., Murphy, T., & VAST Collaboration 2012, in ASP Conf. Ser., Vol. 461, Astronomical Data Analysis Software and Systems XXI, eds. P. Ballester, D. Egret, & N. P. F. Lorente (San Francisco: ASP), 725
Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D. S., & Smith, K. 2011, CSE, 13, 31
Bellock, K. E., Godber, N., & Philip, K. 2021, PyPI
Bertin, E., & Arnouts, S. 1996, A&AS, 117, 393
Beucher, S., & Lantuejoul, C. 1979, International Workshop on Image Processing: Real-time Edge and Motion detection/estimation, Rennes, France, 12
Bonaldi, A., et al. 2021, MNRAS, 500, 3821
Boyce, M. M. 2018, Technical report, Multiwavelength Database Prototype Design Tech Notes. Appendix B: Clustering Algorithm. CIRADA (Internal Report)
Boyce, M. M. 2020, Technical report, Source Extraction Comparison Summary Report, Analysis Requirements Specification Summary. CIRADA (Internal Report)
Buxton, R. 2016, The Complete World of Greek Mythology (London: Thames and Hudson Ltd.)
Condon, J. J., Cotton, W. D., Greisen, E. W., Yin, Q. F., Perley, R. A., Taylor, G. B., & Broderick, J. J. 1998, AJ, 115, 1693
Da Costa, G. S. 1992, in Astronomical CCD Observing and Reduction Techniques, ASP Conference Series, ed. S. B. Howell, 23, 90

Edelsbrunner, H. 2013, A Short Course in Computational Geometry and Topology (Springer)
Fender, R., et al. 2007, PoS
Gordon, Y. A., et al. 2021, ApJS, 255, 30
Greisen, W. 2017, AIPS, Memo 117
Gürkan, G., et al. 2022, MNRAS, 512, 6104
Hale, C. L., Robotham, A. S. G., Davies, L. J. M., Jarvis, M. J., Driver, S. P., & Heywood I. 2019, MNRAS, 487, 1486
Hale, C. L., et al. 2021, PASA, 38, E058
Hales, C. A., Murphy, T., Curran, J. R., Middelberg, E., Gaensler, B. M., & Norris R. P. 2012, MNRAS, 425, 979
Hancock, P. J., Murphy, T., Gaensler, B. M., Hopkins, A., & Curran, J. R. 2012, MNRAS, 422, 1812
Hancock, P. J., Cathryn, M. T., & Hurley-Walker, N. 2018, PASA, 35, E011
Harris, C. R., et al. 2020, Nature, 585, 357
Holschneider, M., Kronland-Martinet, R., Morlet, J., & Tchamitchian, P. 1989, Proceedings of the International Conference, Marseille, France, ed. J. M. Combes,A. Grossmann, & P. Tchamitchian (Springer-Verlag), 286
Hopkins, A. M., Afonso, J., Chan, B., Cram, L. E., Georgakakis, A., & Mobasher B. 2003, AJ, 125, 465
Hopkins, A. M., et al. 2015, PASA, 32, E037
Hunter, J. D. 2007, CSE, 9, 90
Huynh, M. T., Hopkins, A., Norris, R., Hancock, P., Murphy, T., Jurek, R., & Whitting, M. 2012, PASA, 29, 229
Koribalski, B. S., et al. 2020, Ap&SS, 365, 118
Kron, R. G. 1980, ApJS, 43, 305
Lacy, M., et al. 2020, PASP, 132, 035001
Lao, B., An, T., Wang, A., Xu, Z., Guo, S., Lv, W., Wu, X., & Zhang, Y. 2021, SciBu, 66, 2145
Lukic, V., de Gasperin, F., & Brüggen, M. 2019, Galaxies, 8, 3
Lutz, R. K. 1980, CJ, 23, 262
López-Caniego, M., & Vielva, P. 2012, MNRAS, 421, 2139
López-Caniego, M., Herranz, D., González-Nuevo, J., Sanz, J. L., Barreiro, R. B., Vielva, P., Argüeso, F., & Toffolatti, L. 2006, MNRAS, 370, 2047
Makovoz, D., & Marleau, F. R. 2005, PASP, 117, 1113
McConnell, D., et al. 2020, PASA, 37, E048
McKinney, W. 2010, in Proceedings of the 9th Python in Science Conference, ed. S. van der Walt, & J. Millman, 56, doi: https://doi.org/10.25080/Majora-92bf1922-00a
Mohan, N., & Rafferty, D. 2015, Astrophysics Source Code Library, ascl:1107.013
Molinari, S., Schisano, E., Faustini, F., Pestalozzi, M., di Giorgio, A. M., & Liu, S. 2011, A&A, 530, A133
Murphy, T., et al. 2013, PASA, 30, E006
Murphy, T., et al. 2021, PASA, 38, E054
Norris, R. 2017, NatAs, 1, 671
Norris, R. P., et al. 2006, AJ, 132, 2409
Norris, R., et al. 2011, PASA, 28, 215
Norris, R. P., et al. 2021, PASA, 38, E046
Pandas Development Team, 2020, pandas-dev/pandas: Pandas
Petrosian, V. 1976, ApJL, 210, L53
Powell, B. B. 2017, The Poems of Hesiod (University of California Press)
Riggi, S., et al. 2016, MNRAS, 460, 1486
Riggi, S., et al. 2019, PASA, 36, 29
Riggi, S., et al. 2021, MNRAS, 502, 60
Robitaille, T., & Bressert, E. 2012, Astrophysics Source Code Library, ascl:1208.017
Robotham, A. S. G., Davies, L. J. M., Driver, S. P., Koushan, S. P., Taranu, D. S., Casura, S., & Liske, J. 2018, MNRAS, 476, 3137
Sadr, A. V., Vos, E. E., Bassett, B. A., Hosenie, Z., Oozeer, N., & Lochner, M. 2019, MNRAS, 484, 2793
Sault, R. J., Teuben, P. J., & Wright, M. C. H. 1995, in ASP Conf. Ser., Vol. 77, Astronomical Data Analysis Software and Systems IV, ed. R. A. Shaw, H. E. Payne, & J. J. E. Hayes (San Francisco: ASP), 443
Serra, P., et al. 2015, MNRAS, 448, 1922
Spreeuw, J. N. 2010, Dissertation, Astronomical Institute Anton Pannekoek, University of Amsterdam, ISBN 978-90-9024055-8
Swinbank, J. D., et al. 2015, A&C, 11, 25
Umana, G., et al. 2015, MNRAS, 454, 902
van Haarlem, M. P., et al. 2013, A&A, 556, A2
Westmeier, T., et al. 2021, MNRAS, 506, 3962
White, R. L., Becker, R. H., Helfand, D. J., & Gregg, M. D. 1997, ApJ, 475, 479
Whiting, M. 2012, MNRAS, 421, 3242
Whiting, M., & Humphreys, B. 2012, PASA, 29, 371
Wu, C., et al. 2018, MNRAS, 482, 1211

