Publications of the Astronomical Society of Australia (PASA)
doi: 10.1017/pas.2021.xxx.

Hydra: An extensible multi-source-finder comparison and cataloguing tool, currently utilizing Aegean, Caesar, ProFound, PyBDSF, and Selavy.

M. M. Boyce,[1] A. M. Hopkins,[2] S. Riggi,[3] L. Rudnick,[4] M. Ramsay,[1] C. L. Hale,[5] J. English,[1] J. Marvil,[6] M. Whiting,[7] P. Venkataraman,[8] C. P. O'Dea,[1] S. A. Baum,[1] S. Safi-Harb,[1] H. Andernach,[9] J. D. Collier,[10,11] Y. A. Gordon,[1] B. S. Koribalski,[7,11] D. Leahy,[12] M. J. Michałowski,[13] M. Vaccari,[14,15] A. N. Vantyghem,[1] M. Dionyssiou.[8]
[1]Department of Physics and Astronomy, University of Manitoba, 30A Sifton Road, Winnipeg, MB R3T 2N2, Canada
[2]Australian Astronomical Optics, Macquarie University, 105 Delhi Rd, North Ryde, NSW 2113, Australia
[3]INAF, Osservatorio Astrofisico di Catania Via S. Sofia 78, 95123, Catania, Italy
[4]Minnesota Institute for Astrophysics, School of Physics and Astronomy, University of Minnesota, 116 Church Street SE, Minneapolis, MN 55455, USA
[5]School of Physics and Astronomy, University of Edinburgh, Institute for Astronomy, Royal Observatory, Blackford Hill, Edinburgh EH9 3HJ, UK
[6]National Radio Astronomy Observatory, P.O. Box O, Socorro, NM 87801, USA
[7]Australia Telescope National Facility, CSIRO Astronomy and Space Science, PO Box 76, Epping, NSW 1710, Australia
[8]Dunlap Institute for Astronomy and Astrophysics, University of Toronto, 50 St. George Street, Toronto, ON M5S 3H4, Canada
[9]Departamento de Astronomía, DCNE, Universidad de Guanajuato, Callejón de Jalisco s/n, Guanjuato, CP 36023, GTO, Mexico
[10]Inter-University Institute for Data Intensive Astronomy (IDIA), Department of Astronomy, University of Cape Town, Private Bag X3, Rondebosch, 7701, South Africa
[11]School of Science, Western Sydney University, Locked Bag 1797, Penrith, NSW 2751, Australia
[12]Department of Physics and Astronomy, University of Calgary, 2500 University Dr. NW, Calgary, AB T2N 1N4, Canada
[13]Astronomical Observatory Institute, Faculty of Physics, Adam Mickiewicz University, ul. Słoneczna 36, 60-286 Poznań, Poland
[14]Inter-University Institute for Data Intensive Astronomy (IDIA), Department of Physics and Astronomy, University of the Western Cape, Robert Sobukwe Road, 7535 Bellville, Cape Town, South Africa
[15]INAF - Istituto di Radioastronomia, via Gobetti 101, 40129 Bologna, Italy

Abstract

The Evolutionary Map of the Universe (EMU) is a wide-field radio continuum survey whose primary goal is to make a deep 15μJy/beam RMS, intermediate angular resolution $12 - 15''$, 1 GHz survey of the Southern Sky, extending to $+30^\circ$ declination. It is expected to detect and catalogue up to 70 million galaxies, including star forming galaxies (up to $z \sim 1$) and active galactic nuclei. As EMU ramps up, it is of high importance to select a finder with optimized source detection capabilities. We created Hydra to be an extensible multi-source-finder and cataloging tool, which currently includes Aegean, Caesar, ProFound, PyBDSF, and Selavy, that can be used to compare and evaluate different source finders. Hydra provides for the addition of new source finders through containerization and configuration files. Source finders are baselined through optimization of the "percentage real detections" (*i.e.*, minus the percentage difference between detections in the real and inverted images) based on their RMS and island parameters. Hydra provides completeness and reliability plots through observed-deep and generated-shallow 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 source finders in order to choose their desired finder, or a combination thereof. Here we present some initial investigation with Hydra using simulated and EMU pilot survey images.

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

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), delivering data at increasing rates, in the hundreds of gigabytes per second (Whiting & Humphreys, 2012). This places complex requirements on source finders in order to reliably handle compact, 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 source finder fits all these requirements. Hydra is an attempt to get the best of all worlds, by combining the strengths of individual source finders, and improving capability by extending it to include others, as required.

In this paper we provide an overview of radio surveys (§ 1.1) and source finders (§ 1.2), followed by a description of the Hydra software suite (§ 2). Hydra is then used to analyse (*re.* § 3) simulated point-sources (§ 3.1.1), simulated extend-sources (§ 3.2.1), and real sources (§ 3.3), using Aegean, Caesar, ProFound, PyBDSF, and Selavy.[1] This is rounded out with summary and conclusions (§ 4).

1.1 Radio Surveys

In the last two decades or so, radio telescope facilities, such as the Karl G. Jansky Very Large Array (JVLA or, as used herein, VLA), in the Northern hemisphere (*i.e.*, in central New Mexico), hosted surveys, such as, the National Radio Astronomy Observatory (NRAO) VLA Sky Survey (NVSS, Condon et al., 1998), with 82% sky-coverage at 1.4 GHz with $\sigma_{rms} \approx 450\,\mu$Jy/beam (*i.e.*, sensitivity) and $\delta\theta_{res.} \approx 45''$ (*i.e.*, resolution),[2] and the Faint Images of the Radio Sky at Twenty-cm survey (FIRST, Becker et al., 1995; Helfand et al., 2015), with 25% sky-coverage at 1.4 GHz with $\sigma_{rms} \approx 130\,\mu$Jy/beam and $\delta\theta_{res.} \approx 5.4''$.

More recently, improvements to the VLA have allowed it to conduct the VLA Sky Survey (VLASS, Lacy et al., 2020), which has 80% sky-coverage at 3 GHz (*i.e.*, with frequency band, $\Delta\nu \sim 2-4$ GHz) with $\sigma_{rms} \approx 70\mu$Jy/beam and $\delta\theta_{res.} \approx 2.5''$, slated to finish in 2024. VLASS is a continuum survey, providing slow transient detections over 3 epochs over a duration of 7 years, expecting to yield of the order of 2 million sources per epoch (*cf.*, Gordon et al., 2020); a factor of about five larger compared to its predecessor, NVSS ($\sim 400\,000$).

The Giant Metrewave Radio Telescope (GRMT, located in Khodad India) facility's Tata Institute of Fundamental Research (TIFR) GMRT Sky Survey (TGSS, Intema et al., 2017), complements VLASS, covering 90% of the Northern sky at 150 MHz with $\sigma_{rms} \approx 5m$Jy/beam and $\delta\theta_{res.} \approx 25''$, focusing on compact sources (~ 1 million). The Low-Frequency Array (LOFAR, a radio telescope network spanning Europ; Röttgering, 2003) consortium's LOFAR Two-metre Sky Survey (LoTSS, Shimwell et al., 2017), covering 50% of the Northern sky at $\Delta\nu \sim 120-168$ MHz with $\sigma_{rms} \approx 100\mu$Jy/beam and $\delta\theta_{res.} \approx 5''$, further complements continuum research in the areas of massive black holes, galaxies, clusters of galaxies, and large-scale structure (~ 1 million sources).

Rounding out our brief survey of Northern radio telescope facilities, we have the Westerbork Synthesis Radio Telescope (WSRT, located in the Netherlands) and the Allen Telescope Array (ATA, located in California). The WSRT has conducted surveys such as the Westerbork Observations of the Deep APERTIF (APERture Tile In Focus, Oosterloo et al., 2009) Northen-Sky (WODAN, Röttgering, 2010; Riggi et al., 2016), with 0.09% sky-coverage at 1.3 GHz with $\sigma_{rms} \approx 10\mu$Jy/beam and $\delta\theta_{res.} \approx 10''$, and the ATA has conducted surveys such as the ATA Transients Survey (ATATS, Croft et al., 2011), with 1.7% sky-coverage at 1.4 GHz with $\sigma_{rms} \approx 4$mJy/beam and $\delta\theta_{res.} \approx 150''$.

In the Southern hemisphere there is the Square Kilometer Array (SKA) project,[3] whose goal is to build a 1 square kilometer equivalent-dish through aperture synthesis, with a 50 MHz to 14 GHz frequency band, consisting of a series of $50-350$ MHz SKA-low arrays in the Murchison region of Western Australia, and 350 MHz-14 GHz SKA-mid arrays in the Karoo region of South Africa. The SKA is to be built in two phases, SKA1 with 130,000 SKA1-low and 200 SKA1-mid antennas,[4] and SKA2 upgraded to 1 million SKA2-low and 2,000 SKA2-mid antennas. SKA1 is currently underway, since 2018, and, in terms of respective resolution, survey speed, and sensitivity, SKA1-low is expected to be 1.2, 135, and 8 times that of LOFAR, and SKA1-mid is expected to be 4, 50, and 5 times that of the VLA.

The SKA1 precursor telescopes are the Australian SKA Pathfinder (ASKAP,[5] Johnston et al., 2007, 2008) and Murchison Widefield Array (MWA, Lonsdale et al., 2009; Tingay et al., 2013; Wayth et al., 2015) telescopes at the Murchison Radio Observatory (MRO) in the Murchison region of Western Australia, and the Hydrogen Epoch of Reionization Array (HERA, DeBoer et al., 2017) and Karoo Array Telescope (MeerKAT, Jonas, 2009, 2018) telescopes in the Karoo region of South Africa.

MRO is home to ASKAP surveys such as the Evolutionary Map of the Universe (EMU, Norris et al., 2011), with 75% sky-coverage at 1 GHz with $\sigma_{rms} \approx 10\mu$Jy and $\delta\theta_{res.} \approx 10''$, and WALLABY (Koribalski et al. 2020) which is primarily an H I spectral line survey but will also deliver very deep radio continuum images at $\sim$1.4 GHz complementing the lower frequency coverage of EMU, the Variables and Slow Transients (VAST, Banyer et al.,

[1] See Appendix A for notation used herein.
[2] In general, we quote minimum sensitivity and resolution.
[3] https://www.skatelescope.org/
[4] https://www.skatelescope.org/key-documents/
[5] https://www.atnf.csiro.au/projects/askap/index.html

2012; Murphy et al., 2013), with 1.8–24% sky-coverage at $\Delta\nu \sim 1.13 - 1.43$ GHz with $\sigma_{rms} \sim 0.01 - 0.5$mJy/beam and $\delta\theta_{res.} \approx 10''$, with imaging at a $5s$ cadence, and the Rapid ASKAP Continuum Survey (RACS, McConnell et al., 2020), with 83% sky-coverage at $887.5MHz$ (*i.e.*, $\Delta\nu \sim 743.5 - 1031.5$MHz) with $\sigma_{rms} \approx 250\mu$Jy/beam, as well as MWA surveys such as the GaLactic and Extragalactic All-Sky MWA Survey (GLEAM, Wayth et al., 2015; Hurley-Walker et al., 2017), with 60% sky-coverage at $200MHz$ (*i.e.*, $\Delta\nu \sim 72 - 231$ MHz) with $\sigma_{rms} \approx 50$mJy/beam and $\delta\theta_{res.} \approx 2'$.

MeerKAT is conducting surveys such as the MeerKAT International GigaHertz Tiered Extragalactic Exploration (MIGHTEE, Jarvis et al., 2018), with 0.05% sky-coverage at $\Delta\nu \sim 0.9 - 1.67$ GHz with $\sigma_{rms} \sim 1\mu$Jy/beam and $\delta\theta_{res.} \approx 6''$.

These precursor projects are the technology drivers supporting the SKA science goals: *i.e.*, studies of planet formation, signs of exoplanet life, long period gravitational waves, cosmic magnetism, galaxy evolution, fast radio bursts, star formation, and cosmology.

An earlier SKA precursor is the SKA Molonglo Prototype (SKAMP, Adams et al., 2004), a technology upgrade to the Molonglo Observatory Synthesis Telescope (MOST), near the Molonglo River in South Eastern Australia. Examples of surveys performed at this facility are the Sydney University Molonglo Sky Survey (SUMSS, Mauch et al., 2003) and second epoch Molonglo Galactic Plane Survey (MGPS-2, Murphy et al., 2007), collectively providing 22% sky-coverage at 843 MHz with $\sigma_{rms} \approx 8$mJy/beam and $\delta\theta_{res.} \approx 1''$, yielding about 400 000 sources.

Rounding out this brief and incomplete survey of Southern radio telescope facilities, we have the Australia Telescope Compact Array (ATCA) at the Paul Wild Observatory in New South Wales.[6] The ATCA has been used to deliver surveys such as the Stellar Continuum Originating from Radio Physics In Ourgalaxy (SCORPIO, Umana et al., 2015), with 0.0097% sky-coverage at 1.4 GHz with $\sigma_{rms} \approx 30\mu$Jy and $\delta\theta_{res.} \approx 10''$, and the Australia Telescope Large Area Survey (ATLAS, Norris et al., 2006), with 0.0090% sky-coverage at 1.38 GHz with $\sigma_{rms} \approx 10\mu$Jy and $\delta\theta_{res.} \approx 6''$.

As radio telescope technology improves, radio surveys are becoming rich in the number and types of sources detected. EMU is expected to provide up to about 70 million sources (Norris et al., 2011), expanding our knowledge in areas such as galaxy and star formation. This outstrips surveys like VLASS and RACS by a factor of up to 30 (*cf.*, Gordon et al., 2020; McConnell et al., 2020), despite having slightly less sky-coverage. VAST, operating at a cadence of $5s$, 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, extreme scattering events, interstellar scintillation, radio supernovae, and orphan afterglows of gamma-ray bursts (Murphy et al., 2013). In short, surveys are getting to the point where data can no longer be stored, placing a demand on source finder technology, requiring near- or real-time processing strategies.

1.2 Source Finders

The increasing size and data rates of modern radio surveys has increased the need for automated source finding tools with high processing speeds, and good 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 source finders: 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), Python Blob Detector and Source Finder (PyBDSF, formerly Python Blob Detector and Source Measurement or PyBDSM,[7] Mohan & Raffery, 2015), Python Source Extractor (PySE, Spreeuw, 2010; Swinbank et al., 2015), Search and Destroy (SAD, Condon et al., 1998), Selavy (Whiting & Humphreys, 2012), Source Extractor (SExtractor, Bertin & Arnouts, 1996), and SOURCE_FIND (AMI Consortium: Franzen et al., 2011). These have been more recently followed by the introduction of other source finders such as Caesar (Riggi et al., 2016, 2019) and ProFound (Robotham et al., 2018; Hale et al., 2019).

There is also the open-source Source Finding Application (SoFiA), a modular package primarily for application in 3D spectral line surveys, specifically WALLABY (Koribalski et al. 2020). It contains several source finders, source reliability estimates as well as extensive source parametrization as described by Serra et al. (2015). A recent paper by Westmeier et al. (2021) introduces SoFiA-2, an improved parallel implementation of SoFiA for optimal running on the Pawsey Supercomputers.

By and large there is no *'one source finder fits all,'* solution, each is typically optimised for specific tasks (Hopkins et al., 2015). In the grossest sense, there are source finders designed to handle compact sources, including Aegean, APEX, BLOBCAT, CuTEx, Duchamp, PyBDSF, PySE, SAD, Caesar, and Selavy, and those aimed at better capturing the properties of extended

[6] https://www.narrabri.atnf.csiro.au

[7] Software written by Niruj Mahon Ramanujam, Alexander Usov and David Rafferty (see also Hopkins et al., 2015).

sources, such as BLOBCAT, Caesar, IFCA BAF/MF, ProFound, SExtractor, and SOURCE_FIND.

APEX (*re. Spitzer*, Makovoz & Marleau, 2005), CuTEx (*re. Herschel*, Molinari et al., 2010), ProFound (*e.g.* VIKING,[8] Robotham et al., 2018), and SExtractor (a photometry tool, Bertin & Arnouts, 1996) were developed originally to analyse optical or infrared images, whereas Aegean, BLOBCAT, Caesar (*e.g.* SCORPIO, Riggi et al., 2016, 2021), Duchamp, IFCA BAF/MF, PyBDSF (*e.g.* VLASS, Gordon et al., 2020), PySE (*re.* LOFAR, Fender et al., 2007; van Haarlem et al., 2013), SAD (*re.* NVSS, Condon et al., 1998), Selavy, and SOURCE_FIND (*re.* AMI[9] pipeline, AMI Consortium: Zwart et al., 2008) have their origins in radio astronomy.

Atop this broad brush of generalities, some of them have their specializations: *e.g.*, BLOBCAT is designed to handle linear polarizations (Hales et al., 2012), Duchamp is designed for processing data-cubes with HI observations in mind (Whiting, 2012), Caesar (Riggi et al., 2016) and ProFound (Boyce, 2020) are good for handling faint/diffuse extended sources, CuTEx was designed for extracting compact sources under intense background fluctuations (Molinari et al., 2011), and PySE was designed for transient pipeline processing (Fender et al., 2007; van Haarlem et al., 2013). Selavy uses Duchamp as a backend, and is a new breed of source finder – the *"Next Generation"* (NXGEN) – designed for handling high data throughput by utilizing multiple processors (*re.* EMU, Whiting & Humphreys, 2012). Aegean (Hancock et al., 2018) and Caesar (Riggi et al., 2019) are also a part of the NXGENS.

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

For the background estimation, most source finders used in radio astronomy, such as Aegean, PyBDSF, and Selavy, tend to use a sliding box method, where background noise estimates are based on the box and sliding-step sizes. 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 sources (*cf.*, Huyn et al., 2012).

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 (μ) background and RMS (σ) noise used by PyBDSF (Mohan & Raffery, 2015), or median background and Mean Absolute Deviation From the Median (MAD, or MADFM herein: *cf.*, 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 § 2.1.1; Da Costa, 1992; Bertin & Arnouts, 1996; Huyn et al., 2012; Akhlaghi & Ichikawa, 2015; Riggi et al., 2016) over the entire image; whereas, PySE performs σ-clipping locally (see Hopkins et al., 2015). ProFound, an optical source finder, which is making inroads into radio astronomy (Hale et al., 2019), also uses a σ-clipping schema (*re.*, MAKESKYGRID, Robotham et al., 2018). Casaer provides several options, *i.e.* 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 function of pixel location. It is important that these estimates are as optimal as possible, as they can have a significant effect on the performance of a finder (Huyn 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 neighboring 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 (*i.e.*, it uses an island-shaped aperture: *cf.*, 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, though a watershed deblending technique.[10] 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 & Raffery, 2015), and PySE (Hopkins et al., 2015) use variations on this theme.

The component extraction phase is perhaps the most varied in terms of methods. 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 utilized by Duchamp (Whiting & Humphreys, 2012), and, in turn, is also employed by Selavy. These peaks are then fitted by Gaussians (*i.e.*, elliptical Gaussians) producing a component catalogue. Other source finders use a range of Gaussians to fit to an island, using various criteria. PyBDSF (Mohan & Raffery, 2015) and PySE (Spreeuw, 2010; Swinbank et al., 2015) fall into this category; wherein, for example, PySE uses a χ^2 criterion

[8] *i.e.*, VISTA (Visible and Infrared Survey Telescope for Astronomy) Kilo-degree INfrared Galaxy.

[9] *i.e.*, Arcminute Microkelvin Imager.

[10] The term "watershed," refers to drainage basins formed from streams running between mountains (islands) during a rainfall (*a la*, steepest decent, Beucher & Lantuejoul, 1979).

(see Hopkins et al., 2015).

There are also a class of source finders 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 sending 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 source finder algorithms are just the tip of the iceberg of possibilities (*cf.*, Hancock et al., 2012; Hopkins et al., 2015; Bonaldi et al., 2021). In our initial implementation of Hydra we have chosen to explore a representative set of commonly-used source finders, *i.e.*, Aegean, Caesar, ProFound, PyBDSF, and Selavy, bridging some of the major features.

2 HYDRA

Hydra is a software tool capable of running multiple source finders. It is extensible, in that other source finders can be added in a containerized fashion by following a set of straightforward template rules. It provides diagnostic information such as completeness and reliability. Statistical analysis can be based on injected source catalogues from simulated images or on real ("deep") images used as ground-truths for detections in their "shallow" counterparts (*i.e.*, input "deep" images with artificial noise added).

Hydra is innovative in that it minimizes the false detection rate (*re.* Equation 2 in §2.1.3 below) of the source finders by adjusting their detection threshold and island growth parameters (*e.g.*, Hale et al., 2019). It also has an optional feature for RMS box optimization. The software can be upgraded for optimizing other parameters with fairly little effort. This is an essential step in automation, especially when dealing with large surveys such as EMU.

For the purposes of placing the source finders 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 speed (which we also benchmark, *re.* Appendix E). In addition, we keep all of the internal parameters of all of the source finders fixed, instead of tweaking them by hand for each use case. Every effort has been made to keep each source finder as generic as possible regarding their internal parameters, leaving the optimization of there exposed parameters to Hydra.

2.1 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.)*

Figure 1 shows 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 RMS and mean (μ) noise values, and image manipulation, such as inversion and adding noise. Cerberus is an extensible multi-source-finder 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 & Raffery, 2015), and Selavy (Whiting & Humphreys, 2012). Typhon is a tool for optimizing the source finder 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 catalogues, residual images, and region files. Echidna is a planned catalogue stacking and integration tool, internal to Hydra.

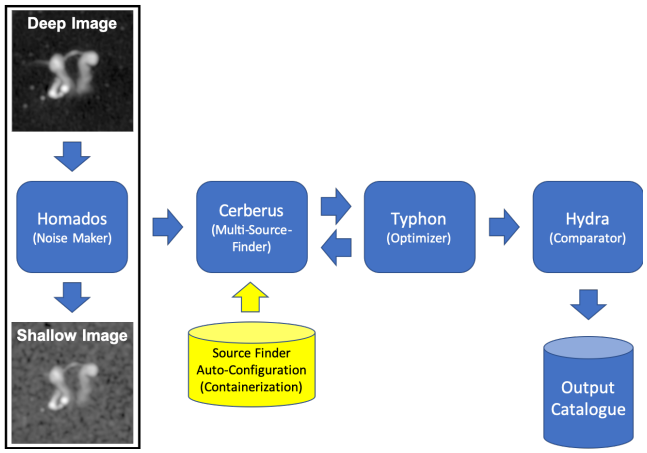


Figure 1. Hydra software suite workflow.

2.1.1 Homados

The main purpose of Homados is to produce a shallow image by adding noise to a reference image. The reference image will often be referred to as the deep image. For the purposes of evaluating source finders these *"deep–shallow"* image pairs can be used to generate completeness and reliability plots (*e.g.*, Hopkins

et al., 2015), assuming source-finder detections in the deep image are real.

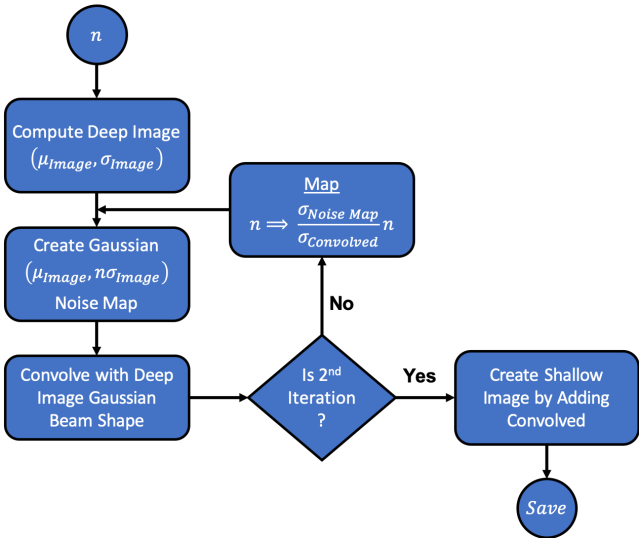


Figure 2. Homados shallow image generation algorithm.

A shallow image is created by adding a deep image's beam convolved with a Gaussian noise map. Figure 2 shows the algorithmic flow chart. It starts by computing the deep image mean and RMS noise, μ_{image} and σ_{image} respectively, using sigma-clipping (Akhlaghi & Ichikawa, 2015). This in turn is used to create a $\mu_{image} \pm n\sigma_{image}$ Gaussian noise map which is then convolved with the deep image's beam shape, defined by its `BMIN`, `BMAJ`, and `BPA` beam parameters (Greisen, 2017) to mimic the correlated noise properties of interferometer images. Here the input parameter n, is the desired shallow image noise level (*i.e.*, $\sigma_{shallow} \sim n\sigma_{deep}$). Because the convolution does not conserve the original noise level the convolved image must be scaled using

$$n \Rightarrow \frac{\sigma_{noise\ \ map}}{\sigma_{convolved}}\, n\,, \tag{1}$$

so as to produce the correct noise level before adding it to deep image. The $\sigma_{noise\ \ map}$ and $\sigma_{convolved}$ values are obtained from the Gaussian noise map and convolved image, respectively, using sigma-clipping.

Figure 3 shows an example Homados shallow image generation run for an ATLAS CDFS DR1 tile (Norris et al., 2006). The noise level was set to $n = 5$, which will be assumed for the rest of this paper. As can be seen, the algorithm provides reasonable results.

The following summarizes the sigma-clipping algorithm used by Homados in its estimation of the noise level of an image:

- Calculate the standard deviation, σ, and the median, m.
- Keep values in the range $[m - \alpha\sigma,\, m + \alpha\sigma]$.

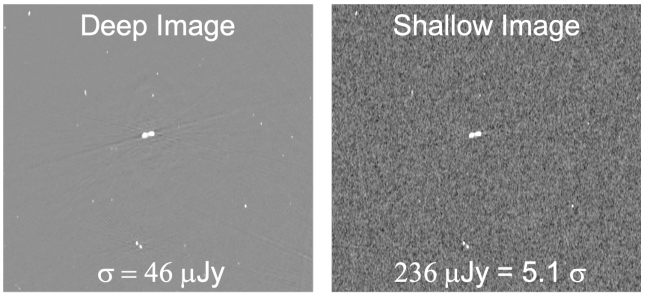


Figure 3. Homados shallow image generation example, using an ATLAS CDFS DR1 2.2° × 2.7° tile (Norris et al., 2006). The figures show deep (left) and shallow (right) images, zoomed in. The noise level, n, was set to 5.

- Repeat the previous steps until

$$\frac{\sigma_{old} - \sigma_{new}}{\sigma_{new}} < tol$$

or

$$N_{iters} > N_{iters}^{max}\,.$$

- Return m, μ, σ, *etc.*

where α is the clipping rate, σ_{old} is from the previous step, σ_{new} is from the current step, tol is the desired tolerance level, N_{iters} is the number of iterations, and N_{iters}^{max} is the maximum number of iterations. Homados has been optimized with default parameter values of $\alpha = 3.25$ and $N_{iters}^{max} = 5$.

Figure 4 shows an example Homados image statistics run for our sample tile. The sigma-clipping algorithm converges fairly rapidly.

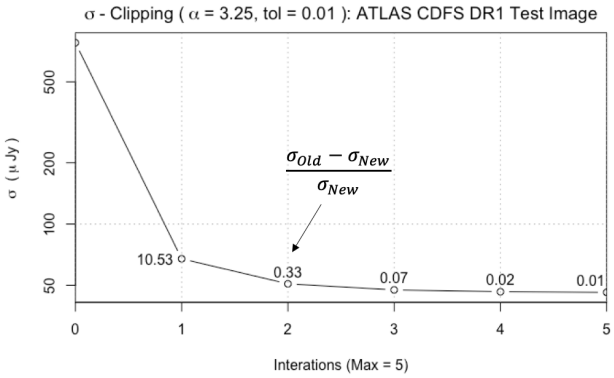


Figure 4. Homados ATLAS CDFS DR1 2.2° × 2.7° deep image sigma-clipping example, showing the estimated noise at each iteration. The intensity ranges before and after sigma-clipping are $\sim [-2080, 85200]\,\mu$Jy and $[-151, 150]\,\mu$Jy, respectively. The final median, mean, and RMS values are $m = -0.254\,\mu$Jy, $\mu = -0.0952\,\mu$Jy, and $\sigma = 46.2\,\mu$Jy, respectively.

In addition to shallow image generation, and image statistics, Homados also performs image inversion. The

first is used for creating completeness and reliability statistics (through Typhon, § 2.1.3), and the latter is used for source-finder optimization (through Cerberus, § 2.1.2).

2.1.2 Cerberus

Cerberus as a standalone tool provides a generic interface to multiple source finders. It is created through a set of Docker[11] and YAML[12] configuration files which are used by a Jinja[13] template-driven code generator to generate containerized source-finders accessible through a Python wrapper script, `cerberus.py` (Figure 5).

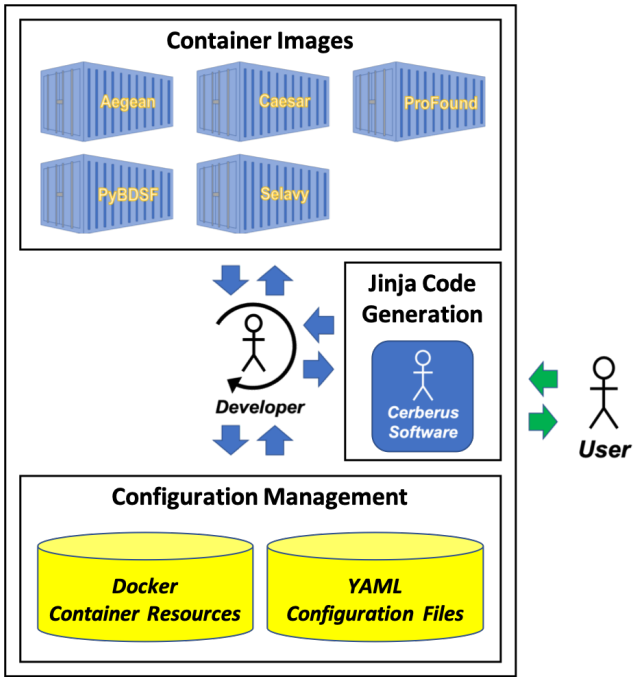


Figure 5. Cerberus code generation workflow.

Currently the code supports Aegean, Caesar, ProFound, PyBDSF, and Selavy source-finder modules. The following output is from template generated code.[14]

```
$ 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 source finders, or modules, can be added by following a set of template rules:

- Create a containerized source finder wrapper:
  - Create a source finder wrapper script
  - Create a Docker build file wrapper
  - Update the master docker-compose build file
  - Build the container image
- Update the `cerberus.py` script: *i.e.*,
  - Create a YAML parameter metadata file
  - Update the master YAML configuration file
  - Run the Jinja script generator tool
- Test the Hydra software suite
- Update the Git[15] repository

Figure 5 summarizes this *"high-level"* workflow: containers for each source finder 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 source finder, in order for the Jinja template-driven script generator to update Cerberus. More details are given in Appendix B; here, we focus on the more salient points.

The key requirement is that each source finder must provide access to two of their main tuning parameters, *"RMS-like"* and *"Island-like"* parameters. The former is related to a *"S/N-like"* detection threshold and the latter is related to an island *"growth-like"* parameter. The following shows the Cerberus command-line interface to the Aegean source finder.

```
$ python cerberus.py aegean --help
Usage: cerberus.py aegean [OPTIONS] FITS_IMAGE_FILE

  Process FITS_IMAGE_FILE using aegean source finder.