Appendix A. Cerberus Code Template Notes

In this appendix we provide more architectural details regarding Cerberus template rules discussed in Section 3.1.2. The intent here is to provide enough detail to give an overall sense of Hydra's extensible nature. Further details can be found in the Hydra user manual.

Fig. A.1. shows a more detailed view of the Cerberus code generation workflow (c.f., Fig. 3). The term *'template'* refers to the overall directory hierarchy, configuration files, and naming conventions. At the lowest level, within the `config` directory, are subdirectories for each of the SFs, containing `*.py` and/or `*.R` script files, `*.dcr` Dockerfiles, and `*.yml` YAML files. The `docker-compose.yml` and `config.yml` files in the main `config` directory provide the glue for building containers and code generation, respectively.

A.1. Containerisation

The general recipe for containerising SFs is as follows.

- Create a Docker build file containing the following:
  - Base operating system environment
  - SF environment with tools
  - SF wrapper script with command-line arguments:
    - Input image path
    - Processing directory path
    - Output directory path
    - Image filename to process
    - RMS-Parameter with default setting
    - Island-Parameter with default setting
    - *RMS box parameters (optional)*
    - FITS catalogue file output flag (default, CSV)
    - Residual image flag
    - Dump flag
    - Help flag
  - Internal directory structure:
    - Script home directory[s]
      - Input subdirectory
      - Processing subdirectory
      - Results subdirectory
  - A container `ENTRYPOINT`
- Update the `docker-compose.yml` configuration file with the container build instructions
- Build the container image

[s]Used for software development and testing.

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

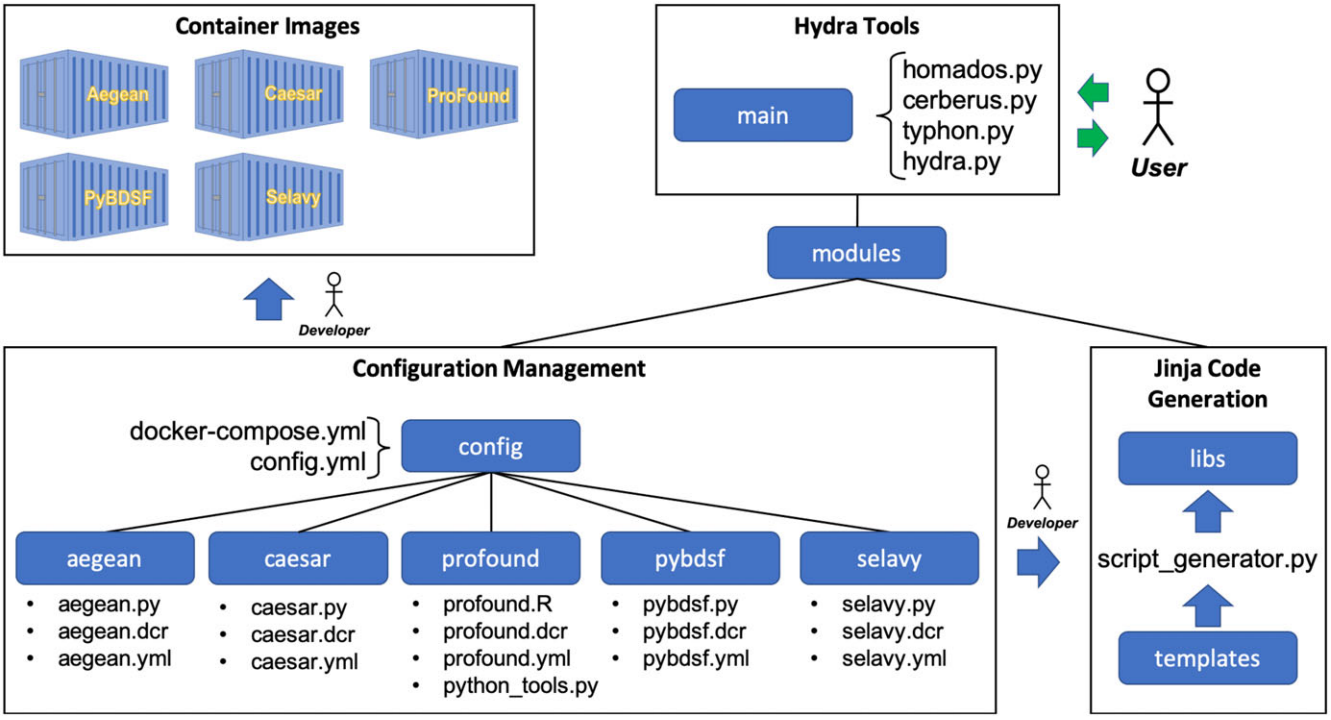


Figure A.1. Detailed breakdown of the Cerberus code generation workflow given in Fig. 3. Fig. A.2 shows a detailed example of an Aegean container image build, utilising `aegean.dcr`, `aegean.py`, and `docker-compose.yml`. Fig. A.3 shows an example of updating `cerberus.py` to include Aegean, through code generation, utilising `aegean.yml` and `config.yml`. All of the information in Configuration Management is accessible to all of the tools within the Hydra software suite.