Options:
  --output-dir TEXT   Results output directory.
  --seedclip FLOAT    Islands 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.
$
```

Here the `seedclip` and `floodclip` represent the former and latter parameters, respectively. The `seedclip` parameter is the clipping value (in σ's) for seeding islands,

[11] `https://www.docker.com`
[12] `https://yaml.org`
[13] `https://jinja.palletsprojects.com`
[14] The command-line interface for all Hydra tools is standardized 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.
[15] `https://git-scm.com`

and the `floodclip` parameter is the clipping value (in σ's) for growing islands (Hancock et al., 2012). Table 1 summarizes the parameters for the currently supported source finders. These parameters are used by Typhon to calibrate the source finders.

Notice also, that the `aegean` command has optional box size and step size parameters, which can be linked in the configuration files. These parameters can be used by Typhon to optimize a source finder's RMS box parameters, prior to optimizing the RMS and island parameters.

2.1.3 Typhon

Typhon is a tool for optimizing the source finders 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:

$$\mathrm{PRD} = \frac{N_{image} - N_{inv.\ image}}{N_{image}}\,100 \tag{2}$$

where N_{image} is the number of detections in the original image and $N_{inv.\ image}$ is the number of detection in the inverted image. 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 false detection rate (Whiting, 2012) can be reduced by optimizing the PRD.

Typhon uses the source finder RMS and island parameters to optimize the PRD. Figure 6 shows Typhon generated PRD curves for a $2^\circ \times 2^\circ$ simulated deep image along with its corresponding shallow image. 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. The 90% to 98% PRD range has been investigated and the former threshold seems to provide reasonable results. Hale et al. (2019) uses a 98% PRD to baseline their source finders, beyond which the detection rate degrades. At that cutoff, however, we tend to find a non-scale-invariant increase in the RMS threshold, due to the rapidly changing shape of the PRD curve.

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

$$1.5\sigma \le \mathrm{RMS} \le \mathrm{RMS}_{max}\,, \tag{3}$$

where

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

The image mean, RMS, and maximum noise parameters μ, σ, and I_{max}^{final}, respectively, are obtained through sigma-clipping, *via* Homados. So for the example given in Figure 4, where $\mu = -0.0952\,\mu$Jy, $\sigma = 46.2\,\mu$Jy, $I_{max}^{final} = 151$, we find $\mathrm{RMS}_{max} = 4\sigma$. The 1.5σ is a practical lower limit for diagnostics, below which the fraction of false detections is excessive. In general, Typhon samples the PRD backwards in the RMS parameter, while varying the island parameter at each step, until the 90% threshold is reached.

The island parameter values are source finder specific and are typically defined over a finite range. This information is stored in Configuration Management (Figure 5), which is accessible to all Hydra scripts. Table 2 shows the island parameter ranges for the currently installed modules. Typhon uses this information along with the soft constraint $\sigma_{island} < \sigma_{RMS}$, defined in Configuration Management,[16] as it searches the parameter space. This soft constraint corresponds to the rule of thumb that the island growth threshold should not exceed the RMS source detection threshold: should $\sigma_{RMS} \le \sigma_{island}$ during optimization, then σ_{island} is forced to $0.999\,\sigma_{RMS}$.

Typhon will also perform an initial RMS box optimization before optimizing the PRD if it is configured to do so. This is of particular importance for extended objects or around bright sources; especially for Gaussian source extraction based finders, such as Aegean, PyBDSF, and Selavy. In fact, for PyBDSF, it is recommended these parameters be optimized.[17] Regardless, using this option serves as another baseline for calibrating source finders (*e.g.*, Huyn et al., 2012; Riggi et al., 2016) for comparison purposes. Here we use this feature for Aegean and PyBDSF.

The algorithm uses BANE, from the Aegean toolbox (Hancock et al., 2018), to produce RMS images as a function of the grid (RMS) box and step sizes, from which it finds an optimal value for a given metric. The optimal RMS box parameters are then fed to the appropriate source finder.

Various metrics can be chosen upon which to optimize the RMS image: μ, median, σ, MADFM[18] and ΣI^2. These metrics were tried for $2^\circ \times 2^\circ$ simulated and EMU-pilot images, by varying the RMS box parameters and visually inspecting the RMS images. The metric optimization that proved the most effective was the minimization of the image mean, μ. Figure 7 shows the variation in RMS box optimization metrics for a $2^\circ \times 2^\circ$ simulated deep image and its corresponding shallow image. (Similar results are found for our EMU images, but with larger box sizes; presumably due to the existence of extended sources.) It is evident that μ produces the most robust results.

The practical boundaries of our RMS box parameters

[16] This option can be turned off or on a source-finder basis. Regardless, it is required for PyBDSF.

[17] See `rms_box` discussion at URL in footnote *d* of Table 1.

[18] NB: Our MADFM estimators are normalized by 0.6744888 (Whiting, 2012)

Table 1 Cerberus RMS and Island parameter definitions in σ's wrt background, with soft constraint $\sigma_{Island} < \sigma_{RMS}$.

Source Finder	RMS Parameter Name	Default	Description	Island Parameter 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`	2.82	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://www.astron.nl/citt/pybdsf/process_image.html`
[e] `https://www.atnf.csiro.au/computing/software/askapsoft/sdp/docs/current/analysis/selavy.html`

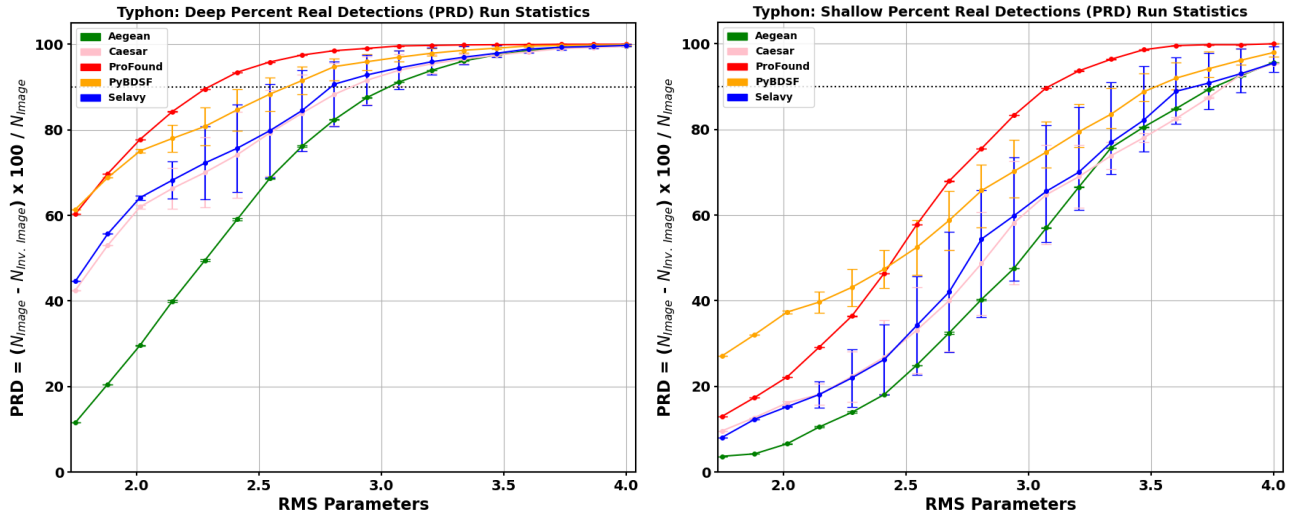


Figure 6. Example Typhon PRD of a $2° \times 2°$ simulated deep image (left) and its corresponding shallow image (right); for colour codes, see Table 20. The variation in the PRD with the RMS parameters (in σ units) is represented along the horizontal axis, and the variation in the PRD with the island parameters is represented by the error bars. The RMS and island parameter names, by source finder, are given in Table 1. The dashed horizontal line indicates the 90% PRD level.

are as follows,

$$\left.\begin{aligned} 3 \leq \frac{\text{box\_size}}{[4(\texttt{BMAJ}+\texttt{BMIN})/2]} \leq 8 \\ \frac{1}{4} \leq \frac{\text{step\_size}}{\text{box\_size}} \leq \frac{1}{2} \end{aligned}\right\}, \qquad (4)$$

over a rough 6×3 box_size by step_size search grid. The $4(\texttt{BMAJ}+\texttt{BMIN})/2$ factor represents the BANE default box size;[19] where we assume a square box, for simplicity. The 1/4 and 1/2 bounds are inspired by the hints given in the PyBDSF manual (*i.e.*, 1/3 and 1/2),[20] and are somewhat pragmatic, in terms of providing a smoothly sliding box. The limits 3 and 8 were determined by trial and error, so as not to produce box sizes too small to be biased

[19] See Aegean footnote *a* in Table 1.
[20] See PyBDSF footnote *d* in Table 1.

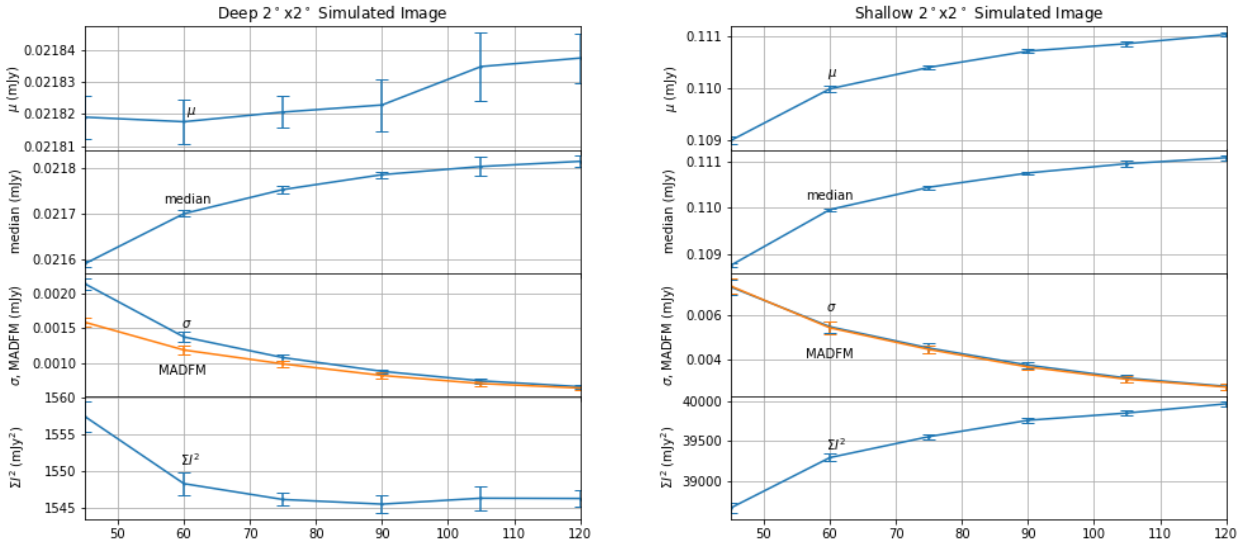


Figure 7. RMS box optimization metrics for a $2^\circ \times 2^\circ$ simulated deep image (left) and its corresponding shallow image (right). The step size variations are represented by the error bars on the plots, with the curves representing their mean values. The simulated beam parameters are `BMAJ` = `BMIN` = $15''$ and `BPA` = 0. We note that, σ and MADFM are divergent, and ΣI^2 is somewhat unstable, within the range of interest. Only min(μ) seems to scale with complexity; as is evident between deep and shallow images, but as well for real world images, as found for our $2^\circ \times 2^\circ$ EMU pilot cutout (not shown).

Table 2 Configured island parameters.

Source Finder	Island Parameter	Range
Aegean	`floodclip`	[2,5]
Caesar	`mergeThr`	[2,3]
ProFound	`tolerance`	[2,5]
PyBDSF	`thresh_isl`	[2,5]
Selavy	`growthCut`	[2,5]

by the presence of bright sources or too large to lose sensitivity to faint sources due to background variations (Riggi et al., 2016). This range is consistent with the rule of thumb that the box size should be 10 to 20 times larger than the beam size, $\sigma_{beam} \sim$ (`BMAJ` + `BMIN`)/2. Equation 4 corresponds to $12 \leq$ box_size$/\sigma_{beam} \leq 32$, with the upper limit higher than the nominal factor of 20 to account for the presence of complex extended sources.

The Typhon optimization algorithm can be summarized as follows.

- If the RMS box optimization is desired:
  - Minimize μ over a 6×3 box_size by step_size search grid, constrained by Equation 4
- Select a centralized $n \times n$ sample-cutout of the input image
  - Select an n^2 rectangular area if the image is oddly shaped
  - Use the full image if it is too small
- Determine the RMS parameter bounds of the cutout (Eq. 3)
- For each source finder:
  - If applicable, set the RMS box parameters to the optimized values
  - Extract the island parameter bounds from Configuration Management (Table 2)
  - Optimize the PRD of the sample cutout:
    - Iterate the RMS parameter backwards from RMS$_{max}$
    - 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 largest value.
- Run the source finders on the full image using their optimized parameters
- Archive the results in a tarball

We have chosen to set $n = 2.5^\circ$ to provide a sufficiently large region of sky to ensure the finder parameters are not biased by small-scale structure in a given image. Appendix C provides details of the source finders and

their settings used to augment our optimization process. In particular, only Aegean and PyBDSF use RMS box optimization inputs from BANE, whereas Caesar and Selavy are internally set by similar optimization schemes. ProFound, however, uses a very different approach instead, drawing on sophisticated optical aperture dilation techniques, suitable for optical and infrared images (Robotham et al., 2018; Kron, 1980).

2.1.4 Hydra

Hydra is the main tool that glues everything together, by running Typhon for deep and shallow images, and producing the following data products:

- Typhon Metrics
  - Deep/Shallow Diagnostic Plots of
    - PRD
    - PRD CPU Times
    - Residual RMS
    - Residual MADFM
    - Residual ΣI^2
  - Table of deep and shallow optimized RMS and island parameters
- Deep/Shallow Catalogues
  - Source finder Catalogues
  - Cluster Catalogue
  - Clump Catalogue
- Optional Simulated Input Catalogue
- Deep/Shallow Cutouts
  - Un/annotated Images
  - Un/annotated Residual Images
- Diagnostic Plots
  - Clump Size Distributions
  - Detections *vs.* S/N
  - Completeness *vs.* S/N
  - Reliability *vs.* S/N
  - S_{out}/S_{in} *vs.* S/N
  - S_{out}/S_{in} Fraction $< 3\sigma$ *vs.* S/N[21]
  - False-Positives *vs.* S/N

  wrt injected, deep, and shallow sources
- Local Web-browser Tool

All of this information is stored in a tarball. With the exception of the source finder, simulated input, and clump catalogues, the local web-browser tool allows the user to explore all of these data products, and is accessible through an `index.html` file in the main `tar` directory.

Figure 8 shows the cutout viewer portion of the Hydra Viewer,[22] with a detailed figure caption. The viewer allows users to inspect input and residual image cutouts centered about all overlapping deep and shallow source finder detections (including optional injected inputs from a simulated catalogue), or clumps (see Figure 67). The user can inspect clumps by their `clump_id`, or by filtering on S/N bins of diagnostic plots, such as completeness and reliability.[23] This allows the user to be able to inspect the cutouts for defects. The clumps are further broken down into subclumps, which represent all overlapping deep and shallow detections for a give source finder (or a set of simulated inputs; Figure 68). In addition, matches are provided for the closest overlaps between source finder detections (and injected sources; *cf.*, numbered cyan boxes in Figure 67). All of this information is stored in a cluster catalogue, along with other information as outlined in Figure 8 (see also § D.1).

The cluster catalogue is linked to all of the deep and shallow source finder and simulated catalogues, tying all of this information together. In addition, it is also linked to a clump catalogue that provides statistical information about the clumps, such as residual RMS per cutout per unit area, best residual statistics, number of components, *etc.* All of this is achieved through a clustering algorithm, summarized in Appendix D. Completeness and reliability plots are also computed using this technique, which is outlined in the appendix.

3 ANALYSIS

In this section we examine the source finders using simulated compact and extended sources, subsequently repeating the analysis using real data. In the latter case, an EMU survey image cutout is used as an exemplar of real data containing compact and non-negligible extended source populations. The images have been chosen to be small enough so as to easily explore the catalogues by hand, but large enough to allow Hydra's optimizer to work effectively and provide reasonable output statistics. We find that a $2^\circ \times 2^\circ$ image is optimal for this purpose. While we use real data in our analysis, our focus is on demonstrating the performance of the Hydra software, and not on producing specific catalogues for these datasets at this time. That will be the subject of future work. For clarity in this discussion, source components are referenced using their `clump_id` and `match_id`'s: see infographic, Figure 9.

3.1 Simulated Compact Sources

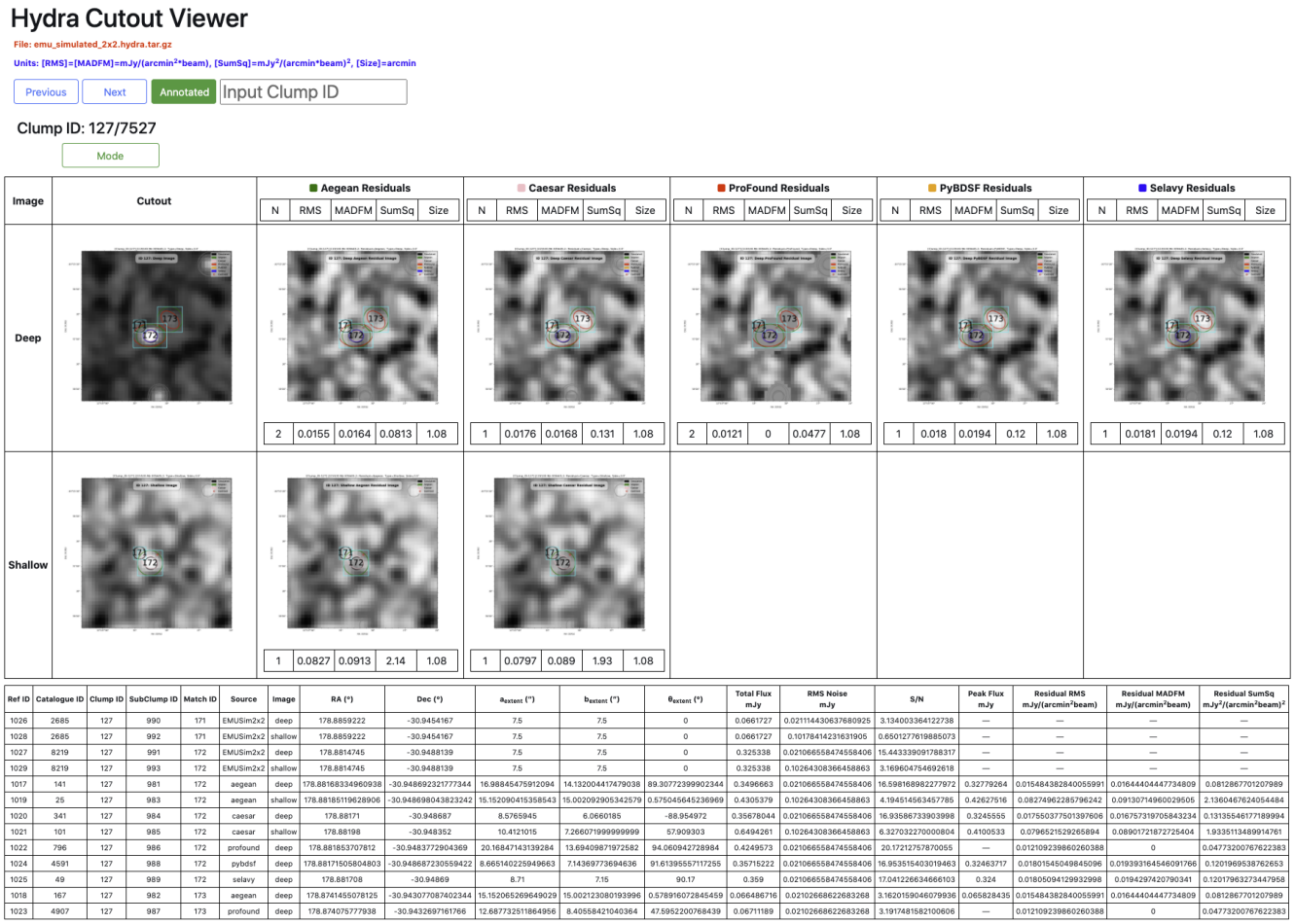
3.1.1 Simulated Data

The simulated image, shown in Figure 10, is produced in two steps, generation of a noise image, and then addition of artificial sources. We use MIRIAD to generate the artificial noise image, using the following steps. The

[21] *Re.*, Equation 7.

[22] NB: The figure is for gleaning the layout, not the details.

[23] Only deep-shallow completeness, $\mathcal{C}_{DS}$, and reliability, $\mathcal{R}_{DS}$, filtering is supported in the current viewer. See Table 25 for other completeness and reliability metrics.



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	0.016444044477348609	0.0812867701207989
1019	25	127	983	172	aegean	shallow	178.88185119628906	-30.948698043823242	15.152090415358543	15.002092905342579	0.5750456452369969	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	16.93586733903998	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	178.881853707812	-30.9483772904369	20.168471431392284	13.69409871972582	94.060942728984	0.4249573	0.021066558474558406	20.17212757870055	—	0.012109239860260388	0	0.04773200767622383
1024	4591	127	988	172	pybdsf	deep	178.88171505804803	-30.948687230559422	8.66514022594966 3	7.143697736946 36	91.61395557117255	0.35715222	0.021066558474558406	16.953515403019463	0.32463717	0.01801545049845096	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	0.12017963273447958
1018	167	127	982	173	aegean	deep	178.8741455078125	-30.943077087402344	15.152065269649029	15.002123080193996	0.578916072845459	0.066486716	0.021026686226832 68	3.162015904607993 6	0.065828435	0.015484382840055991	0.016444044477348609	0.0812867701207989
1023	4907	127	987	173	profound	deep	178.874075777938	-30.9432697161766	12.687732511864956	8.40558421040364	47.5952200768439	0.06711189	0.021026686226832 68	3.191748158210060 6	—	0.012109239860260388	0	0.04773200767622383

Figure 8. 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 completeness and reliability, by using the Mode button (see Figure 72). The central panel contains deep (top) and shallow (bottom) square image cutouts (first column) and source finder residual image cutouts (remaining columns), centered about a given clump's centroid. Here the annotation is turned on, with the neighboring 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, source finder catalogue cross-reference ID, clump ID, subclump ID, match ID, source finder or injected catalogue name, image depth, RA (°), Dec (°), semi-major axis (″), semi-minor axis (″), position angle (°), total flux (mJy), BANE RMS noise (mJy), S/N (total flux over BANE RMS noise), peak flux (mJy), residual RMS ($mJy/(\mathrm{arcmin}^2$ beam)), residual MADFM ($mJy/(\mathrm{arcmin}^2$ beam)), and residual ΣI^2 ($(mJy/(\mathrm{arcmin}^2$ beam$))^2$). The residual information is also shown below each cutout, along with the number of components (N), and cutout size (of side Size, in arcmin). The cutout sizes are all the same size as the maximum clump size, limited to a maximum of 3′, above which true cutout sizes are shown. Note that the residual information is normalized by the cutout area (arcmin^2), as it is on a per cutout bases. The clump ID refers to the union (overlap) over all source finder deep and shallow detections (including injected sources from a 2° × 2° simulated catalogue, `EMUSim2x2`), whereas the subclump ID is wrt each source finder. The Match ID (indicated by the cyan boxes) refers to the closest overlap between source finder components. The information in this figure is illustrative only.

IMGEN task was used to produce a 1800×1800 pixel image, with 4″ pixels, (*i.e.*, a 2° × 2° field) populated by random Gaussian noise of RMS 20 μJy. 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.

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. 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 calculated from the source count model. Flux densities for each artificial source were assigned randomly between the extrema of the bin in which it lies. Flux positions were also assigned randomly, with no attempt to mimic the clustering properties of real sources. For the 2° × 2° field with a flux limit of 50 μJy, we end up with a list of 9,075 artificial sources.

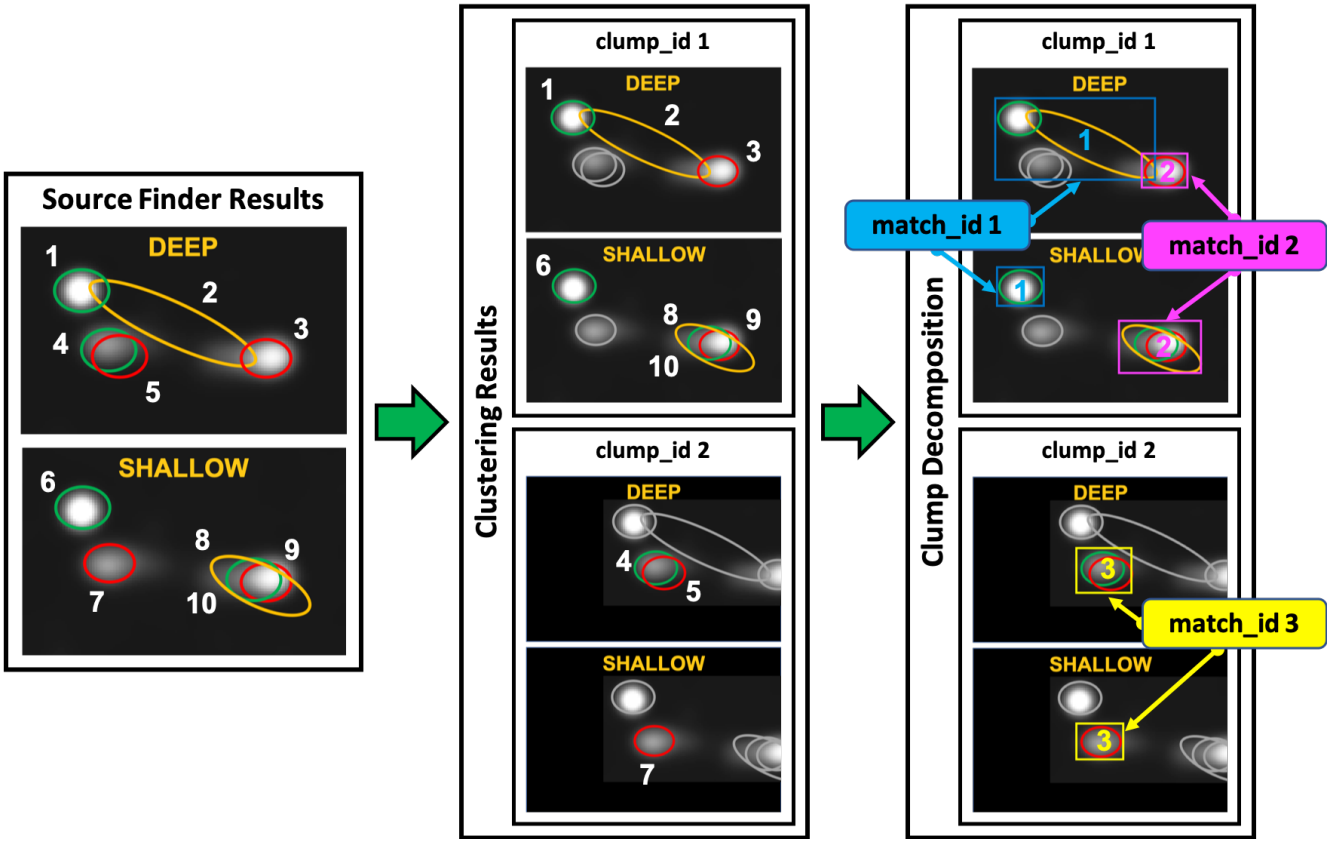


Figure 9. Clustering Process Infographic: The left panel shows the results of 3 hypothetical source finders (*i.e.*, reg, green, and gold). The middle panel shows the results after clustering, resulting in two clumps, assigned `clump_id` 1 (upper panel) and `clump_id` 2 (lower panel). A clump is defined as the spatial-wise overlap between source finder detections (*i.e.*, components), in the deep and shallow images together. The components are numbered, indicating which clumps they end up in. For instance, components 1, 2, and 3 are linked together, in the deep image, and, in turn, 1 overlaps with 6, and 3 overlaps with 8, 9, and 10, in the shallow image, forming the components of `clump_id` 1; likewise, `clump_id` 2 is composed of components 4, 5, and 7. Clumps are centered in the Hydra Viewer, with components that do not belong greyed out. The right panel shows the results after the clumps are decomposed into closest (*i.e.*, center-to-center) matches between source finders, in the deep and shallow images, such that there is only one source finder within a match in the deep and/or shallow image/s. These matched sets are assigned `match_id`'s, with boxes enclosing the extremities of the components. The Hydra Viewer displays these numbers at the center 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}. For more details on clustering, see Appendix D.

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

3.1.2 Results

Tables 3 and 4 show the Hydra run metrics for the 2° × 2° simulated image of point sources. The RMS box optimization parameters, given in Table 3, are used as inputs for Aegean, PyBDSF, and Selavy. The RMS and island parameters, given in Table 4, are from the Typhon optimization routine, extracted at the 90% PRD level (*re.* Figure 73 top left). Also shown are the residual image RMS, MADFM, and ΣI^2 statistics. These are all of comparable magnitude for both deep and shallow images, with Selavy being slightly higher in the deep image.

Figure 11 shows the completeness ($\mathcal{C}$) and reliability ($\mathcal{R}$) for the deep ($\mathcal{D}$) and shallow ($\mathcal{S}$) images. Distributions of $\mathcal{C}_\mathcal{D}$, $\mathcal{R}_\mathcal{D}$, $\mathcal{C}_\mathcal{S}$, $\mathcal{R}_\mathcal{S}$, $\mathcal{C}_{\mathcal{DS}}$, and $\mathcal{R}_{\mathcal{DS}}$ as a function of S/N are given. The $\mathcal{C}_\mathcal{D}$, $\mathcal{R}_\mathcal{D}$, $\mathcal{C}_\mathcal{S}$, and $\mathcal{R}_\mathcal{S}$ values are all calculated relative to the known injected sources.

To link the detected and the injected sources we use a clustering approach (see details in § D.3). These metrics are consequently a reflection of positional accuracy, as

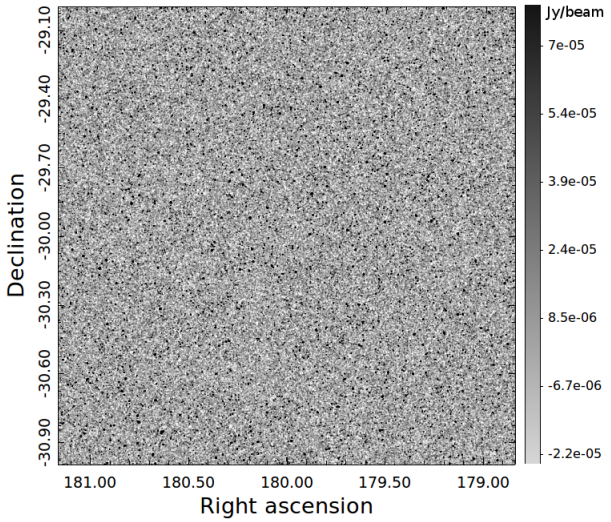


Figure 10. Simulated map with point-like sources. The coordinates are arbitrarily set.

Table 3 Hydra μ, `box_size`, and `step_size` run-parameters for the $2^\circ \times 2^\circ$ simulated deep/shallow point-source images. The box and step size parameters were used as inputs for Aegean, PyBDSF, and Selavy.[a]

Image	μ (μJy)	`box_size` (pixels)	`step_size` (pixels)
Deep	21.81	60	30
Shallow	108.2	45	22

[a]The Selavy module only accepts `box_size`.

the detected and injected components are required to overlap in order to be correctly associated. The alternative method is to use a cutoff distance in a catalogue cross-match (*e.g.*, Huyn et al., 2012; Hopkins et al., 2015; Hale et al., 2019). Depending on the cutoff distance adopted, this approach may lead to associations between adjacent clumps that may not actually be related. The clustering approach aims to mitigate against this effect.

The $\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$ parameters (Figure 11) represent completeness and reliability calculated differently, ignoring for the moment the knowledge of the injected source properties. Instead, for these parameters we assume that the detections arising from a given finder in the deep image represent the real sources, and the results from the same finder in the shallow image are used to calculate completeness and reliability. This approach is explored here as a benchmark, anticipating the next stage of investigation when the finders are applied to real images where the intrinsic underlying source population is not known. It is useful to investigate these results using our simulated image, where we do know the underlying injected sources, in order to establish any issues or systematics that may be a result of this approach.

The $\mathcal{C}$ and $\mathcal{R}$ distributions can be explored using the Hydra Viewer (Figure 8) in its Completeness and Reliability Modes, respectively, and we use this functionality in exploring the performance of the different finders.

All finders show a similar behaviour in general, with completeness beginning to drop away from 100% at S/N as high as ~ 30, and dropping rapidly below S/N~ 5. False sources are typically limited to about 10% of the sample down to S/N~ 5, but can appear in some finders at S/N up to as high as ~ 30. We explore the origins of these effects in further detail below. In general terms, regarding simulated point sources, 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 finders 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, especially in the presence of close nearby sources, or nearby noise spikes, by quite significant amounts in some cases. For ProFound this arises due to the blending of neighbouring sources. Very similar behaviour is seen when treating the sources detected by a given finder using the "deep" image as the reference in calculating completeness and reliability in the "shallow" image ($\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$).

Here we explore some of the failure modes at different S/N levels. Starting at the extreme high S/N end, at S/N $\sim$ 1600, we find 2 injected sources that Caesar and Selavy have failed to detect. One of these is shown in Figure 12,[24] and lies right at the edge of the image. Another, at S/N $\sim$ 43 in the shallow version of the image (Figure 13), is detected by Caesar in the deep image but not in the shallow image, while Selavy misses it in both cases. It is likely that sources at or close to the edge of an image are problematic for some finders, either due to the background estimates there, or to issues related to fitting sources that may be partially truncated.

Some of the incompleteness seen at high S/N values is not a failing of the source finders, but is instead a consequence of overlapping input sources. The S/N $\sim$ 20 bin provides an illustration of a scenario where some input sources are effectively impossible to find. We have a case where three injected sources lie sufficiently close to each other that they appear as a single point-like source (Figure 14). Here `match_id`'s 6200, 6202, and 6203, with injected fluxes of 0.0529, 0.4032, and 0.6928 mJy, re-

[24]NB: Image scale annotation is not supported in the current version of the Hydra Viewer.

Table 4 Typhon run metrics for $2^\circ \times 2^\circ$ simulated image of point sources.

Source Finder	Image Depth	Parameters RMS		Island		Detected N	Residuals RMS	MADFM	ΣI^2
Aegean	Deep	`seedclip`	3.074	`floodclip`	3.070	6,016	2.00E-05	1.90E-05	1.27E-03
Caesar	Deep	`seedThr`	3.074	`mergeThr`	3.000	4,084	1.90E-05	1.60E-05	1.14E-03
ProFound	Deep	`skycut`	2.412	`tolerance`	2.409	4,997	1.80E-05	1.60E-05	1.10E-03
PyBDSF	Deep	`thresh_pix`	2.809	`thresh_isl`	2.806	5,991	2.20E-05	1.90E-05	1.58E-03
Selavy	Deep	`snrCut`	3.206	`growthCut`	3.203	3,225	4.50E-05	2.00E-05	6.54E-03
Aegean	Shallow	`seedclip`	3.868	`floodclip`	3.864	747	1.10E-04	1.10E-04	3.94E-02
Caesar	Shallow	`seedThr`	4.000	`mergeThr`	3.000	657	1.09E-04	1.06E-04	3.88E-02
ProFound	Shallow	`skycut`	3.074	`tolerance`	3.070	642	1.09E-04	1.07E-04	3.87E-02
PyBDSF	Shallow	`thresh_pix`	3.735	`thresh_isl`	3.732	598	1.11E-04	1.09E-04	3.98E-02
Selavy	Shallow	`snrCut`	4.000	`growthCut`	3.996	427	1.14E-04	1.10E-04	4.22E-02

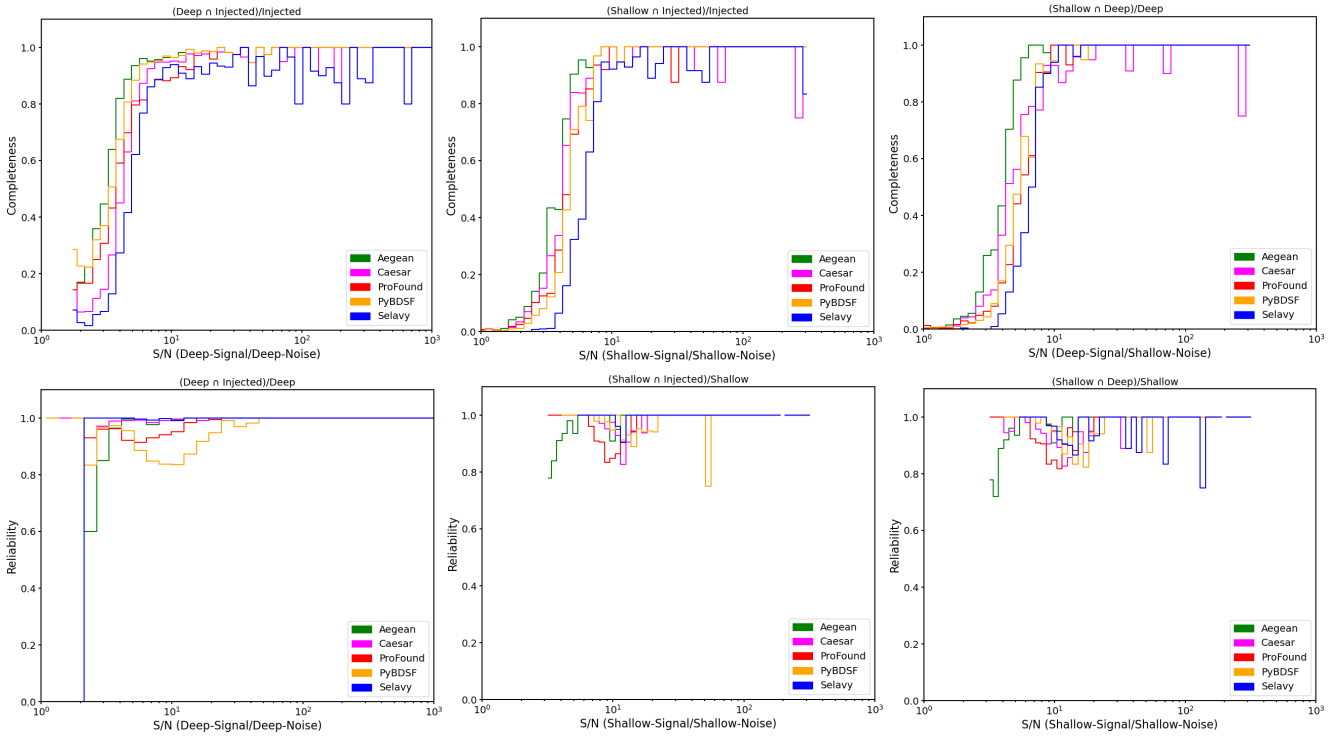