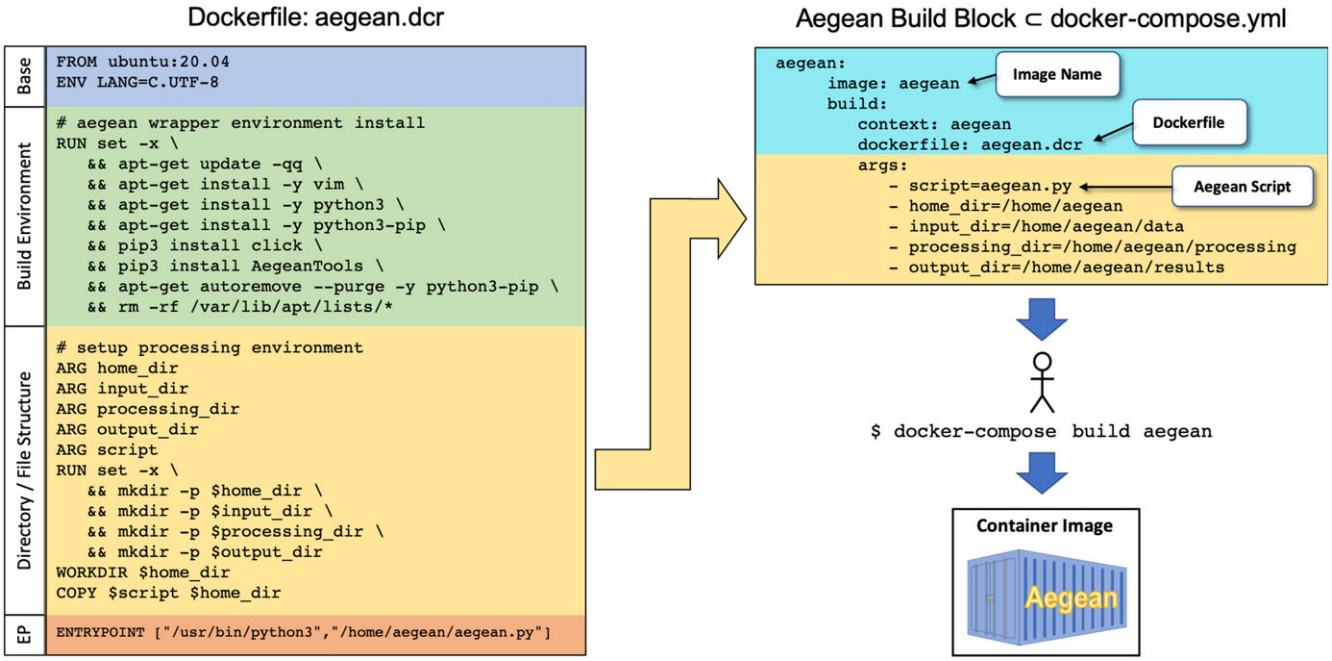


Figure A.2. Example of Aegean containerisation. The Dockerfile, `aegean.dcr`, has four main parts, (1) a base Ubuntu 20.04 operating system, (2) an `AegeanTools` toolbox build environment, (3) a `home_dir`, `input_dir`, `processing_dir`, and `output_dir` directory structure along with a local `script`, `aegean.py`, and (4) an `ENTRYPOINT` (EP) through which `aegean.py` can be externally accessed by `cerberus.py`. The `docker-compose.yml` configuration file contains an Aegean build block (`aegean:`), which has two main parts, (1) a part defining the image name (`aegean`) along with a pointer the Dockerfile (`aegean/aegean.dcr`), and (2) a part containing the directory file structure to be built along with a pointer to `aegean.py`. The container image is built, using this information, with the `docker-compose` command.

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

The input and output directories serve as external mount points, used by the `cerberus.py` wrapper script: that is, the input directory contains the input image, the processing directory contains the SF wrapper script scratch files, and the results directory contains output catalogues, region files, etc.[t] The container `ENTRYPOINT` allows `cerberus.py` external access to the internal script. Aegean is perhaps one of the simplest SFs to containerise, with details shown in Fig. A.2.

As can be seen, Aegean comes as part of an `AegeanTools` toolbox within an Ubuntu 20.04 operating system. The Dockerfile, container directory structure, and local `aegean.py` container wrapper script are defined in the `docker-compose.yml` configuration file. Everything related to Aegean containers, directories, scripts, etc., are all prefixed with `aegean`. Also, `aegean.py` can be accessed internally within the container, for example,

```
/home/aegean# python3 aegean.py --help
Usage: aegean.py [OPTIONS] \
       INPUT_DIR \
       PROCESSING_DIR \
       OUTPUT_DIR FITS_IMAGE_FILE
  Aegean image processing tool.
  inputs:
        INPUT_DIR: location of image_filename.fits
        PROCESSING_DIR: location of scratch directory
        OUTPUT_DIR: location to place results
        FITS_IMAGE_FILE: image_filename.fits
  outputs:
        OUTPUT_DIR/image_filename.aegean.csv
        OUTPUT_DIR/image_filename.aegean.reg
Options:
  --seedclip FLOAT    Island seeding parameter.
  --floodclip FLOAT   Island growing parameter.
  --box-size INTEGER  RMS Box Size (requires: step-size).
  --step-size INTEGER RMS Step Size (requires: box-size).
  --fits              Output FITS catalogue.
  --residual          Output residual and module files.
  --dump              Dump out all processing files.
  --help              Show this message and exit.
/home/aegean#
```

or externally outside of the container, that is,

```
$ docker run --rm -t aegean --help
```

the latter being used by `cerberus.py` (Section 3.1.2). After implementing the above template rules a new container image can be built using the `docker-compose` command within the `config` directory.

A.2. Code Generation

For code generation a metadata file needs to be created (e.g., `aegean.yml`), and then linked to the master configuration file, `config.yml`. This information is then used for code generation through the Jinja template engine. Fig. A.3 shows an example workflow for creating the Aegean module.

All scripts within the Hydra software suite have access to the Configuration Management system in order to perform operations in a generic fashion. For example, `cerberus.py` utilises the

[t]The dump flag (`--dump`) copies the contents of the internal processing directory to the external results directory, which can be used for debugging purposes.

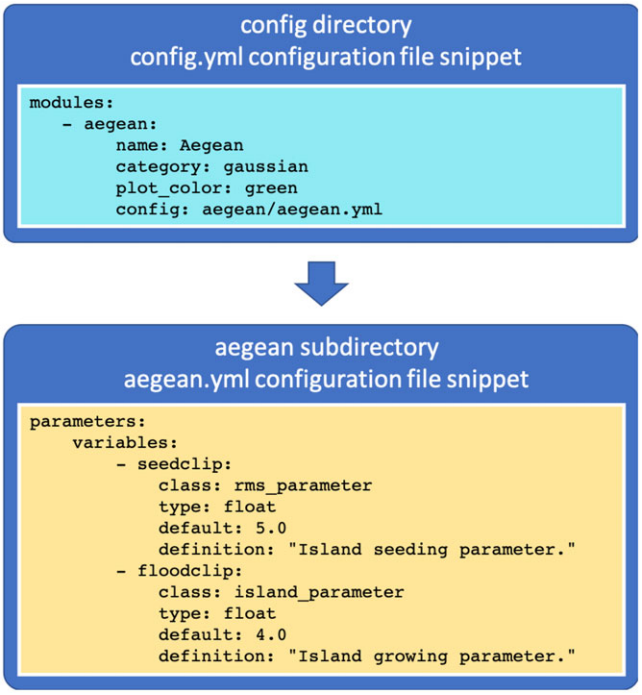


Figure A.3. Example of the addition of an Aegean module for `cerberus.py` through code generation. In short, the developer creates an `aegean.yml` metadata file and links it to the master configuration file, `config.yml`. Then by running the `script_generator.py` script, in the `libs` directory, the module is installed.

docker-compose configuration file for linking calls to the source-finder container images, `typhon.py` utilises the metadata files for parameters and constraints used for source-finder optimisation, `hydra.py` utilises the metadata files for catalogue processing, and so on.

B. Source Finder Implementation Notes

In this section we briefly overview the SFs currently supported by Hydra and their relevant settings.

It should be noted that Aegean, Caesar, and Selavy have various multiprocessor mode implementations wherein large images are split into manageable chunks and processed in parallel in order to reduce the overall processing time (see Hancock et al. 2018; Riggi et al. 2019; Whiting & Humphreys 2012, respectively for details). For the current implementation of Hydra we have chosen to leave these modes disabled. These modes provide various methods for dealing with background noise computations and source detection, which become problematic near edges of sub-images, and consequently have tendencies to bias the statistics especially when comparing against non-multiprocessor SFs, such as ProFound and PyBDSF.

B.1. Aegean

The `AegeanTools` toolbox contains two main items of relevance to this discussion, a background and RMS noise computation script, BANE, and a source finding script, `Aegean` (Hancock et al. 2018). BANE uses a sliding box-car method, with grid-based box and step size parameters, wherein RMS noise estimates are calculated using sigma-clipping. Aegean itself uses a faster, but less accurate, 'zones' algorithm. Aegean can also use the output from BANE, which is the implementation adopted here.