Figure 11. Completeness and reliability plots for the $2^\circ \times 2^\circ$ simulated point-source image, of $\mathcal{C}_\mathcal{D}$ *vs.* $\mathcal{D}$-signal/$\mathcal{D}$-noise (top-left), $\mathcal{R}_\mathcal{D}$ *vs.* $\mathcal{D}$-signal/$\mathcal{D}$-noise (bottom-left), $\mathcal{C}_\mathcal{S}$ *vs.* $\mathcal{S}$-signal/$\mathcal{S}$-noise (top-middle), $\mathcal{R}_\mathcal{S}$ *vs.* $\mathcal{S}$-signal/$\mathcal{S}$-noise (bottom-middle), $\mathcal{C}_{\mathcal{DS}}$ *vs.* $\mathcal{D}$-signal/$\mathcal{S}$-noise (top-right), and $\mathcal{R}_{\mathcal{DS}}$ *vs.* $\mathcal{S}$-signal/$\mathcal{S}$-noise (bottom-right); for colour codes, see Table 20. The deep/shallow injected noise maps used for determining S/N were generated using BANE.

spectively, appear as a single compact unresolved source. All source finders detect it, with measured flux densities of 1.1118, 1.1113, 1.2481, 1.1296, and 1.1230 mJy, for Aegean, Caesar, ProFound, PyBDSF, and Selavy, respectively. These values are consistent with the total injected flux $\sim 1.1489\,\mathrm{mJy}$.

All source finders except ProFound make detections in the shallow image with the same order of flux measurements as in the deep image. We intentionally chose not to use flux density in our cross-matching in the calculations of completeness and reliability. Had we done so, none of the input sources would have been identified as a match here, whereas with positional matching alone the combined input source was recognised.

A similar effect arises in the way reliability is calculated when a source has its flux density measured incorrectly. One of the injected sources, with flux density of $I_{\mathrm{Injected}} \sim 291\,\mu\mathrm{Jy}$, is erroneously measured by Caesar to have $I_{\mathrm{Caesar}} \sim 0.664\,\mu\mathrm{Jy}$ (Figure 15). This is likely to be a consequence of a poor deblending of the two adjacent sources, initially identified by Caesar as a single island. The position is accurate, so it is counted

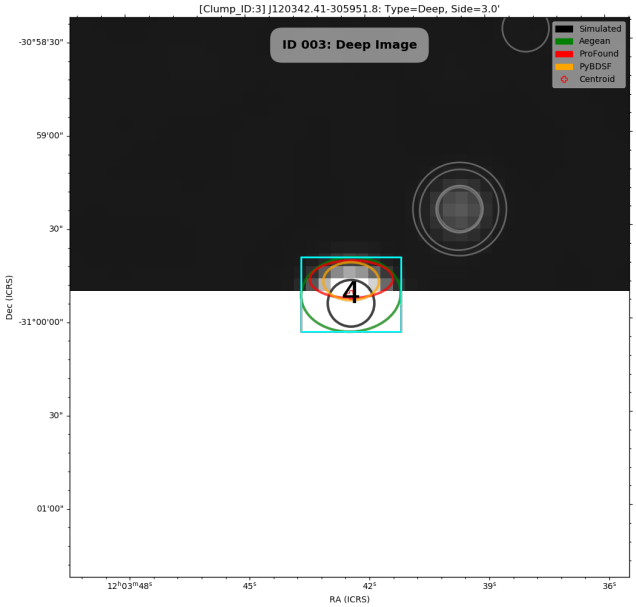


Figure 12. $0.534' \times 0.534'$ cutout for an injected source at the edge of the $2° \times 2°$ simulated point-source deep image; for colour codes, see Table 20. Only Aegean, ProFound, and PyBDSF were able to detect it. This information was extracted from the S/N $\sim$ 1600 ± 110 bin in $\mathcal{C}_\mathcal{D}$ (Figure 11).

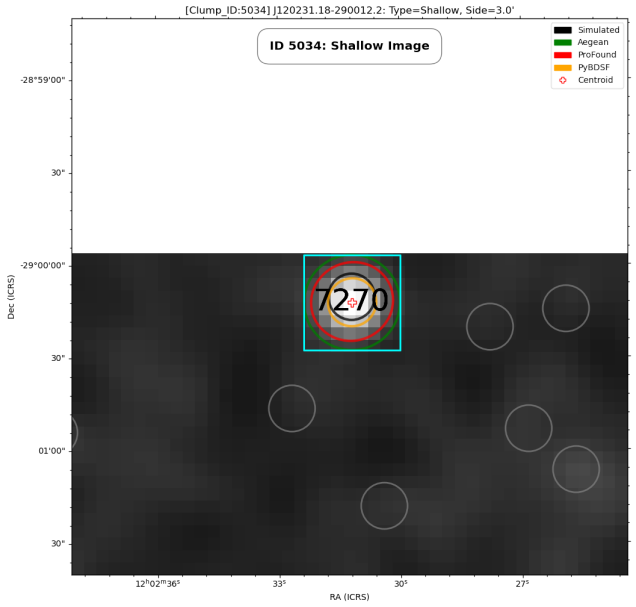


Figure 13. $0.518' \times 0.518'$ cutout for an injected source at the edge of the $2° \times 2°$ simulated point-source shallow-image; for colour codes, see Table 20. Caesar and Selavy did not detect the source. This information was extracted from the S/N $\sim$ 42.6 ± 2.9 bin in $\mathcal{C}_\mathcal{S}$ (Figure 11).

as a correct match, but because $\mathcal{R}$ is calculated using the measured S/N, this source contributes to $\mathcal{R}_\mathcal{D}$ at an unphysical value of S/N $\sim$ 0.033. For this source, Aegean, PyBDSF, and Selavy correctly estimate the flux, *i.e.* $281.9^{+4.2}_{-8.4}\,\mu$Jy, and it contributes to $\mathcal{R}_\mathcal{D}$ for

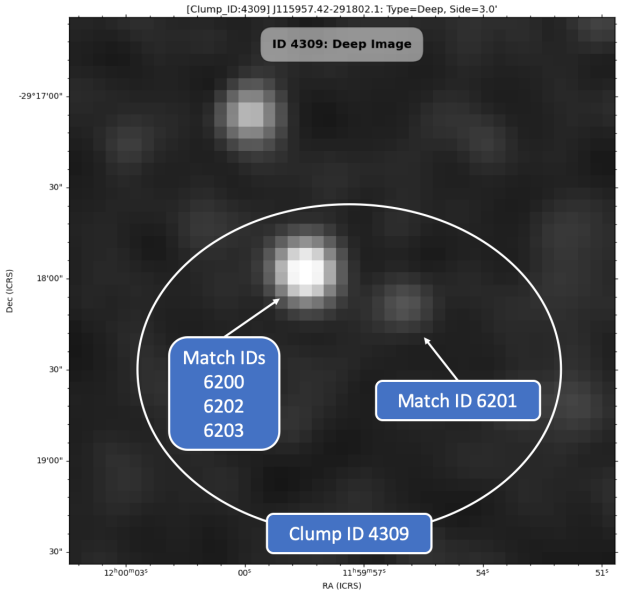


Figure 14. $1.21' \times 1.21'$ cutout for injected source `match_id`'s 6200 through 6203 of `clump_id` 4309 for the $2° \times 2°$ simulated point-source deep-image. Only ProFound missed detection of the isolated source at `match_id` 6201. The remaining injected sources overlap to make up a single unresolved compact object, which is identified as `match_id` 6203 by all source finders. This information was extracted from the S/N $\sim$ 19.6 ± 1.3 bin in $\mathcal{C}_\mathcal{D}$ (Figure 11).

those finders at S/N $\sim$ 12. In this case, again, requiring flux-density matching would lead to this source being deemed a false detection, although it has been detected at the correct location. ProFound does not separately detect `match_id` 6668 (Figure 16) as it has blended it with `match_id` 6669. Comparing Figures 15 and 16 we can see the similar islands detected by ProFound and Caesar, although ProFound's boundary is tighter, which provides a clear indication of why only a single match was identified.

An example of a spurious source at reasonably high S/N ($\sim$ 12), due to a noise spike, is shown in Figure 17. This false source is detected by all source finders. As image (and survey) sizes become larger, such false sources will be detected at higher rates, as the number simply scales with the number of resolution elements sampled (Hopkins et al., 2002).

PyBDSF sees the largest number of false detections within this S/N bin (a false/true ratio of 86/522). Figure 18 shows an example of one of these, illustrating a major failure mode for PyBDSF. It tends to overestimate source sizes due to inclusion of nearby noise spikes. At low or modest S/N even small noise fluctuations can lead it to expand its island in fitting to the source. Figure 19 shows another example, here with an excessively large size ($\sim 67''$), associated with the dip in the S/N $\sim$ 12 bin in $\mathcal{R}_\mathcal{S}$ (Figure 11). The Clump Size Mode of the Hydra Viewer (see Figure 69) is helpful in identifying such examples. Figure 20 shows an example where a real

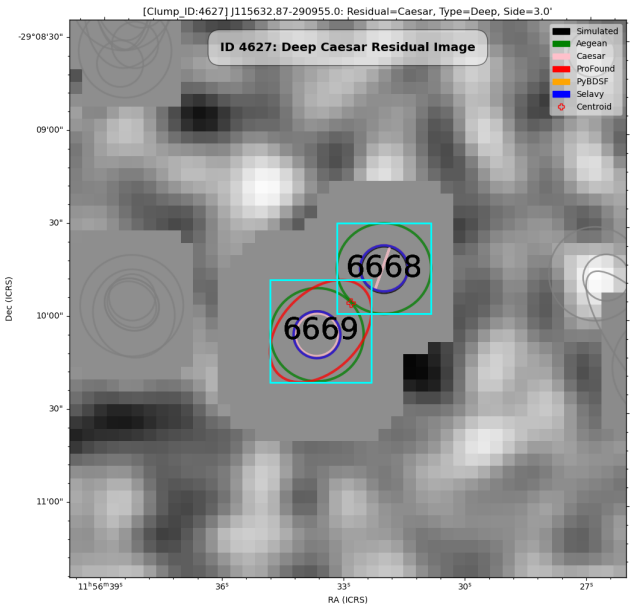


Figure 15. Casear `clump_id` 4627 residual-image $0.916' \times 0.916'$ cutout for the $2^\circ \times 2^\circ$ simulated point-source deep-image; for colour codes, see Table 20. Caesar underestimates the flux density for `match_id` 6668 ($0.00066\,m$Jy, compared to $0.29\,m$Jy for the injected-source), resulting in an artifact in the calculated reliability, placing this source at an artificially low S/N. This information was extracted from the S/N $\sim 0.0331 \pm 0.0036$ bin in $\mathcal{R}_\mathcal{D}$ (Figure 11).

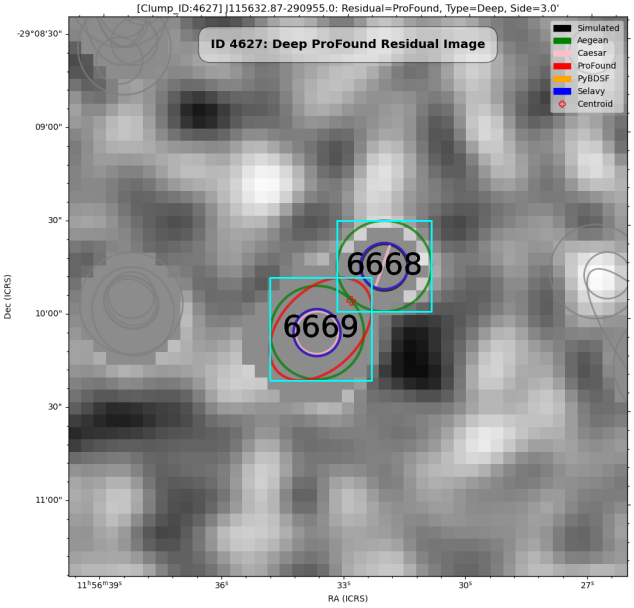


Figure 16. ProFound `clump_id` 4627 (*c.f.* Figure 15) residual-image $0.916' \times 0.916'$ cutout for the $2^\circ \times 2^\circ$ simulated point-source deep-image; for colour codes, see Table 20. ProFound provides a single flux-weighted component that blends these two adjacent sources, and which is best matched to `match_id` 6669. This leads to the result that `match_id` 6668 is deemed undetected in the $\mathcal{C}_\mathcal{D}$ statistics.

injected source is the centre of a 10-component PyBDSF clump, spanning $\sim 156''$.

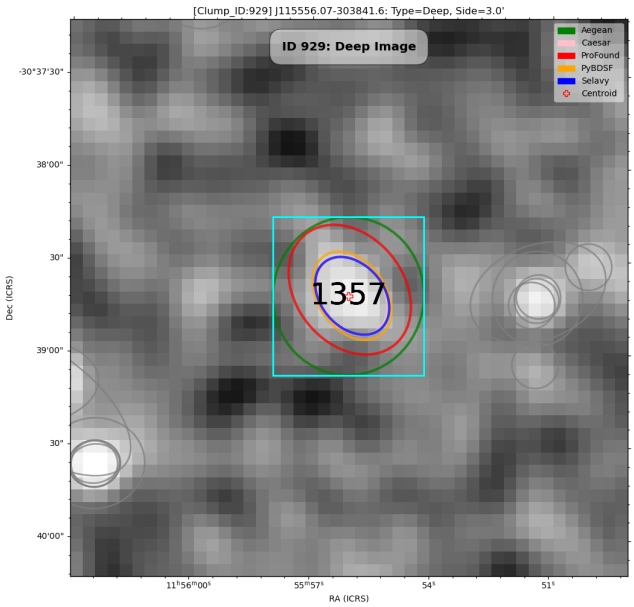


Figure 17. $0.851' \times 0.851'$ cutout of `clump_id` 929 of the $2^\circ \times 2^\circ$ simulated point-source deep-image; for colour codes, see Table 20. The detections within this clump are anomalous, as there is no injected source. This information was extracted from the S/N $\sim 12.4 \pm 1.4$ bin of $\mathcal{R}_\mathcal{D}$ (Figure 11).

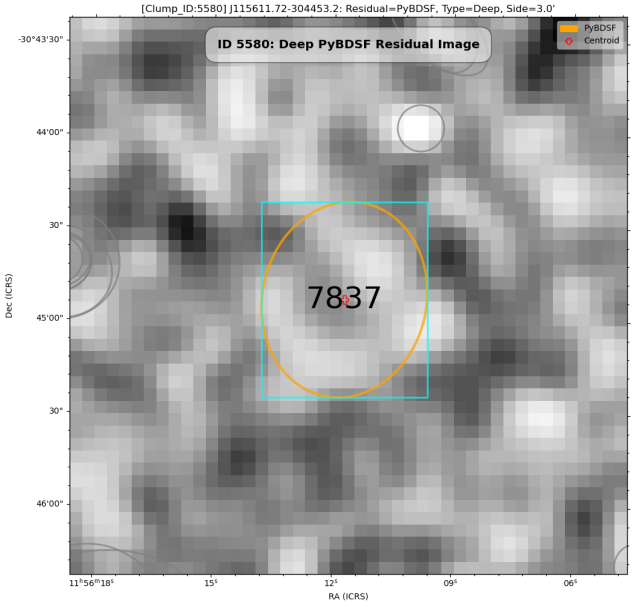


Figure 18. PyBDSF $1.05' \times 1.05'$ residual-image cutout of `clump_id` 5580 of the $2^\circ \times 2^\circ$ simulated point-source deep-image; for colour codes, see Table 20. This is a spurious detection, as there is no injected source. This information was extracted from the S/N $\sim 12.4 \pm 1.4$ bin of $\mathcal{R}_\mathcal{D}$ (Figure 11).

Figure 21 shows an example of a case where a robust detection in the deep image appears at low S/N (S/N ~ 1.1) in the shallow image. This scenario is expected to explain most of the incompleteness seen at low S/N. In this example Caesar does not detect the shallow

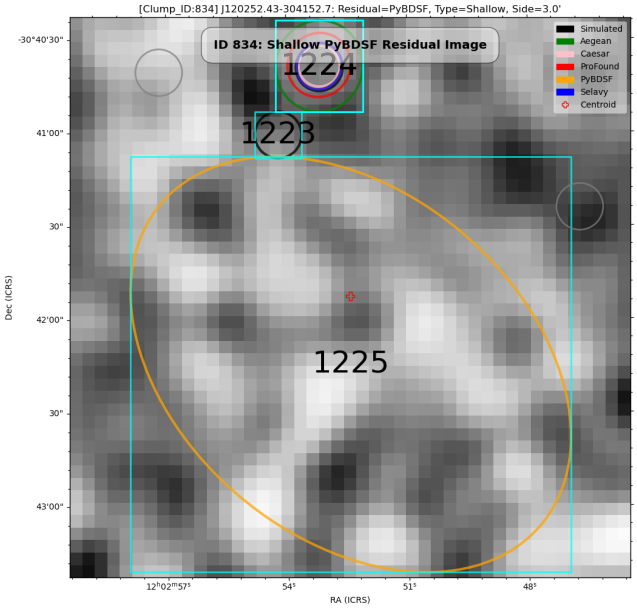


Figure 19. PyBDSF $2.96' \times 2.96'$ residual-image cutout of `clump_id` 834 of the $2° \times 2°$ simulated point-source shallow-image; for colour codes, see Table 20. The detection by PyBDSF at `match_id` 1225 is spurious, as there is no injected source at this location. This information was extracted from the S/N $\sim 12.4 \pm 1.4$ bin of $\mathcal{R}_{\mathcal{D}}$ (Figure 11).

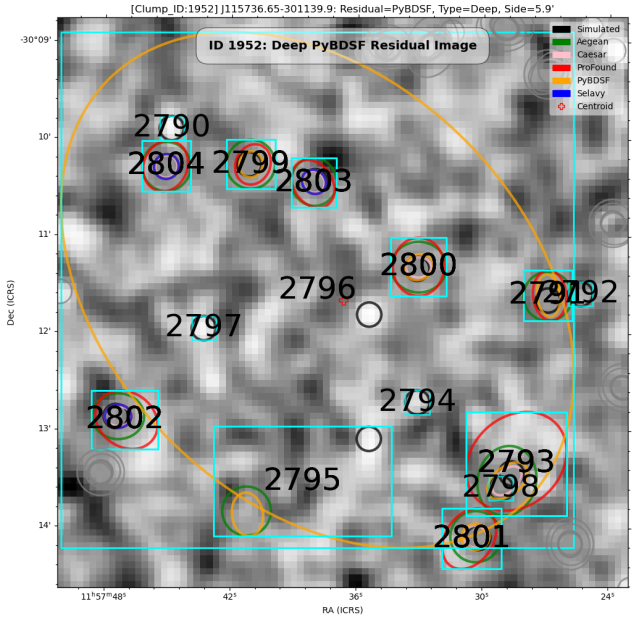


Figure 20. PyBDSF $5.89' \times 5.89'$ residual-image cutout of `clump_id` 1952 of the $2° \times 2°$ simulated point-source deep-image; for colour codes, see Table 20. The large-footprint detection by PyBDSF at `match_id` 2796 demonstrates one its most frequent failure modes. Its flux density is 1.84 mJy, as compared to 0.0798 mJy for the injected-source. This information was obtained using the Hydra Viewer in Clump Size Mode (see Figure 69).

source corresponding to its deep detection at `match_id` 1286. Caesar is the only finder to correctly identify this source in the deep image. The failure of Caesar to detect the neighbouring injected source at `match_id` 1285 in either deep or shallow images is associated with the deblending issues noted above. Figure 22 shows an example of the residual image for the same clump using Selavy. The Aegean residual image here is similar. ProFound is similar to Caesar, but with a slightly smaller footprint. We note that the subtraction of the component at `match_id` 1287 emphasises the remaining injected sources at 1285 and 1286.

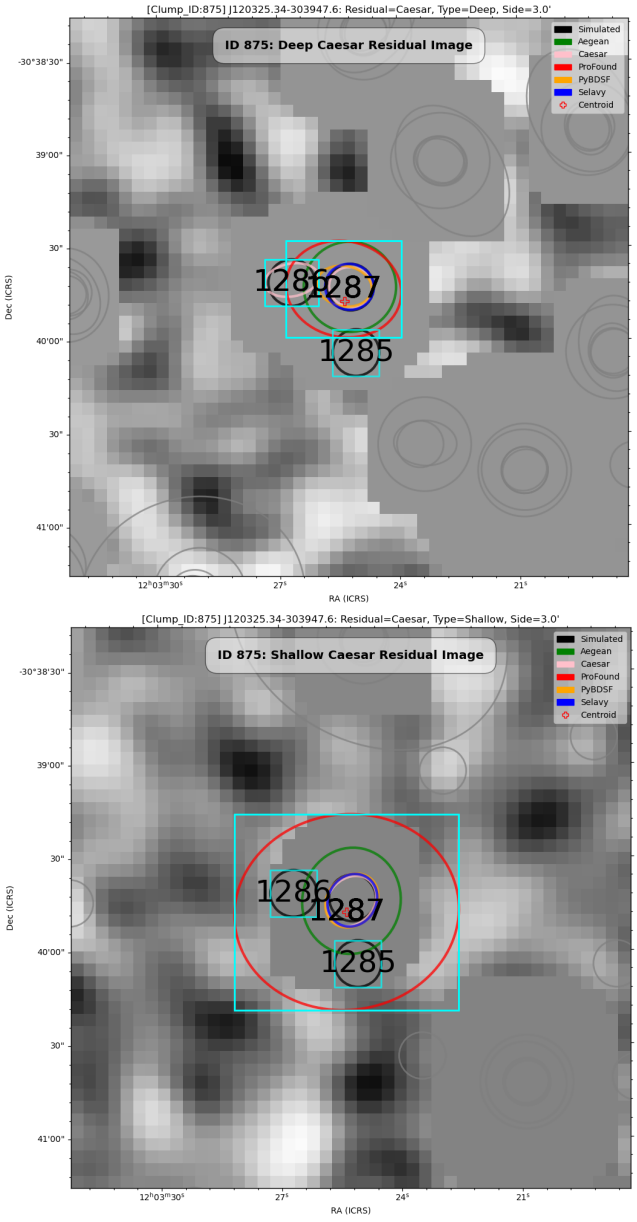


Figure 21. Caesar $1.2' \times 1.2'$ residual-image deep (top) and shallow (bottom) cutouts of `clump_id` 875 of the $2° \times 2°$ simulated point-source deep/shallow-image; for colour codes, see Table 20. Caesar does not detect the shallow source corresponding to its deep detection at `match_id` 1286. This information was extracted from the S/N $\sim 1.136 \pm 0.075$ bin of $\mathcal{C}_{\mathcal{DS}}$ (Figure 11).

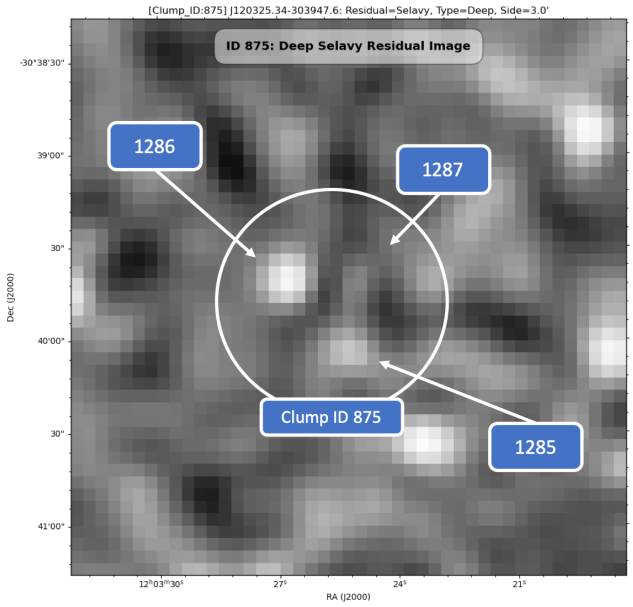


Figure 22. Selavy $1.2' \times 1.2'$ residual-image cutout of `clump_id` 875 (*c.f.*, Figure 21) of the $2° \times 2°$ simulated point-source deep-image. Selavy only made a detection at `match_id` 1287.

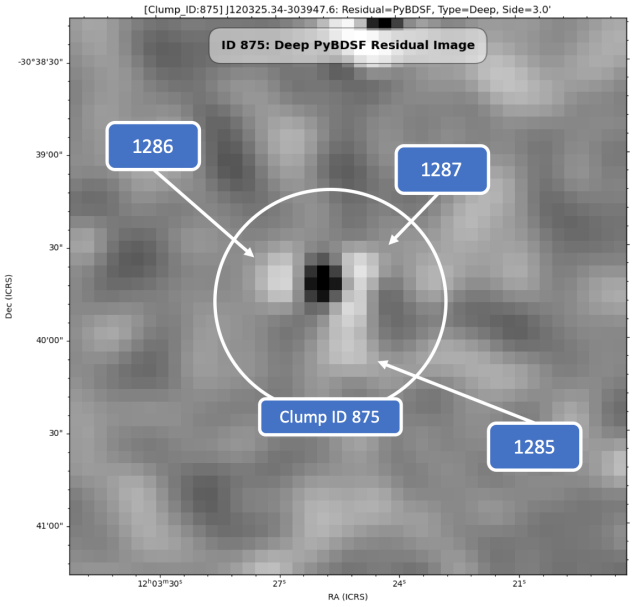


Figure 23. PyBDSF $1.2' \times 1.2'$ residual-image cutout of `clump_id` 875 (*c.f.*, Figure 21) of the $2° \times 2°$ simulated point-source deep-image. PyBDSF only made a detection at `match_id` 1287.

The existence of initially undetected sources in the residual images suggests that running a second iteration of the finders on such residual images may potentially improve on the completeness of the delivered catalogues. This may not be practical for Caesar or ProFound, however, which subtract out all of the flux within an island; although, residual images created from their island components may be an option. The residuals from the Gaussian-fitting finders may be suitable for such an approach, although there is a challenge arising from over-subtraction where the initial source list includes overestimated sources (*e.g.*, Figure 23). The location of that detection is slightly offset from `match_id` 1287 towards the flux-centroid of the three injected sources (1285, 1286, 1287), indicating that the detection has attempted to fit to all three jointly. The "hole" visible here is due to PyBDSF overestimating the flux density due to the presence of the adjacent sources. In this circumstance, a second iteration of a source finder on the residual image could exacerbate the situation by adding further spurious detections. (This is quite evident for the extended-component in the PyBDSF residual deep-image, at `match_id` 114, shown in Figure 37. It clearly exposes `match_id`'s 109 and 111; however, 112 and 113 are indistinguishable from the exposed spurious sources.)

In exploring the S/N$\sim$ 21 bin, we find an elongated Gaussian fit to an island by Caesar at `match_id` 5806, which has no corresponding shallow detection (Figure 24). This is likely due to a fitting problem with the irregularly shaped island, for which it has only detected 1 of 3 injected sources. All source finders make detections in this region of the deep image, but not the shallow due to low S/N (*i.e.*, $\mathrm{S/N}_{\mathrm{injected}} \sim 1.03^{+0.30}_{-0.37}$). Figure 25 shows a similar artifact that contrasts Caesar with ProFound. Here the island for Caesar is smaller than for ProFound, as it misses the detection of an injected source at `match_id` 195.

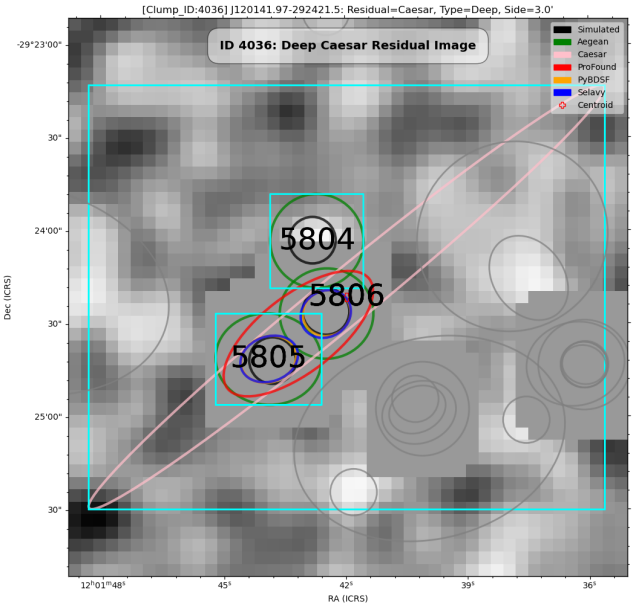


Figure 24. Caesar $2.77' \times 2.77'$ residual-image cutout of `clump_id` 4036 of the $2° \times 2°$ simulated point-source deep-image; for colour codes, see Table 20. Caesar's Gaussian fit at `match_id` 5806 extends well beyond its island. Its flux density estimate is 2.18 mJy compared to 0.14 mJy for the injected source. This information was extracted from the S/N $\sim 20.7 \pm 1.4$ bin of $\mathcal{C}_{\mathcal{DS}}$ (Figure 11).

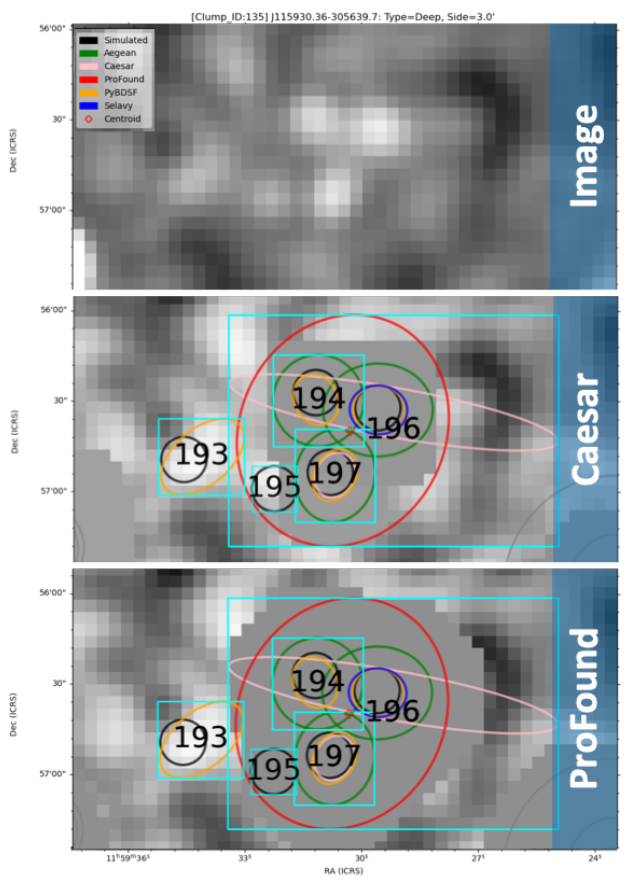


Figure 25. $2.38' \times 2.38'$ cutout-region, with image (top), Caesar residuals (middle), and ProFound residuals (bottom), of `clump_id` 135 of the $2^\circ \times 2^\circ$ simulated point-source deep image; for colour codes, see Table 20. Caesar and ProFound overestimate the flux densities at `match_id` 196, with 0.91 mJy and 0.60 mJy, respectively, as compared to 0.12 mJy for the injected source. This information was obtained using the Hydra Viewer in Clump Size Mode (*c.f.*, Figure 69).

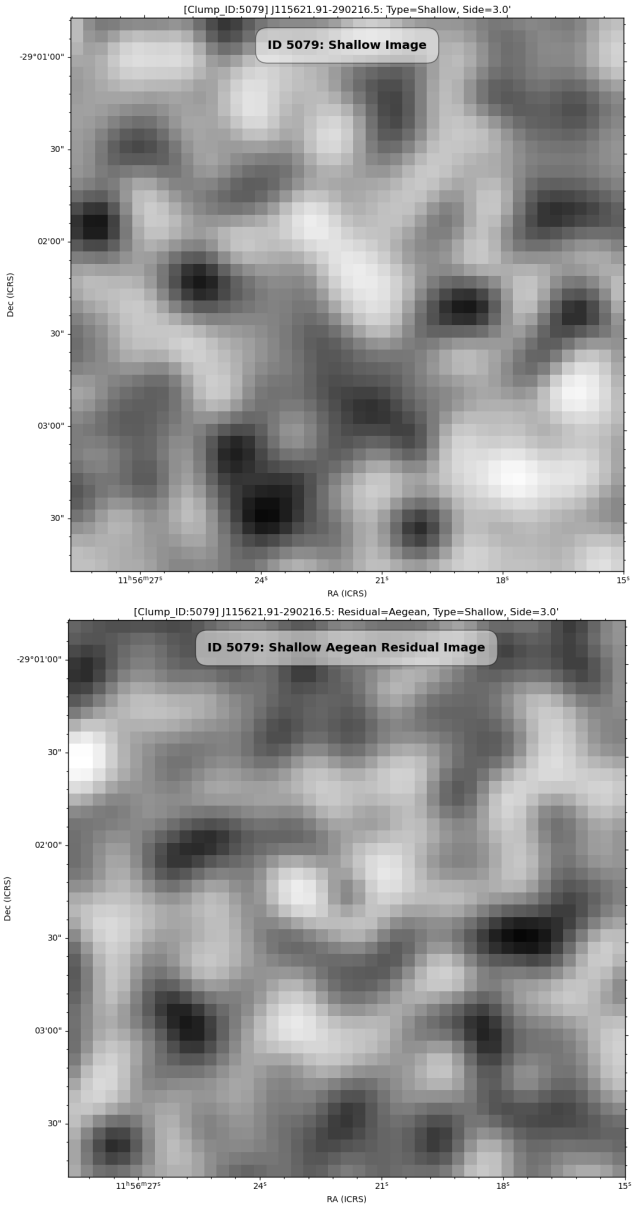


Figure 26. $0.505' \times 0.505'$ image (top) and Aegean-residual (bottom) cutouts of `clump_id` 5079 of the $2^\circ \times 2^\circ$ simulated point-source shallow-image. The hole in the centre of the bottom cutout corresponds to a detection by Aegean. This is a spurious detection, due to a random noise fluctuation, as there is no injected source at this location. This information was extracted from the S/N $\sim 3.71 \pm 0.17$ bin of $\mathcal{R}_{\mathcal{DS}}$ (Figure 11).

At low S/N in general we expect to find spurious detections. In the S/N $\sim$ 3.7 bin of $\mathcal{R}_{\mathcal{DS}}$ (Figure 11), taking Aegean as an example, there are 18 real and 7 spurious detections. These correspond to noise fluctuations (*e.g.*, Figure 26) – as expected. At higher S/N too there can still be spurious detections, often in the vicinity of injected sources, perhaps arising from the combination of true underlying source flux distorted by bright nearby noise spikes. Inspecting the S/N$\sim$ 38 and S/N$\sim$ 74 bins we see shallow detections coincident with injected sources, but with no deep counterparts. Figure 27 shows such an example for Caesar.

This summary covers the main forms of incompleteness and false detections associated with $\mathcal{C}$ and $\mathcal{R}$. Aegean tends to perform quite well, except in the case of noise spikes at low S/N (*e.g.*, Figure 26). Caesar tends to overestimate the footprint of injected sources which are close in proximity, failing to resolve them into individual components (*e.g.*, Figures 24 and 25). ProFound provides a footprint closely matching the underlying flux distribution, although with no attempt to infer the flux of known objects that extends below the detection threshold (*e.g.*, Figure 25). ProFound is also not designed to resolve segments (*i.e.*, *"islands"*) into several components. PyBDSF has the tendency to fit large islands of relatively low total flux density at low S/N (*e.g.*, Figures 18 through 20). Selavy has trouble detecting sources near the edge of images (*cf.*, Figures 12 and 13).

Figure 28 shows the distribution of major axes for sources measured in the deep and shallow images. The

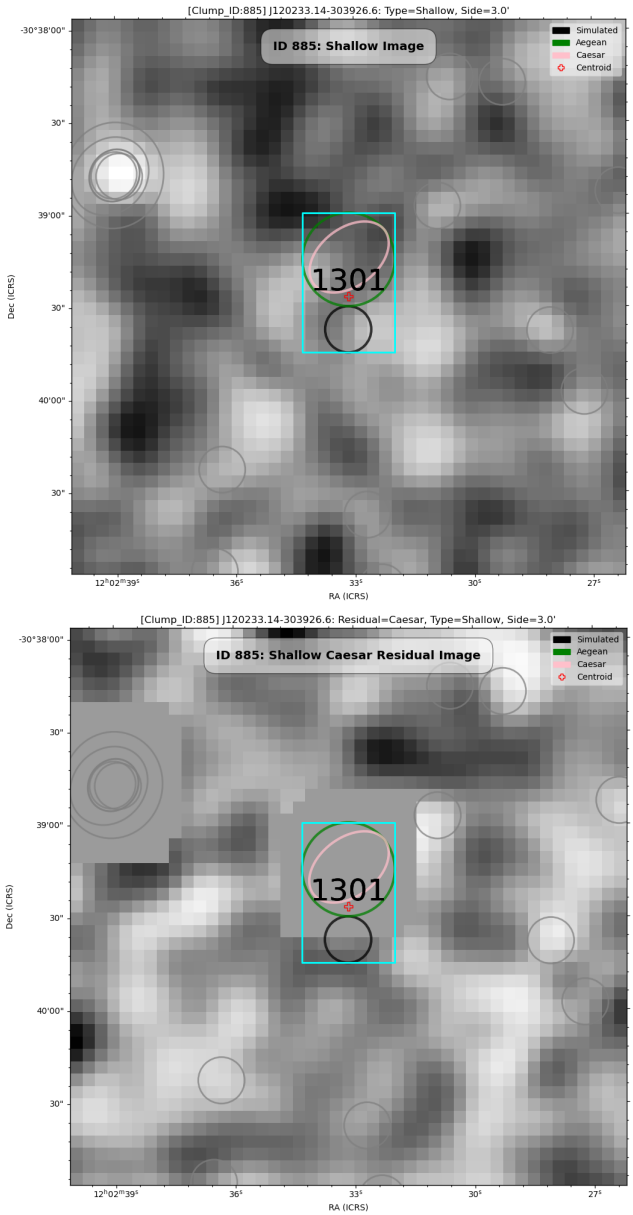


Figure 27. $0.905' \times 0.905'$ image (top) and Caesar residual (bottom) cutouts of `clump_id` 885 of the $2° \times 2°$ simulated point-source shallow image; for colour codes, see Table 20. This shows a spurious detection by Caesar in the vicinity of an injected source, for which there is no deep counterpart. This information was extracted from the S/N $\sim 38.5 \pm 1.8$ bin of $\mathcal{R}_{\mathcal{DS}}$ (Figure 11).

injected sources are all point-like with a fixed size (grey histogram bar). This illustrates the tendency of some finders (PyBDSF, ProFound) to overestimate source sizes through blending or incorporation of noise spikes, compared to the other finders.

3.1.3 Performance statistics

We use the deep/shallow image pairs to establish the validity of using finder-detected sources in the deep image as the reference for calculating completeness ($\mathcal{C}_{\mathcal{DS}}$) and

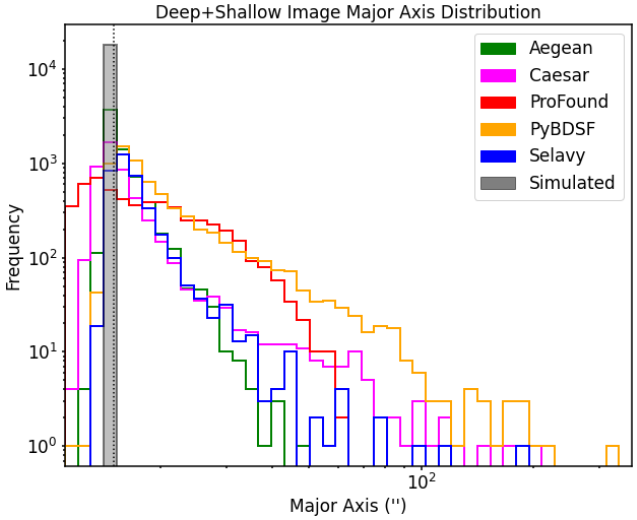


Figure 28. Major axis distributions for the $2° \times 2°$ deep and shallow point-source simulated images. The vertical dashed-line represents the beam size.

reliability ($\mathcal{R}_{\mathcal{DS}}$) statistics for that finder in the shallow counterpart. Given we have the underlying known injected sources we can assess how well this approach works.

We do this here by defining the goodness of completeness, $\tilde{\mathcal{C}}_{\mathcal{DS}}$, and goodness of reliability, $\tilde{\mathcal{R}}_{\mathcal{DS}}$ (details in § D.3). These metrics are calculated by eliminating false-detections, leaving only the fraction of real deep-detections recovered in the shallow image ($\tilde{\mathcal{C}}_{\mathcal{DS}}$), and the fraction of real shallow-detections also detected in the deep image ($\tilde{\mathcal{R}}_{\mathcal{DS}}$). Figure 29 shows $\tilde{\mathcal{C}}_{\mathcal{DS}}$ and $\tilde{\mathcal{R}}_{\mathcal{DS}}$ as a function of S/N. Comparing with $\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$ in Figure 11, $\tilde{\mathcal{C}}_{\mathcal{DS}}$ is largely unchanged, while $\tilde{\mathcal{R}}_{\mathcal{DS}}$ does not show the dip in reliability at around S/N$\sim$ 10 seen for most finders in $\mathcal{R}_{\mathcal{DS}}$. It also excludes the decline in reliability 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 deep image that are (not surprisingly) missed in the shallow image. To quantify these results we define residuals between these values:

$$\delta\mathcal{C}_{\mathcal{DS}} = \tilde{\mathcal{C}}_{\mathcal{DS}} - \mathcal{C}_{\mathcal{DS}} \tag{5}$$

and

$$\delta\mathcal{R}_{\mathcal{DS}} = \tilde{\mathcal{R}}_{\mathcal{DS}} - \mathcal{R}_{\mathcal{DS}}\,. \tag{6}$$

In general, these quantities 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 (and hence S/N bin) between the deep and shallow images. Figure 30 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.

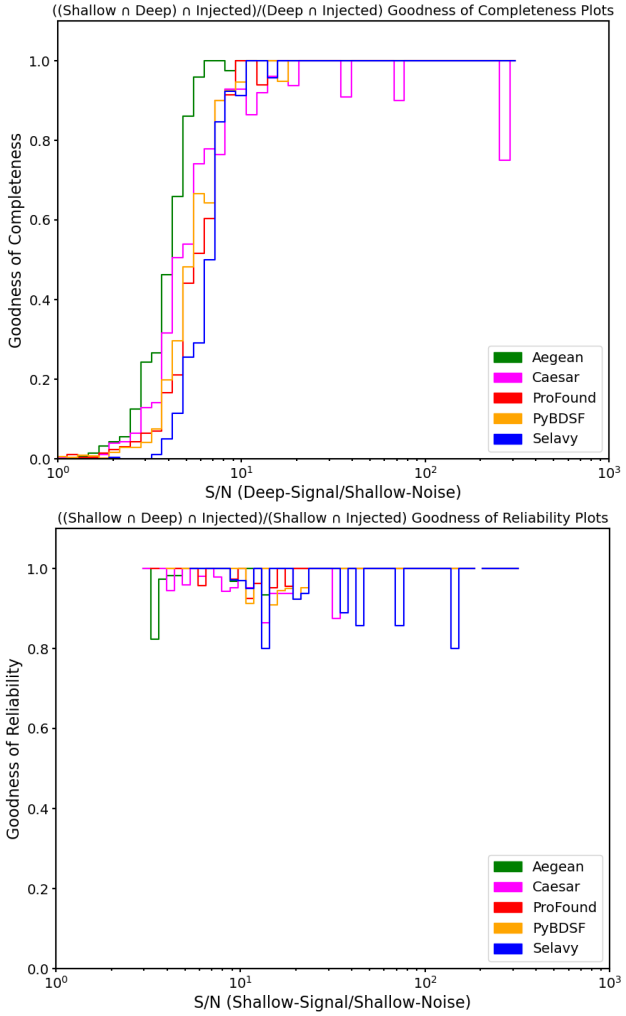


Figure 29. Goodness of completeness (top) and reliability (bottom) for the $2^\circ \times 2^\circ$ point-source simulated image.

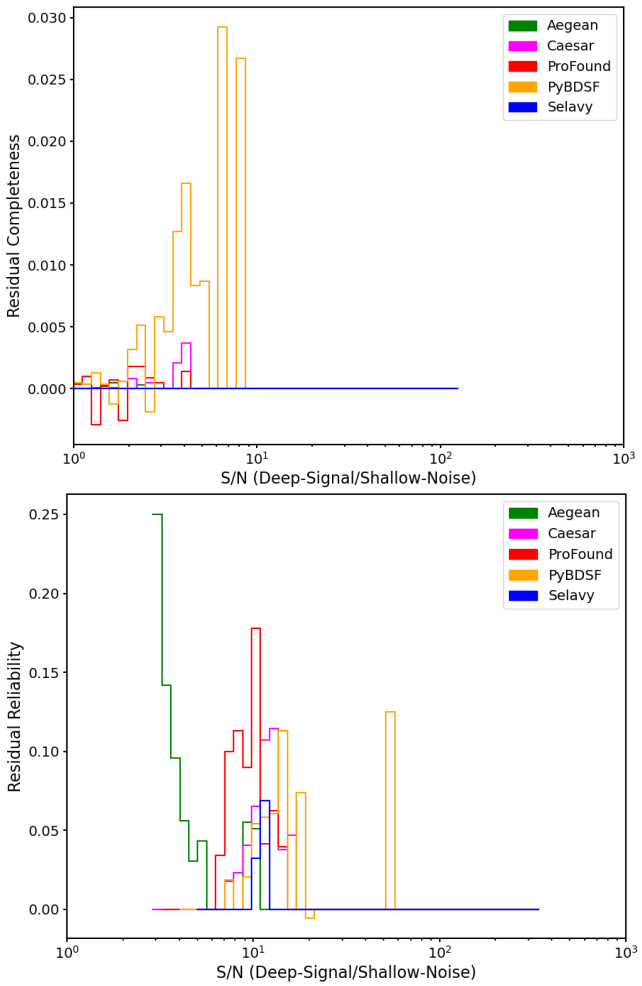


Figure 30. Residual completeness (top) and reliability (bottom) for the $2^\circ \times 2^\circ$ point-source simulated image.

Recognising these limitations, while also noting that for several finders $\delta\mathcal{C}_{\mathcal{DS}}$ 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 artifacts and other hard to simulate systematics.

Another measure of source finder performance is the ratio of measured to injected flux density, S_{out}/S_{in}. Figure 31 shows S_{out}/S_{in} as a function of S_{in} (expressed as S/N) for each source finder. The horizontal (dotted) lines represent $S_{out} = S_{in}$, and the solid and dashed curves about these lines are the 1σ and 3σ deviations from S_{in}, respectively. The dot-dashed curves are the detection thresholds (*i.e.*, the RMS parameters in Table 4, corresponding to 90% PRD), and the (curved) dotted lines represent nominal 5σ thresholds (*c.f.*, Hopkins et al., 2015). The sharp vertical cut off below S/N ~ 2 is a consequence of having no artificial input sources fainter than that level.

Also shown are histograms giving the distribution of S/N in the false-positives for each source finder. The latter panel emphasises in another way the finder characteristics we have seen in earlier reliability distributions. Specifically, Aegean picks up false detections predominantly close to the S/N threshold, with PyBDSF and ProFound having false detections peaking between about S/N$\sim 5-10$, arising from blended sources and overestimated source sizes. The low numbers of false sources seen by Caesar and Selavy correspond to their drop off in completeness at higher S/N compared to other finders. In particular, Aegean displays higher levels of completeness, but at the expense of reliability at low S/N, with the converse being true for Selavy and Ceasar. In this analysis, Aegean shows arguably the best performance, as is evident from its false-positive distribution being limited to low S/N, generally reliable flux densities seen in the limited scattering beyond $S_{out}/S_{in} > 3\sigma$, and

good completeness to low S/N. This is not surprising, though, as Aegean is well designed for identifying and fitting point sources, and the simulated data used here do not include anything else.

To quantify the reliability of the flux density measurements, we look at the scatter in the distribution of S_{out}/S_{in} as a function of S/N. We first define the fraction of objects $r_{n\sigma}(\hat{S}_i)$ above a S/N limit, $\hat{S}_i$, with S_{out}/S_{in} lying between $1-n\sigma$ and $1+n\sigma$:

$$r_{n\sigma}(\hat{S}_i) = \frac{\left|\left\{\frac{S_{out}}{S_{in}}\middle| 1-\frac{n}{\hat{S}_{in}} \leq \frac{S_{out}}{S_{in}} \leq 1+\frac{n}{\hat{S}_{in}} \ni \hat{S}_{in} \geq \hat{S}_i\right\}\right|}{\left|\left\{\frac{S_{out}}{S_{in}}\middle| \hat{S}_{in} \geq \hat{S}_i\right\}\right|}. \quad (7)$$

Using this, we define the scatter in S_{out}/S_{in} outside an $n\sigma$ range as

$$s_{n\sigma}(\hat{S}_i) = 1 - r_{n\sigma}(\hat{S}_i)\,. \quad (8)$$

Aegean and Selavy tend to show the least scatter (Table 5). The scatter for PyBDSF and Selavy seen here is slightly higher than that reported by Hopkins et al. (2015), who saw 3σ scatters at a 5σ threshold of 4.9% for PyBDSF and 3.3% for Selavy.

Table 5 3σ scatter for flux-ratio plots in Figure 31.

Source	3σ Scatter		
Finder	$s_{3\sigma}(3)$	$s_{3\sigma}(5)$	$s_{3\sigma}(10)$
Aegean	3.49%	3.91%	4.27%
Caesar	6.93%	7.11%	6.65%
ProFound	14.55%	16.70%	21.74%
PyBDSF	8.85%	6.20%	6.64%
Selavy	6.43%	6.92%	6.63%

All source finders, with the exception of Selavy, are seen to measure sources down to the specified RMS threshold (the dot-dashed line). Selavy was given a 3.2σ threshold (Table 4), but only appears to recover sources down to a 5σ level. Apart from this detail, the global performance of the finders in terms of measured flux density is in line with expectations, with the examples of different failure modes described above (§ 3.1.2) contributing to the scatter seen here.

3.2 Simulated Extended Sources

3.2.1 Simulated Data

Following a similar procedure as in Section 3.1.1, 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. 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 3.1.1).

A total of 9,974 artificial sources were injected, corresponding to a source density of 2,500 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 SCORPIO 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 Figure 32, was produced by convolving this input sky model using the CASA *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 $(\mathrm{S/N})_{min} \sim 1.3$, in this new simulated image 20.6% of the injected sources fall below S/N = 1. This is by design, to provide a realistic looking image, and to test the impact on the finders of having real sources lying below the noise level.

3.2.2 Results

Tables 6 and 7 show the Hydra run metrics for the 2° × 2° simulated image of extended sources. The RMS and island parameters are from the Typhon optimization routine, extracted at the 90% PRD level. PyDBSF has the most detections for both the deep and shallow images, whereas Selavy has the least. Also shown are the residual image RMS, MADFM, and ΣI^2 statistics. The statistics are comparable between most finders, although Selavy shows higher values.

Table 6 Hydra μ, `box_size`, and `step_size` run-parameters for the 2° × 2° simulated deep/shallow extended-source images. The box and step size parameters were used as inputs for Aegean, PyBDSF, and Selavy.[a]

Image	μ (μJy)	`box_size` (pixels)	`step_size` (pixels)
Deep	68.01	120	60
Shallow	325.3	60	30

[a] The Selavy module only accepts `box_size`.

Figure 33 shows $\mathcal{C}_{\mathcal{D}}$, $\mathcal{R}_{\mathcal{D}}$, $\mathcal{C}_{\mathcal{S}}$, $\mathcal{R}_{\mathcal{S}}$, $\mathcal{C}_{\mathcal{DS}}$, and $\mathcal{R}_{\mathcal{DS}}$. These figures show poorer performance overall compared to those for the simulated point-source image (Figure 11). There are also several artifacts 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 finders. Here Aegean appears to perform the best, with Selavy showing much poorer performance. The sharp drop seen in $\mathcal{C}_{\mathcal{DS}}$ for all finders

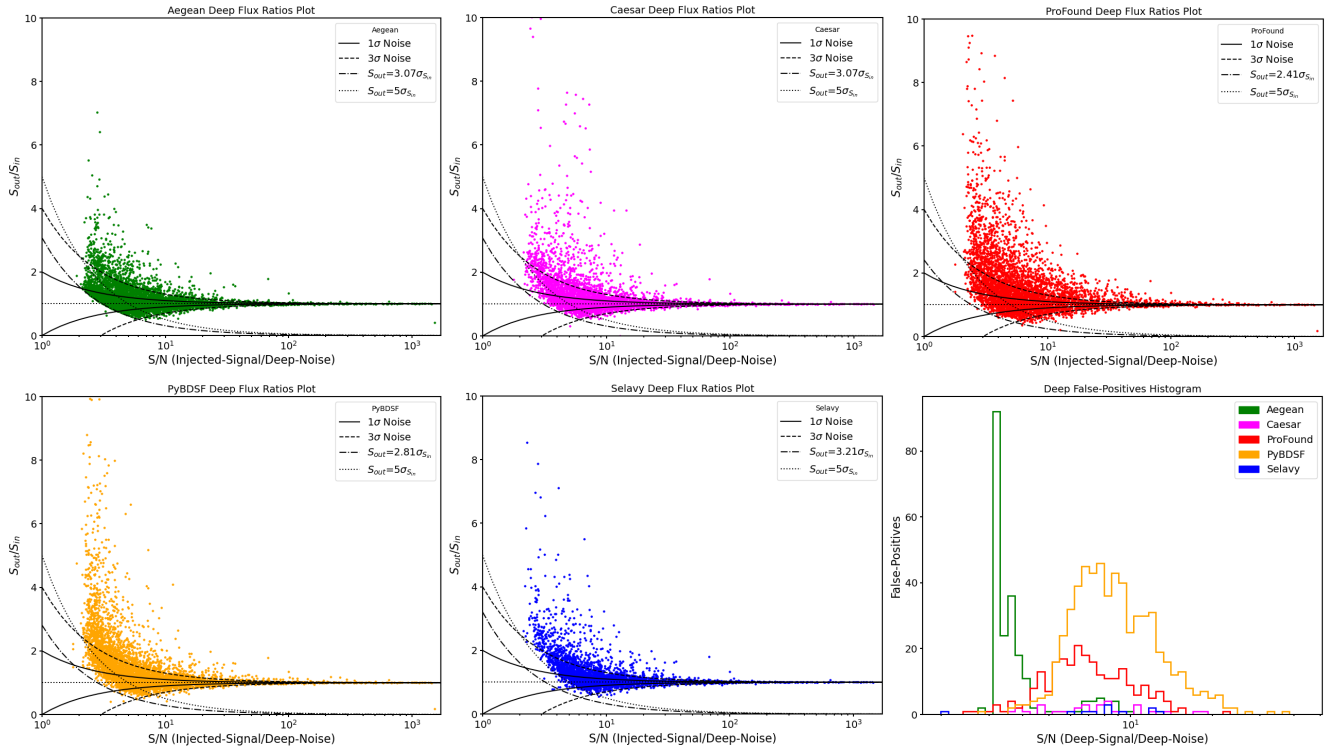


Figure 31. Flux-ratios and false-positive distribution as a function of injected S/N for the $2^\circ \times 2^\circ$ simulated point-source deep image; for colour codes, see Table 20. The 1σ (solid) and 3σ (dashed) curves are RMS noise (σ) deviations from the flux-ratio = 1 lines (dotted). Also shown are the detection threshold (dot-dashed curves; see Table 4) and nominal 5σ threshold (dotted curves).

Table 7 Typhon run metrics for $2^\circ \times 2^\circ$ simulated extended-source images.

Source	Image	Parameters				Detected	Residuals		
Finder	Depth	RMS	Island			N	RMS	MADFM	ΣI^2
Aegean	Deep	`seedclip`	3.074	`floodclip`	3.070	4,112	6.70E-05	5.60E-05	0.0144
Caesar	Deep	`seedThr`	3.206	`mergeThr`	3.000	3,618	5.40E-05	4.40E-05	0.0094
ProFound	Deep	`skycut`	2.412	`tolerance`	2.409	3,730	5.20E-05	4.30E-05	0.0087
PyBDSF	Deep	`thresh_pix`	2.809	`thresh_isl`	2.806	4,688	1.06E-04	5.40E-05	0.0366
Selavy	Deep	`snrCut`	3.206	`growthCut`	3.203	2,544	9.82E-04	5.80E-05	3.1231
Aegean	Shallow	`seedclip`	3.603	`floodclip`	3.599	1,287	3.21E-04	3.17E-04	0.3360
Caesar	Shallow	`seedThr`	3.603	`mergeThr`	3.000	1,297	3.10E-04	2.95E-04	0.3143
ProFound	Shallow	`skycut`	2.941	`tolerance`	2.938	1,138	3.11E-04	2.98E-04	0.3146
PyBDSF	Shallow	`thresh_pix`	3.338	`thresh_isl`	3.335	1,312	3.16E-04	3.13E-04	0.3244
Selavy	Shallow	`snrCut`	3.735	`growthCut`	3.732	787	6.23E-04	3.20E-04	1.2604

reveals a shortcoming of the deep-shallow comparison approach, identified in § 3.1.3 above. This is exacerbated in the simulation containing extended sources, where the likelihood of a finder characterising a given source differently in the shallow image from the deep version is increased. This inconsistency causes detected sources not to be matched between the two, leading to the poor inferred completeness and reliability.