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

The Aegean SF uses a flood-fill algorithm to identify islands above a detection threshold. It then implements a deblending process to determine the number of local maxima through the discrete 2D Laplacian (i.e., curvature) kernel

$$L^2_{xy} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \tag{B1}$$

to identify flux peaks for localised fitting of Gaussian components (Hancock et al. 2012). Its flood-fill algorithm utilises two parameters, a seed threshold parameter, σ_s (i.e., `seedclip`), above which to seed an island, and a flood threshold parameter, σ_f (i.e., `floodclip`), above which to grow an island, such that, $\sigma_s \geq \sigma_f$ (see Table 2). It then convolves the image with Equation (B1), producing a curvature map, from which it implements Gaussian fits to local depressions (i.e., negative curvature bowls, corresponding to local flux density maxima) within the islands.

The implementation of the Aegean (version 2.2.4) module for Cerberus provides the box-car and flood-fill parameters, as per module design requirements.

Here we use the source component catalogue: that is, source island information is not included in this version of Hydra.[u] Of particular interest are the integrated flux densities and fitted component sizes.

B.2. Caesar

Caesar does its source finding using a flood-fill method to obtain blobs (i.e., 'islands'), from which child-blobs (or 'nested blobs') are (optionally) extracted using elliptical-Gaussian-based Laplacian (Equation (B1)) or χ^2 filters (Riggi et al. 2016, 2019). Compact sources, that is, childless-blobs, are subtracted out to leave a residual map with extended sources that can be extracted either though a wavelet transform, saliency, hierarchical-clustering, or active-contour filter.

The RMS and island parameters come in two sets, one for the parents, `seedThr` and `mergeThr`, respectively, and one for the children, `nestedBlobPeakZThr` and `nestedBlobPeakZMergeThr`, respectively. As we are optimising these parameters externally (through the PRD, Equation (1)), we set `compactSourceSearchNIters` = 1 to prevent decrements in `seedThr` by `seedThrStep`. In our implementation, we also search for child-blobs (i.e., `searchNestedSources = true`), with their RMS and island parameters set to the same values as their parents (Riggi et al. 2019). For child-blob filtering we use the Laplacian method (i.e., `blobMaskMethod` = 2, with `fitSources = true`), and for source extraction we use the saliency filter method (i.e., `extendedSearchMethod` = 2; for algorithms, see Riggi et al. 2016).

It should be noted here that searching for child-blobs is not necessary for an image consisting of only point sources; however, for consistency, we prefer to have the same settings for both point and extended sources, for the purposes of comparing performance against other SFs. The only potential impact of this approach is in extra processing time.

Caesar also does background and RMS noise optimisation through any of the following metrics, μ/σ, median/MADFM, biweight, and clipped median/σ. Consequently it does not require pre-tuning like PyBSDF, for example. Here we have chosen the median/MADFM metric (i.e., `bkgEstimator` = 2, with `useLocalBkg = true` and `useBeamInfoInBkg = true`), as it is similar to the pre-optimisation scheme option used by Typhon.

Table B.1 summarises all of the internal settings of Cerberus's Caesar module. Note that we have also set some of the residual image processing flags, so as to remove all source types (`removedSourceType = -1`) with the appropriate thresholding (i.e., `residualZHighThr = seedThr` and `residualZThr = mergeThr`).

The output source components are obtained from the source component catalogue, that is, we do not include the source island catalogue in this version of Hydra. Of particular interest are the integrated flux densities and fitted component sizes.

Table B.1. Caesar (Version 1.1.5) module settings.

Parameter	Value
`useLocalBkg`	`true`
`bkgEstimator`	2
`useBeamInfoInBkg`	`true`
`searchCompactSources`	`true`
`compactSourceSearchNIterse`	1
`searchNestedSources`	`true`
`extendedSearchMethod`	4
`blobMaskMethod`	2
`nestedBlobPeakZThr`	`seedThr`
`nestedBlobPeakZMergeThr`	`mergeThr`
`fitSources`	`true`
`computeResidualMap`	`true`
`removeNestedSources`	`true`
`removedSourceType`	-1
`residualZHighThr`	`seedThr`
`residualZThr`	`mergeThr`
`saveResidualMap`	`true`
`residualMapFITSFile`	`residual.fits`