We find in this scenario that a large number of the detections arise from an overlapping or close association between injected sources, or where an injected source falls near a positive noise fluctuation. This occurs in about $25 \pm 10\,\%$ of cases (more for some finders, less for others, and varying with S/N), consistent with the reported completeness and reliability statistics. Figure 34 shows an example of a very low S/N detection being influenced by an even fainter adjacent source. Here, only ProFound and PyBDSF, which have the lowest detection thresholds (Table 7), make detections, and only in the deep image. Both finders treat the two adjacent injected sources as a single source. Figure 35 shows an example of noise spikes appearing as low S/N detections. Again, only ProFound and PyBDSF make detections, and only in the deep image. There is little difference in apparent

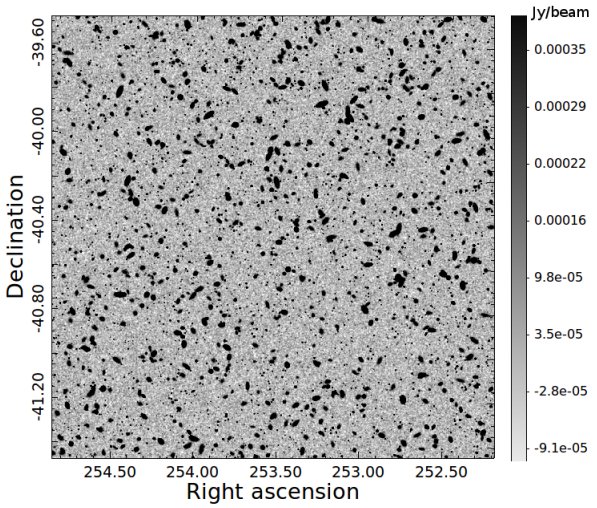


Figure 32. Simulated image with both point-like and extended sources. The sky coordinates are arbitrarily chosen.

flux density between the injected source detection at `match_id` 6607 and the spurious adjacent detections at `match_id`'s 6608 and 6608.

Figure 36 shows an example of injected sources spanning $0.28 < \mathrm{S/N} < 80$ in `clump_id` 50, with details in Table 8. This is an informative clump, as it encapsulates a moderately rich environment in which to explore detection anomalies. The residuals from the four finders that make detections here are shown in Figure 37.

Table 8 Summary of injected sources in Figure 36. The Detected column indicates if at least one source finder has detected an injected source.

$\mathcal{C}_\mathcal{D}$ S/N Bins (Figure 33)	Injected S/N	RMS (mJy)	`match_id` (Figure 36)	Detected (Figure 37)
0.318 ± 0.036	0.287	0.130	113	False
0.784 ± 0.088	0.810	0.129	112	False
2.42 ± 0.27	2.30	0.115	109	False
3.80 ± 0.43	4.01	0.114	111	False
36.2 ± 4.0	37.4	0.0949	110	True
71.2 ± 8.0	69.2	0.114	114	True

All source finders except Selavy make detections, but only at `match_id`'s 110 and 114. Selavy does, however, detect `match_id` 114 in the shallow image, and Figure 38 shows the residual shallow image for Selavy, for this clump. This may arise from Selavy's underlying Duchamp-style thresholding (Whiting, 2012).

From the flux-ratios in Figure 31, Selavy appears to produce a catalogue with an effective detection threshold of 5σ, regardless of its RMS parameter setting. Recalling that the extended sources have a maximum peak flux density of 1 mJy, as opposed to 1 Jy for their point-source counterparts, this suggests the issue may be related to the extended low surface brightness emission around such objects. There is a roughly 10% degradation in $\mathcal{C}_\mathcal{D}$, comparing the results for point-sources alone (Figure 11) to the simulation including extended-sources (Figure 33). While this degradation approximates the fraction of extended sources injected, not all the missed sources are extended. It does seem likely, though, that it is the extended sources that contribute disproportionately to the shortfall in $\mathcal{C}_\mathcal{D}$.

Comparing $\mathcal{C}_\mathcal{S}$ to $\mathcal{C}_\mathcal{D}$ (Figure 33), the distributions are consistent with the addition of the 5σ noise level (shifting the S/N axis by 5 units), for most finders (*e.g.*, the value of $\mathcal{C}_\mathcal{D}$ at $\mathrm{S/N} = 10$ is similar to the value of $\mathcal{C}_\mathcal{R}$ at $\mathrm{S/N} = 5$). Selavy shows a different behaviour here. Its performance in the shallow image appears to be better than in the deep image. At the lowest S/N end this is related to faint low surface brightness objects lying below the detection threshold in the shallow image, and not contributing to the $\mathcal{C}_\mathcal{S}$ and $\mathcal{C}_\mathcal{R}$ distributions. At moderate and high S/N, though, the reason for the difference is less clear. It may be that faint extended emission around bright objects misleads Selavy in the deep image, but being pushed below the noise level in the shallow image it makes the bright central region of those sources more easily detectable.

It is clear from the details in Table 8 that the sources not explicitly detected by any finder are at low S/N. Their detection is also confounded by the overlap with the brightest source (`match_id` 114). Aegean and ProFound have handled this by extending the size of the ellipse associated with `match_id` 114, influenced by the additional low S/N emission. This leads to over-estimation by Aegean of the source flux density and mischaracterisation of its orientation (evidenced by the oversubtraction in its residual). ProFound simply associates all the flux with the single object, which will lead to a flux density overestimate compared to the input flux density of `match_id` 114. Caesar searches for blobs nested inside large islands, allowing a more accurate characterisation of such overlapping sources. These residual statistics are demonstrated in Table 7. Here the residual statistic values are significantly lower for ProFound and Caesar than the other finders. The PyBDSF residual deep-image in Figure 37, which emphasises the undetected sources at `match_id`'s 109, 111, 112, and 113, suggests that a second detection pass on such residual images may have merit. How best to implement such an approach remains a challenge, as it may introduce new false detections if residuals such as seen here with Aegean dominate. This is likely to be exacerbated for extended objects.

At the high S/N end some failure modes already seen in the point-source simulation are repeated, such as missing sources at the edges of images, and fainter sources (although high S/N in their own right) that neighbour or overlap with brighter ones. An example of the latter is given in Figure 39. This is also an example of Caesar missing a bright detection ($\mathrm{S/N} \sim 220$) entirely. Even at the brightest end, sources can be missed by finders. An

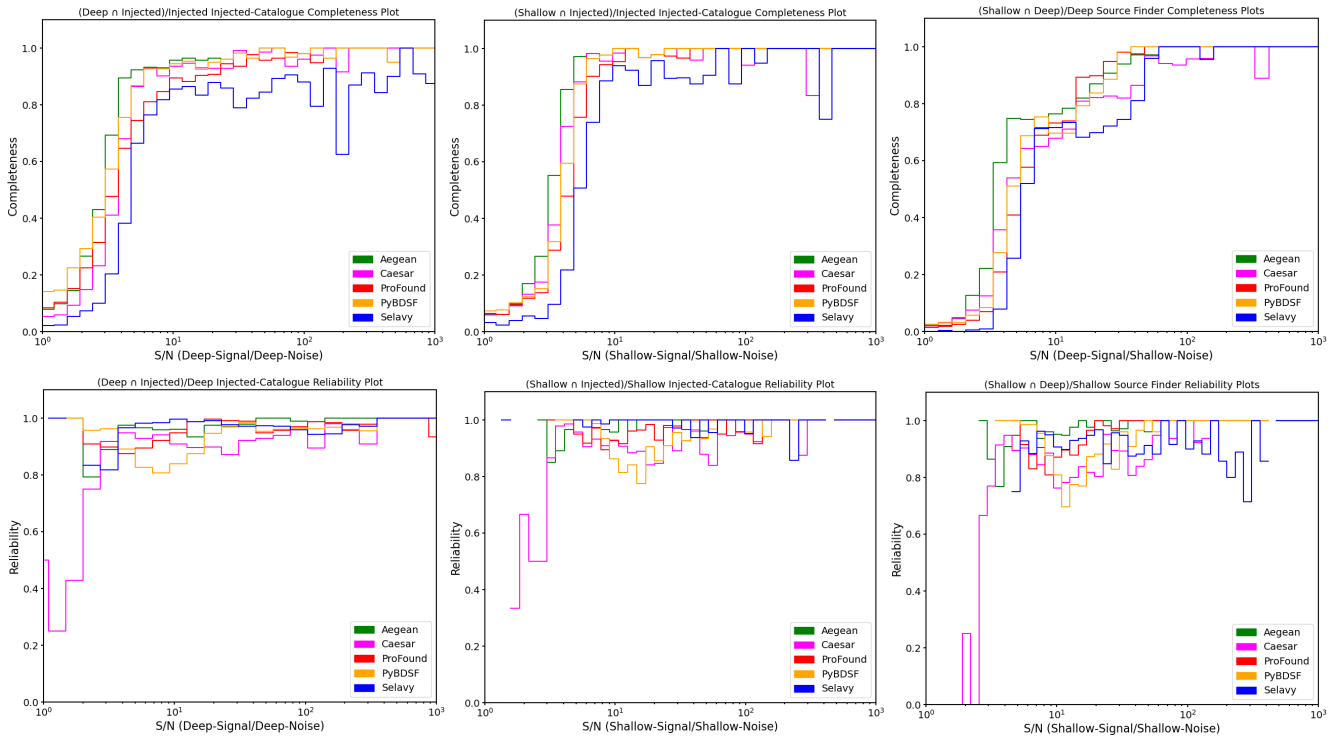


Figure 33. Completeness and reliability plots for the $2^\circ \times 2^\circ$ simulated image of extended and point sources, of $\mathcal{C}_\mathcal{D}$ *vs.* $\mathcal{D}$-signal/$\mathcal{D}$-noise (top-left), $\mathcal{R}_\mathcal{D}$ *vs.* $\mathcal{D}$-signal/$\mathcal{D}$-noise (bottom-left), $\mathcal{C}_\mathcal{S}$ *vs.* $\mathcal{S}$-signal/$\mathcal{S}$-noise (top-middle), $\mathcal{R}_\mathcal{S}$ *vs.* $\mathcal{S}$-signal/$\mathcal{S}$-noise (bottom-middle), $\mathcal{C}_{\mathcal{DS}}$ *vs.* $\mathcal{D}$-signal/$\mathcal{S}$-noise (top-right), and $\mathcal{R}_{\mathcal{DS}}$ *vs.* $\mathcal{S}$-signal/$\mathcal{S}$-noise (bottom-right); for colour codes, see Table 20. The deep/shallow injected noise maps used for determining S/N were generated using BANE.

example is Selavy failing to detect one of the two injected sources in S/N $\sim$ 4130 bin. Oddly, Selavy does detect this source in the shallow image, suggesting the failure in the deep image may be related to its background estimation in this case.

Turning to $\mathcal{R}_\mathcal{D}$ and $\mathcal{R}_\mathcal{S}$, at the lowest S/N levels (S/N $\lesssim$ 3) Caesar appears to be reporting large numbers of false sources, leading to poor reliability. All of these artifacts are of the same nature as those in Figures 24 and 25. They spatially coincide with the injected sources, but overestimate the flux density due to fitting an extreme ellipse with artificially large major axis. A catastrophic example of this effect is found at S/N $\sim$ 260, shown in Figure 40.

Figure 41 shows the major axis distributions for deep and shallow detections. This shows the clear delineation between compact and extended components in the simulation (*c.f.*, Figure 28). The detection of the extended components is evident. PyBDSF tends to most accurately recover the extended source sizes, but, along with Caesar also has the most outliers. These can be attributed to size overestimates due to inclusion of noise spikes or adjacent sources in the fitted sizes. All other finders, including ProFound, tend to underestimate extended source sizes. This may be a consequence of the low surface brightness wings of extended sources lying below detection thresholds. Selavy again shows a different characteristic from the other finders, appearing to underestimate the sizes for the mildly extended sources, while approaching the performance of Caesar and PyBDSF for the most extended ones. Components less than 15″ are attributed primarily to noise spikes, but also occasionally to underestimating the sizes of real sources.

The remaining failure modes are similar to those discussed already for compact sources (§ 3.1.2), associated with deblending and the presence of noise spikes. Adding extended sources exacerbates these issues in general.

3.2.3 Performance statistics

Figure 42 shows the residual completeness and reliability for our extended source simulation (*c.f.*, Figure 30). The $\delta\mathcal{C}_{\mathcal{DS}}$ and $\delta\mathcal{R}_{\mathcal{DS}}$ show similar characteristics to that seen for compact sources alone. As before, $\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$ should provide reasonable approximations to $\mathcal{C}$ and $\mathcal{R}$, and with the same limitations as already described. In particular, the potential for a source finder to inconsistently characterise a complex source between the deep and shallow images is a primary source of error here.

Figure 43 shows S_{out}/S_{in} as a function of S_{in} (expressed as S/N) for each source finder. The horizontal (dotted) lines represent $S_{out} = S_{in}$, and the solid and dashed curves about these lines are the 1σ and 3σ devi-

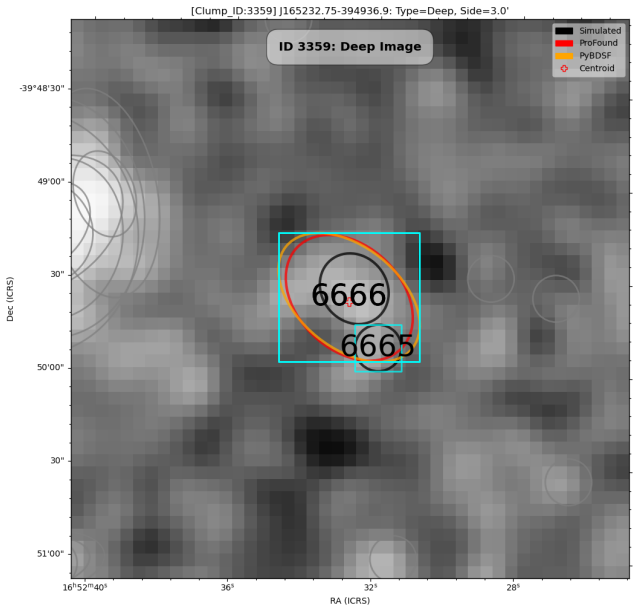


Figure 34. 1.58′×1.58′ cutout of `clump_id` 3359 of the 2°×2° simulated extended source deep image; for colour codes, see Table 20. There are two injected sources at `match_id`'s 6665 (S/N ∼ 0.070) and 6666 (S/N ∼ 0.336). Only `match_id` 6666 is detected by ProFound (S/N ∼ 7.468) and PyBDSF (S/N ∼ 12.446). This information was extracted from the S/N ∼ 0.318 ± 0.036 bin of $\mathcal{C}_{\mathcal{D}}$ (Figure 33).

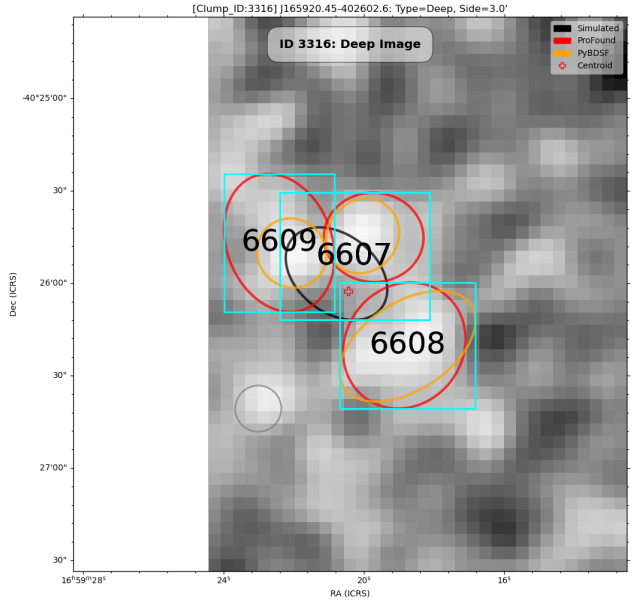


Figure 35. 1.58′ × 1.58′ cutout of `clump_id` 3316 of the 2° × 2° simulated extended source deep image; for colour codes, see Table 20. There is an injected source at `match_id` 6607 (S/N ∼ 0.34) which is detected by ProFound (S/N ∼ 4.82) and PyBDSF (S/N ∼ 6.39). The remaining detections, by both source finders, at `match_id`'s 6608 and 6609 are spurious, due to noise fluctuations. This information was extracted from the S/N ∼ 0.318 ± 0.036 bin of $\mathcal{C}_{\mathcal{D}}$ (Figure 33).

ations from S_{in}, respectively. The dot-dashed curves are

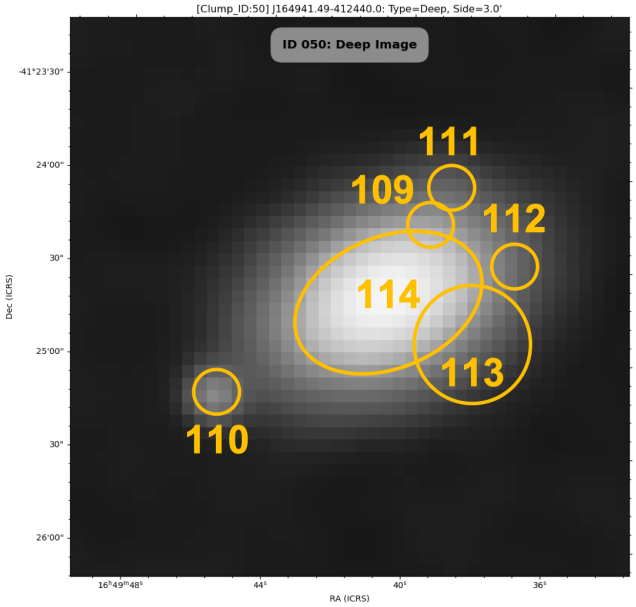


Figure 36. 2.51′ × 2.51′ cutout of `clump_id` 50 of the 2° × 2° simulated extended source deep image. The orange ellipses indicate injected sources, summarised in Table 8. The associated source finder detections are shown in Figure 37. These sources span 0.28 < S/N < 80 in $\mathcal{C}_{\mathcal{D}}$ (Figure 33).

the detection thresholds (*i.e.*, the RMS parameters in Table 7, corresponding to 90% PRD), and the (curved) dotted lines represent nominal 5σ thresholds (*c.f.*, Hopkins et al., 2015). Also shown are histograms giving the distribution of S/N in the false-positives for each source finder. The behavior of these diagnostics is similar to the case for compact sources (Figure 31), apart from the absence of the hard cutoff at S/N ∼ 2, arising from the inclusion of injected sources to much lower S/N. Again, Selavy seems to be limited to detecting sources only above a nominal 5σ threshold, despite being run with `snrCut`= 3.2 (Table 7).

Aegean and Selavy appear to perform best in the flux density characterisation, with the lowest scatter (Table 9). The scatter is noticeably more significant, though, compared to the point-source case (§ 3.1.3). This increase in scatter is attributed largely to flux density overestimates arising from overlapping sources not being well deblended, or the incorporation of adjacent faint sources or noise spikes in fitting a source. It is also worth noting that each of Caesar, ProFound, and PyBDSF systematically increase in their overestimate of flux densities as the S/N decreases, in contrast to the performance of Aegean, where the measured flux densities reliably hug the detection threshold line to the faintest levels. This is likely a consequence of adjacent noise fluctuations leading to overestimates of source size, which becomes more likely at progressively lower S/N.

Having explored simulated sources, both compact and extended, we now turn to an investigation drawing on

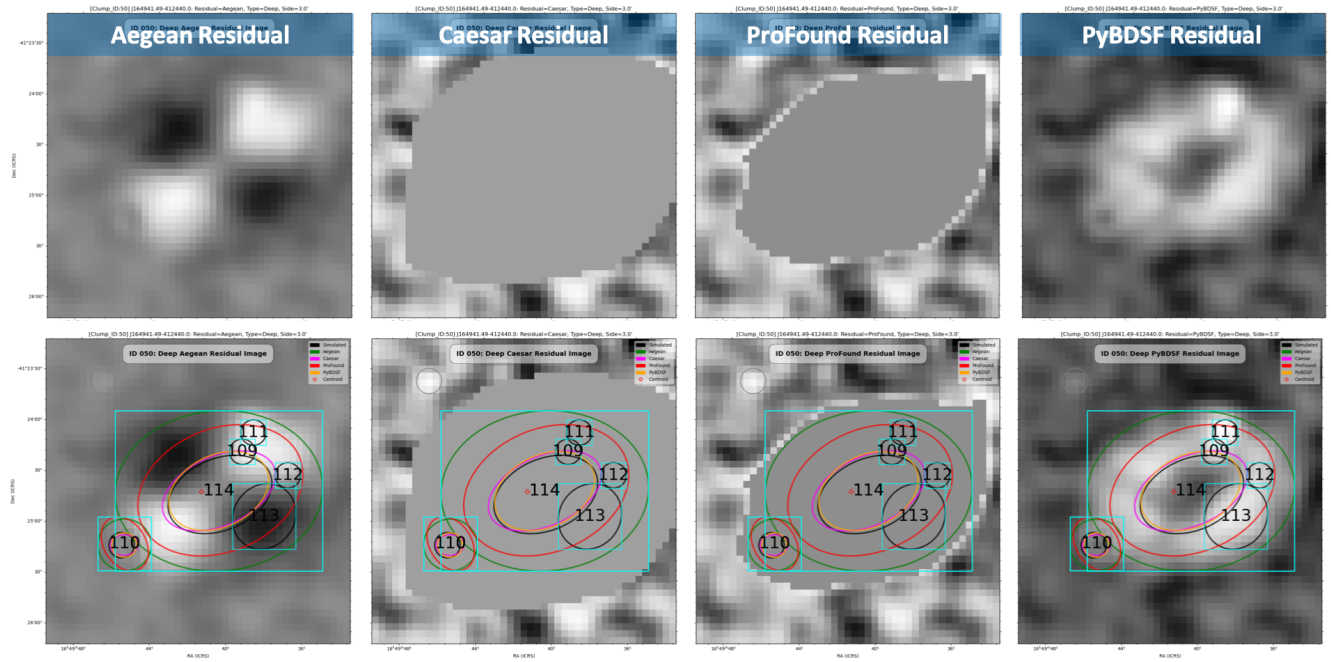


Figure 37. $2.51' \times 2.51'$ cutout, without (top) and with (bottom) annotation, of `clump_id` 50 of the $2° \times 2°$ simulated extended source deep image; for colour codes, see Table 20. The corresponding injected sources are shown in Figure 36. All source finders make detections, except Selavy. The remaining finders make detections at `match_id`'s 110 and 114 only.

Table 9 3σ scatter for flux-ratio plots in Figure 43.

Source Finder	3σ Scatter $s_{3\sigma}(3)$	$s_{3\sigma}(5)$	$s_{3\sigma}(10)$
Aegean	9.20%	7.98%	7.48%
Caesar	11.02%	9.92%	9.08%
ProFound	14.20%	15.93%	20.28%
PyBDSF	10.80%	9.55%	9.57%
Selavy	10.25%	8.91%	9.06%

real observational data.

3.3 Analysis of EMU Data

Figure 44 shows a $2° \times 2°$ cutout from the centre of an EMU pilot sample tile that is used for this analysis. This sample provides a good mixture of compact, extended, and complex sources including diffuse extended emission. Comparing the sample cutout with Figures 10 and 32 highlights the differences with real data, primarily through the addition of complex and diffuse sources as well as imaging artifacts.

Tables 10 and 11 show the Hydra run metrics for the sample. The RMS and island parameters are from the Typhon optimization routine, extracted at the 90% PRD level. Also shown are the residual image RMS, MADFM, and ΣI^2 statistics. The number of detected components range from 5,880 to 11,484 in the image (Table 11). The latter component count is from ProFound, which has the lowest RMS threshold at 2.28. Selavy detects the fewest components, with the highest threshold of 2.94. In terms of the RMS residual statistics, ProFound, closely followed by Caesar, show the lowest values, likely a consequence of their approach of subtracting all the flux associated with detected islands. Next is Aegean, then Selavy, then PyBDSF. The results are the same in the shallow version of the image, except that PyBDSF performs better than Selavy in this case.

Table 10 Hydra μ, `box_size`, and `step_size` run-parameters for the $2° \times 2°$ cutout of the EMU pilot survey tile. The box and step size parameters were used as inputs for Aegean, PyBDSF, and Selavy.[a]

Image	μ (μJy)	`box_size` (pixels)	`step_size` (pixels)
Deep	35.28	360	135
Shallow	171.23	108	40

[a] The Selavy module only accepts `box_size`.

Figure 45 shows the deep-shallow completeness and reliability ($\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$) as a function of S/N. With real data, we have no underlying "true" source list to refer to, so we rely on using the sources detected in the deep version of the image as the reference for assessing completeness and reliability in the shallow version of the image. We adopt this approach in the absence of a better alternative, noting the limitations identified in the analysis of the simulations above, but reassured by the relatively small $\delta\mathcal{C}_{\mathcal{DS}}$ and $\delta\mathcal{R}_{\mathcal{DS}}$ in general (§ 3.1.2 and

Table 11 Typhon run metrics for $2^\circ \times 2^\circ$ EMU pilot tile sample cutout.

Source	Image	Parameters				Detected	Residuals		
Finder	Depth	RMS		Island		N	RMS	MADFM	ΣI^2
Aegean	Deep	`seedclip`	2.676	`floodclip`	2.674	8,538	9.10E-05	2.60E-05	0.1088
Caesar	Deep	`seedThr`	2.809	`mergeThr`	2.806	7,838	2.70E-05	2.30E-05	0.0100
ProFound	Deep	`skycut`	2.279	`tolerance`	2.277	11,484	2.40E-05	1.90E-05	0.0085
PyBDSF	Deep	`thresh_pix`	2.941	`thresh_isl`	2.938	8,292	1.08E-03	2.60E-05	15.0605
Selavy	Deep	`snrCut`	2.941	`growthCut`	2.938	5,880	5.66E-04	2.70E-05	4.1532
Aegean	Shallow	`seedclip`	3.603	`floodclip`	3.599	926	1.92E-04	1.69E-04	0.4801
Caesar	Shallow	`seedThr`	3.735	`mergeThr`	3.000	885	1.69E-04	1.66E-04	0.3689
ProFound	Shallow	`skycut`	3.735	`tolerance`	3.732	778	1.69E-04	1.65E-04	0.3700
PyBDSF	Shallow	`thresh_pix`	3.735	`thresh_isl`	3.732	794	4.09E-04	1.69E-04	2.1648
Selavy	Shallow	`snrCut`	3.603	`growthCut`	3.599	789	5.27E-04	1.69E-04	3.5938

§ 3.2.2). Some degradation in the quality of these metrics is expected, however, due to the presence of diffuse emission and artifacts in the real data. Bearing this in mind, it is evident that the detections are fairly complete for all finders, with high completeness ($\mathcal{C}_{\mathcal{DS}} \gtrsim 0.9$) above S/N $\gtrsim 10$ dropping to $\mathcal{C}_{\mathcal{DS}} \sim 0.5$ by S/N ~ 5. Reliability is also generally high ($\mathcal{R}_{\mathcal{DS}} \gtrsim 0.9$) for most finders, although Selavy's performance here is notably poorer ($0.7 \lesssim \mathcal{R}_{\mathcal{DS}} \lesssim 0.9$) across almost the full range of S/N. All finders drop in reliability below S/N $\sim 10 - 20$.

Figure 46 shows a result that is typical for compact sources in this image, robustly detected by all finders, for an example at S/N ~ 12. There are also the typical effects for sources lying at the edges of images, as shown in Figure 47 (*c.f.*, Figure 12), as well as the expected artifacts due to noise fluctuations, as shown in Figure 48 at low S/N. Different effects are seen when considering real sources with diffuse emission.

Figure 49 shows an example of a bright source, slightly extended and with a tail of diffuse emission. Caesar decomposes this object into two sources in the deep image, but not in the shallow (Figure 50). The remaining source finders detect this as a single source in both the deep and shallow images. The diffuse emission is washed out in the shallow image, leading to the flux densities measured for this source by all finders being underestimated compared to their results in the deep image.

Figure 51 shows an example of a source with complex structure including diffuse emission at S/N ~ 25. Each source finder apart from ProFound characterises this source (and a fainter neighbouring source to the south) by decomposing it into two components. Each finder's two-component solution is different though, leading to the three separate `match_id`'s 1697, 1698, 1699. All finders identify `match_id` 1697, corresponding to the brightest portion of the source. Selavy and Caesar identify (differently) the emission to the northeast as a separate component (`match_id` 1699). PyBDSF and Aegean identify the faint neighbour to the south (`match_id` 1698) as the second component, although Aegean drastically overestimates its extent. These additional components tend not to be identified by finders in the shallow image, due to the diffuse emission being masked at the higher noise level. This appears to be the primary cause for the reduction in $\mathcal{C}_{\mathcal{DS}}$ seen at high S/N.

We now consider an example of a source with a combination of complex structures, `clump_id` 2293, shown in Figures 53 and 54, with details in Table 12. Here we have chosen to list the MADFM as our representative statistic to characterise residuals (e.g., Riggi et al., 2019)), although the other statistics are still computed. This complex of emission provides a good illustration of how the different source finders perform with multiple overlapping, extended and diffuse structures.

The system is shown in Figure 52, with bright compact features labelled (a) through (e), and diffuse extended emission as (χ), (ϵ), and (λ). This main complex has the label B2293 (not shown), and is identified as a separate clump here from the bright component labelled C2294. While the emission in C2294 seems highly likely to be related to the emission from B2293, our approach to associating detections into clumps relies on their measured sizes touching or overlapping[25]. The compact nature of C2294 leads the finders to characterise it with ellipses that do not overlap with any ellipse in B2293, and these are consequently marked as independent clumps. The source at `match_id` 2667, indicated in the figure, does end up being associated with clump B2293, although it is not physically linked to the complex system (B2293+C2294).

It corresponds to `match_id` 2668 (*i.e.*, $26.6'' \times 26.6''$) of `clump_id` 2294. The clump information is provided in the lower partition of Table 12; the annotations are greyed out, in the aforementioned figures. In general, we

[25]This condition is a result of setting $f = 1$ to define the "skirt" size in Equation 10. Setting $f = 1.5$ more effectively clumps components of complex structures together, but can also lead to linking unassociated neighbouring components. We chose to fix $f = 1$ in our analysis throughout for convenience.

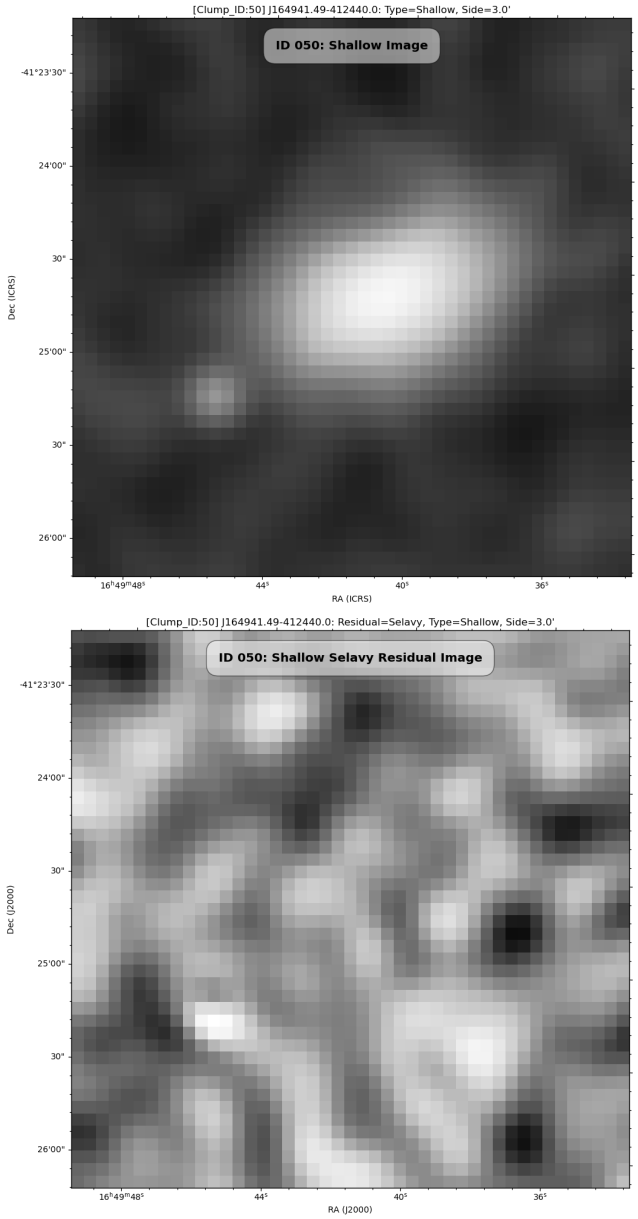


Figure 38. $2.51' \times 2.51'$ image (top) and Selavy residual-image (bottom) cutouts of `clump_id` 50 (*re.*, Figures 36 and 37) of the $2° \times 2°$ simulated extended-source shallow-image. Figures 36 shows the location of the injected-sources for this image, wherein Selavy makes detections at `match_id`'s 110 and 114; as well as, the other source finders. Caesar is the only source finder that detects the remaining sources, except at `match_id` 109.

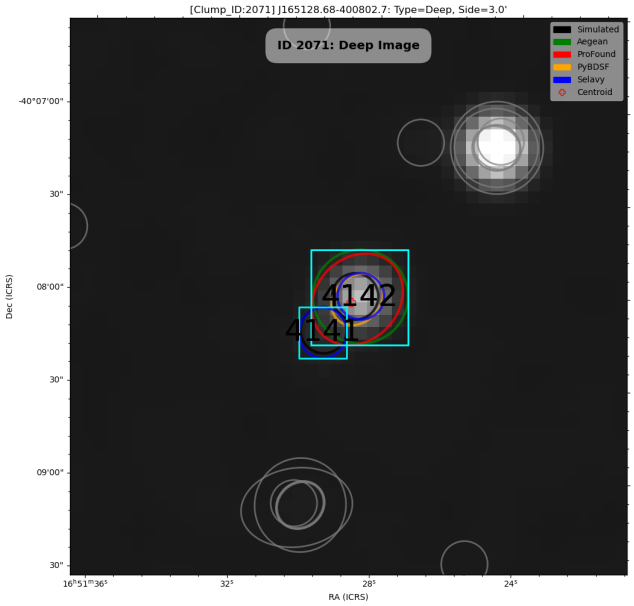


Figure 39. $0.643' \times 0.643'$ image cutout of `clump_id` 2071 of the $2° \times 2°$ simulated extended source deep image; for colour codes, see Table 20. All source finders detected the injected source at `match_id` 4142 ($(\mathrm{S/N})_{injected} \sim 226$), except for Caesar. Only Selavy detects the adjacent fainter source, `match_id` 4141, where $(\mathrm{S/N})_{injected} \sim 19$. This information was extracted from the $\mathrm{S/N} \sim 220 \pm 25$ bin of $\mathcal{C}_{\mathcal{D}}$ (Figure 33).

will focus our attention on the bent-tail source (B2293), referring to the compact source (C2294) as required.

The following discussion shall be implicitly in reference to the information provided in Figures 52, 53, and 54, and Table 12, with explicit reference where clarity is required.

The interpretation of (ε) and (χ) as purely diffuse emission is somewhat subjective, as there may be unresolved sources. The shallow image tends to expose (ε) as a possible source, when contrasted with its deep counterpart. Indeed, Caesar does detect a small ($a \sim 10.9''$, $b \sim 6.24''$) faint ($135\,\mu$Jy) source at `match_id` 2662. However, there is no corresponding shallow detection. Visual inspection (*re.*, image-stretching) seems to indicate (χ) is a strong outflow from (a), potentially ramming into intergalactic medium and/or a compact object, producing a distinctive bent-tail. None of the source finders make a detection, however (χ) does appear to be exposed in the residual images of Aegean (deep+shallow) and PyBDSF (deep). Selavy does subtract out (χ) (`match_id` 2661, shallow) and C2294 (`match_id` 2660, deep+shallow), both in the deep and shallow residual-image cutouts, exposing the would-be-outflow. However, this is at the expense overcompensating its source detection. As for (λ), both visual inspection and the residual-images indicate it as diffuse emission. We shall now examine the results of each source finder.

Aegean nicely matches C2294 (`match_id` 2668) and the identified source at (a) (`match_id` 2660) for the deep and shallow images. As noted in previous discussions (*re.* Figure 51), it performs poorly for sources with diffuse emission; as is evident in its deep detections, at `match_id`'s 2661, 2662 and 2663. The component footprints overestimate the extent of B2293's components along the outflow from (χ) to (λ). The largest of them being at `match_id` 2663, with $a \sim 157.56''$ and $b \sim 52.04''$. It is centered about source (d), with its major axis biased along $(\lambda) - (e)$ (*i.e.*, $\theta \sim 38.32°$) as if to compensate

Table 12 Cluster table information for `clump_id`'s 2293 (upper partition) and 2294 (lower partition). The images shown in Figures 53 and 54 are for the upper partition. The source component footprint parameters, a, b, and θ, correspond to the major axis, minor axis, and position angle, respectively. The S/N is wrt the shallow image (*a la* BANE). The MADFM's are computed, within the source component footprints and normalized wrt their areas, for the residual images.

Match ID	Source Finder	Image Depth	RA (°)	Dec (°)	a ($''$)	b ($''$)	θ (°)	Flux (mJy)	S/N	MADFM mJy/(I^2beam)
2660	Aegean	Deep	320.614	-56.011	33.75	22.85	82.447	51.0417	105.4360	2.4513e-03
2660	Aegean	Shallow	320.614	-56.012	31.88	22.87	82.612	49.0351	101.2908	1.0319e-02
2660	Caesar	Deep	320.614	-56.011	53.35	19.46	-79.362	52.5567	108.5653	5.5643e-04
2660	Caesar	Shallow	320.612	-56.012	39.18	19.61	-76.716	37.7645	79.6115	5.0744e-03
2660	ProFound	Deep	320.619	-56.013	43.97	26.57	109.235	69.8364	136.9028	6.1224e-05
2660	ProFound	Shallow	320.624	-56.016	68.82	35.74	141.154	85.2258	159.2764	5.6682e-03
2660	PyBDSF	Deep	320.619	-56.017	128.89	62.74	155.166	126.1139	246.0558	1.2288e-02
2660	Selavy	Deep	320.615	-56.012	42.40	23.71	102.080	61.3140	125.3935	2.6793e-03
2660	Selavy	Shallow	320.613	-56.012	31.47	22.05	100.170	46.5150	97.0618	1.0414e-02
2661	Aegean	Deep	320.642	-56.017	55.90	48.05	-84.490	17.5477	37.3929	2.4513e-03
2661	Aegean	Shallow	320.642	-56.017	73.95	27.27	-49.783	17.7016	38.2606	1.0319e-02
2661	Caesar	Deep	320.638	-56.016	165.70	78.08	8.087	73.9215	151.3647	5.5643e-04
2661	Caesar	Shallow	320.644	-56.020	248.30	90.18	20.210	90.2743	195.1955	5.0744e-03