B.3. ProFound

ProFound uses a watershed deblending process, wherein it systematically searches for the highest flux pixel and expands outwards and downwards in flux to some cutoff, creating a segment (i.e., 'island'), before proceeding to the next highest flux pixel, and so on (Robotham et al. 2018). The end result is the formation of 'flux-mountains' (segments) with peaks and valleys (boundaries between segment groups). After the segments have been determined, it then dilates them until convergence is reached, as determined by a Kron/Petrosian-like dilation kernel (see Section 2), while assigning overlapping segment fluxes to the ones with the most flux (e.g., Fig. B.1). The segment formation threshold is determined by a `skycut` parameter, which corresponds to our RMS parameter, and the segment partitioning is determined by a `tolerance` parameter, which corresponds to our island parameter.

ProFound was designed for optical images and so there are some nuances when it comes to applying it to radio image data

[u] that is, the `--island` flag is not set. For more details, see Aegean footnote *a* in Table 2.

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

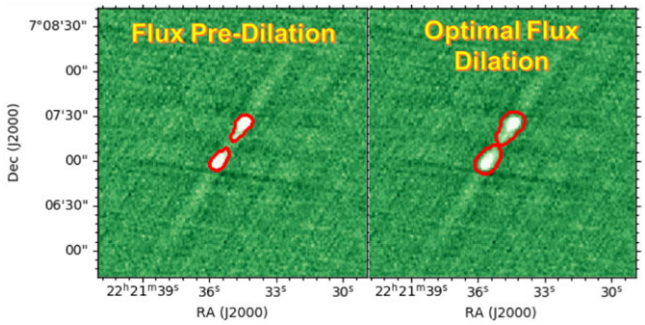


Figure B.1. Example of ProFound dilation process of a small 3′ × 3′ VLASS Epoch 1.1 Quick Look (QL) image cutout, centred at J222135+070712. The cutout was extracted from a QL image tile, available at NRAO (https://science.nrao.edu), and then processed using ProFound in R Studio.

(see Hale et al. 2019). Table B.2 summarises all of the settings internal to the ProFound module,[v] used herein. In addition, a considerable amount of wrapper code was required so as to extract the appropriate 'radio catalogue like' information from its internal hierarchical data structure. Of particular interest are the total flux density and component size, which are pixel sums and flux-weighted fits,[w] respectively, for a given segment (Robotham et al. 2018). Consequently these are not directly comparable to FWHM measurements from Gaussian fits in other SFs.

B.4. PyBDSF

PyBDSF identifies islands by collecting pixels greater than a given flux threshold, `thresh_pix`,[x] and then expands outwards from those pixels in octets above a given island-boundary threshold, `thesh_isl`. Our RMS and island parameters correspond to former and latter thresholds, respectively. After the islands have been determined, it performs multiple Gaussian fits to each island (Hancock et al. 2012).

We follow the same generic recipe as Hale et al. (2019) to accommodate extended sources, as outlined in Table B.3. The `atrous_do` parameter selects the à trous wavelet decomposition (Holschneider et al. 1989) module as one of several options for post-processing. These include shapelet decomposition, à trous wavelet decomposition, PSF variation, polarisation, and spectral index modules. Setting `flag_maxsize_bm` = 100 along with `atrous_do = True` allows for Gaussians greater than the beam size and of varying scales, respectively. Setting `mean_map = "zero"` sets the background mean to zero, enhancing the detection of extended emission.

The PyBDSF module also provides the `rms_box` tuple, so that the RMS box and step sizes can be optimised by Typhon.

The output source components are obtained from the source catalogue: that is, `catalog_type = 'srl'` *via* the `write_image(...)` command. The catalogue does not include empty islands (i.e., `incl_empty = False`). Of particular interest are the total flux densities and component sizes, which are expressed as integrated Stokes I and FWHM's, respectively.

[v] See also ProFound footnote *c* in Table 2.

[w] For the major axis, it is the *'weighted standard deviation along the major axes (i.e., the semi-major first moment, so ∼2 times this would be a typical major axis Kron radius) in units of pix.'* [22]

[x] Re. `process_image(...)` of footnote *d* in Table 2.

Table B.2 ProFound (version 1.13.1, with R version 4.0.3) module default settings, where `box = c(100, 100)`. It should be noted that results can differ radically between versions of ProFound and R. Furthermore, the default settings mentioned in the documentation can differ considerably from what is actually in the source code. This table includes all settings that are deemed important for reproducing our results.