2661	Selavy	Shallow	320.634	-56.014	112.44	46.23	163.650	60.2680	118.3390	1.0414e-02
2662	Aegean	Deep	320.636	-56.024	83.96	64.66	-55.037	17.7797	34.2307	2.4513e-03
2662	Caesar	Deep	320.627	-56.021	10.93	6.24	-78.426	0.1349	0.2410	5.5643e-04
2662	PyBDSF	Shallow	320.624	-56.018	111.50	56.04	162.988	156.9845	293.7207	1.7932e-02
2663	Aegean	Deep	320.639	-56.027	157.56	52.04	38.318	3.2044	6.4656	2.4513e-03
2663	Aegean	Shallow	320.639	-56.029	106.85	35.06	-4.002	21.3041	44.3798	1.0319e-02
2663	ProFound	Deep	320.641	-56.030	61.32	32.77	29.961	21.4079	46.5896	6.1224e-05
2664	Aegean	Deep	320.630	-56.048	98.68	47.38	5.068	14.5981	43.9117	2.4513e-03
2664	Aegean	Shallow	320.627	-56.052	36.73	34.34	-12.335	6.0130	21.4578	1.0319e-02
2664	ProFound	Deep	320.621	-56.053	47.60	35.67	5.535	7.5893	28.1344	6.1224e-05
2664	ProFound	Shallow	320.625	-56.054	41.23	34.49	173.862	10.5205	39.3395	5.6682e-03
2664	Selavy	Shallow	320.628	-56.052	50.22	43.97	128.550	10.8400	37.7821	1.0414e-02
2665	Aegean	Deep	320.640	-56.055	19.85	17.86	-84.110	0.8030	3.3544	2.4513e-03
2665	ProFound	Deep	320.641	-56.056	29.00	23.98	178.631	2.4034	10.3152	6.1224e-05
2666	Aegean	Deep	320.599	-56.056	57.28	28.06	85.426	0.9795	4.0272	2.4513e-03
2666	ProFound	Deep	320.593	-56.056	19.18	16.12	31.535	0.3273	1.3524	6.1224e-05
2666	PyBDSF	Deep	320.593	-56.056	17.38	15.17	80.152	0.1721	0.7110	1.2288e-02
2667	Aegean	Deep	320.666	-55.989	18.68	14.42	2.981	0.1361	0.5255	2.4513e-03
2667	Caesar	Deep	320.666	-55.989	22.20	17.07	-76.756	0.1552	0.5994	5.5643e-04
2667	Caesar	Shallow	320.626	-56.012	59.98	23.76	-62.079	22.7529	42.1717	5.0744e-03
2667	ProFound	Deep	320.651	-55.995	18.49	9.79	54.245	0.1416	0.3971	6.1224e-05
2667	PyBDSF	Deep	320.666	-55.989	19.98	14.83	7.938	0.1606	0.6202	1.2288e-02
2667	Selavy	Deep	320.666	-55.989	19.92	14.88	12.690	0.1610	0.6216	2.6793e-03
2668	Aegean	Deep	320.595	-56.004	18.82	18.40	-87.136	20.9655	53.8435	1.3433e+00
2668	Aegean	Shallow	320.595	-56.004	18.65	18.41	87.954	20.9354	53.7664	1.1211e+00
2668	Caesar	Deep	320.595	-56.004	27.80	16.61	-68.410	20.6564	53.0499	0.0000e+00
2668	Caesar	Shallow	320.595	-56.004	25.56	20.72	-48.870	21.5519	55.3497	0.0000e+00
2668	ProFound	Deep	320.596	-56.004	16.68	16.01	141.799	21.8717	55.5098	0.0000e+00
2668	ProFound	Shallow	320.596	-56.004	16.05	14.91	132.551	21.5683	54.7398	0.0000e+00
2668	PyBDSF	Shallow	320.595	-56.004	19.06	18.04	123.511	20.8894	53.6482	7.8357e+00
2668	Selavy	Deep	320.595	-56.004	18.53	17.57	99.270	19.9640	51.2715	1.1786e+00
2668	Selavy	Shallow	320.595	-56.004	19.04	18.21	123.250	21.3080	54.7232	4.7455e-01

for outflow. As for its shallow counterpart, its footprint estimates, $a \sim 106.85''$ and $b \sim 35.06''$, and orientation, $\theta \sim -4.00°$, better characterize the $\{(e), (\alpha), (d), (\beta)\}$ complex, as the diffuse emission is washed out. In short,

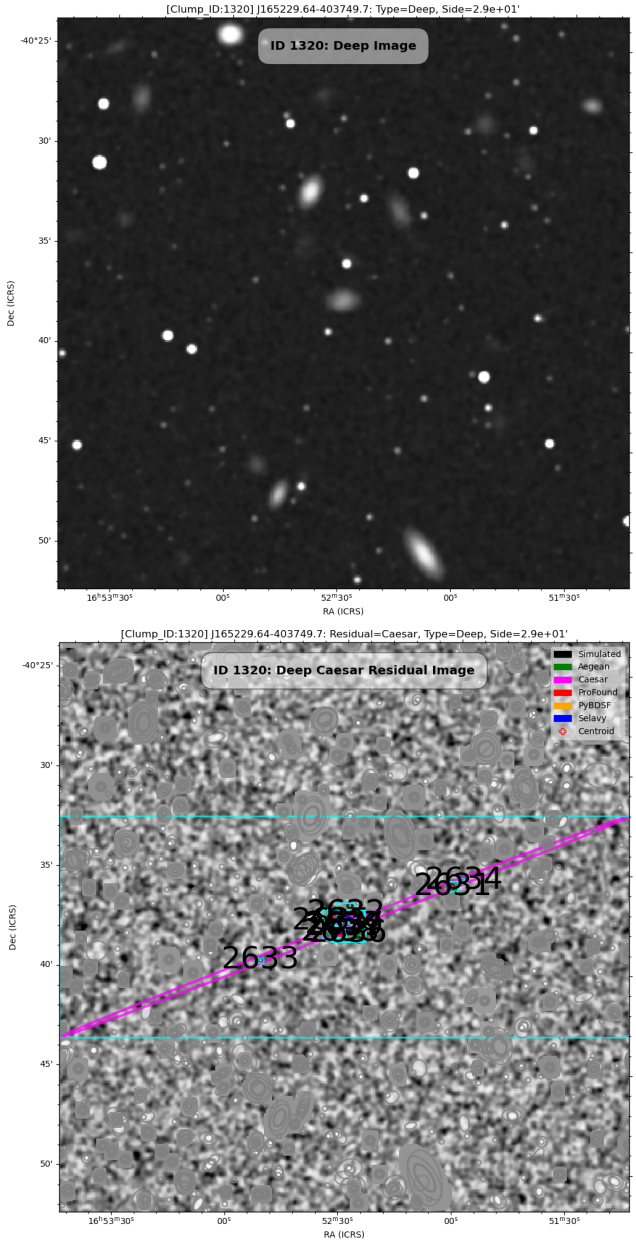


Figure 40. 28.5′ × 28.5′ image (top) and Caesar residual-image (bottom) cutouts of `clump_id` 1320 of the 2° × 2° simulated extended source deep image; for colour codes, see Table 20. Caesar overestimates the flux density at `match_id` 2639, with 18 mJy, compared to 0.074 mJy for the injected source. Its respective semi-major and semi-minor axes are 920″ and 11″, compared to 20″ and 10″ for the injected source. This information was extracted from the S/N ∼ 259 ± 39 bin of $\mathcal{C}_{\mathcal{S}}$ (Figure 33).

for sources with diffusion emission, Aegean tends to be biased towards the flux-weighted center, with its overall footprint compensating for the total flux within. In turn, this affects the position and size accuracy of the region it is trying to characterize.

The deep and shallow S/N at `match_id` 2663 are 6.5 and 44.4, respectively. So we find no contribution to the S/N ∼ 6 bin of $\mathcal{C}_{\mathcal{DS}}^{Aegean}$ (∼ 0.8, Figure 45 top). How-

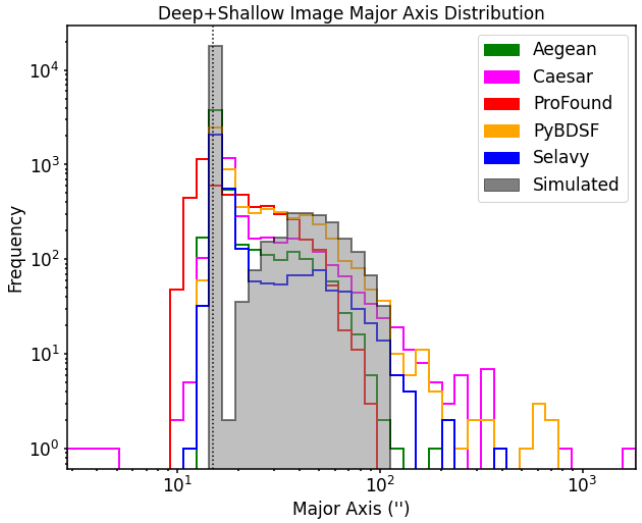


Figure 41. Major axis distributions for the 2° × 2° deep and shallow extended-source simulated images. The vertical dashed-line represents the beam size.

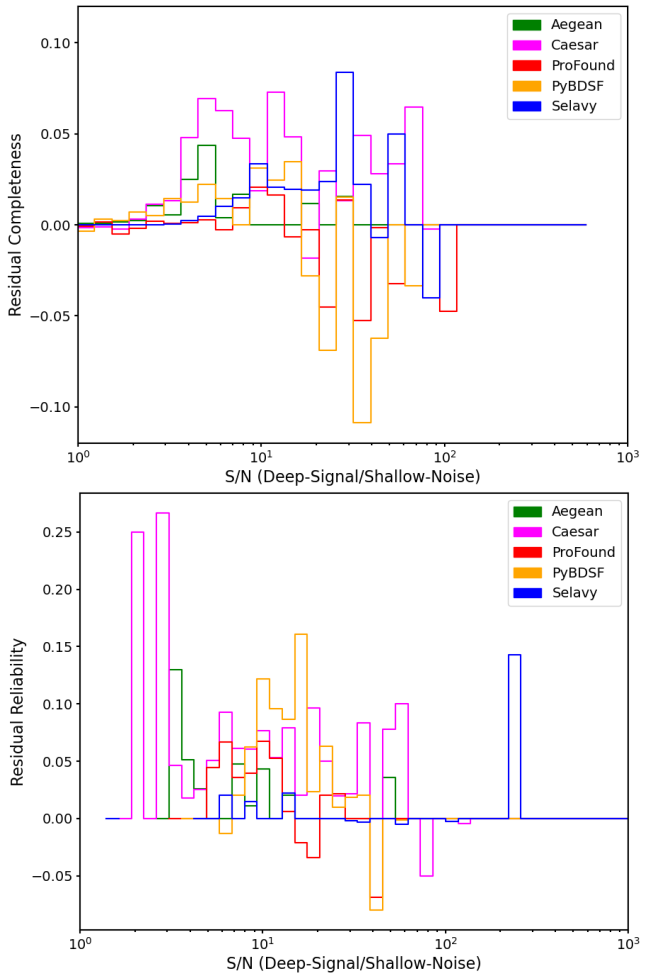


Figure 42. Residual completeness (top) and reliability (bottom) for the 2° × 2° extended source simulated image.

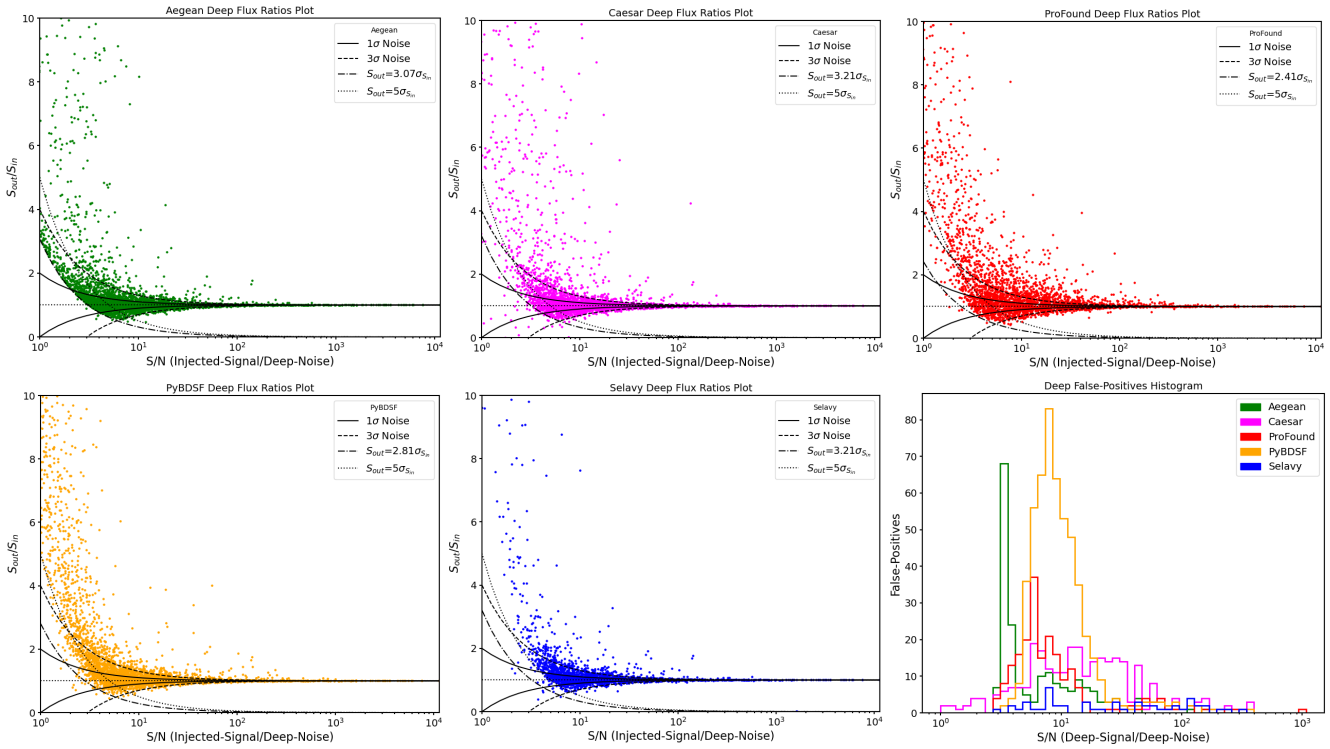


Figure 43. Flux-ratios and false-positive S/N distribution as a function of injected S/N for the $2^\circ \times 2^\circ$ simulated extended source deep image; for colour codes, see Table 20. The 1σ (solid) and 3σ (dashed) curves are RMS noise (σ) deviations from the flux-ratio = 1 lines (dotted). Also shown are the detection threshold (dot-dashed curves; see Table 7) and nominal 5σ threshold (dotted curves).

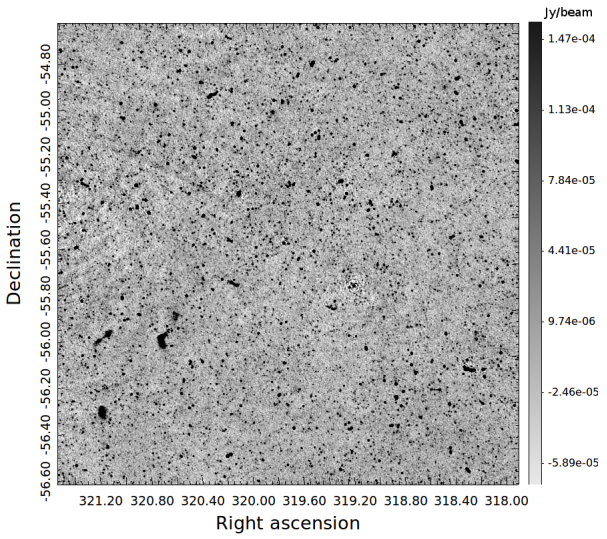


Figure 44. $2^\circ \times 2^\circ$ central cutout from an EMU pilot sample tile.

ever, given the complexity of B2293, it is not surprising; as not all of the flux is accounted for in the region of `match_id` 2663. Accounting for all of the fluxes, intersecting deep (*i.e.*, `match_id`'s 2660 through 2665) and shallow (*i.e.*, `match_id`'s 2660, 2661, 2663, and 2664) components, about `match_id` 2663, we find total fluxes of 87.19504 *m*Jy and 94.053734 *m*Jy, respectively (*cf.*, Table 12). Naïvely, dividing by a shallow noise of 0.4956 *m*Jy,[26] at `match_id` 2663, we find S/N's of roughly 176 and 190, respectively. This would correspond to a contribution to S/N [165, 197) (*i.e.*, S/N $\sim$ 181) bin of our $\mathcal{C}_{\mathcal{DS}}$ plot, assuming such a metric. Similar considerations could also be given for $\mathcal{R}_{\mathcal{DS}}$.

Caesar picks up C2294 (`match_id` 2668) and (a) (`match_id` 2660) for the deep and shallow images. In the deep image, the outflow (χ) is folded in with its estimate of the source (a). In the shallow image, it resolves "(a)", into (a) and (χ) (`match_id` 2667). Recall that Caesar is designed to pickup diffuse emission (*cf.*, Riggi et al., 2016), which is evident in the extent of its residual-image footprint (representing the parent-blob), shown in Figure 53 (last column). It correctly characterizes the extent and shape of the diffuse emission. The individual components (child-blocs) are sifted out in its second iteration. So it looks like this is a matter of fine tuning, to correctly characterise (a) and its outflow (χ).[27] Regardless, all of the source finders were tuned as generalists, given that fine tuning by hand is impractical

[26] This can be estimated by dividing the total flux density by the S/N for the deep image at `match_id` 2663, given in Table 12.

[27] Something a more sophisticated multi-parameter (*i.e.*, > 2) version of Hydra should be able to manage. In short, a second set of RMS and island parameters would be required for Caesar's child-blobs (*cf.*, Table 21).

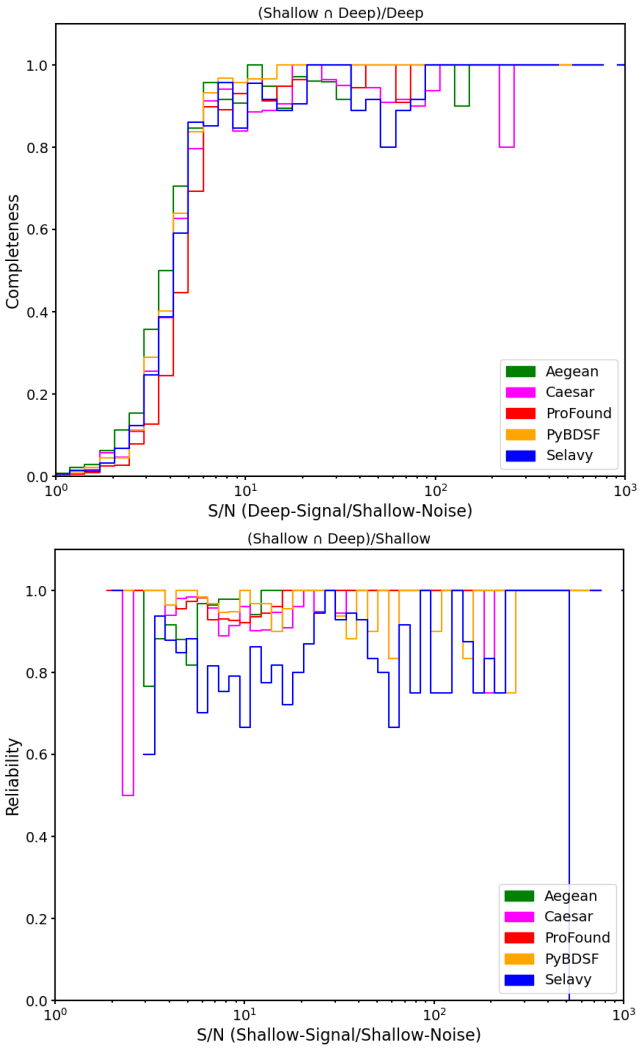


Figure 45. Deep-shallow completeness and reliability plots of $\mathcal{C}_{\mathcal{DS}}$ *vs.* $\mathcal{D}$-signal/$\mathcal{S}$-noise (top) and $\mathcal{R}_{\mathcal{DS}}$ *vs.* $\mathcal{S}$-signal/$\mathcal{S}$-noise (bottom), respectively, for the $2^\circ \times 2^\circ$ EMU pilot tile sample cutout; for colour codes, see Table 20. The deep/shallow noise maps used for determining S/N were generated using BANE.

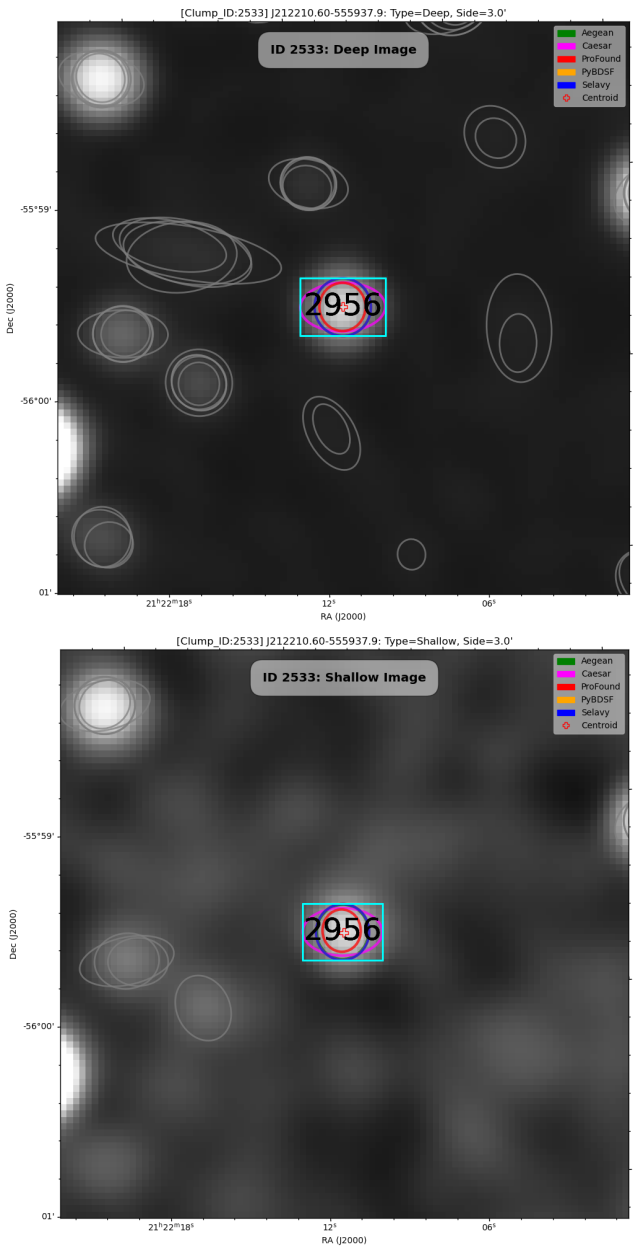


Figure 46. $0.447' \times 0.447'$ cutouts of `clump_id` 2533 of the $2^\circ \times 2^\circ$ EMU deep (top) and shallow (bottom) sample images; for colour codes, see Table 20. All source finders detect the source at `match_id` 2956 in both the deep and shallow images.

when dealing with millions of sources (*re.* EMU, Norris et al., 2011).

With the exception of detecting (ε) in the deep image (`match_id` 2662 with S/N $\sim 0.2\,m$Jy), Caesar does poorly in terms of characterizing sources downstream of (χ). It attempts to fit everything to a single component (`match_id` 2661), both in the deep ($a \sim 166''$, $b \sim 78.1''$, and $\theta \sim 8.09^\circ$ with S/N $\sim 73.9\,m$Jy) and shallow ($a \sim 248''$, $b \sim 90.2''$, and $\theta \sim 20.2^\circ$ with S/N $\sim 90.3\,m$Jy) images. These correspond to S/N bins 152 and 181, respectively,[28] and so do not contribute to $\mathcal{C}_{\mathcal{DS}}$ (Figure 45 top); even taking into consideration components that intersect with `match_id` 2661. This failure mode is reminiscent of the simulated point source case, shown in Figures 24 and 25 (*cf.*, Figure 40 for extended case), but biased by diffuse emission. Comparing intersecting components with `match_id`'s 2663 and 2661, for Aegean and Caesar, respectively, we find a 9% percent difference in S/N, tacitly assuming Aegean is the ground truth.[29] So the size of Caesar's deep-shallow footprints appears to account for the flux along (χ) − (λ), but over estimates its extent.

ProFound does a decent job at characterizing the features of B2293+C2294 in the deep image, but not the

[28] Using a shallow noise estimate of $0.488\,m$Jy (*cf.*, footnote 26).

[29] That is, taking an average deep–shallow percentage difference, between the source finders.

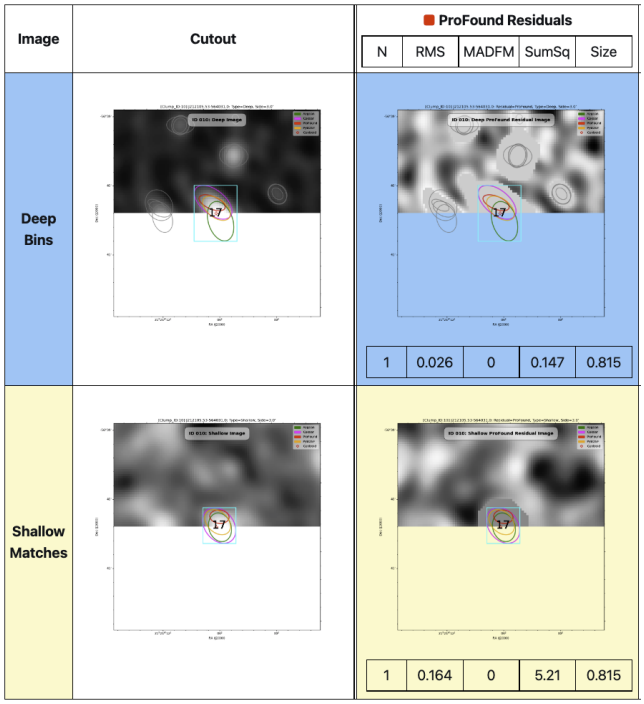


Figure 47. $0.815' \times 0.815'$ EMU image (left) and ProFound residual-image (right) cutouts of `clump_id` 10 of the $2° \times 2°$ EMU deep (top) and shallow (bottom) sample images; for colour codes, see Table 20, and for annotations, see Figure 8. All source finders detect the source at `match_id` 17 in both the deep and shallow images. Only ProFound's detections are consistent enough between the deep and shallow images for the source to be counted in the completeness statistic.

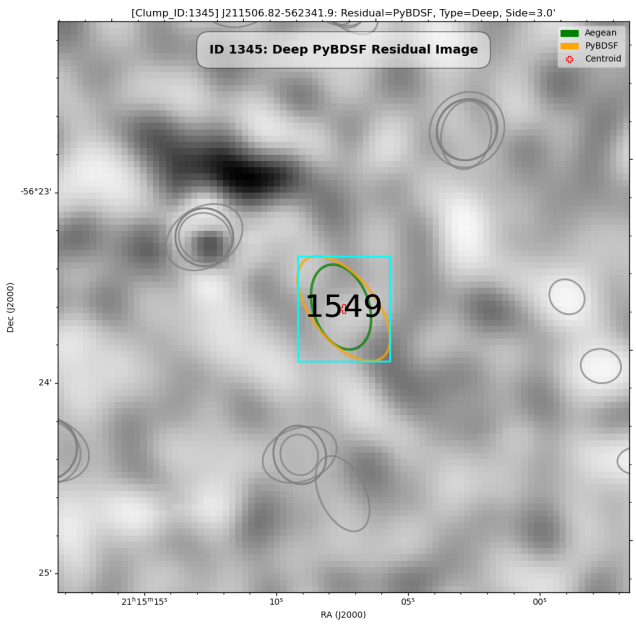


Figure 48. PyBDSF $0.552' \times 0.552'$ residual image cutout of `clump_id` 1345 of the $2° \times 2°$ EMU deep sample image; for colour codes, see Table 20. Aegean and PyBDSF detect a deep source (`match_id` 1549) here, although visually this object is consistent with a noise spike. With no corresponding shallow detection, such results contribute to the inferred incompleteness at low S/N.

shallow. In the former case, it does not pick up on the fine features, but it does characterize the extent of the emission (*re.* the residual images, Figure 54 top left), and the overall shape and orientation of B2293's source elements. It tends to glom sources with outflows: *i.e.*, it associates $(a) + (\chi)$, $(e) + (\varepsilon) + (d) + (\lambda)$, (b), and (c) with `match_id`'s 2660, 2663, 2664, and 2665, respectively. This is consistent with how ProFound functions, it finds diffuse islands and characterizes them one-to-one with flux-weight components. So its behaviour should not come as a surprise.

As for the shallow image, ProFound does not do as well, because the diffuse emission is washed out. Comparing the deep and shallow residual images in Figure 54 (top and bottom left, respectively), it becomes clear, as there is a significant shrinkage in its footprint (this also happens with Caesar, but to a lessor degree, *cf.*, Figure 53 right column). As such, B2293 is broken down into 2 gross components, everything along outflow $(a) - (\lambda)$ at `match_id` 2660, and $(b) + (c)$ at `match_id` 2664. Focusing on shallow–`match_id` 2660, we see it appears to be a merge of deep–`match_id`'s 2660 and 2663, resulting in a flux loss of $\sim 6.02\,m$Jy. Combining the deep detections, and contrasting it with the shallow one, we have S/N's of 171 and 159 (given a shallow noise of 0.534, at `match_id`'s 2660), which corresponds to $\mathcal{C}_{\mathcal{DS}}$ S/N's bin 181 and 151, respectively (*cf.*, Figure 45 top). The flux loss may account for this discrepancy (a similar argument could also be made for Caesar), albeit this argument is somewhat contrived – *"ouroborossian"*.

PyBDSF does not fair well for extended sources with diffuse emission, it tends to glom everything together. In the deep image it gloms the B2294+C2294 system into one component (`match_id` 2660), whereas in the shallow image it is separates out into two components, B2263 (`match_id` 2662) and C2294 (`match_id` 2668). Clearly it is representing the system as a single island in the deep, as it is shrouded in diffuse emission, and resolving into two island in the shallow, as the diffuse emission between B2294 and C2294 is reduced. For our settings, PyBDSF uses the number of peaks found within the island as its initial guess for the number of Gaussians to fit (naïvely $\sim \mathcal{O}(6)$, *cf.* Figure 52), iteratively reducing them until a good fit is achieved (*cf.*, § C.4). Presumably, it is the diffuse emission that is affecting the fit quality. This may also explain the large failure modes in our simulated point source image, shown in Figures 18 through 20. In the latter Figure, PyBDSF fits a large region of injected sources obscured by background noise, not unlike our diffuse emission case.

Selavy correctly characterizes C2294 (`match_id` 2668) in the deep and shallow images, but does poorly with B2294. B2294 is characterized by a single component in the deep image, at `match_id` 2660, and by three com-

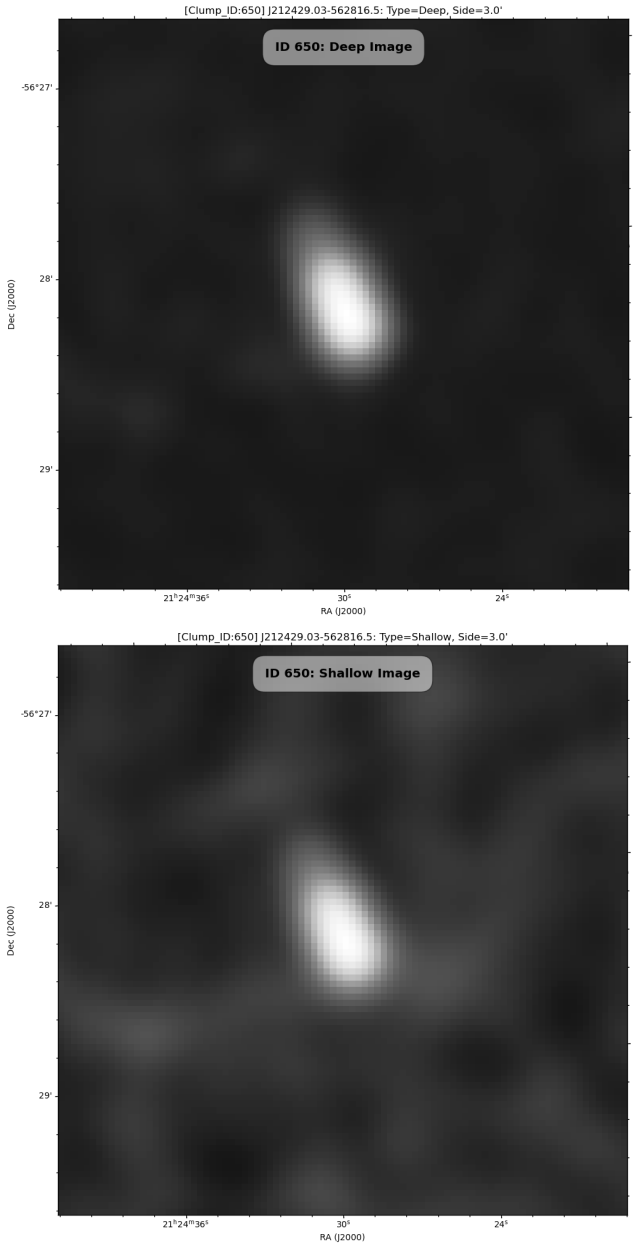


Figure 49. $0.615' \times 0.615'$ cutouts of `clump_id` 650 of the $2° \times 2°$ EMU deep (top) and shallow (bottom) sample images.

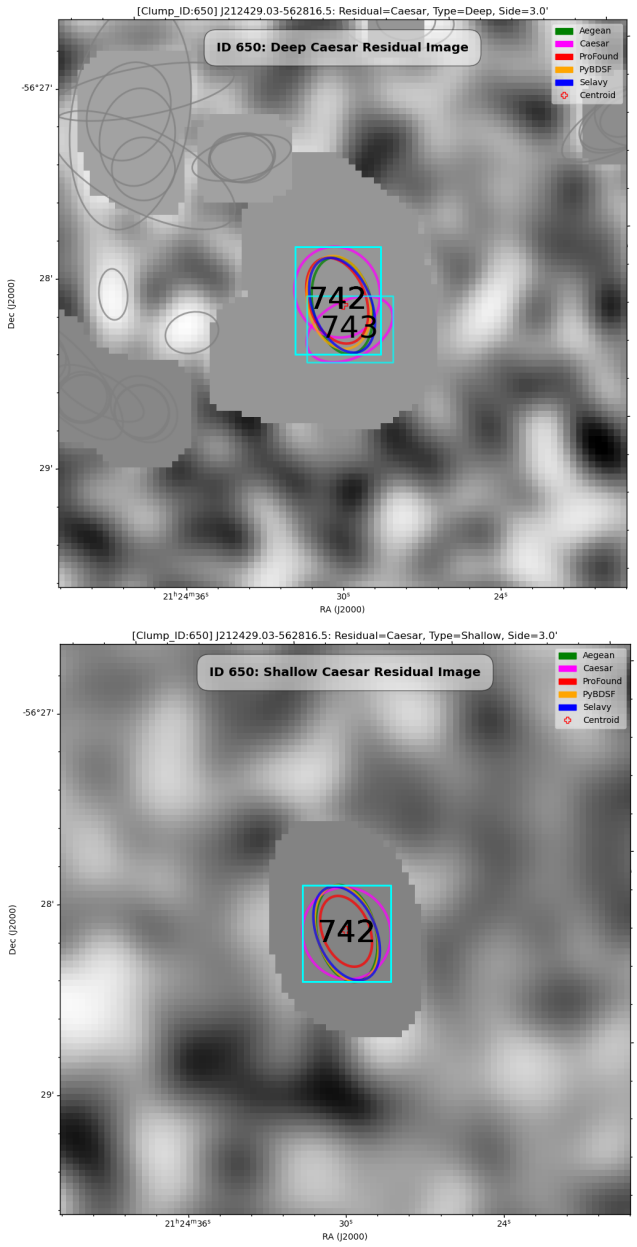


Figure 50. Caesar $0.615' \times 0.615'$ residual-image cutouts of `clump_id` 650 of the $2° \times 2°$ EMU deep (top) and shallow (bottom) sample images (*c.f.*, Figure 49); for colour codes, see Table 20. All source finders make deep and shallow detections at `match_id` 742. Caesar separately detects the bright peak at `match_id` 743 in the deep image, but with no corresponding shallow match. This contributes to a reduction in the inferred completeness for Caesar at S/N $\sim$ 12.

ponents in the shallow image, at `match_id`'s 2660, 2661, and 2667. It appropriately characterizes $(a)+(\chi)$ in the deep and (χ) in the shallow, at `match_id` 2660; as expected, given some of the diffuse emission is washed out in the latter. However, as for the remainder, it completely misses everything downstream of (χ) in the deep image, whereas the shallow image has two components, along $(\chi)-(\lambda)$ (`match_id` 2661) and $(b)+(c)$ (`match_id` 2667). Selavy uses methods similar to PyBDSF (*cf.*, § C.5), and so presumably the fit has failed in the deep image, but was slightly more successful in the shallow image (due to a reduction in diffuse emission). In fact, this is particularly troubling for Selavy when it comes to $\mathcal{R}_{\mathcal{DS}}$ (Figure 45 bottom): *e.g.*, the shallow detection at `match_id` 2661 has no deep counterpart (*i.e.*, is not considered real), resulting in a degradation in the S/N $\sim$ 125 bin. This has even been observed for a point source with a diffuse halo, in the S/N $\sim$ 65 bin.

In going from simulated to real images, we find the major distinction comes in when considering diffuse sources.

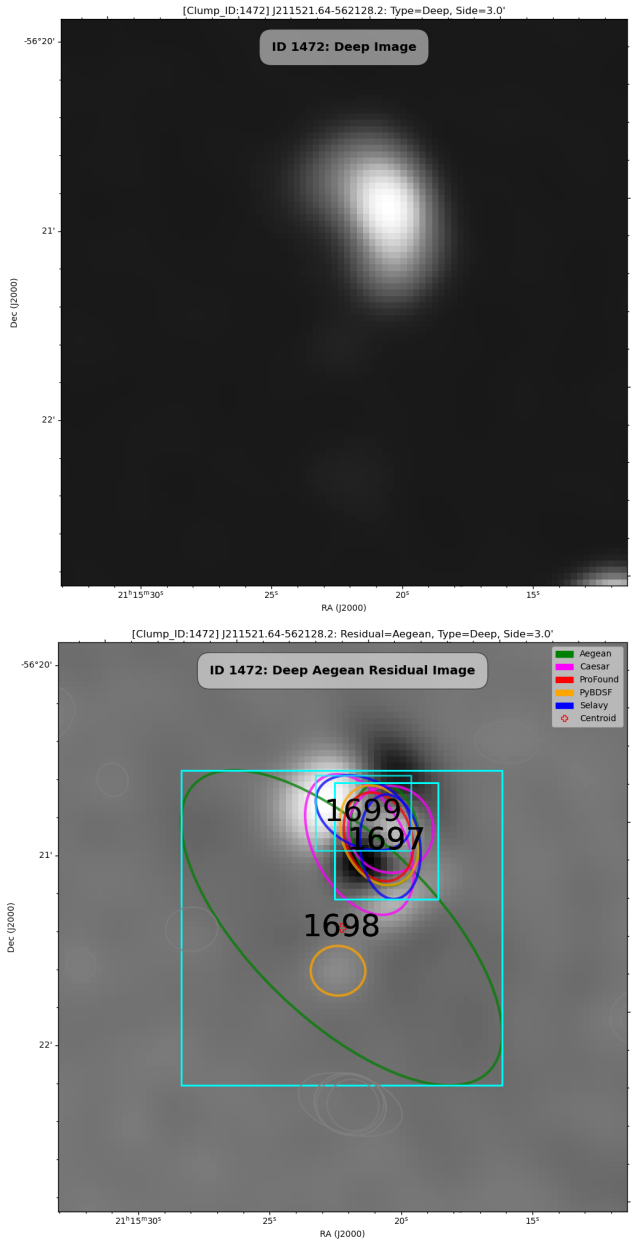


Figure 51. $1.69' \times 1.69'$ EMU image (top) and Aegean residual-image cutouts of `clump_id` 1472 of the $2° \times 2°$ EMU deep sample image; for colour codes, see Table 20. Here, Aegean overestimates the extent of the diffuse source (top middle) at `match_id` 1698 (bottom middle): *i.e.*, $(a, b, \theta) \sim (130'', 57.6'', 45.8°)$, with major, minor, and position-angle components, respectively. There is no corresponding shallow match. This leads to a degradation in the $\mathcal{C}_{\mathcal{DS}}$ S/N ~ 25 bin (Figure 45 top).

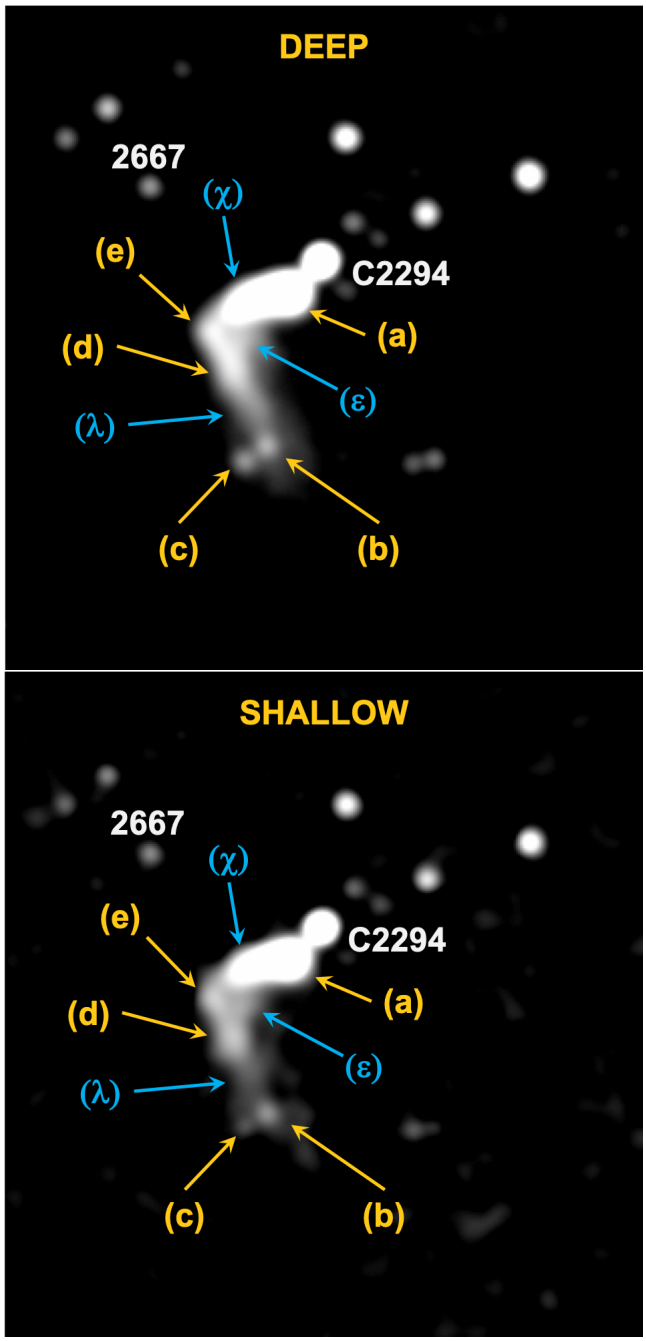


Figure 52. Deep (top) and shallow (top) image cutouts of `clump_id` 2293 (*re.*, Figures 53, 54, and Table 12) obtained from the $2° \times 2°$ EMU pilot sample cutout. The elements of the main clump (B2293) are labelled (a) through (e) for bright compact emission, and (χ), (ϵ), and (λ) for diffuse emission. Also shown is `match_id` 2667, which is part of this clump. While highly likely to be related to the emission from B2293, the clump labelled C2294 is separated in our analysis. Its compact nature leads the finders to characterise it with ellipses that do not overlap with any ellipse in B2293, resulting in these being labelled as independent clumps.

ProFound performs the best when it comes characterizing complex sources with diffusion emission: *i.e.*, it deblends them into flux mountains, but does not resolve them further – by design. Caesar performs similarly. However, it is hobbled from optimal performance, as its parent and child segment RMS and island parameters have the same settings. This is due to the current limitations of Hydra (*re.* footnote 27). In short, given the SCORPIO survey analysis by Riggi et al. (2016), it is expected to outperform ProFound in terms of its deblending power; however, it may not be able to scope

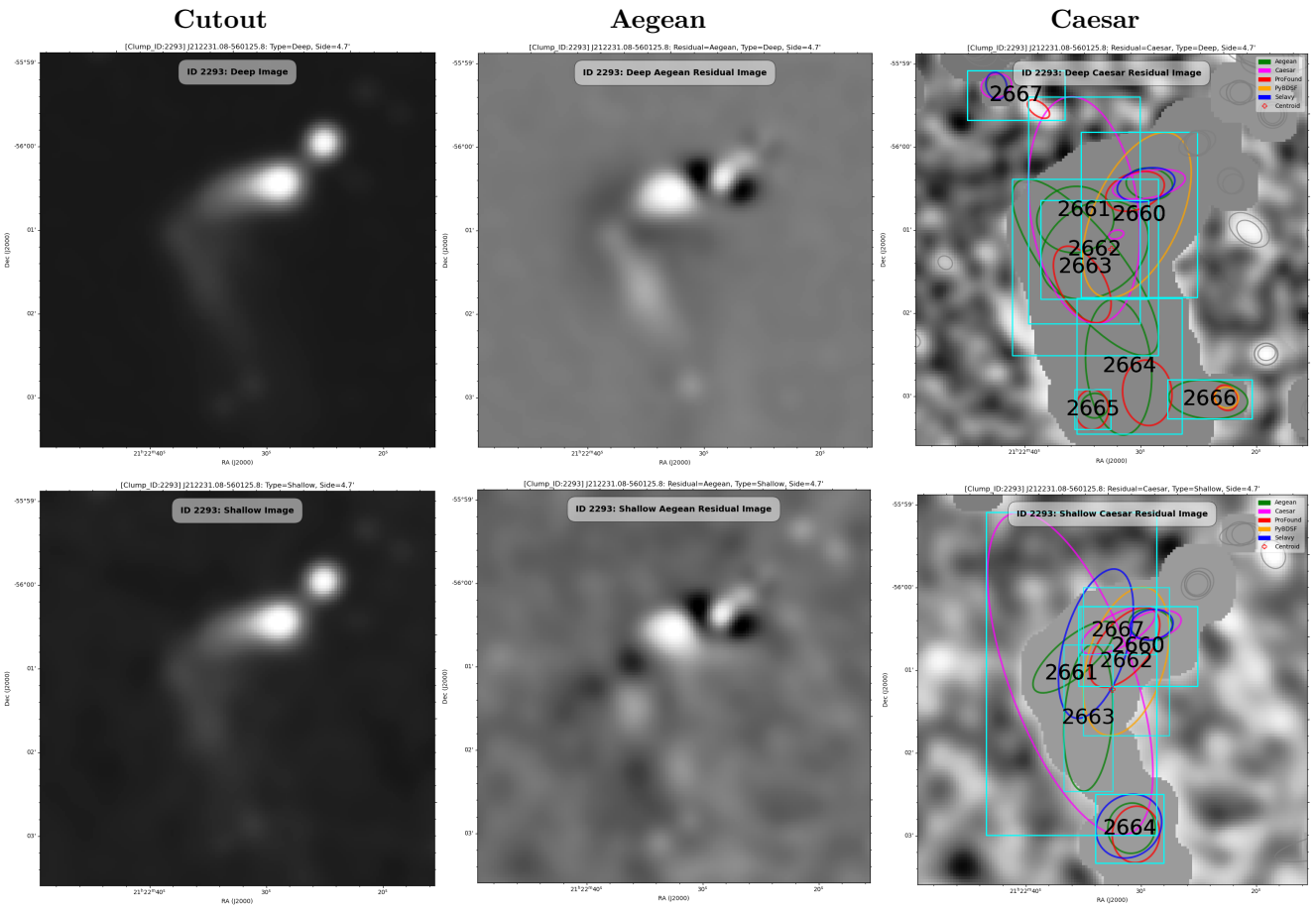


Figure 53. $4.73' \times 4.73'$ image cutout (left column), along with Aegean (middle column) and Caesar (right column) residual-image cutouts, of `clump_id` 2293 of the $2° \times 2°$ EMU deep (top row) and shallow (bottom row) sample images; for colour codes, see Table 20. Only the Caesar residual-images are annotated with `match_id` information (*re.* Table 12), as they are not obscured. See Figure 54 for continuation.

up as much diffuse emission (*cf.*, Caesar and ProFound footprints, in Figure 53 right column and Figure 54 left column, respectively). PyBDSF and Selavy have similar performance issues, especially when it comes to $\mathcal{R}_{\mathcal{DS}}$, and even more so for Selavy (*cf.*, Figure 45 bottom). They both tend to glom sources embedded in regions of diffuse emission into single components; especially in deep images, and slightly less so in their shallow counterparts, wherein some of the diffuse emission is washed away. Selavy performance is slightly worse, when comes to fitting failures of complex sources in deep images, which results in more degradation in $\mathcal{R}_{\mathcal{DS}}$. Aegean slightly differs from PyBDSF and Selavy, in that it uses a curvature map to pin its Gaussian fits to valleys of negative curvature, which is prone to being offset by strong outflows, resulting improper characterization of source position and extent. Although, its flux estimates tend to be somewhat reasonable, which is reflected in an improvement in $\mathcal{R}_{\mathcal{DS}}$ over the latter two source finders.

Figure 55 compares the major axes distribution of the source finders for our $2° \times 2°$ EMU pilot sample cutout. The distribution for ProFound is fairly symmetric with no outliers. Aegean, PyBDSF, and Selavy have roughly the same tail, which is skewed from ProFound with increasing value. Caesar has a large tail. This is consistent with the observed irregularities of the source finders in our B2293+C2294 bent-tail mini-lab, wherein there is a miss-characterization of the position and extent of its constituent elements; especially, along the outflow $(\chi) - (\beta)$ (Figure 52). Comparing this with the distributions for simulated extended-sources (Figure 41), we notice that the tail structure is roughly the same for all source finders, in the case of simulated sources, but diminished wrt the case for real source.[30] This is a reflection of the effects of diffuse emission. The distribution about the neighbourhood of the beam, is consistent between the real and simulated cases; wherein confusion settles in, as a dips below the beam size.

[30] The simulated point-source distribution, shown in Figure 28, indicates an additional component to the skewness of the simulated extended-source distribution. It does not really add much to the flavour of our discussion; except perhaps, that the skewness is due to the convergence of random noise fluctuations.

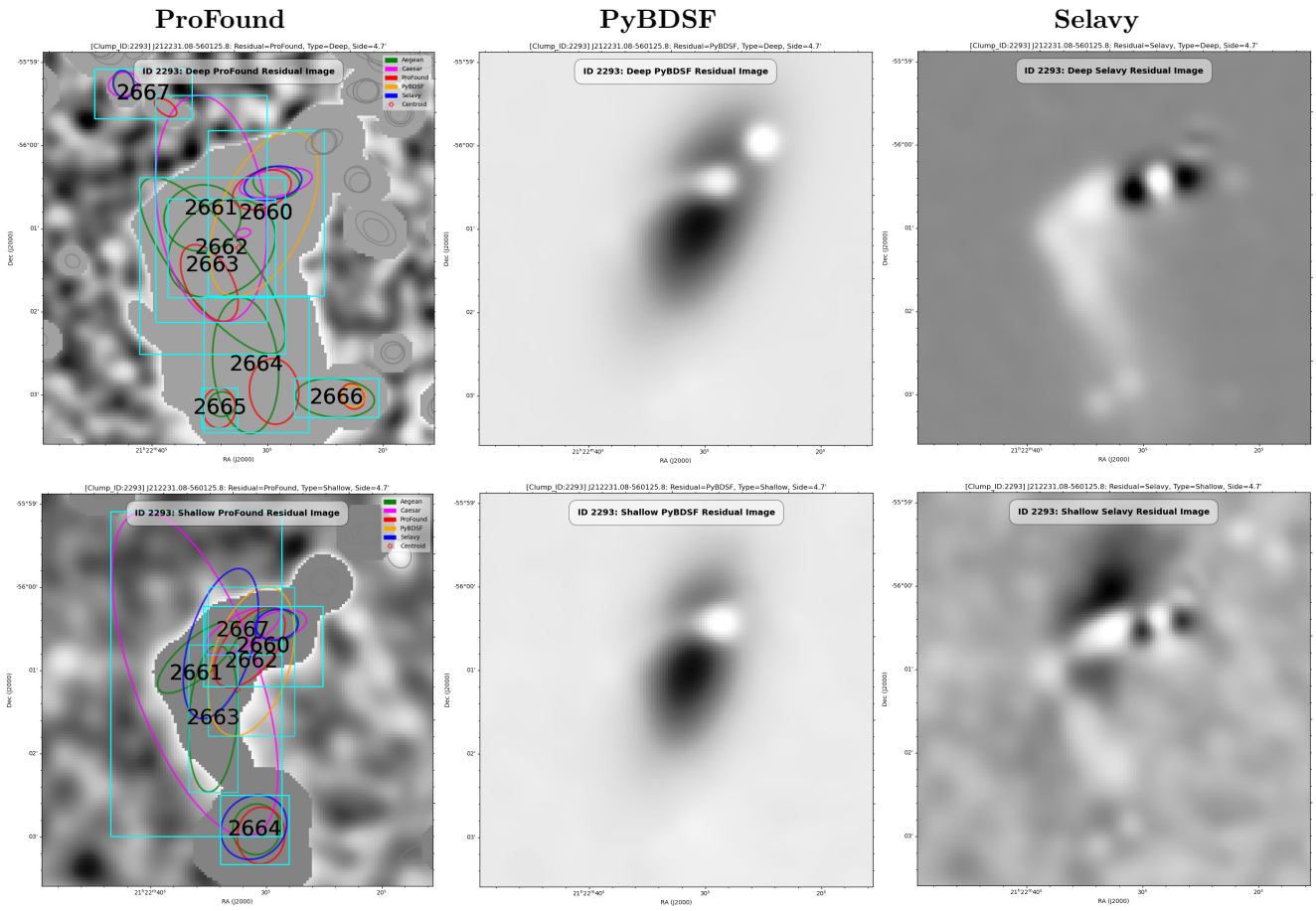


Figure 54. $4.73' \times 4.73'$ ProFound (left column), PyBDSF (middle column) and Selavy (right column) residual-image cutouts of `clump_id` 2293 of the $2° \times 2°$ EMU deep (top row) and shallow (bottom row) sample images; for colour codes, see Table 20. Only the ProFound residual-images are annotated with `match_id` information (*re.* Table 12), as they are not obscured. This figure is a continuation of Figure 53.

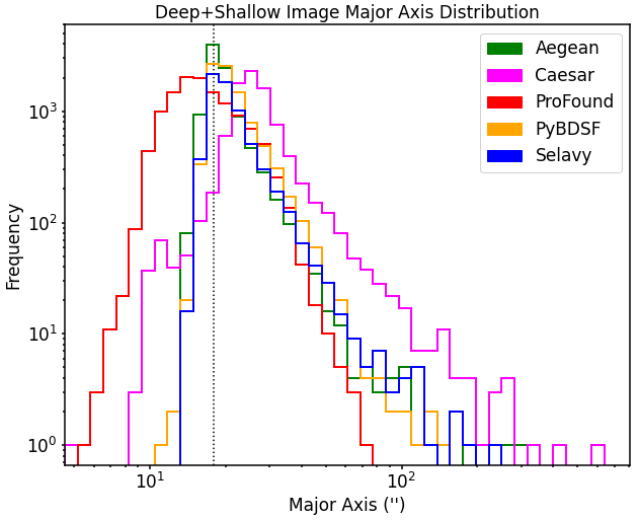


Figure 55. Major axis distributions for the $2° \times 2°$ EMU pilot sample cutout. The vertical dashed-line represents the beam size.

3.3.1 Performance statistics

Figure 56 shows S_{out}/S_{in} as a function of S_{in} (expressed as S/N) for each source finder. The horizontal (dotted) lines represent $S_{out} = S_{in}$, and the solid and dashed curves about these lines are the 1σ and 3σ deviations from S_{in}, respectively. The dot-dashed curves are the detection thresholds (*i.e.*, the RMS parameters in Table 11, corresponding to 90% PRD), and the (curved) dotted lines represent nominal 5σ thresholds (*c.f.*, Hopkins et al., 2015). Also shown are histograms giving the distribution of S/N in the false-positives for each source finder. The scatter is similar to that of the simulated extended-sources case (*cf.*, Figure 31). In the case of real sources, Selavy surpasses the 5σ barrier, down to the desired 3.6σ `snrCut` (in Table 11).

Table 13 shows sample values of 3σ scatter in the S/N $\sim 1 - 10$ regime (*cf.*, § 3.1.3). With the exception of ProFound, the scatter is pretty much the same across the board. Selavy has the highest false-positive distribution, which is also reflected in its $\mathcal{R}_{\mathcal{DS}}$ plot (Figure 45).

For the simulated images we were able to determine

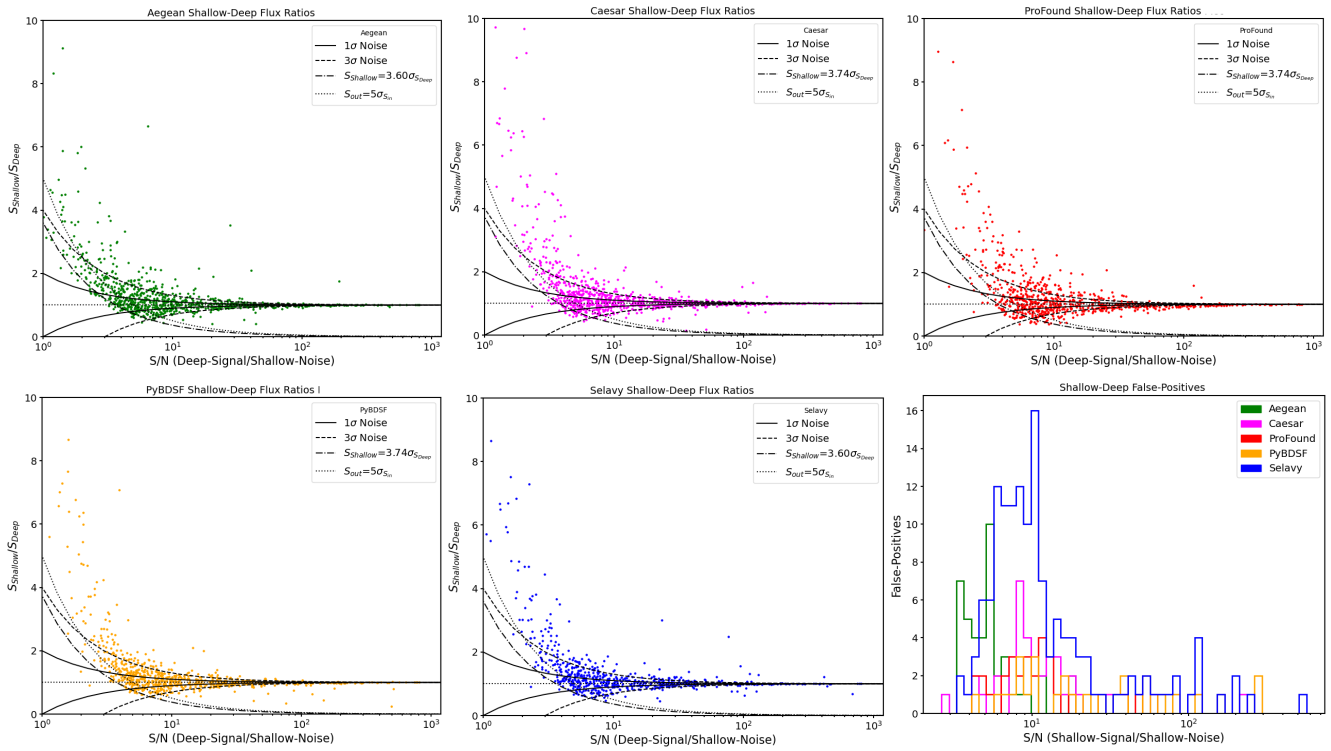


Figure 56. Flux-ratios and false-positive distribution as a function of injected S/N for the $2^\circ \times 2^\circ$ EMU pilot tile sample cutout; for colour codes, see Table 20. The 1σ (solid) and 3σ (dashed) curves are RMS noise (σ) deviations from the flux-ratio = 1 lines (dotted). Also shown are the detection threshold (dot-dashed curves; see Table 7) and nominal 5σ threshold (dotted curves).

Table 13 3σ scatter for flux-ratio plots in Figure 56.

Source	3σ Scatter		
Finder	$s_{3\sigma}(3)$	$s_{3\sigma}(5)$	$s_{3\sigma}(10)$
Aegean	3.40%	4.88%	5.59%
Caesar	4.62%	4.33%	6.76%
ProFound	6.60%	6.94%	10.72%
PyBDSF	3.43%	3.18%	5.88%
Selavy	3.94%	3.77%	5.99%

the "goodness" of source finder detections, through various metrics, as the sources are known. For real sources, this luxury is no longer afforded. Regardless, we have developed metrics that give us a good sense of the quality of $\mathcal{C}_{\mathcal{DS}}$, $\mathcal{R}_{\mathcal{DS}}$, flux-ratios, and false-positives (*via.*, $\delta\mathcal{C}_{\mathcal{DS}}$ and $\delta\mathcal{R}_{\mathcal{DS}}$; *cf.*, Figures 30 and 42). Other metrics can be developed through cross source-finder comparisons.

Figure 57 shows detection confidence charts, indicating coincident detections between source finders. The stacked-plot on the far left, shows coincidence frequency between all 5 source finders, followed by the adjacent stacked-plot showing coincidence frequency between 3 source finders, *etc.* This metric uses a "majority rules" process to determine if a detection is real. The more votes boosts the confidence. So if all detectors are totally undecided, chances are the detections are spurious, and the more detectors there are, the more likely this is the case. Conversely, if all detectors agree, chances are the detections are real, and the more detectors there are, the more likely this is the case.

For example, the stacked-plot on the far top-right (Figure 57) indicates that ProFound probably has the most spurious deep–detections, of all of the source finders. Indeed, a cursory search of the corresponding 3,373 image cutouts, shows this to be a case. A vast majority of the signals appear to be due to random noise convergence. However, some of them are ambiguous (*e.g.*, Figure 58).

The up side of this, is that for deep images, majority votes can be used to determine ground truths. So now metrics, such as, $\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$, can be plotted based on vote counts, providing an entire suite of tools for getting a sense of the "goodness" of the data. Indeed, a cursory look at the deep-shallow confidence charts (Figure 57) indicates that $\mathcal{R}_{\mathcal{DS}}^{ProFound}$ is biased, as a vast majority deep detections by ProFound are spurious. However, the small population in the corresponding shallow chart, indicates that most of these anomalies are washed out (as confirmed by a cursory search; *e.g.*, there is no corresponding shallow detection of the anomaly shown in Figure 58).

Other diagnostic tools are already available in the Hydra Viewer, such as, clump and S/N distributions.

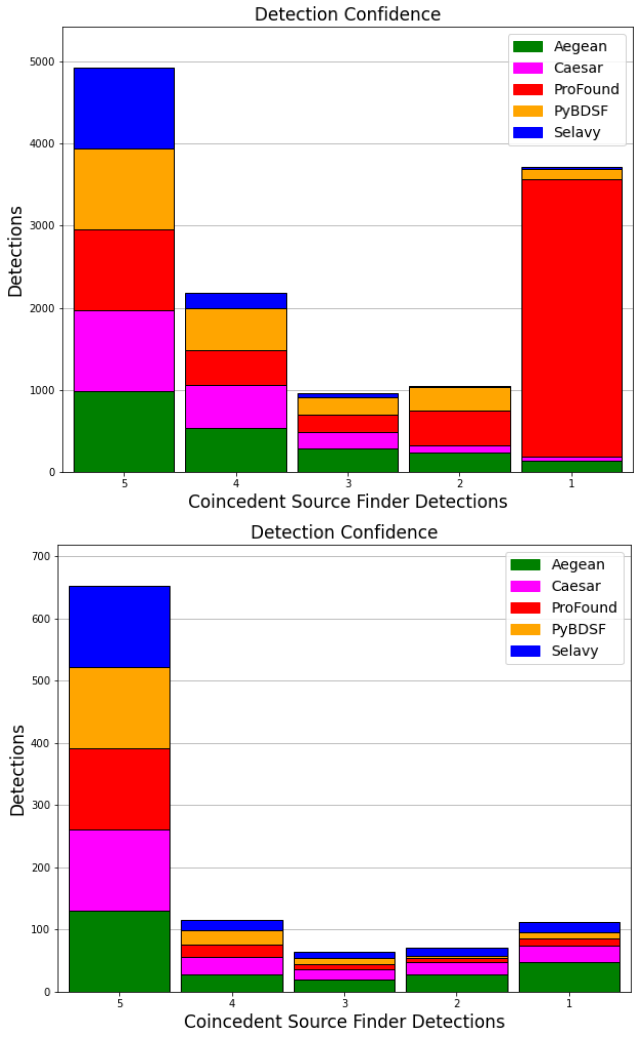


Figure 57. Deep (top) and shallow (bottom) detection confidence charts. The stacked plots indicate agreements between source finder on detections. So from left to right, 5 source finders agree, 4 source finders agree, *etc.*

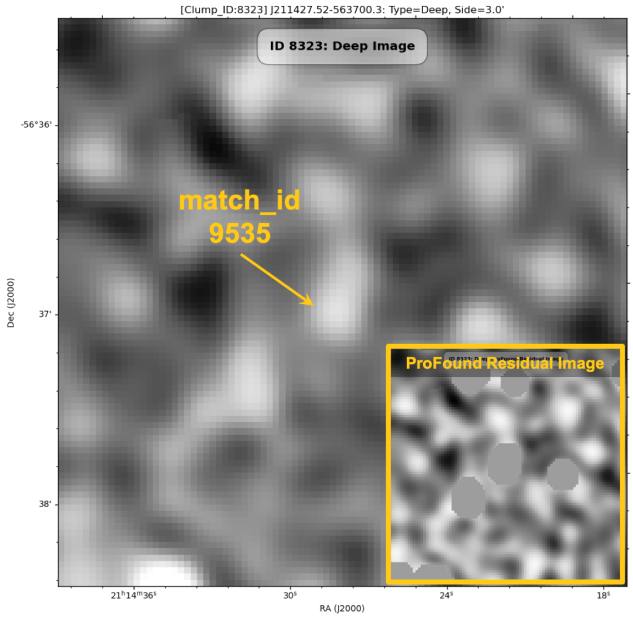


Figure 58. Example of an ambiguous detection (`match_id` 9535) by ProFound in a $0.256' \times 0.256'$ EMU image cutout at `clump_id` 8323 (residual image inset). At first glance, it appears to be part a vary faint ring structure; something for which ProFound is uniquely suited. However, upon further investigation, the size of the anomaly is on the order of EMU's beam size ($18''$). So clearly it is a spurious detection. This example was found by exploring the discordant bin of the deep detection confidence chart (Figure 57 top-right stacked plot). Regardless, one should be careful (*cf.*, Figure 59).

The aforementioned concepts could be integrated into the viewer, making it a robust exploratory software suite. This would allow users to check data quality, categorize sources, search for interesting objects, *etc.*

3.3.2 Processing Residual Images

Given the information at hand, we now revisit the question (*re.* § 3.2.2) of processing residual images: *i.e.*, to try to recover as much flux as possible, while correctly characterizing there source in terms of position and extent. Table 14 shows normalized MADFM statistics for the $2^\circ \times 2^\circ$ EMU pilot sample and $4.73' \times 4.73'$ `clump_id` 2293 cutouts. This suggest 3 approaches:

1. process all $2^\circ \times 2^\circ$ residual images,
2. process the $2^\circ \times 2^\circ$ residual image with the best MADFM,
3. process an aggregate clump-based residual-image

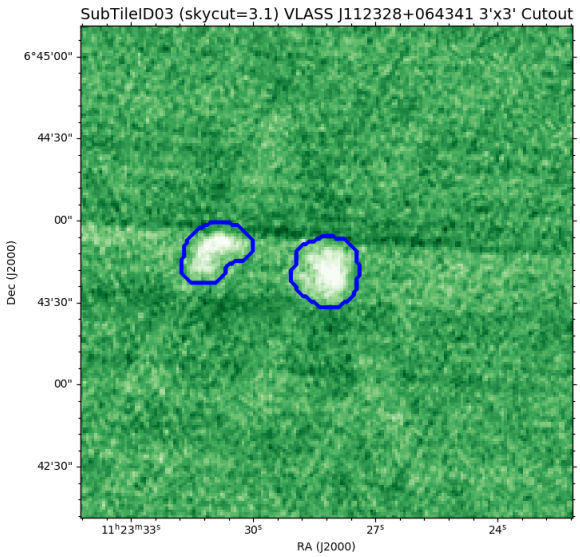


Figure 59. Example of a diffuse VLASS source, uniquely detected by ProFound in a comparison study with PyBDSF (Boyce, 2020). The source was found in a $3' \times 3'$ region centered at J112328+064341 (see Figure 65 for resource information).

consisting of the best MADFM's on a clump-by-clump bases,

where we focus our attention on the deep residual-images. Approach 1 can be immediately ruled out, as it is combinatorically impractical (*i.e.*, at total of 25 PRD optimization pathways would be required).

Table 14 Residual-image statistics for $2^\circ \times 2^\circ$ EMU pilot sample and $4.73' \times 4.73'$ `clump_id` 2293 cutouts (extracted from Tables 11 and 12, respectively). The MADFM's are normalized by cutout area, and are in units of mJy/(arcmins^{2}beam). N is the component count.

Residual Image		EMU Sample		Cutout 2293	
Source	Image	$2^\circ \times 2^\circ$		$4.73' \times 4.73'$	
Finder	Depth	N	MADFM	N	MADFM
Aegean	Deep	8,538	2.60E-05	8	2.45E-03
Caesar	Deep	7,838	2.30E-05	4	5.56E-04
ProFound	Deep	11,484	1.90E-05	6	6.10E-05
PyBDSF	Deep	8,292	2.60E-05	3	1.23E-02
Selavy	Deep	5,800	2.70E-05	2	2.68E-03
Aegean	Shallow	926	1.69E-04	4	1.03E-02
Caesar	Shallow	885	1.66E-04	3	5.07E-03
ProFound	Shallow	778	1.65E-04	2	5.67E-03
PyBDSF	Shallow	794	1.69E-04	1	5.67E-03
Selavy	Shallow	789	1.69E-04	3	1.04E-02

Examining the MADFM's in Tables 4, 7, and 14, it is evident that ProFound will practically win out every time, followed closely by Caesar. This is inherent in their non-Gaussian nature. In order to place them on equal footing, one would have to consider using residual images created by their Gaussian components. The analysis of the bent-tail system, B2293+C2294 (Figure 52), indicates that ProFound would be on a competitive footing with the other source finders. It would more than likely excel for complex extended sources with diffuse emission. Caesar, on the other hand, would not fair as well, given Hydra's 2-parameter optimization space.[27] For the moment, let us forego using Caesar and ProFound, given the state of Hydra's current infrastructure.

Using approach 2, we see that Aegean and PyBDSF tie (*cf.*, Table 11). Rounding errors aside, one could randomly chose one of the winners, or the one with the lowest RMS metric (*cf.*, Table 14), *etc.* The residual images will include any artifacts that come with the chosen source finder. However, one could use the confidence detection chart (Figure 57) to mitigate this risk.

Approach 3 is perhaps more unconventional, as it is an aggregate of results from the different source finders. However, it is likely to contain fewer artifacts. Figure 60 shows the distribution of winning clump residual-image cutouts (*i.e.*, the source finders with the lowest MADFM on a clump-by-clump bases) for Aegean, PyBDSF, and Selavy. For low MADFM, the aggregate residual-image is roughly a homogeneous mixture of results from the different source finders, with the larger of the contributions from Selavy. For increasing MADFM, contributions from Selavy decease, followed by PyBDSF, and lastly Aegean. For our bent-tail system, the aggregate would include 8 `clump_id` 2293 (which contains B2293) components from Aegean (*cf.*, Table 14) and 1 `clump_id` 2294 (which only contains C2294) component from Selavy (*cf.*, Table 12 bottom partition). The combined effect is not 100% clear, in general, and so further investigation would be required.

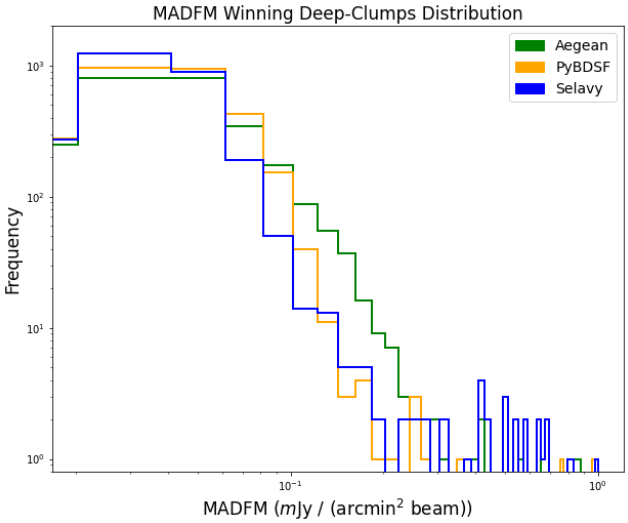


Figure 60. Distribution of clumps with the lowest MADFM wrt Aegean, PyBDSF, and Selavy.

A simpler approach would be to identify isolated components, and subtract them out to produce a residual image, exposing more complex structure. (This is pretty much how Caesar works, by reprocessing residual image to characterize underlying complexity.) Further to that, one could restrict the size to around the beam size (*i.e.*, compact sources), so as to reduce the introduction of artifacts (*cf.*, Figure 37 right column). One could do this iteratively; successively reprocessing residual images, terminating subject to some criterion. Either approach 2 or 3 could be considered in this regard.

The results from either of the latter two approaches would then have to be folded in with the existing cluster catalogue, and appropriately tagged. A considerable amount of analysis would be required before this method could be justified. Regardless, this was one of the initial driving forces behind Hydra.

3.4 Source Finder Speed Testing

The Hydra tool was also used to analyse the processing speeds of the different source finders. The details of the analysis are in Appendix E. The bottom row of Figure 73, in the appendix, shows PRD CPU times

for simulated point-sources (left), simulated extended-sources (middle), and the EMU sample cutout (right). The statistics are for the deep images only, as their shallow cousins tell a similar story. The CPU times are recorded for their RMS and island parameters down to the 90% PRD cutoff (top row). Table 15 shows the processing speeds.

Table 15 Source finder processing times for simulated point-sources, simulated extended-sources, and real sources (*i.e.*, EMU sample cutout). These numbers are rough orders of magnitude, for deep images only (*cf.*, Figure 73).

Source Finder	CPU Time (s)		
	Point	Extended	Real
Aegean	3×10^2	6×10^2	2×10^2
Caesar	1×10^3	4×10^3	5×10^2
ProFound	9×10^2	9×10^2	4×10^2
PyBDSF	9×10^2	9×10^2	4×10^2
Selavy	1×10^3	4×10^3	3×10^3

The NxGen source finders are orders of magnitude slower than ProFound and PyBDSF. Granted they are restricted to single-process mode (*re.* § E), however it is reasonable to expect comparable processing times. This suggests there is room for improvement, upon which they should run considerably faster in multi-process mode.[31]

3.5 Discussion

UNDER CONSTRUCTION

3.5.1 Typhon Statistics

Hydra has been used to baseline Aegean, Caesar, ProFound, PyBDSF, and Selavy in terms of their RMS and island parameters, at a PRD 90% cutoff. The Aegean and PyBDSF source finder `box_size` and `step_size` parameters were primed using BANE, with Selavy being primed by the former. The source finders were baselined for deep and shallow images, separately.

Tables 3, 6, and 10 show the `box_size` and `step_size` parameters for the simulated points-sources, simulated extended-sources, and the EMU sample cutout, respectively. Table 16 summarizes the results, with the box parameters converted from pixels to arc-seconds.

For the simulated deep point-sources, μ ($\sim 22\mu$Jy) is in accord with the design RMS floor noise of $20\,\mu$Jy (§ 3.1.1), indicating appropriate RMS box parameters settings (Table 3). For the simulated deep extended-sources, μ is high by design, so as to compromise noise estimates between compact and extended sources (*cf.*, § 2.1.3). The RMS boxes extend roughly 90% and 80% beyond the maximum size of the simulated point and extended sources, respectively; as ascertained, upon contrasting

[31] Software architecture issues aside.

Table 16 Hydra optimized μ, `box_size`, and `step_size` run-parameters for the $2^\circ \times 2^\circ$ deep/shallow simulated point-source (top partition), simulated extended-source (middle partition), and EMU sample (bottom partition) images, extracted from Tables 3, 6, and 10, respectively. The box and step size parameters were used as inputs for Aegean, PyBDSF, and Selavy.[a]

Image	μ (μJy)	`box_size` ($''$)	`step_size` ($''$)
Deep	21.81	240	120
Shallow	108.2	180	88
Deep	68.01	480	240
Shallow	325.3	240	120
Deep	35.28	720	270
Shallow	171.23	216	80

[a] The Selavy module only accepts `box_size`.

Table 16 with the major axis distributions, shown in Figures 28 (point-sources) and 41 (extended-sources). Finally, we note for the simulated images, the ratio of the box to step sizes is 1/2, for both deep and shallow.

Examining Table 16 further, we notice that, $\mu_{\mathcal{D}}/\mu_{\mathcal{S}} \sim \mathcal{O}(0.5)$ for both the simulated and EMU sample images. This indicates a consistency in our RMS box parameter calculations. Also $\mu_{\rm EMU} \sim 35\,\mu$Jy, lies between the simulated point and extended sources, which is a reflection of the population cross-section of radio sources. The major axis distribution, in Figures 55, suggests a source cutoff of $\mathcal{O}(120)''$, intuiting (*re.* Table 16) an RMS box extending roughly 80% beyond the maximum source size. In short, the RMS box optimization between simulated and real images is consistent, given the metric, μ. Finally, we note for the real deep and shallow images, the ratio of the box to step size is 1/3 (*cf.*, discussion *re.* Equation 4 in § 2.1.3).

Table 17 contains optimized RMS and island source-finder parameters for deep/shallow simulated point-source, simulated extended-source, and EMU sample images, extracted from Tables 4, 7, and 11, respectively. Figure 15, in Appendix E, shows the Typhon PRD response curves (top row) for the deep images (left, middle, and right, respectively), wherein the parameters are extracted just before the 90% PRD threshold; recall, the optimization process runs backwards from RMS_{max} (Equation 3). The shallow response curves (not shown) are similar, but approach the 90% PRD threshold more rapidly.

For a given source finder, and for a given image depth, the RMS and island parameters roughly equate between the simulated point and extended source images (*re.* the upper left two panels in Table 17). This is also reflected the PRD response curves (Figure 15).

Figure 61 shows a summary of the deep and shallow detections (*i.e.*, N in Table 17) for the different images.

Table 17 Deep/shallow Typhon optimized RMS and island source-finder parameters for the $2^\circ \times 2^\circ$ simulated point-source, simulated extended-source, and EMU sample images, extracted from Tables 4, 7, and 11, respectively, where N is the number of detections.

Source	Image	Point Sources			Extended Sources			EMU Sample		
Finder	Depth	RMS	Island	N	RMS	Island	N	RMS	Island	N
Aegean	Deep	3.074	3.070	6,016	3.074	3.070	4,112	2.676	2.674	8,538
Caesar	Deep	3.074	3.000	4,084	3.206	3.000	3,618	2.809	2.806	7,838
ProFound	Deep	2.412	2.409	4,997	2.412	2.409	3,730	2.279	2.277	11,484
PyBDSF	Deep	2.809	2.806	5,991	2.809	2.806	4,688	2.941	2.938	8,292
Selavy	Deep	3.206	3.203	3,225	3.206	3.203	2,544	2.941	2.938	5,880
Aegean	Shallow	3.868	3.864	747	3.603	3.599	1,287	3.603	3.599	926
Caesar	Shallow	4.000	3.000	657	3.603	3.000	1,297	3.735	3.000	885
ProFound	Shallow	3.074	3.070	642	2.941	2.938	1,138	3.735	3.732	778
PyBDSF	Shallow	3.735	3.732	598	3.338	3.335	1,312	3.735	3.732	794
Selavy	Shallow	4.000	3.996	427	3.735	3.732	787	3.603	3.599	789

The relative proportions of N, between source finders, remains roughly the same between point-source and extend-source simulated images. However, in going to the real (EMU) image, the relative proportion for ProFound increases significantly. This is most likely due to an increase in spurious detections, as surmised from the detection confidence charts (Figure 57). The most likely reason for this jump, is a lack of fidelity in the noise distribution for our simulated images (*cf.*, Figures 26 and 58).

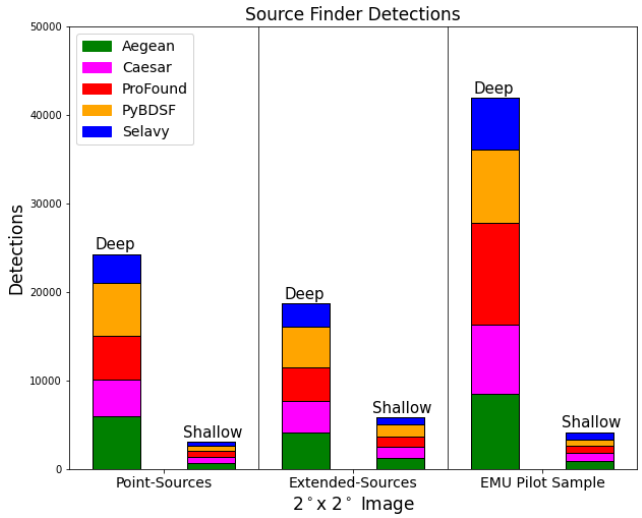


Figure 61. Stacked plot summary of deep and shallow source finder detections (N) given in Table 17.

Selavy on the other hand, has relatively low component counts for all images. The flux-ratio plots for the simulated images, Figures 31 (point-sources) and 43 (extended-sources), show RMS thresholds of 5σ, which are above the target thresholds of 4.0σ and 3.7σ, respectively (Table 17).[32] As a result, there is also a late rise in all of the completeness plots, Figures 11 (point-sources, top row) and 33 (extend-sources, top-row), for the simulated images. However, for the EMU sample image case, the flux-ratio meets the target threshold, Figure 56 (*cf.*, Table 17). Also the rise in completeness, Figure 45 (top), is in cadence with the other source finders. This change is reflected as an increase in the relative component count of Selavy, in the deep stacked plot, in the right-hand panel of Figure 61.

3.5.2 Cutout Case Studies

Table 18 summarizes several case studies, that were carried out in the analysis of the simulated and real images. They cover the most frequently encountered anomalies, while visually exploring the cutouts using the Hydra Viewer. They have been sorted into rough categories: *i.e.*, edge detection, de/blending, faint sources, component size errors, bright sources, and diffuse emission. With the exception of diffuse emission, all of the artefacts can be found in all of the images, to varying degrees.

Detecting sources at the edge of images is problematic, as it is difficult to estimate the noise and sometimes the source is even cutoff. Regardless, it is part of the sample population of artefacts that should be accounted for. For sources cutoff near the edge, Aegean tends to extrapolate their shape, while ProFound estimates the remaining flux, and PyBDSF treats them as complete sources, Figure 12. Their response is significantly improved when sources are just touching the edge, Figure 13. As compared to the remaining source finders, a cursory scan shows Caesar does edge detection at a slightly less frequency, whereas Selavy is no where to be found. As

[32] Figures 31 and 43 are for the deep image case. However, this is also true for the shallow case, and the case where the deep image is assumed to be the ground truth. These figures are not shown, as they do not add any additional insight to the analysis.

Table 18 Case studies for deep ($\mathcal{D}$) and/or shallow ($\mathcal{S}$) simulated point-source (Pts), simulated extended-source (Ext), and EMU image (Img) source detections by Aegean (Ag), Caesar (Ca), ProFound (Pr), PyBDSF (Pd), and Selavy (Se). The "Fig" column provides the figure number, while the "Depth" column indicates the image types shown.

			Depth		Source Finder					
Case Study	Fig	Img	$\mathcal{D}$	$\mathcal{S}$	Ag	Ca	Pr	Pd	Se	Notes
Edge Detection	12	Pts	Y	N	Y*	N	Y	Y	N	*Extrapolated off edge.
	13	Pts	Y	N	Y	N	Y*	Y	N	*Good match.
	35	Ext	Y	N	N	N	Y*	Y*	N	*3 false around injected.
	47	EMU	Y	Y	Y	Y	Y*	Y	Y	*Only $\mathcal{C}_{\mathcal{DS}}$ contribution.
Blended Sources	14	Pts	Y	N	Y	Y	Y	Y	Y	3 Sources Blended.
	34	Ext	Y	N	N	Y*	Y*	N	N	*Blends 2 injected sources.
	36	Ext	Y	N	Y	Y	Y	Y	Y	Large masking source.
Deblending Issues	16	Pts	Y	N	N	N	Y*	N	N	*Close source missed.
	21	Pts	Y	Y	Y	Y*	Y	Y	Y	*Shallow source missed.
	22	Pts	Y	Y	Y	Y	Y	Y	Y*	*Detects 1 of 3.
	23	Pts	Y	N	Y	Y	Y	Y*	Y	*Centriod biased.
Faint Detection	14	Pts	Y	N	Y	Y	N	Y	Y	*Re.* `match_id` 6201.
Noise Spike Detection	17	Pts	Y	N	Y	Y	Y	Y	Y	All false detections.
	18	Pts	Y	N	N	N	N	Y*	N	*False detection.
	26	Pts	N	Y	Y*	N	N	N	N	*Only in shallow.
	27	Pts	N	Y	Y*	Y*	N	N	N	*Near an injected source.
	48	EMU	Y	N	Y*	N	N	Y*	N	*Nearby source, $8''$ away.
Undersized Components	15	Pts	Y	N	N	Y*	N	N	N	*Flux underestimate.
Oversized Components	19	Pts	N	Y	N	N	N	Y*	N	*Noise spikes: $a \sim 160''$.
	20	Pts	Y	N	N	N	N	Y*	N	*Injected /w noise: $a \sim 6'$.
	24	Pts	Y	N	N	Y*	N	N	N	*Very eccentric: $a \sim 3.6'$.
	25	Pts	Y	N	N	Y*	Y*	N	N	*Flux overestimated.
	40	Ext	Y	N	N	Y*	N	N	N	*Fit failed: $a \sim 0.5°$.
Bright deep source missed	38	Ext	N	Y	N	N	N	N	Y*	*Only shallow detected.
	39	Ext	Y	N	N	Y*	N	N	N	*Detected by others.
Good Detection	46	EMU	Y	Y	Y	Y	Y	Y	Y	Bright source..
Blended with Diffuse Emission	50	EMU	Y	Y	Y	Y*	Y	Y	Y	*Deblends deep.
	51	EMU	Y	N	Y*	Y	Y	Y*	Y†	*Bad fit. †Deblends deep.
Diffuse Emission Case Study	52	EMU	Y	Y	Y	Y	Y	Y	Y	See § 3.3.

ProFound is capable of characterizing irregularly shaped objects, in real images, it tends to contribute to $\mathcal{C}_{\mathcal{DS}}$ for cutoff sources, Figure 47. In short, it accounts for the flux in the deep and shallow image, wherein the deep image is assumed to be the ground truth.

For the simulated images, there are example cases of unresolved compact sources: *e.g.*, Figure 14. These are detected as a single source, wherein the total flux is accounted for. Blending is also encountered for overlaid point and extended sources, with the former having slightly lower S/N, Figure 34. For simple cases of real sources with diffuse emission, such as a compact object with a diffuse tail (Figure 49), Aegean, ProFound, PyBDSF, and Selavy tend to characterize them with a single flux-weight component, whereas Caesar tends to resolve them in deep images, but not shallow images where the emission is diminished Figure 50.

Deblending issues can occur in systems with relatively low S/N neighbouring sources of a more bright object, causing a weighted offset from the bright object, as shown in the example for PyBDSF in Figure 23. This causes a degradation in $\mathcal{R}_{\mathcal{D}}$, and likely $\mathcal{R}_{\mathcal{DS}}$ in the case of real images (*i.e.*, due to a $\mathcal{DS}$ S/N mismatch, as is the case in this example). In such cases, PyBDSF tends to bias its position estimates towards flux-weighted centers. The other source finders, tend not to be biased in such a fashion (*cf.*, Figure 22). However, unlike the other source finders, ProFound tends to weight its component footprint to include low flux sources, Figure 16. In the case of a compact system with two roughly opposing diffuse tails (*i.e.*, Figure 51 top), say (*i.e.*, slightly more complicated than our previous example), Aegean, ProFound, and PyBDSF correctly characterize its core, whereas ProFound appropriately deblends its core and

diffuse emission separately (*i.e.*, its apparent core with diffuse halo), and Selavy alternatively deblends it into two components to account for its slight bend (*i.e.*, its apparent jet structure; *cf.*, Figure 51 bottom).

One of Caesar's most commonly encountered artefacts are highly eccentric and (sometimes excessively) large components with high flux: *e.g.*, Figures 24 and 40. This seems related to deblending issues with low S/N sources in local proximity (*e.g.*, compare the footprints of Caesar and ProFound in Figure 24). Similarly, one of PyBDSF's most commonly encountered artefacts are slightly eccentric and (sometimes excessively) large components with low flux: *e.g.*, Figures 19 and 20. In this case, however, it seems related to low smooth-rolling hills of flux and, especially so, when the region is peppered with low S/N sources (*cf.*, Figure 20): *i.e.*, it tends to glom proximally located noise spikes (*cf.*, Figures 18 and 19).

All of the source finders detect noise spikes that humans equally mistaken for true sources; as shown in Figure 17, for a simulated image. However, in real images, such noise spikes tend to be diminished to a point where humans can tell they are not real sources, but source finders can sometimes trip up, Figure 48. (There are also those grey areas, *e.g.* Figure 58.) This brings into question the fidelity of the noise distribution in our simulations. Regardless, in either case, none of the signals are detected in the shallow image, so they do not contribute to $\mathcal{C}_{\mathcal{DS}}$ or $\mathcal{R}_{\mathcal{DS}}$.

There are some oddities, that occur with less frequency, such as, Caesar underestimating flux (Figure 15) or Selavy detecting bright sources in the shallow (Figure 38), but not deep (Figure 37), images, whereas all the other source finders succeed. In the former case, it is a deblending issue with add mixtures of low and high S/N sources in close proximity. However, for the latter case it is not clear.

The real image contains sources with diffuse emission, which is not reflected in our simulations. So the analysis of the EMU sample image (Figure 44) focused on this aspect. De/blending use cases were examined for simple one and two tailed compact-source systems (*i.e.*, Figures 50 and 51, respectively). It was observed that the diffuse emission was washed out in the shallow image, causing a degradation in $\mathcal{C}_{\mathcal{DS}}$, but not $\mathcal{R}_{\mathcal{DS}}$. However, we note here, that the core peak flux is unaffected (*i.e.*, $S^{\mathcal{D},\mathcal{S}}_{peak} \sim 4.2$ and 13 mJy, respectively), except for a slight decease in the deep image for Selavy (*i.e.*, $S^{\mathcal{D}}_{peak} \sim 3.2$ and 10 mJy,[33] respectively).[34] This was followed by a detailed analysis of a bent-tailed system; shown in Figure 52, along with a complete set of Hydra Viewer cutouts for the main clump, shown in Figures 53 (part a) and 54 (part b).

. . . currently under construction . . .

4 SUMMARY AND CONCLUSIONS

PENDING...

5 ACKNOWLEDGEMENTS

M. Boyce, S. Baum, Y. Gordon, D. Leahy, C. O'Dea, M. Ramsay, S. Safi-Harb, and A. Vantyghem acknowledge partial support from the Natural Sciences and Engineering Research Council (NSERC) of Canada.

M. Michałowski acknowledges the support of the National Science Centre, Poland through the SONATA BIS grant 2018/30/E/ST9/00208.

The Canadian Initiative for Radio Astronomy Data Analysis (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 Australian SKA Pathfinder is part of the Australia Telescope National Facility (ANTF) which is managed by the Commonwealth Scientific and Industrial Research Organisation (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.

A NOTATION

Table 19 contains a list of symbols used herein for convenience. Table 25 (§ D.2) contains a summary of definitions related to the completeness and reliability metrics defined in Table 19.

Table 20 provides the colour coding corresponding to the source finders used throughout the various figures in this paper. This table is provided for clarity, even though the figures also have legends.

[33] *i.e.*, averaged from two tail-core deblending, with $S^{\mathcal{D}}_{peak} \sim 9.5$ and 11.4 mJy for each tail-core, but with $S^{\mathcal{S}}_{peak} \sim 11.4\,m$Jy for both cores.

[34] NB: Peak fluxes for ProFound are not provided in the current version of Hydra.

Table 19 Table of symbols.

Symbol	Meaning
PRD	Percentage Real Detections
$\Rightarrow$	Maps-into
$\lceil x \rceil$	Round x up
I	Pixel Flux Density per Beam
μ	Mean
m	Median
σ	RMS
MADFM	Median Absolute Deviation From the Median
$\mathcal{D}$	Deep Image
$\mathcal{S}$	Shallow Image
$\mathcal{DS}$	Deep-Shallow Images
$\mathcal{C}$	Completeness
$\mathcal{C}_{\mathcal{D}}$	$\mathcal{D}$ Completeness
$\mathcal{C}_{\mathcal{S}}$	$\mathcal{S}$ Completeness
$\mathcal{C}_{\mathcal{DS}}$	$\mathcal{DS}$ Completeness
$\tilde{\mathcal{C}}_{\mathcal{DS}}$	$\mathcal{DS}$ Goodness of Completeness
$\delta\mathcal{C}_{\mathcal{DS}}$	$\mathcal{DS}$ Residual Completeness
$\mathcal{R}$	Reliability
$\mathcal{R}_{\mathcal{D}}$	$\mathcal{D}$ Reliability
$\mathcal{R}_{\mathcal{S}}$	$\mathcal{S}$ Reliability
$\mathcal{R}_{\mathcal{DS}}$	$\mathcal{DS}$ Reliability
$\tilde{\mathcal{R}}_{\mathcal{DS}}$	$\mathcal{DS}$ Goodness of Reliability
$\delta\mathcal{R}_{\mathcal{DS}}$	$\mathcal{DS}$ Residual Reliability
S	Flux Density
$\hat{S}$	S/N
$r_{n\sigma}$	$n\sigma$ Flux Fraction
$s_{n\sigma}$	$n\sigma$ (Flux) Scatter $(1 - r_{n\sigma})$

Table 20 Source finder colour annotation.

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

B CERBERUS CODE TEMPLATE NOTES

In this appendix we provide more architectural details regarding Cerberus template rules discussed in § 2.1.2. Enough detail is provided to get an overall sense of Hydra's extensible nature. Further details can be found in the Hydra user manual.

Figure 62 shows a more expanded view of the Cerberus code generation workflow give in Figure 5. 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 source finders, 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.

B.1 Containerization

The general recipe for containerizing source finders is as follows.

- Create a Docker build file containing the following:
  - Base operating system environment
  - Source finder environment with tools
  - Source finder 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: *i.e.*,
    - Script home directory[35]
      - 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

The input and output directories serve as external mount points, used by the `cerberus.py` wrapper script: *i.e.*, the input directory contains the input image, the processing directory contains the source finder wrapper script scratch files, and the results directory contains output catalogues, region files, *etc.* The container `ENTRYPOINT` allows `cerberus.py` external access to the internal script. Aegean is perhaps one of the simplest source finders to containerize: Figure 63 shows the details.

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, *e.g.*,

[35] Used for software development and testing.

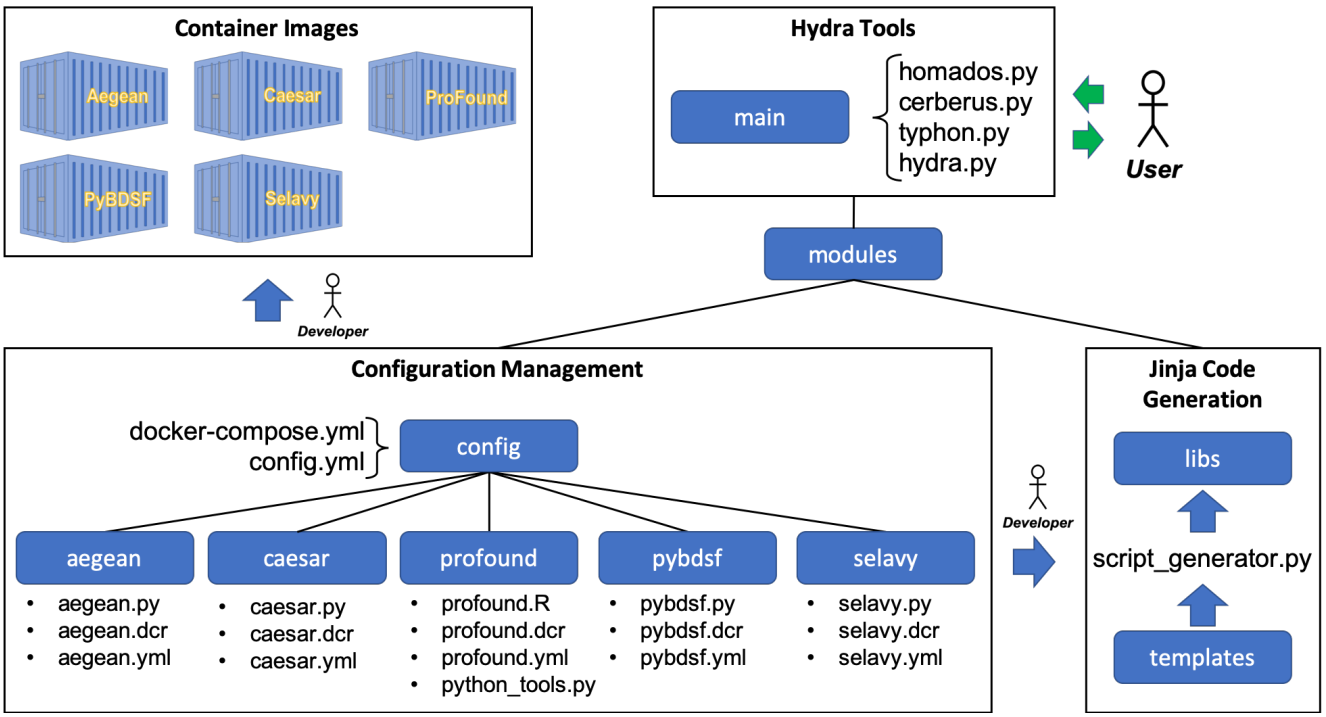


Figure 62. Detailed breakdown of the Cerberus code generation workflow given in Figure 5. Figure 63 shows a detailed example of an Aegean container image build, utilizing `aegean.dcr`, `aegean.py`, and `docker-compose.yml`. Figure 64 shows an example of updating `cerberus.py` to include Aegean, through code generation, utilizing `aegean.yml` and `config.yml`. All of the information in Configuration Management is accessible to all of the tools within the Hydra software suite.

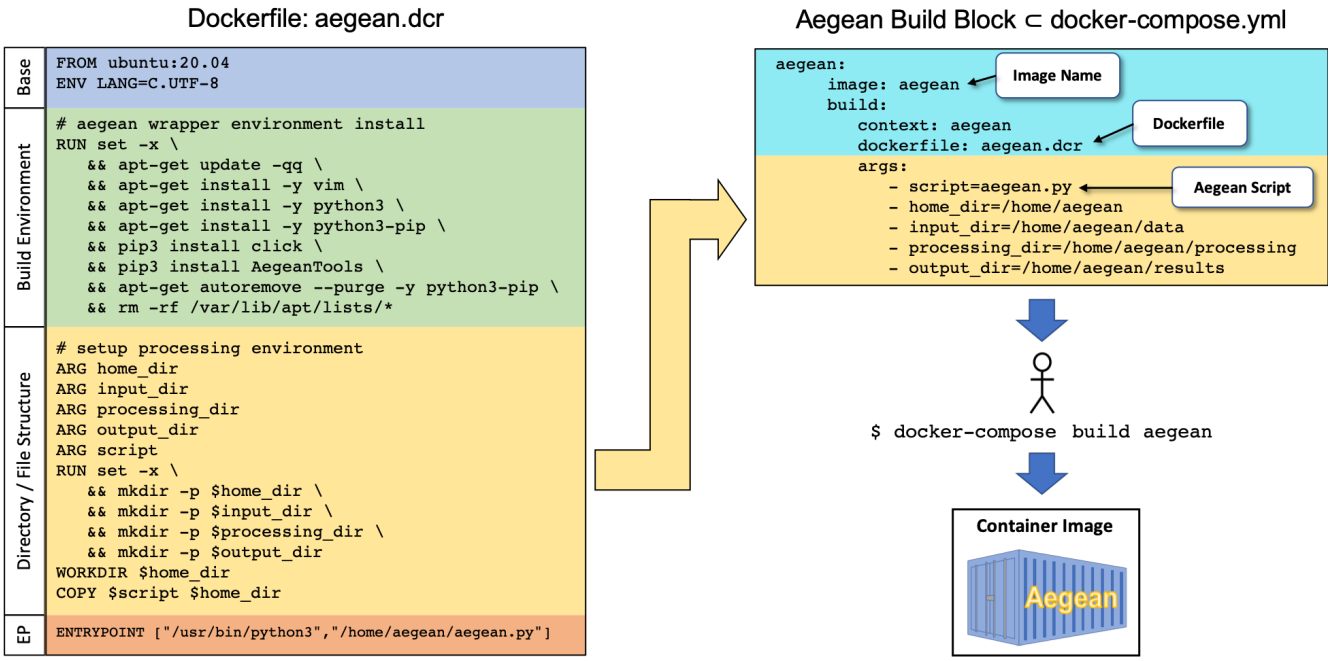


Figure 63. Example of Aegean containerization. The Dockerfile, `aegean.dcr`, has four main parts, a base Ubuntu 20.04 operating system, an `AegeanTools` toolbox build environment, a `home_dir`, `input_dir`, `processing_dir`, and `output_dir` directory structure along with a local `script`, `aegean.py`, and 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, a part which defines the image name (`aegean`) and a pointer the Dockerfile (`aegean/aegean.dcr`), and a part which contains the directory file structure to be built and a pointer to `aegean.py`. The container image is built, using this information, *via* the `docker-compose` command.