Parameter	Value	Parameter	Value
`pixcut`	3	`iterative`	`FALSE`
`ext`	2	`doclip`	`TRUE`
`reltol`	0	`shiftloc`	`FALSE`
`cliptol`	Inf	`paddim`	`TRUE`
`sigma`	1	`verbose`	`TRUE`
`smooth`	`TRUE`	`plot`	`FALSE`
`SBN100`	100	`stats`	`TRUE`
`size`	5	`rotstats`	`TRUE`
`shape`	`"disc"`	`boundstats`	`TRUE`
`iters`	6	`nearstats`	`TRUE`
`threshold`	1.05	`groupstats`	`TRUE`
`magzero`	0	`group`	`NULL`
`pixscale`	1	`groupby`	`"segim"`
`redosegim`	`FALSE`	`offset`	1
`redosky`	`TRUE`	`haralickstats`	`FALSE`
`redoskysize`	21	`sortcol`	`"segID"`
`box`	`box`	`decreasing`	`FALSE`
`grid`	`box`	`lowmemory`	`FALSE`
`type`	`"bicubic"`	`keepim`	`TRUE`
`skytype`	`"median"`	`watershed`	`"ProFound"`
`skyRMStype`	`"quanlo"`	`pixelcov`	`FALSE`
`roughpedestal`	`FALSE`	`deblendtype`	`"fit"`
`sigmasel`	1	`psf`	`NULL`
`skypixmin`	`prod(box)/2`	`fluxweight`	`"sum"`
`boxadd`	`box/2`	`convtype`	`"brute"`
`boxiters`	0	`convmode`	`"extended"`
`iterskyloc`	`TRUE`	`fluxtype`	`"Raw"`
`deblend`	`FALSE`	`app_diam`	1
`df`	3	`Ndeblendlim`	Inf
`radtrunc`	2		

Table B.3 PyBDSF (version 1.9.1) module settings.

Parameter	Value
`atrous_do`	`True`
`flagging_opts`	`True`
`flag_maxsize_bm`	100
`mean_map`	`"zero"`
`interactive`	`False`
`quiet`	`False`

B.5. Selavy

Selavy is a 'single-pass' raster-scan, or thresholding, type SF (see Lutz 1980), with Duchamp (Whiting 2012), a 3D SF, at its heart (Whiting & Humphreys 2012). Here we are interested in its 2D spatial search features (i.e., `searchType = spatial`).

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press

The algorithm works downwards by growing regions of detection through a threshold parameter (see Fig. 3 of Whiting & Humphreys 2012), `snrCut`, our RMS parameter, after which they can be further extended downwards and outwards through an optional `growthCut` parameter, our island parameter. Further post-processing options are available, such as producing components from multi-Gaussian fitting: that is, `doFit = True`, to turn the fitting option on, `fitTypes = [full]`, to fit all degrees of freedom, and `numGaussFromGuess = True`, to provide an initial guess from the number of distinct peaks found within a given region during thresholding.[y]

Selavy also has various options for background estimates, such as typical μ/σ or more robust median/MADFM based statistics (Whiting 2012). We use a variable sliding box method (`VariableThreshold = True`) with robust statistics (`flagRobustStats = True`) and `Selavy.VariableThreshold.boxSize = (rms_box-1)/2`, where `rms_box` is determined by Typhon.

[y] See Selavy footnote *e* in Table 2.

Table B.4. Selavy (version 1.1.0) module settings.

Parameter	Value
`Selavy.imagetype`	`fits`
`Selavy.flagLog`	`True`
`Selavy.flagDS9`	`True`
`Selavy.Fitter.doFit`	`True`
`Selavy.Fitter.fitTypes`	`[full]`
`Selavy.Fitter.numGaussFromGuess`	`True`
`Selavy.searchType`	`spatial`
`Selavy.VariableThreshold`	`True`
`Selavy.flagRobustStats`	`True`
`Selavy.flagGrowth`	`True`

Table B.4 summarises all of the internal settings of Cerberus's Selavy module.

The output source components are obtained from the source component catalogue: that is, we do not include the source island catalogue in this version of Hydra. Of particular interest are the integrated flux densities and fitted component sizes.

https://doi.org/10.1017/pasa.2023.24 Published online by Cambridge University Press