```
/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, *i.e.*,

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

the latter being used by `cerberus.py` (*cf.*, § 2.1.2). So, after implementing the above template rules, which we have demonstrate by example, a new container image can be built using the `docker-compose` command within the `config` directory.

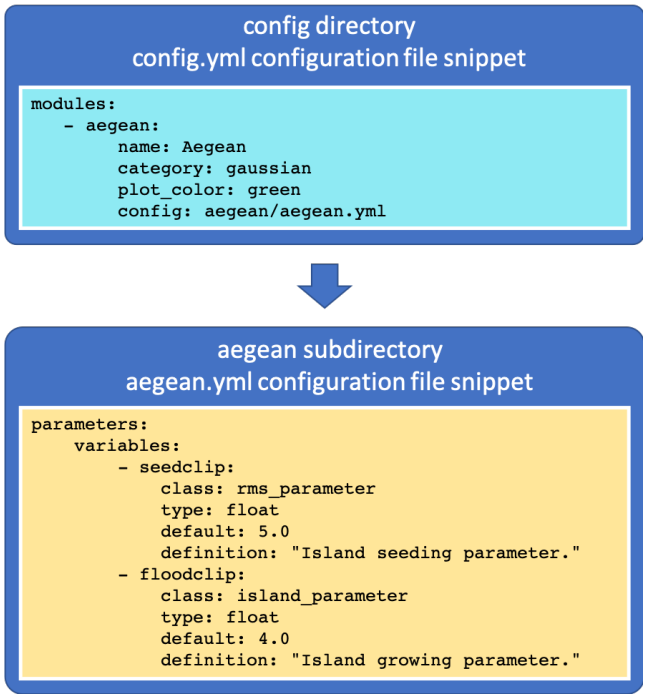


Figure 64. Example of the addition of an Aegean module for `cerberus.py` through code generation. In short, the developer creates an `aegean.yml` medadata file and links it to the master configuration file, `config.yml`. Then by running the `script_generator.py` script, in the `libs` (libraries) directory, the module is installed.

B.2 Code Generation

For code generation a medadata 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, *via* the Jinja template engine. Figure 64 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` utilizes the docker-compose configuration file for linking calls to the source-finder container images, `typhon.py` utilizes the medadata files for parameters and constraints used for source-finder optimization, `hydra.py` utilizes the medadata files for catalogue processing, and so on.

C SOURCE FINDER IMPLEMENTATION NOTES

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

In should be noted that Aegean, Caesar, and Selavy have various multiprocessor mode implementations, wherein large images are broken into manageable chunks, in order to reduce processing time (see, Hancock et al., 2018; Riggi et al., 2019; Whiting & Humphreys, 2012). For this particular 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 (*cf.*, *op. cit.*); especially, when comparing with non-multiprocessor source finders, such as, ProFound and PyBDSF.

C.1 Aegean

The `AgeanTools` toolbox contains to main items of interest, 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 estimates are done *via* sigma-clipping; whereas, Aegean uses a fast, but less accurate, zones algorithm. Aegean can also use the output from BANE, which is the implementation adopted here.

The Aegean source finder uses a flood-fill algorithm to determine islands, whereupon it uses a de-blending process to determine the number of local maxima, through

the discrete 2D Laplacian (*i.e.*, curvature) kernel

$$L^2_{xy} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad (9)$$

for localized fitting of Gaussian components (Hancock et al., 2012). Its flood-fill algorithm utilizes 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, $\sigma_s \geq \sigma_f$ (*cf.*, Table 1). It then convolves the image with Equation 9, producing a curvature map, wherein it constrains Gaussian fits to local depressions (*i.e.*, negative curvature bowls, corresponding to local maxima) within the islands.

The implementation of the Aegean module for Cerberus exposes the box-car and flood-fill parameters.

C.2 Caesar

Caesar does its source finding using a flood-fill method to obtain blobs (*i.e.*, *"islands"*), from which child-blobs (*i.e.*, *"nested blobs"*) are (optionally) extracted using elliptical-Gaussian based Laplacian (*cf.*, Equation 9) or χ^2 filters (Riggi et al., 2016, 2019). Compact sources, *i.e.*, childless-blobs, are subtracted out leaving 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 comes in two sets, one for the parents, `seedThr` and `mergeThr`, respectively, and one for the children, `nestedBlobPeakZThr` and `nestedBlobPeakZMergeThr`, respectively. As we are optimizing these parameters externally (*a la* PRD, Equation 2), 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 their parents (*cf.*, 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).

In 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 contrasting with other source finders. Regardless, the only hit we are taking here – in principle – is processing time.

Caesar also does background and RMS noise optimization through either of the following metrics, μ/σ, median/MADFM, biweight, and clipped median/σ. As such, 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-optimization scheme option used by Typhon.

Table 21 summarizes 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`).

Table 21 Caesar 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`

C.3 ProFound

ProFound uses a watershed de-blending 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 § 1.2), while assigning overlapping segment fluxes to the ones with the most flux: *e.g.*, Figure 65. 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 astronomy and so there are some nuances when it comes to applying it to radio (see, Hale et al., 2019). Table 22 summarizes all

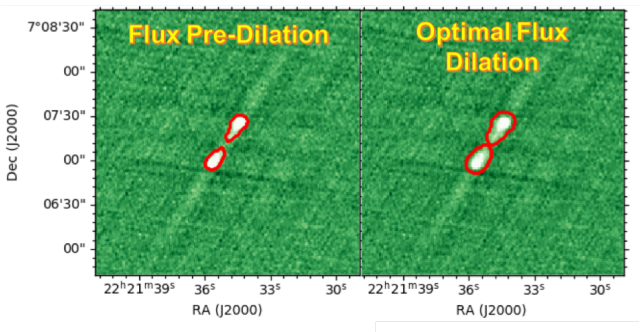


Figure 65. Example of ProFound dilation process of a small $3' \times 3'$ VLASS Epoch 1.1 Quick Look (QL) image cutout centered at J222135+070712. The cutout was extracted from a QL image tile, available at NRAO (https://science.nrao.edu), and then processed using ProFound in R Studio.

of the settings internal to the ProFound module,[36] 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.

C.4 PyBDSF

PyBDSF determines islands by collecting pixels greater than a given flux threshold, `thresh_pix`, and then expands outwards from those pixels in octets above a given island-boundary threshold, `thesh_isl`;[20] wherein, our RMS and island parameters correspond to former and latter thresholds, respectively. After the island have been determined, it performs multiple Gaussian fits to each island (*cf.*, Hancock et al., 2012).

We follow the same generic recipe as Hale et al. (2019) to accommodate extended sources, as outlined in Table 23. The `atrous_do` selects the à trous wevelet decomposition (Holschneider et al., 1989) module as one of several options for post processing: *i.e.*, shapelet decomposition, à trous wavelet decomposition, PSF variation, polarization, 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 exposes the `rms_box` tuple, so that the RMS box and step sizes can be optimization by Typhon.

C.5 Selavy

Selavy is a *"single-pass"* raster-scan, or thresholding, type source finder (*cf.*, Lutz, 1980), with Duchamp (Whiting, 2012), a 3D source finder, at its heart (Whiting & Humphreys, 2012). Here we are interested in its 2D spatial search features (*i.e.*, `searchType = spatial`). The algorithm works downwards by growing regions of detection through a threshold parameter (see Figure 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: *i.e.*, `doFit = True`, to turn the fitting option on, `fitTypes = [full]`, to fit all degrees of free-

[36] See ProFound footnote *c* in Table 1.

Table 22 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 radically differ between versions of ProFound and R. Also the default settings mentioned in the documentation can differ considerably from what is actually in the source code. As such, this table includes *"all"* settings that are deemed important to 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 23 PyBDSF module settings.

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

dom, and `numGaussFromGuess = True`, to provide an initial guess from the number of distinct peaks found within a given region during thresholding.[37]

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`), wherein `Selavy.VariableThreshold.boxSize` = (`rms_box`-1)/2, where `rms_box` determined by Typhon.

Table 24 summarizes all of the internal settings of Cerberus's Selavy module.

Table 24 Selavy 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`

D THE CLUSTERING ALGORITHM AND APPLICATION NOTES

Clustering is used for grouping together radio components which have elliptical footprints, or extents, in an overlapping chain-like fashion (Boyce, 2018). This has the advantage of grouping components together, into clumps, that potentially belong to extended sources. In addition, it allows for local comparison between different source finders, deep–shallow detections, injected sources and detections, *etc.* It also serves to determine global cutout sizes, completeness and reliability, and so on.

Two components C_i and C_j are said to overlap if their distance $\Delta(C_i, C_j)$ is less than the overlap of their elliptical footprints to within some skirt factor f: *i.e.*,

$$\Delta(C_i, C_j) \leq f \times (r_i + r_j) \tag{10}$$

is our distance metric, where

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

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

and

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

where a_μ is the semi-major axis (`BMAJ`), b_μ is the semi-minor axis (`BMIN`), and θ_μ is the position angle (`PA`), such that $\mu = i, j$. Herein, we set $f = 1$. Figure 66 outlines the derivation of Equation 10.

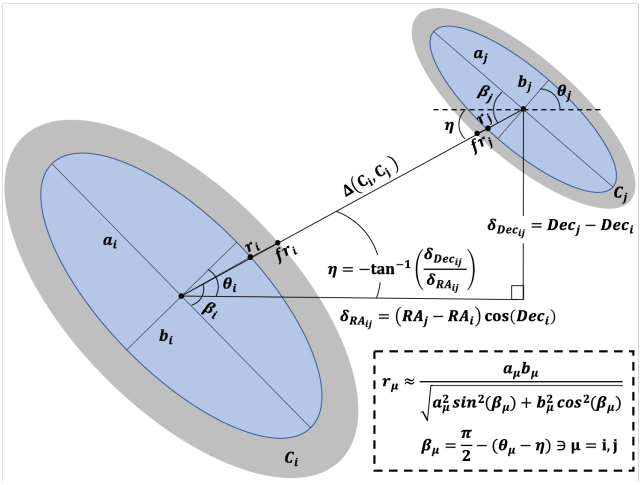


Figure 66. Derivation of the distance metric used for clustering. Here we assume that the space is locally flat, so that $\Delta(C_i, C_j) \approx (\delta^2_{\mathtt{RA}_{ij}} + \delta^2_{\mathtt{Dec}_{ij}})^{1/2}$, where $\delta_{\mathtt{RA}_{ij}} = (\mathtt{RA}_j - \mathtt{RA}_i)\cos(\mathtt{Dec}_i)$ and $\delta_{\mathtt{Dec}_{ij}} = \mathtt{Dec}_j - \mathtt{Dec}_i$. The distances from the centers of components C_i and C_j, along a ray between them, is given by r_i and r_j, respectively: *i.e.*, r_μ, which is a standard geometrical expression in terms of angle $\beta_\mu = \pi/2 - (\theta_\mu - \eta)$ wrt the ray and the semi-major axis a_μ, where θ_μ is the position angle and $\eta = -\tan^{-1}(\delta_{\mathtt{Dec}_{ij}}/\delta_{\mathtt{RA}_{ij}})$. The grey area outside the ellipses is the skirt, whose extent is determined by f. Putting all of this together yields Equation 10.

We now look at applications of clustering, pertaining to cluster and clump catalogues, and completeness and reliability plots. The examples herein, are for pedagogical purposes only.

D.1 The Cluster Catalogue

From a Hydra software implementation perspective, the source finder component catalogue RA, Dec, semi-major, semi-minor, and position angle columns are mapped to internal `ra`, `dec`, `extent_semimajor`, `extent_semiminor`, and `extent_angle` clustering parameters, respectively, through Configuration Management, Figure 62. For example, the PyBDSF output component catalogue[20] columns `RA`, `DEC`, `Maj`, `Min`, and `PA` are mapped to the aforementioned parameters, respectively, with the appropriate `Maj`/2, `Min`/2, and unit conversations. This information is part of the cluster catalogue, indicating clumps of source finder deep–shallow detections (including injected sources) by clump ID. Other common elements, such as, total and peak fluxes are also included.

[37] See Selavy footnote[e] in Table 1.

Figure 67 shows an example of a clump defined by Aegean, ProFound, PyBDSF, and Selavy detections centered about a 3′ square EMU pilot sample cutout: see infogragphic, in Figure 9, for a functional definition of a clump. In this particular example, there are no overlapping shallow detections, otherwise they would be included as part of the clump. Note that we have 10 deep detections in this clump, and so they would be expressed as 10 separate rows in the cluster catalogue, all with the same clump ID in the `clump_id` column, with a `deep` designation for each component in the `image_type` column. Also included, is a reference to the corresponding catalogue, *via* the `catalogue_id`, `source` (source finder or injected catalogue name), and `image_type` (redundant for injected sources) columns.

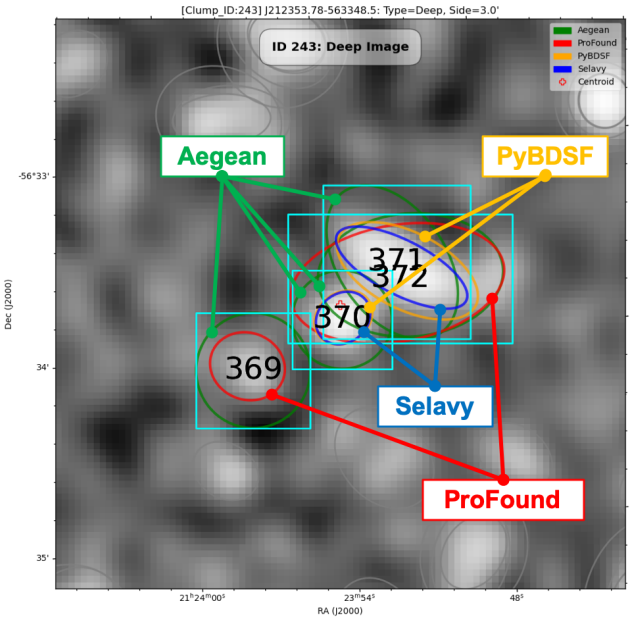


Figure 67. EMU pilot `clump_id` 243 example. The clump consists of 4 Aegean, 2 ProFound, 2 PyBDSF, and 2 Selavy overlapping deep image catalogue components. If there are overlapping components in the shallow catalogues, these would also be part of this clump (*cf.*, Figure 9). This example does not include the Caesar module.

The cluster catalogue also contains subclump ID's (*i.e.*, a `subclump_id` column) indicating deep–shallow clumping within a given deep–shallow source finder catalogue set. Figure 68 shows the subclump decomposition for the example given in the previous figure. Here Aegean's subclump consists of 4 overlapping components, whereas as the rest are 1 component subclumps. There are no shallow detections contributing to the subclumps in this particular example.

The numbers in the center of the cyan boxes of Figures 67 and 68 are the match IDs, indicating the closest deep–shallow source–finder (including injected) catalogue–set matches: see Figure 9, for functional definition of match-set decomposition (*re.* match ID's). They are expressed in the `match_id` column of the cluster table, and provide some indication of completeness and reliability on a local scale. For example, match ID 369 contains one Aegean and one ProFound detection, whereas the other source finders have no detection in this box. So one could carry out completeness and reliability studies between source finders (*cf.*, Figure 57). However, we do not do this here, due to the number of permutations, and so, to first order, we focus on statistics related to injected sources and deep–shallow detections.

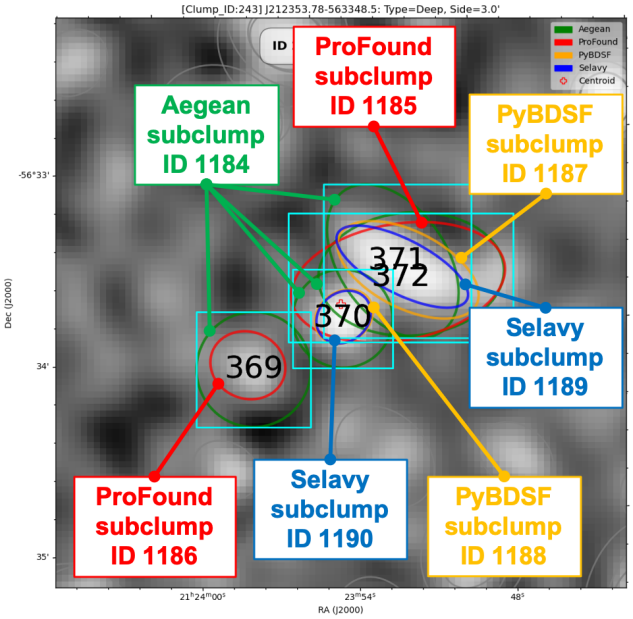


Figure 68. EMU pilot `clump_id` 243 subclumps example: *re.*, Figure 67. Subclumps are defined as deep–shallow clumping for a given source finder (*cf.*, Figure 9). Here `clump_id` 243 is decomposed into 1 Aegean (ID 1184), 2 ProFound (IDs 1185, 1186), 2 PyBDSF (IDs 1187, 1188), and 2 Selavy (IDs 1189, 1180) subclumps. `subclump_id`'s are not shown as part of the cutout annotations, so as to avoid clutter.

Section 3.3 shows a detailed example of a cluster catalogue excerpt, Table 12, for a bent-tail system, Figure 52, along with accompanying Hydra Viewer cutouts, Figures 53 and 54. Figure 9 is provided as a guide to aid in the translation of information between the table and the cutouts.[38]

D.2 The Clump Catalogue

The table at the bottom of Figure 8 shows most of the information stored in the cluster table, that can be used for analysing the performance of source finders on a

[38] NB: Individual on/off annotation control for each source finder is not supported in the current version of the Hydra Viewer: *i.e.*, it is all on or all off. Unfortunately, this can be disconcerting when there are too many overlapping elements.

local scale: *i.e.*, residual cutout RMS, MADFM, and ΣI^2 per unit area ($'^2$), *etc.* The clump table consists of rows, by unique `clump_id`'s, of cluster centroid positions, cutout sizes ($'$), total number of components, number of components per source finder, source finders with the best residual statistics, *etc.* Figure 69 shows an example of a deep clump distribution, extracted from the clump table; accessible through the Hydra Viewer, or its tarball archive.

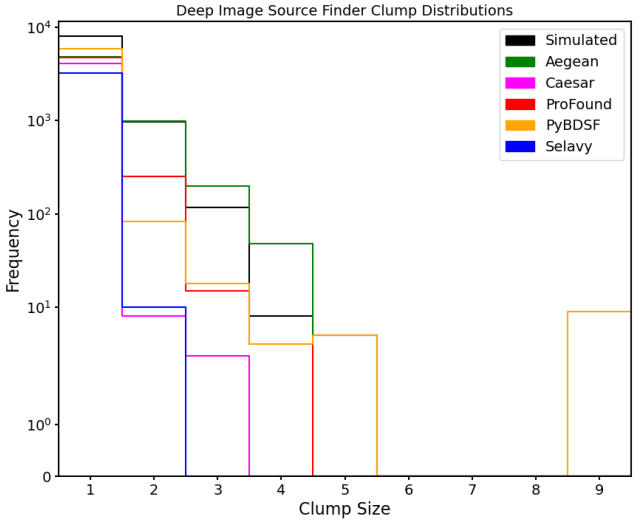


Figure 69. Example of a clump distribution for a $2^\circ \times 2^\circ$ simulated deep-image. The Clump Size bins can be explored with the Hydra Viewer, *via.* its Mode button (*cf.*, Figure 8).

D.3 Completeness and Reliability

Completeness is the fraction of real detections to real sources, whereas reliability is the fraction real detections to detected sources (*cf.*, Figure 70). Here we define these metrics in terms of *"real"* injected (simulated) sources ($\mathcal{J}$) *vs.* deep and shallow detections ($\mathcal{D}$ and $\mathcal{S}$, respectively), and *"assumed-real"* deep detections *vs.* shallow detections. In the former case, we take the fraction of real–deep or real–shallow detections ($\mathcal{D} \cap \mathcal{J}$ or $\mathcal{S} \cap \mathcal{J}$, respectively) to the injected sources for our completeness,

$$\mathcal{C}_{\mathcal{D}} = \frac{\mathcal{D} \cap \mathcal{J}}{\mathcal{J}} \tag{11}$$

or

$$\mathcal{C}_{\mathcal{S}} = \frac{\mathcal{S} \cap \mathcal{J}}{\mathcal{J}}\,, \tag{12}$$

respectively, and the fraction of real–deep or real–shallow detections to corresponding deep or shallow detections for our reliability

$$\mathcal{R}_{\mathcal{D}} = \frac{\mathcal{D} \cap \mathcal{J}}{\mathcal{D}} \tag{13}$$

or

$$\mathcal{R}_{\mathcal{S}} = \frac{\mathcal{S} \cap \mathcal{J}}{\mathcal{S}}\,, \tag{14}$$

respectively. In the latter case, we take the fraction real–shallow detections ($\mathcal{S} \cap \mathcal{D}$) to deep detections for our completeness,

$$\mathcal{C}_{\mathcal{DS}} = \frac{\mathcal{S} \cap \mathcal{D}}{\mathcal{D}}\,, \tag{15}$$

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

$$\mathcal{R}_{\mathcal{DS}} = \frac{\mathcal{S} \cap \mathcal{D}}{\mathcal{S}}\,. \tag{16}$$

Figure 70 provides a visual representation of all of the aforementioned quantities.

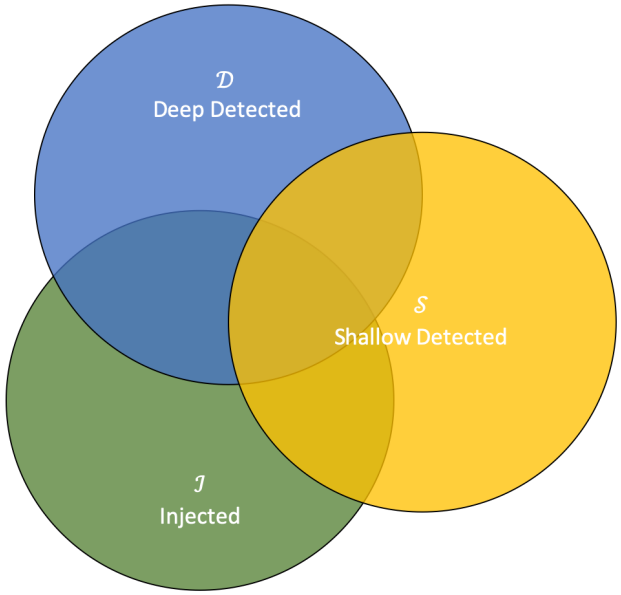


Figure 70. Venn diagram of completeness and reliability, for sets of deep ($\mathcal{D}$), shallow ($\mathcal{S}$), and injected ($\mathcal{J}$) sources.

We can even 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,

$$\tilde{C}_{\mathcal{DS}} = \frac{(\mathcal{D} \cap \mathcal{J}) \cap (\mathcal{S} \cap \mathcal{J})}{\mathcal{D} \cap \mathcal{J}}\,, \tag{17}$$

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

$$\tilde{R}_{\mathcal{DS}} = \frac{(\mathcal{D} \cap \mathcal{J}) \cap (\mathcal{S} \cap \mathcal{J})}{\mathcal{S} \cap \mathcal{J}}\,. \tag{18}$$

Table 25 summaries all the aforementioned completeness and reliability metrics, *re.*, Equations 11 through 18.

We perform these calculations by computing overlaps between input and detection extents, to obtain real detections, through spatial clustering, *vis-à-vis* Equation 10.

Figure 71 shows examples of deep–shallow source component overlaps, $\mathcal{S} \cap \mathcal{D}$, *re.*, $\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$. Matches are done pair-wise, within clumps, between the closest centers of overlapping deep–shallow components. This method is more precise than the conventional $\sim \mathcal{O}(3'')$ separation cutoff method (*cf.*, Hopkins et al., 2015; Riggi et al., 2019), as it ensures source components always overlap. The $(\mathcal{S} \cap \mathcal{D}) : \mathcal{D}$ and $(\mathcal{S} \cap \mathcal{D}) : \mathcal{S}$ ratios are then binned with respect to the $\mathcal{D}$ and $\mathcal{S}$ fluxes, respectively, producing $\mathcal{C}_{\mathcal{DS}}$ *vs.* $\mathcal{D}$ completeness and $\mathcal{R}_{\mathcal{DS}}$ *vs.* $\mathcal{S}$ reliability step-plots.

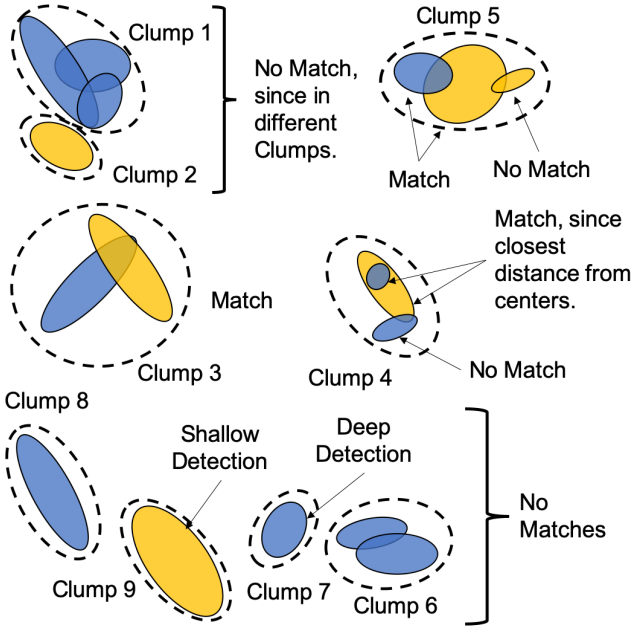


Figure 71. Examples of deep (blue) and shallow (amber) source component overlaps, *re.*, $\mathcal{C}_{\mathcal{DS}} = (\mathcal{S} \cap \mathcal{D})/\mathcal{D}$ and $\mathcal{R}_{\mathcal{DS}} = (\mathcal{S} \cap \mathcal{D})/\mathcal{S}$. Real-shallow detections are indicated by overlapping pair-wise deep-shallow detections ($\mathcal{S} \cap \mathcal{D}$), whose centers are closest. The dash-lines indicate clumps of component extent overlays. Recall $f = 1$ (*re.*, Equation 10), so clumps 1 and 2 are not connected.

One can explore S/N bins of $\mathcal{C}_{\mathcal{DS}}$ and $\mathcal{R}_{\mathcal{DS}}$ step-plots through the Hydra Viewer, Figure 8. Figure 72 shows an example snippet for Aegean in Completeness Mode. As it can be seen, clustering is powerful tool in terms of computing quantities, such as, completeness and reliability, and grouping the information accordingly, for close visual inspection of image cutouts and examination of local parameters, *vis-à-vis* the cluster catalogue.

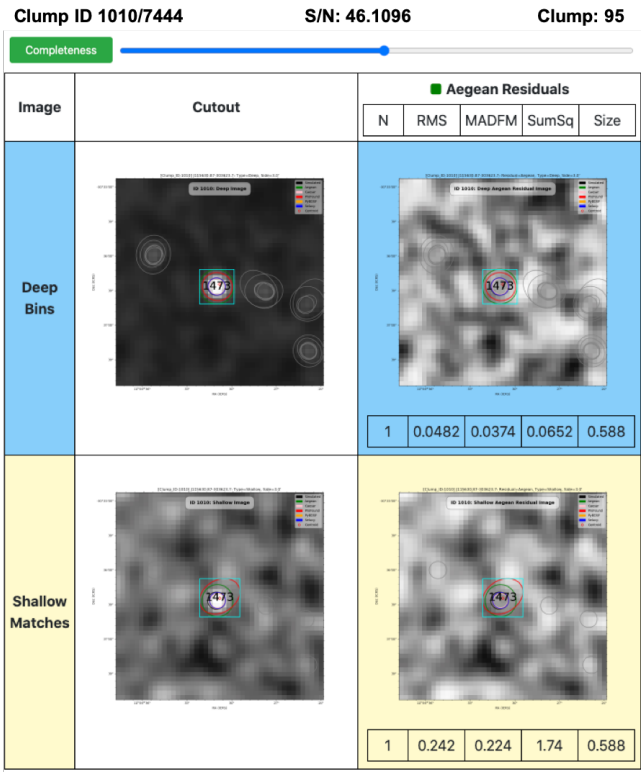


Figure 72. Example snippet of the Hydra Viewer in Completeness Mode. The slider is currently set for exploring the 46.1096 S/N bin of the deep-shallow completeness plot, which contains 95 clumps. The user can navigate the clumps within this bin by using the left and right arrow keys, or the Previous and Next buttons (not shown). The blue and yellow colour codes indicate the inputs (Deep Bins) and detections (Shallow Matches), respectively. The colours are flipped in Reliability Mode (with the Bins and Matches annotations interchanged).

E SOURCE FINDER SPEED PERFORMANCE ANALYSIS

Figure 73 shows software PRD optimization (top) contrasted with CPU times (bottom) for the different source finders. Aegean, Caesar, and Selavy have multiproces-

Table 25 Completeness/reliability metrics, *vis-à-vis* Figure 70, in terms of deep ($\mathcal{D}$), shallow ($\mathcal{S}$), and injected ($\mathcal{J}$) sources.

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

sor capabilities. These features are turned off so as to place all source finders on equal footing for performance comparison purposes. The assumption being, that they should perform as well as the non-multiprocessing source finders for reasonably sized images. Also, in multiprocessor mode, these source finders slice up images and marshal them for processing, which biases the statistics (Whiting & Humphreys, 2012; Hancock et al., 2018; Riggi et al., 2019). This was the primary reason for using single processor mode.

For the simulated images, ProFound and PyBDSF compute the PRD, on par, on $\mathcal{O}(90)\,s$. For real images they are on $\mathcal{O}(40)\,s$. Aegean is on $\mathcal{O}(300)\,s$ for simulated point-sources, $\mathcal{O}(600)\,s$ for simulated extended-sources, and $\mathcal{O}(200)\,s$ for real sources. For the simulated images, Caesar and Selavy compute the PRD, on par, on $\mathcal{O}(1,000)\,s$ for point-sources, and $\mathcal{O}(4,000)\,s$ for extended-sources. For real sources, Selavy runs on $\mathcal{O}(3,000)\,s$, while Caesar drops down to $\mathcal{O}(500)\,s$. The PRD CPU time is the time to process both the non-inverted and inverted images (*re.* Equation 2).

These source finders vary in CPU times over 3 decades: ProFound and PyBDSF in the 10's, Aegean in 100's, and Caesar and Selavy in the 1,000's. Caesar and Selavy are more sophisticated than the others source finders, and are tuned for optimal performance, at the expense of a CPU hit. PyBDSF is more of the traditional kind of source finder,[20] whereas Aegean is NxGen (Hancock et al., 2018). ProFound has its origins in optical astronomy, and is novel in its use of irregularly shaped apertures (Robotham et al., 2018).

Firstly, it is surprising that ProFound outperforms all of the radio based source finders. The authors have stated that it was specifically developed in R for efficiency (Robotham et al., 2018). Secondly, it is surprising that Caesar is 2 orders of magnitude slower than ProFound, given their similarities. Granted, it does a two stage operation, deblending in a similar manner to ProFound, and then parent-blob processing, which one would reasonably expect to run on $\mathcal{O}$(PyBDSF)-time (*cf.*, Riggi et al., 2016). However, one would only expect a 2–3 times reduction in processing time. In short, it looks like there is considerable room for improvement.

Selavy is up there with Caesar in CPU times. This should not be the case, as its not as sophisticated. However, it is novel in that is uses Duchamp at its core (Whiting, 2012; Whiting & Humphreys, 2012), which is a data cube (3D) source finder. This could be its *Achilles' heel*. Aegean is even less sophisticated.

In short, there appears to be definite room for improvement for the aforementioned multiprocessor based source finders – in single thread mode. In turn, one would expect this should dramatically increase their performance in multiprocessor mode.

REFERENCES

AMI Consortium: Franzen T. M. O., et al., 2011, MNRAS, 415, 2699

AMI Consortium: Zwart J. T. L., et al., 2008, MNRAS, 391, 1545

Adams T. J., Bunton J. D., Kesteven M. J., 2004, Exp. Astron., 17, 279

Akhlaghi M., Ichikawa T., 2015, ApJS, 220

Astropy Collaboration et al., 2013, A&A, 558, A33

Astropy Collaboration et al., 2018, AJ, 156, 123

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, and N.P.F. Lorente (San Francisco: ASP), p. 725

Becker R. H., White R. L., Helfand D. J., 1995, ApJ, 450, 559

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, Franck, p. 12

Bonaldi A., et al., 2021, MNRAS, 500, 3821

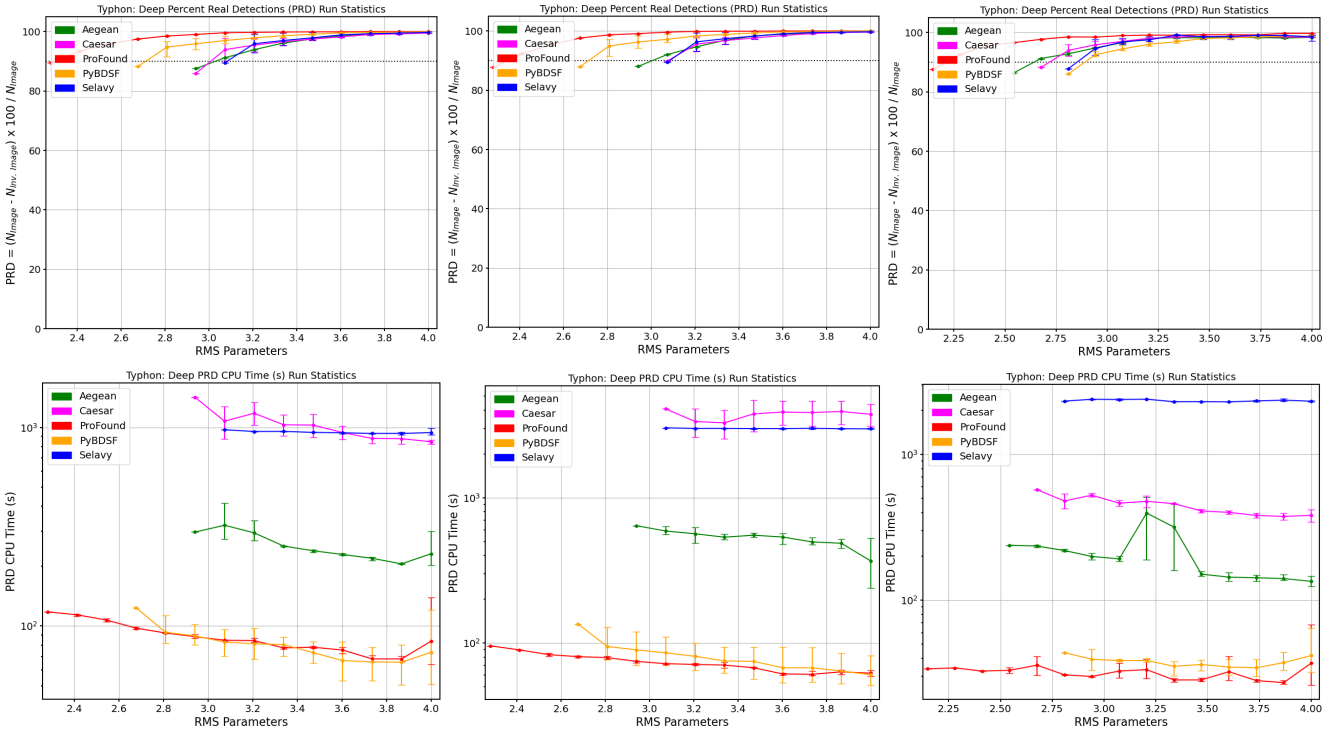


Figure 73. Source finder PRD optimization (top row) and CPU time (bottom row) statistics for simulated-point (left column), simulated-extended (middle column) and real (right column) deep-images. The optimized run statistics are provided in Tables 4, 7, and 11, respectively. The processing was done on a 2GHz 16 core (single threaded) Intel Xeon Processor with 60G of RAM running Ubuntu 10.04.3 LTS – Go Bionic Beaver!

Boyce M., 2018, Technical report, Multiwavelength Database Prototype Design Tech Notes. Appendix B: Clustering Algorithm. CIRADA (Internal Report)

Boyce M., 2020, Technical report, Source Extraction Comparison Summary Report, Analysis Requirements Specification Summary. CIRADA (Inernal Report)

Buxton R., 2016, The Complete World of Greek Mythology. Thames and Hudson Ltd, London

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

Croft S., Bower G. C., Keating G., Law C., Whysong D., Williams P. K. G., Wright M., 2011, ApJ, 731

Da Costa G. S., 1992, Astronomical CCD Observing and Reduction Techniques, ASP Conference Series, ed. Steve B. Howell, 23, 90

DeBoer D. R., et al., 2017, PASP, 129

Fender R., et al., 2007, PoS

Gordon Y. A., et al., 2020, RNAAS, 4

Greisen W., 2017, AIPS, Memo 117

Hale C. L., Robotham A. S. G., Davies L. J. M., Jarvis M. J., Driver S. P., Heywood I., 2019, MNRAS, 487

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

Hancock P. J., Cathryn M. T., Hurley-Walker N., 2018, PASA, 35

Helfand D. J., White R. L., Becker R. H., 2015, ApJ, 801

Holschneider M., Kronland-Martinet R., Morlet J., Tchamitchian P., 1989, Proceedings of the International Conference. Springer-Verlag.

Hopkins A. M., Miller C. J., Connolly A. J., Genovese C., Nichol R. C., Wasserman L., 2002, AJ, 123, 1086

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

Hurley-Walker N., et al., 2017, MNRA, 464, 1146

Huyn M. T., Hopkins A., Norris R., Hancock P., Murphy T., Jurek R., Whitting M., 2012, PASA, 29

Intema H. T., Jagannathan P., Mooley K. P., Frail D. A., 2017, A&A

Jarvis M. J., et al., 2018, PoS, MeerKAT2016, 006

Johnston S., et al., 2007, PASA, 24, 174

Johnston S., et al., 2008, Exp. Astron., 22, 151

Jonas J. L., 2009, IEEEP, 97, 1522

Jonas J., 2018, PoS, MeerKAT2016, 001

Kron R. G., 1980, ApJS, 43

Lacy M., et al., 2020, PASP, 132

Lonsdale C. J., et al., 2009, Proc. IEEE, 97, 1497

Lutz R. K., 1980, Comput. J., 23

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
Mauch T., Murphy T., Buttery H. J., Curran J., Hunstead R. W., Piestrzynski B., Robertson J. G., Sadler E. M., 2003, MNRAS, 342
McConnell D., et al., 2020, PASA, 37
Mohan N., Raffery D., 2015, Astrophysics Source Code Library, ascl:1107.013
Molinari S., et al., 2010, PASP, 122
Molinari S., Schisano E., Faustini F., Pestalozzi M., di Giorgio A. M., Liu S., 2011, A&A, 530
Murphy T., Mauch T., Green A., Hunstead R. W., Piestrzynska B., Kels A. P., Sztajer P., 2007, MNRAS, 382, 382
Murphy T., et al., 2013, PASA, 30
Norris R., 2017, Nature Astronomy, 1, 671
Norris R. P., et al., 2006, AJ, 132
Norris R., et al., 2011, PASA, 28, 215
Oosterloo T., Verheijen M. A. W., van Cappellen W., Bakker L., Heald G., Ivashina M., 2009, Proceedings of Wide Field Astronomy & Technology for the Square Kilometre Array (SKADS 2009), eds. Chateau de Limelette, Belgium.
Petrosian V., 1976, Astrophys. J. Lett., 210
Powell B. B., 2017, The Poems of Hesiod. University of California Press
Riggi S., et al., 2016, MNRAS, 460
Riggi S., et al., 2019, PASA, 36
Riggi S., et al., 2021, MNRAS, 502
Robotham A. S. G., Davies L. J. M., Driver S. P., Koushan S. P., Taranu D. S., Casura S., Liske J., 2018, MNRAS, 476
Röttgering H., 2003, NewAR, 47, 405
Röttgering H. J. A., 2010, Proceedings of the ISKAF2010 Science Meeting.
Shimwell T. W., et al., 2017, A&A
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
Tingay S. J., et al., 2013, PASA, 30
Umana G., et al., 2015, MNRAS, 454, 902
Wayth R. B., et al., 2015, PASA, 32
Whiting M., 2012, MNRAS, 421, 3242–3256
Whiting M., Humphreys B., 2012, PASA, 29, 371
van Haarlem M. P., et al., 2013, A&A, 556